Современная криптография

Теория и практика

Modern Cryptography

Theory and Practice

Wenbo Mao

Hewlett-Packard Company

Pearson Education
Prentice Hall Professional Technical Reference
Upper Saddle River, New Jersey 07458
www.phptr.com

Современная криптография

Теория и практика

Венбо Мао

Компания Hewlett-Packard

Москва • Санкт-Петербург • Киев
2005

ББК 32.973.26–018.2.75
М24
УДК 681.3.07

Издательский дом "Вильямс"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция канд. физ.-мат. наук *Д.А. Клюшина*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com
115419, Москва, а/я 783, 03150, Киев, а/я 152.

Мао, Венбо.

М24 Современная криптография: теория и практика. : Пер. с англ. — М. : Издательский дом "Вильямс", 2005. — 768 с. : ил. — Парал. тит. англ.

ISBN 5–8459–0847–7 (рус.)

Книга, написанная ведущим специалистом по компьютерной безопасности компании HP Венбо Мао, посвящена актуальным проблемам современной криптографии. Автор критикует "учебные" криптографические алгоритмы и описывает принципы разработки криптосистем и протоколов повышенной стойкости. В ней изложены математические основы криптографии, описаны промышленные стандарты криптографических протоколов, включая IPSec, IKE, SSH, TLS (SSL) и Kerberos, приведены формальные доказательства сильной стойкости практичных схем шифрования, цифровой подписи, зашифрованной подписи и аутентификации, а также проанализированы протоколы с нулевым разглашением. Книга предназначена для профессионалов в области криптографии и компьютерных систем защиты информации.

ББК 32.973.26–018.2.75

ISBN 5–8459–0847–7 (рус.)
ISBN 0–13–066943–1 (англ.)

Оглавление

Содержание

Список иллюстраций

Список алгоритмов, протоколов и атак

Посвящается

$$\text{Ronghui} \parallel \text{Yiwei} \parallel \text{Yifan}$$

$$= (\text{荣卉} \| \text{弋桅} \| \text{依帆}) \oplus 101101110\cdots$$

Аннотация

Во многих книгах по криптографии описываются так называемые основные, или учебные, криптографические схемы и протоколы. Особенно это касается схем и протоколов, основанных на алгоритмах с открытым ключом. В этой книге принят иной подход: в ней намного больше внимания уделяется *прикладным* аспектам криптографии. В частности, здесь объясняется, почему "учебная криптография" пригодна только для идеального мира, где все данные являются случайными, а плохие парни ведут себя смирно. Кроме того, в ней описаны многочисленные атаки на "учебные" схемы, протоколы и системы, основанные на вполне реальных сценариях. Эти примеры демонстрируют неприемлемость "учебной криптографии" для реальных приложений. В книге выделен и подробно описан набор криптографических схем, протоколов и систем, многие из которых стали стандартными, детально анализируются принципы, на которых они основаны, обсуждается их практическая полезность и проверяется уровень их стойкости, в том числе путем формального доказательства. Особый раздел книги посвящен теоретическим основам современной криптографии.

Предисловие

Человечество вступило в эру, когда деловая переписка, финансовые транзакции и обмен правительственными документами все чаще осуществляется с помощью открытых компьютерных систем связи, таких как Internet. Любой человек, где бы он ни находился, может воспользоваться этими возможностями. Это является огромным преимуществом компьютерных сетей. Перечислим некоторые области деятельности, в которых использовался, используется или может использоваться оперативный обмен данными.

> Банковское дело, оплата счетов, электронная коммерция, биржевые торги, аукционы, уплата налогов, азартные игры, микроплатежи (т.е. оплата за отдельную загрузку файла), электронная идентификация, дистанционный доступ к медицинским записям, корпоративные сети, хранение и обработка секретных данных, сертифицированная доставка документов, безопасный обмен конфиденциальными документами, заключение контрактов, программная отметка времени (time-stamping), подтверждение подлинности документов, голосование, развлечения, лицензирование, заказ и продажа билетов, интерактивные игры, цифровые библиотеки, управление правами доступа, борьба с пиратством. . .

Читатель может продолжить этот список.

Все это возможно, только если обеспечена безопасность обмена данными через открытые сети. Применение криптографии позволяет эффективно решить эту проблему. Наиболее распространенными криптографическими средствами, обеспечивающими безопасность связи, являются шифрование, цифровые подписи и аутентификация пользователя на основе пароля. Однако, как мы неоднократно убедимся в дальнейшем, даже самые надежные криптографические средства имеют недостатки, которые могут привести к серьезным последствиям для безопасности данных. Более того, во многих приложениях, подобных перечисленным выше, основные криптографические средства не обеспечивают достаточной степени защиты.

С течением времени электронная коммерция, деловая активность и сфера дистанционных услуг усложняются, предъявляя к средствам защиты данных, пере-

даваемых через открытые сети, все более высокие требования.[1] Таким образом, для проектирования, реализации, анализа и поддержки систем защиты информации и криптографических протоколов необходимо все больше профессионалов. К их числу относятся как системные администраторы, системные инженеры и разработчики программного (аппаратного) обеспечения, предназначенного для защиты информации, так и криптографы.

На протяжении последних лет, работая консультантом по защите информации и криптографическим системам в компании Hewlett-Packard Laboratories в Бристоле (Bristol), автор обнаружил, что требования, предъявляемые к профессионалам в этой области, возрастают, а количество специалистов сокращается. В результате во многих случаях за проектирование и реализацию систем защиты безопасности и криптографических протоколов берутся дилетанты, ориентированные на решение прикладных задач и мало смыслящие в криптографии. Это происходит потому, что разработка криптографических систем и протоколов представляет собой трудную задачу даже для опытного криптографа.

Работа, которой занимался автор, предоставила ему исключительные возможности для исследования многих систем защиты информации и криптографических протоколов. Некоторые из них были разработаны дилетантами в этой области, хотя и предназначались для серьезных приложений. Зачастую автор обнаруживал в таких системах черты "учебной" криптографии. Как правило, это проявлялось в том, что разработчики использовали криптографические алгоритмы и схемы так, как они описаны в учебниках. Непосредственное шифрование (direct encryption) пароля (секретное число, состоящее из небольшого количества цифр) с помощью одного из основных алгоритмов шифрования с открытым ключом (например, "RSA") представляет собой типичный пример "учебной" криптографии. Обнаруживая следы "учебной" криптографии в серьезных приложениях, автор убедился, что многие люди, занимающиеся разработкой и реализацией систем защиты информации, не понимают грозящей им опасности.

Для того чтобы создавать надежные системы защиты, мало владеть методами "учебной" криптографии. Именно поэтому автор написал учебник по "неучебной" криптографии. Основные цели книги перечислены ниже.

- Описать широкий спектр криптографических алгоритмов, схем и протоколов, уделив особое внимание *практическим*, а не учебным версиям.

---

[1] Компания Gartner Group прогнозирует, что в 2004 г. суммарный доход от электронных форм деловой активности, получаемый в результате сделок между компаниями (business to business — B2B), а также между компаниями и поставщиками (business to consumer — B2C), в Европейском Союзе достигнет 2,6 трлн. долл. [5] (с вероятностью 0,7), что в 28 раз превышает уровень 2000 г. [5]. Кроме того, компания eMarketer [104], сообщает, что в 2002 г. потери финансовых организаций в США вследствие краж, осуществленных в результате несанкционированного доступа в компьютерные сети, составили 1,4 млрд. долл., а прогнозируеый уровень годичного прироста потерь может достичь 29%.

- Продемонстрировать слабость методов “учебной” криптографии на примерах многочисленных атак, описав типичные приемы их проведения.
- Изложить основные принципы разработки, анализа и реализации криптографических систем и протоколов, сделав акцент на стандартах.
- Изучить формальные методы доказательства стойкости и практической пригодности криптографических систем и протоколов.
- Подробно изложить теоретические основы современной криптографии для читателей, желающих иметь систематизированные знания.

Обзор

Современная криптография — обширная область знаний, сложившаяся в результате интенсивных исследований на протяжении последних тридцати лет. Книга посвящена одному из аспектов криптографии: описанию прикладных криптографических схем и протоколов, а также исследованию их стойкости.

Книга состоит из шести разделов.

Часть I содержит две главы (1, 2), представляющие собой элементарное введение в криптографию и основы защиты информации. В главе 1 демонстрируется эффективность криптографии при решении сложных задач, связанных с обменом данными. В ней рассматривается простой криптографический протокол “подбрасывания монеты по телефону” (первый из описанных в книге протоколов). Далее в главе изучаются культурные и экономические вопросы, связанные с криптографией. В главе 2 на примере целого ряда простых протоколов аутентификации (authentication) продемонстрирован неприятный факт: от ошибок никто не застрахован.

Поскольку первая часть книги является довольно элементарной, она предназначена для новичков.

Часть II состоит из четырех глав (3–6), содержащих математические основы криптографии. Она представляет собой самостоятельное описание математических понятий, используемых в книге. Читатели, которых интересует лишь ответ на вопрос “Как?”, т.е. способы использования прикладных криптографических схем и протоколов, могут пропустить эту часть без особого ущерба для понимания остальных глав. Читатели, которых интересует ответ на вопрос “Почему?”, т.е. причины стойкости этих схем и протоколов, найдут в этой части книги достаточно интересный материал. Представляя основные принципы криптографических схем и протоколов, мы будем выявлять слабые места одних и доказывать преимущества других. В процессе доказательства мы всегда будем ссылаться на конкретные математические факты, приведенные в части II.

Эту часть книги можно также использовать в качестве систематизированного учебника по теоретическим основам современной криптографии.

Часть III состоит из четырех глав (7–10). В них описаны основные криптографические алгоритмы, а также способы обеспечения конфиденциальности данных и защиты их целостности. Глава 7 посвящена симметричным криптографическим схемам. Глава 8 описывает средства несимметричной криптографии. В главе 9 рассматривается степень защиты, характерная для основных асимметричных криптографических функций, применяемых в идеальном мире, в котором данные являются совершенно случайными. Глава 10 содержит способы обеспечения целостности данных.

Поскольку схемы и методы, рассматриваемые здесь, имеют фундаментальный характер, многие из них фактически относятся к категории "учебных" и, следовательно, являются *ненадежными*. Чтобы выявить слабые места описанных схем, демонстрируются многочисленные примеры атак на них. Для читателей, которые не планируют глубоко изучать практическую криптографию, обеспечивающую повышенный уровень защиты, эта часть должна послужить тревожным сигналом, свидетельствующим об общей ненадежности "учебных" схем.

Часть IV состоит из трех глав (11–13). Они посвящены аутентификации, представляющей собой важное понятие прикладной криптографии и защиты информации. В этих главах рассмотрен широкий спектр тем. Глава 11 включает технические основы, принципы, несколько основных протоколов и стандартов, распространенные приемы атак и средства их предотвращения. В главе 12 рассматриваются четыре хорошо известные протокольные системы аутентификации и их применения в реальных приложениях. Глава 13 содержит описание современных и оригинальных приемов, пригодных для применения в открытых системах.

Эта часть будет особенно полезной для практиков, например, для администраторов систем защиты данных на производстве, а также разработчиков программного и аппаратного обеспечения, заботящихся о безопасности информации.

Часть V содержит четыре главы (14–17). В них описаны средства, позволяющие формально и строго обосновать стойкость средств криптографии с открытым ключом (шифрование (encryption), цифровая подпись (signature) и зашифрованная подпись (signcryption)), а также формальные методы анализа протоколов аутентификации. В главе 14 вводятся формальные определения понятий, связанные с повышенной стойкостью защиты (strong security properties). Следующие две главы содержат прикладные аналоги "учебных" криптографических схем, введенных в части III, сопровождаемые формальным доказательством их сильной стойкости. Глава 17 описывает методы

формального анализа протоколов аутентификации, выходящие за рамки методов, рассмотренных в части IV.

Часть VI завершает книгу. Она состоит из двух глав (18, 19), содержащих техническую информацию, и краткого заключения (глава 20). Основная информация изложена в главе 18. В ней рассматривается класс криптографических протоколов, известных под названием *протоколов с нулевым разглашением* (zero-knowledge protocols). Эти протоколы являются важной частью систем защиты информации в разнообразных приложениях, связанных с электронной коммерцией и виртуальным бизнесом. Они предназначены для верификации заявленных свойств секретных данных (например, для проверки соответствия некоторым требованиям) и обеспечения строгой конфиденциальности по отношению к претенденту. Протоколы с нулевым разглашением вводятся для иллюстрации разнообразных реальных ситуаций, в которых нарушается конфиденциальность, целостность, аутентификация и возможно отречение от авторства. В заключительной технической части книги (глава 19) завершается решение той задачи, которую мы сформулировали в первой части: "подбрасывание монеты по телефону". В результате мы получим протокол, обладающий как очевидной стойкостью, так и эффективностью, позволяющей применять его в реальных приложениях.

Излишне говорить, что описание каждой прикладной криптографической схемы или протокола должно начинаться с обоснования причины, по которой непригоден ее "учебный" аналог. Эти причины мы всегда будем демонстрировать с помощью атак, выявляющих слабые места этих схем и протоколов. Кроме того, эти описания должны завершаться строгим анализом, подтверждающим заявленную стойкость предложенных решений. Таким образом, при описании атак и слабых мест защиты в некоторых частях книги мы неизбежно должны использовать математические и логические утверждения, выводы и преобразования.

Хотя считается, что овладеть методами прикладной криптографии в короткий срок и без усилий невозможно, наша книга — не научное сочинение, интересное лишь специалистам в области криптографии. Все факты, описанные и проанализированные в книге, криптографам хорошо известны. Однако автор полагает, что эти темы вполне доступны и неспециалистам, поскольку книга содержит большое количество примеров и пояснений и сопровождается самостоятельным математическим и справочным материалом.

Книга будет полезной следующему кругу читателей.

- Студентам, завершившим или завершающим первый курс обучения по специальности "компьютерные науки", "информатика" или "прикладная математика" и планирующим продолжить свое образование в области защиты информации. Они могут рассматривать эту книгу как учебник по прикладной криптографии.

- Инженерам по защите информации в высокотехнологичных компаниях, отвечающим за проектирование и создание систем защиты информации. Если применение методов "учебной" криптографии в академических исследованиях не слишком опасно, поскольку может вызвать лишь небольшие затруднения, то в реальных информационных системах их использование может привести к катастрофическим последствиям. По этой причине читатели обязаны знать о непригодности книжных криптографических методов в реальной жизни. Более того, эти читатели должны хорошо понимать принципы защиты информации, лежащие в основе прикладных схем и протоколов, а также уметь правильно применять их на практике. Часть, содержащая математические основы криптографии, делает книгу пригодной для самостоятельного обучения.
- Администраторам по защите информации на предприятиях, а также разработчикам программного и аппаратного обеспечения, для которых важны вопросы безопасности. Для этих читателей часть I является простым и элементарным курсом, посвященным культурным и экономическим аспектам криптографии; части III и IV представляют собой сокращенный справочник по криптографии и основам защиты информации. Эти три части содержат описания многочисленных криптографических схем и протоколов, а также разнообразных приемов атак и их предотвращения, о которых следует знать. Данные части книги не отягощены излишними теоретическими подробностями и вполне доступны этой категории читателей.
- Аспирантам, начинающим свои исследования в области криптографии или компьютерных систем. Эти читатели оценят справочник, в котором изложены основные понятия, связанные со стойкостью защиты информации. Книга позволит им быстрее освоиться в этой обширной области знаний. Для них части II, IV, V и VI окажутся вполне содержательным обзором, который позволит сориентироваться в потоке литературы по криптографии и сформулировать свои собственные научные интересы.

Некоторые части книги (в частности, главы I, II, III и VI) можно рассматривать как учебник по прикладной криптографии, предназначенный для студентов, специализирующихся по компьютерным наукам, информатике и прикладной математике.

Благодарности

Я глубоко признателен Фенг Бао (Feng Bao), Колину Бойду (Colin Boyd), Ричарду Де Милло (Richard DeMillo), Стивену Гэлбрайту (Steven Galbraith), Дитеру Голлману (Dieter Gollmann), Киту Харрисону (Keith Harrison), Маркусу Личу (Marcus Leech), Хельгеру Липмаа (Helger Lipmaa), Хои-Квонг Ло (Hoi-Kwong Lo),

Хавьеру Лопесу (Javier Lopez), Джону Мэлоун-Ли (John Malone-Lee), Кэри Мельтцер (Cary Meltzer), Кристиану Пакуин (Christian Paquin), Кенни Патерсону (Kenny Paterson), Дэвиду Пойнтчевалу (David Pointcheval), Винсенту Риймену (Vincent Rijmen), Найджелу Смарту (Nigel Smart), Дэвиду Солдере (David Soldera), Паулю ван Ооршоту (Paul van Oorschot), Сержу Воденэ (Serge Vaudenay) и Стефеку Забе (Stefek Zaba). Эти люди потратили немало своего времени, рецензируя отдельные главы. Их неоценимые комментарии, критические замечания и предложения улучшили содержание книги.

Кроме того, отвечая на мои вопросы, в создание книги внесли свой вклад: Михир Белларэ (Mihir Bellare), Ян Камениш (Jan Camenisch), Нейл Данбар (Neil Dunbar), Йэйр Франкел (Yair Frankel), Шаи Халеви (Shai Halevi), Антуан Жё (Antoine Joux), Марк Жойе (Marc Joye), Чэли Кауфман (Chalie Kaufman), Адриан Кент (Adrian Kent), Хьюго Кравчик (Hugo Krawczyk), Кэтрин Мидоуз (Catherine Meadows), Билл Манро (Bill Munro), Фонг Нгуен (Phong Nguyen), Радя Перлман (Radia Perlman), Марко Рикка (Marco Ricca), Рональд Райвест (Ronald Rivest), Стив Шнайдер (Steve Schneider), Игорь Шуп (Igor Shoup), Игорь Шпарлински (Igor Shparlinski) и Моти Юнг (Moti Yung).

Я хотел бы выразить благодарность Джилл Хэрри (Jill Harry) из издательства Prentice-Hall PTR и Сьюзан Райт (Susan Wright) из издательства HP Professional Books за предложение написать книгу, а также за моральную и профессиональную поддержку на протяжении долгого периода работы над рукописью. Я также признателен Дженнифер Блекуэл (Jennifer Blackwell), Роберту Кэрролу (Robert Carroll), Бренде Маллиган (Brenda Mulligan), Джастину Сомма (Justin Somma) и Мэри Судул (Mary Sudul) из издательства Prentice-Hall PTR, а также Уолтеру Брюсу (Walter Bruce) и Пэт Пекари (Pat Pekary) из издательства HP Professional Books.

Я благодарен моим коллегам из компании Hewlett-Packard Laboratories Bristol, в частности, Дэвиду Боллу (David Ball), Ричарду Кардвеллу (Richard Cardwell), Ликун Чену (Liqun Chen), Йену Коулу (Ian Cole), Гарету Джоунсу (Gareth Jones), Стефану Пирсону (Stephen Pearson) и Мартину Садлеру (Martin Sadler) за техническую поддержку, предоставление литературных источников и менеджмент.

Бристоль, Англия
Май 2003

Ждем ваших отзывов!

Вы, уважаемый читатель, и есть главный критик и комментатор этой книги. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию следующих книг. Наши координаты:

E-mail: `info@williamspublishing.com`

WWW: `http://www.williamspublishing.com`

Адреса для писем:

из России: 115419, Москва, а/я 783

из Украины: 03150, Киев, а/я 152

Часть I

Введение

Первая часть книги состоит из двух вводных глав. В них излагаются основные понятия криптографии и защиты информации, описываются свойства среды, в которой люди обмениваются конфиденциальной информацией, появляются персонажи, действующие в этой среде, рассматривается стандартный образ действий "плохих парней", формулируются принципы функционирования криптографических систем и систем защиты информации, а также указываются причины их крайней ненадежности.

Эта часть содержит элементарные сведения и предназначена для новичков.

Глава 1

Защита информации в игре “орел или решка”

Для начала рассмотрим пример, иллюстрирующий применение криптографии при решении несложной задачи. Описывая этот пример, мы преследуем три цели.

- Продемонстрировать эффективность и практичность криптографии при решении прикладных задач.
- Дать предварительное понятие об основах криптографии.
- Сформировать определенное отношение к процессу разработки криптографических систем, обеспечивающих защиту информации.

Сначала рассмотрим тривиальную задачу и ее не менее тривиальное решение, представляющее собой всем известную игру с двумя участниками. Однако довольно скоро нам станет ясно, что эта игра не так проста, если игроки не видят друг друга. Если участники находятся в разных местах, нельзя гарантировать, что игра проходит честно. В этом случае они не могут доверять друг другу.

Стремление гарантировать соблюдение правил игры побуждает нас защитить игроков от мошенничества. Способ, которым достигается эта цель, широко применяется для защиты связи в открытых сетях — это сокрытие информации с помощью криптографии.

Средства прикладной криптографии позволяют получить качественное решение нашей первой задачи, связанной с защитой информации. Это позволит нам в дальнейшем перейти к обсуждению критериев качества криптографических систем (раздел 1.2). Кроме того, мы рассмотрим подоплеку и культурные аспекты тех областей, в которых необходимо применять технологии защиты конфиденциальной информации.

1.1 Орел или решка

Сформулируем простую задачу. Алиса и Боб[1] хотят провести вечер вместе, но не могут решить, куда идти: в кино или в театр. Чтобы разрешить возникший спор, они договорились подбросить монету: этот способ жеребьевки известен всем.

Алиса берет монету и говорит Бобу: "Ты выбираешь сторону, а я подбрасываю монету". Боб соглашается, и Алиса подбрасывает монету в воздух. Затем они вместе смотрят, какой стороной вверх упала монета. Если выпал результат, выбранный Бобом, то Боб решает, куда идти, в противном случае Боб подчиняется Алисе.

В работах, посвященных процедурам обмена информацией, многосторонняя игра, подобная описанной выше, носит специальное название — *протокол*. Строго говоря, протоколом называется точно определенная последовательность действий, выполняемых несколькими участниками. Следует подчеркнуть, что протокол всегда предполагает существование нескольких участников; если вся процедура полностью выполняется одним участником, ее нельзя называть протоколом.

1.1.1 Первое знакомство с криптографией

Теперь представьте, что Алиса и Боб выполняют описанный выше протокол, разговаривая по телефону. Алиса предлагает Бобу: "Выбери сторону монеты, а я ее подброшу и скажу, выиграл ты или нет". Разумеется, Боб не согласится, поскольку не может проверить результат жеребьевки.

Добавим в этот протокол немного криптографии и вернемся к варианту жеребьевки по телефону. В результате мы получим криптографический протокол — первый в нашей книге! Давайте рассматривать нашу "криптографию" как математическую функцию $f(x)$, определенную на множестве целых чисел и обладающую следующими волшебными свойствами.

Свойство 1.1. Волшебная функция f

I. Для каждого целого числа x мы можем легко вычислить значение $f(x)$, однако по заданному значению $f(x)$ невозможно восстановить его прообраз x, например, определить, каким числом является x — четным или нечетным.

II. Невозможно найти пару чисел (x, y), таких что $x \neq y$ и $f(x) = f(y)$.

В свойстве 1.1 употребляются наречия "легко" и "невозможно". Их смысл необходимо разъяснить. Поскольку оба слова определяют степень трудности какого-то действия, их взаимное соотношение вполне понятно. Однако, поскольку

[1] Алиса и Боб являются широко известными персонажами в области криптографии, криптографических протоколов и защиты информации; они будут участниками большинства криптографических протоколов, обсуждаемых в книге.

Протокол 1.1. Подбрасывание монеты по телефону

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Боб договорились о следующем:

а) волшебная функция f обладает свойствами 1.1;

б) если x — четное число, то $f(x)$ означает ОРЕЛ, в противном случае $f(x)$ означает РЕШКА.

(* *Внимание.* Условие *б* является слабым местом протокола, см. упражнение 1.2. *)

1. Алиса выбирает большое случайное целое число x, вычисляет значение $f(x)$ и сообщает его Бобу по телефону.
2. Боб сообщает Алисе свою догадку о числе x: четное оно или нечетное.
3. Алиса называет Бобу число x.
4. Боб проверяет значение $f(x)$ и убеждается в правоте или неправоте своей догадки.

мы считаем, что функция $f(x)$ обладает волшебными свойствами, можно использовать эти слова в их общеупотребительном значении. В главе 4 будет сформулировано математическое определение понятий "легко" и "невозможно". Одна из основных целей книги — установить различные количественные значения слов "легко", "трудно" и даже "невозможно". Фактически, как мы убедимся после изучения главы 19, содержащей окончательную реализацию протокола жеребьевки с помощью монеты, слово "невозможно" в формулировке свойств 1.1 имеет два совершенно разных количественных значения.

Предположим, что Алиса и Боб согласились применить волшебную функцию $f(x)$. Допустим также, что они договорились четное число считать эквивалентом ОРЛА, а нечетное — РЕШКИ. Теперь они готовы выполнить наш первый криптографический протокол 1.1.

Очевидно, что протокол "Подбрасывание монеты по телефону" вполне работоспособен. Проведем элементарный "анализ стойкости" этого протокола. (Предупреждение: мы заключили слова "анализ стойкости" в кавычки, поскольку этот анализ далеко не полон.)

1.1.1.1 Элементарный "анализ стойкости"

Во-первых, свойство II функции $f(x)$ означает, что Алиса не может найти два разных числа x и y, одно из которых было бы четным, а другое — нечетным (этот факт можно выразить формулой $x \neq y (\mathrm{mod}\, 2)$), таких что $f(x) = f(y)$. Следовательно, сообщив значение $f(x)$ Бобу (шаг 1), Алиса фиксирует свой выбор

числа x и не может от него отказаться. На этом жеребьевка с помощью монеты завершается.

Во-вторых, благодаря свойству I, Боб не может определить, является прообраз, известный Алисе, четным или нечетным при заданном значении $f(x)$. Следовательно, предположение Боба, сделанное им на втором шаге протокола, является настоящей догадкой (т.е. догадкой, сделанной при полном отсутствии информации). На этом этапе Алиса может убедить Боба, верна ли его догадка, назвав число x(шаг 3). Боб может проверить это, самостоятельно вычислив функцию $f(x)$ (шаг 4) и сравнив ее с числом, полученным от Алисы на первом шаге, предполагая, что функция $f(x)$ обладает заявленными свойствами. Кроме того, жеребьевка является корректной, если число x извлекается из достаточно большого множества, чтобы вероятность угадать четность числа x не превышала 50%.

Следует подчеркнуть, что, проводя наш "анализ стойкости", мы многое упростили и о многом умолчали. В результате текущая версия протокола далека от совершенства. Некоторые упрощения и умолчания мы обсудим в следующей главе. Криптографические средства, необходимые для правильной и точной реализации данного протокола, а также методы анализа его стойкости будут рассмотрены в дальнейших главах. Мы отложим конкретную реализацию протокола 1.1, основанную на свойствах волшебной функции $f(x)$, до заключительной части книги, содержащей техническую информацию (глава 19). Для того чтобы провести формальный анализ стойкости конкретной реализации протокола, нам необходимо овладеть соответствующими методами.

1.1.2 Первые сведения об основах криптографии

Хотя наш первый протокол весьма прост, его можно назвать полноценным криптографическим протоколом, поскольку волшебная функция, положенная в его основу, представляет собой основной ингредиент современной криптографии: **однонаправленную функцию** (one-way function). Два волшебных свойства этой функции ставят две **трудноразрешимые вычислительные** задачи (computationally intractable): одну — перед Алисой, а вторую — перед Бобом.

Элементарный анализ стойкости протокола 1.1 показывает, что однонаправленная функция предоставляет надежное средство выбора места отдыха. Обобщим это утверждение.

> *Существование однонаправленной функции гарантирует существование стойкой криптографической системы.*

В настоящее время хорошо известно, что справедливо и обратное утверждение.

> *Существование стойкой криптографической системы предполагает существование однонаправленной функции.*

Широко распространено мнение, что однонаправленные функции действительно существуют. Это внушает уверенность в том, что мы можем защитить информацию. Наш оптимизм часто подкрепляется повседневным опытом: многие процессы, протекающие в реальном мире, носят однонаправленный характер. Рассмотрим следующее физическое явление (хотя у него нет точного математического аналога): уронить стакан на пол и разбить его на мелкие осколки довольно легко, поскольку при этом происходит рассеивание определенного объема энергии (т.е. тепла, акустических волн и даже слабого света) в окружающей среде. Обратный процесс, заключающийся в собирании рассеянной энергии и воссоединении разбросанных осколков, чтобы получился целый стакан, представляет собой весьма трудную, скорее, даже неразрешимую проблему. (Если бы это было возможно, восстановленная энергия вернула бы стакан на ту высоту, с которой он упал!)

Класс математических функций, обладающих свойствами однонаправленности, необходимыми для современной криптографии, будет рассмотрен в главе 4.

1.1.3 Основы информационной безопасности: не только трудноразрешимые вычислительные задачи

Итак, мы утверждаем, что для защиты информации необходимы определенные математические свойства. Более того, мы оптимистично заявили, что верно и обратное: математические свойства предполагают (т.е. гарантируют) защиту информации.

Однако в реальности последнее утверждение не всегда справедливо! Безопасность информации в реальных приложениях зависит от многих аспектов. Для подтверждения своих слов рассмотрим наш первый протокол.

Следует заметить, что в ходе анализа стойкости протокола 1.1 мы проигнорировали многие важные аспекты. Фактически сам по себе протокол 1.1 представляет собой значительно упрощенную спецификацию. В нем не учтены некоторые детали, влияющие на степень надежности. Это позволило нам не задавать некоторые неудобные вопросы.

Например, мы могли бы спросить: действительно ли Алиса вынуждена фиксировать свой выбор числа x? И наоборот, действительно ли Боб вынужден фиксировать свою догадку о четности или нечетности числа x? Под словом "вынужден" мы имеем в виду следующее: может ли голос по телефону гарантировать строгие математические свойства, обеспечивающие защиту информации? Можно было бы поинтересоваться, достаточно ли хороший генератор случайных чисел использует Алиса при выборе случайного числа x? Качество этого генератора крайне важно для многих серьезных приложений, в которых необходимо принимать правильное решение.

Все эти детали не упоминаются в упрощенной спецификации протокола и, следовательно, являются неявными допущениями. Если бы этот протокол формулировался для серьезной задачи, в него следовало бы включить *явные* инструкции. Например, оба участника могут записывать ответ партнера, т.е. значение $f(x)$ и догадку о четности или нечетности числа x, а затем воспроизводить его в случае спора.

Довольно часто криптографические системы и протоколы, в особенности учебные, формулируются в упрощенном виде. Это позволяет достичь ясности изложения, особенно если некоторые предположения кажутся очевидными. Однако иногда неявные предположения или соглашения могут порождать недоразумения и приводить к непредсказуемым последствиям. Таким образом, "ясность изложения" достигается за счет безопасности. Это может сделать систему защиты информации уязвимой для атак и свести уровень безопасности к нулю. Проверить выполнение неявных предположений особенно трудно. Важность явного формулирования предположений при разработке и проектировании криптографических систем мы обсудим в разделе 1.2.5.

1.1.4 Современная роль криптографии: обеспечение игры по правилам

Некогда криптография была прерогативой правительств. Военные и дипломатические организации использовали ее для обеспечения секретности своих сообщений. В настоящее время роль криптографии изменилась. Теперь в ее функции входит защита информации, т.е. обеспечение соблюдения правил в "играх" с гораздо бóльшим количеством участников. Отчасти поэтому мы начали книгу с описания простой игры.

Разумеется, выбор места для отдыха — не самое серьезное приложение криптографии, а жеребьевку по телефону можно считать развлечением. Однако существует много "игр", предусматривающих обмен информацией, которые следует рассматривать весьма серьезно. По мере развития виртуального бизнеса и электронной коммерции расширяется обмен данными через открытые сети. Во многих случаях такой обмен информацией можно рассматривать как разновидность "игры". (В предисловии мы перечислили сферы, в которых обмен данными осуществляется через открытые сети; все они предусматривают некоторые интерактивные действия участников, подчиняющихся определенным правилам, которые можно назвать "правилами игры".) Эти "игры" могут быть весьма серьезными!

Как правило, "игроки", участвующие в таких играх, изолированы друг от друга и общаются между собой через открытые сети, которые, как известно, слабо защищены. Удаленность друг от друга и низкий уровень защиты могут спровоцировать некоторых "игроков" (например, они могут оказаться непрошеными гостями) на какое-нибудь хитроумное нарушение правил. Обычно "игроки" нарушают

правила, чтобы получить незаконные привилегии, например, раскрыть конфиденциальную информацию, скрыто модифицировать данные, фальсифицировать факты, аннулировать долговые обязательства, повредить отчетную документацию, снизить уровень защиты или совсем уничтожить ее и т.п. Важная роль, которую играют средства связи в современном бизнесе, электронной коммерции и сфере виртуальных услуг (а также во многих других областях, связанных с секретными разработками компаний, персональной информацией, военными действиями и государственными секретами), исключает получение незаконных привилегий игроками, не соблюдающими правил.

Описывая криптографический протокол "Подбрасывание монеты по телефону", мы показали, каким образом простая игра порождает криптографический протокол, гарантирующий требуемый уровень безопасности. Наш пример продемонстрировал эффективность средств криптографии, обеспечивающих соблюдение правил игры. Действительно, применение криптографии — это эффективный и *единственный практичный* способ обеспечения безопасного обмена информацией между компьютерами и в сетях связи. Криптографические протоколы представляют собой процедуру обмена информацией, защищенную криптографическими средствами и, следовательно, гарантирующую соблюдение правил. Электронная коммерция, виртуальный бизнес и другие сферы, использующие средства связи, крайне нуждаются в средствах, предотвращающих постоянный соблазн "нарушить правила". Это привело к появлению многих криптографических систем и протоколов, которые являются предметом нашей книги.

1.2 Критерии качества криптографических систем и протоколов

Начнем с фундаментального вопроса.

Что такое хорошая криптографическая система или протокол?

Совершенно очевидно, что ответить на этот вопрос нелегко! Слово "*хороший*" может иметь много разных значений, и, следовательно, ответов на поставленный вопрос будет много. Основная задача книги — дать точный ответ на этот фундаментальный вопрос. Однако в первой главе мы можем предложить лишь приблизительные варианты ответа.

1.2.1 Стойкость защиты в прикладных системах

Рассмотрим первый криптографический протокол, описанный в разделе 1.1.1. Его можно назвать хорошим, потому что он очень прост. Некоторым читателям, возможно, уже известны многочисленные однонаправленные функции хэширования, применяемые на практике, например, хэш-функция SHA-1 (см. 10.3.1). Эту

функцию легко реализовать даже с помощью карманного калькулятора. Ее результатом является строка битов, состоящая из 160 бит, или 20 байт (1 байт = 8 бит). Используя шестнадцатеричную схему кодирования (см. пример 5.17), эту строку можно зашифровать в виде 40 шестнадцатеричных символов[2]. Алисе или Бобу будет совсем нетрудно продиктовать это число по телефону или записать на клочке бумаги. Такая реализация вполне безопасна, если Алиса и Боб спорят о том, куда идти — в театр или в кино. Если Алиса захочет "сжульничать", она должна будет решить нетривиальную задачу: найти числа $x \neq y (\mathrm{mod}\, 2)$, такие что $f(x) = f(y)$. Перед Бобом также стоит непростая задача: определить по заданному значению $f(x)$, четным или нечетным является число x.

Однако наши суждения о реализации протокола "Подбрасывание монеты по телефону" с помощью функции SHA-1 основаны на несерьезности последствий, к которым может привести рассматриваемая игра. Во многих более важных приложениях (которые мы рассмотрим в разделе 1.2.4) подобные протоколы требуют от хэш-функции больше надежности, чем может обеспечить функция SHA-1. Следует заметить, что функция, обладающая свойствами 1.1, в которых слово "невозможно" употребляется в буквальном смысле, является *абсолютно надежной* однонаправленной функцией. Такую функцию нелегко реализовать. Хуже того, даже ее существование до сих пор остается под вопросом (хотя мы придерживаемся оптимистичного мнения (см. раздел 1.1.2), условия существования однонаправленной функции будут рассмотрены в главе 4). Следовательно, в более серьезных приложениях, предусматривающих жеребьевку, хэш-функции нельзя считать приемлемыми — нужны более строгие криптографические средства. С другой стороны, если вы спорите о том, где провести вечер, тяжеловесные криптографические системы совершенно излишни.

Следует заметить, что существуют приложения, в которых слишком сильная защита может нарушить правильное функционирование всей системы безопасности. Например, Райвест (Rivest) и Шамир (Shamir) предложили схему для осуществления микроплатежей под названием MicroMint [242], в которой для повышения эффективности сознательно используются определенные недостатки алгоритма шифрования. Эта система платежей основана на вполне разумном предположении, что только крупная организация, обладающая большими ресурсами (например, большой банк), может эффективно подготовить большое количество "коллизий" для практических хэш-функций. Иначе говоря, только такая организация способна вычислить k разных чисел $(x_1, x_2, \ldots, x_k)$, удовлетворяющих условию

$$f(x_1) = f(x_2) = \cdots = f(x_k).$$

[2]Множество шестнадцатеричных символов состоит из цифр и букв $\{0, 1, 2, \ldots, 9, A, B, \ldots, F\}$, соответствующих шестнадцати четырехбитовым числам.

Набор чисел $(x_1, x_2, \ldots, x_k)$ называется *коллизией однонаправленной функции* f. Пара коллизий может быть эффективно проверена, поскольку функция f допускает эффективное вычисление. Коллизии можно рассматривать как результат работы крупного провайдера услуг и, следовательно, считать сертифицированными значениями. Алгоритм DES (Data Encryption Standard — стандарт шифрования данных, см. раздел 7.6) считается приемлемым алгоритмом реализации такой однонаправленной функции [242], порождающим относительно небольшое пространство выходов (64 бит). Итак, в отличие от обычного криптографического использования однонаправленных функций, в которых коллизия практически всегда является результатом успешной атаки на систему (например, на протокол "Подбрасывание монеты по телефону"), в системе MicroMint коллизии используются для того, чтобы обеспечить микроплатежи! Очевидно, стойкая однонаправленная функция, имеющая значительно большее пространство выходов (т.е. намного превышающее 64 бит), сводит к нулю возможности предоставления таких услуг даже для крупных организаций. К числу таких функций относится функция SHA-1, пространство выходов которой состоит из 160 бит. (В разделе 3.6 мы изучим аспекты, связанные с вычислительной сложностью поиска коллизий хэш-функций.)

Вполне понятно, что применение громоздких криптографических технологий при разработке систем защиты данных (например, многослойное шифрование, произвольное использование цифровых подписей, обращение за интерактивными услугами только к доверенному посреднику и даже к большому количеству таких посредников), может создать ложное впечатление полной безопасности. К тому же, иногда такие системы легче разрабатывать. Однако излишнее усердие в этом вопросе нежелательно, поскольку более сильная система защиты повышает сложность всей системы в целом. В свою очередь, более сложная система может затруднить анализ стойкости (поскольку она больше подвержена ошибкам) и реализацию системы защиты информации. Кроме того, такая система менее эффективна, а для ее внедрения и поддержки требуются дополнительные средства.

Гораздо более интересно и сложно разработать криптографическую или другую систему защиты информации, использующую лишь самые необходимые средства для достижения требуемого уровня безопасности. Это весьма важный элемент, который следует учитывать при анализе качества таких систем.

1.2.2 Уверенность в безопасности, основанная на исследовании эталонов

Как убедиться в стойкости криптографического алгоритма? Можно ли утверждать, что алгоритм надежен, так как никто не смог его взломать? К сожалению, *нет*. В принципе, если алгоритм еще никем не взломан, можно лишь сказать, что пока мы не знаем, как это сделать. В криптографии взломанные алгоритмы

иногда играют роль эталона. Если стойкий алгоритм не обладает свойствами уже взломанного алгоритма, мы не можем даже утверждать, что он более надежен.

Несмотря на это, существует несколько исключений. В большинстве случаев взлом криптографического алгоритма или схемы сводится к решению некоторых математических задач, например, к решению определенного уравнения или вычислению обратной функции. Эти математические задачи считаются "трудноразрешимыми". Формальное определение "трудноразрешимой" задачи будет дано в главе 4. Если придерживаться неформальной, однако вполне корректной точки зрения, можно сказать, что математическая задача является трудно разрешимой, если ее невозможно решить ни одним известным методом за разумное время.

Существует великое множество хорошо известных трудноразрешимых задач, которые часто используются в современной криптографии, в частности, в асимметричных методах (см. разделы 8.3–8.14). Например, в криптографии с открытым ключом к неразрешимым задачам относятся задача факторизации целых чисел (integer factorization problem), задача дискретного логарифмирования (discrete logarithm problem), задача Диффи–Хеллмана (Diffie–Hellman problem) и некоторые родственные задачи (мы сформулируем и обсудим их в главе 8). Эти задачи можно рассматривать как эталоны, поскольку они имеют долгую историю и в настоящее время считаются действительно трудноразрешимыми.

В настоящее время стандартный способ доказательства высокой стойкости криптографических алгоритмов заключается в формальном доказательстве того факта, что атака на алгоритм эквивалентна решению одной из хорошо известных трудноразрешимых задач. Доказательство представляет собой эффективное математическое преобразование или последовательность преобразований, сводящих атаку на алгоритм к решению задачи-эталона. Такое эффективное преобразование называется **редукцией** (reduction). Поскольку мы практически уверены, что трудные задачи решить невозможно (особенно за время, которое отводится на атаку), мы можем прийти к вполне обоснованному выводу, что искомой атаки не существует. Такой способ доказательства называется "сведением к противоречию": поиску простого решения трудной задачи.

Криптографические алгоритмы и протоколы считаются *хорошими*, если формально доказана их стойкость к особой разновидности мощных атак, называемых *адаптивными* (adaptive attacks). В дальнейшем мы будем называть *практической стойкостью* (*fit-to-application security*) свойство алгоритма, доказанное путем формального сведéния мощных атак к решению трудноразрешимых задач.

Одной из основных задач книги является доказательство практической стойкости большого количества криптографических алгоритмов и протоколов.

1.2.3 Практическая эффективность

Говоря, что математическая задача является эффективно разрешимой, мы имеем в виду, что время, необходимое для решения задачи, полиномиально зависит от ее размера. Формальное определение понятия эффективности, позволяющее точно измерить ее величину, будет введено в главе 4.

Не вдаваясь в количественные аспекты эффективности, можно сказать, что задачи разделяются на два класса: разрешимые и трудноразрешимые. Это определение играет фундаментальную роль в современной криптографии, основанной на понятии вычислительной сложности. Очевидно, криптографический алгоритм должен быть разработан так, чтобы с одной стороны он был разрешимым, а с другой — трудноразрешимым для постороннего или атакующего.

Однако следует отметить, что такое определение охватывает чрезвычайно широкое поле количественных измерений. Если время решения задачи по отношению к законному пользователю является полиномиальным, но слишком большим, то понятие "эффективности" теряет практический смысл. Таким образом, хороший криптографический алгоритм должен быть *практически эффективным* (practically efficient). Точнее говоря, полином, оценивающий стоимость ресурсов для пользователя, должен быть малым (т.е. иметь невысокую степень, как указано в главе 4).

В главе 14 мы обсудим работы, посвященные разработке формально обоснованных стойких криптосистем с открытым ключом. Причиной, побудившей создать эти алгоритмы, стала ненадежность основных алгоритмов шифрования с открытым ключом. (Мы называем такие ненадежные схемы "учебными", поскольку их примитивные версии рассматриваются в большинстве учебников по криптографии; они будут описаны в части III.) Однако в большей части новаторских работ по формально обоснованным стойким криптосистемам с открытым ключом изучался метод побитового шифрования (bit-to-bit encryption method) [125, 210, 241]. Некоторые из них содержали экстраординарные попытки доказательства правильного шифрования каждого отдельного бита [210] и описание способов аутентификации с открытым ключом [241]. Несмотря на то что ранние работы внесли важный вклад в теорию доказательства сильной стойкости, системы, которые в них предлагались, оказались неэффективными для приложений. После главы 14 мы продолжим изучение работ, посвященных формальному доказательству стойкости криптосистем с открытым ключом и схем цифровой подписи. Криптографические схемы, предложенные в этих работах, обладают не только повышенной стойкостью, но и прикладной эффективностью. Это действительно очень хорошие криптографические схемы.

Криптографический протокол — это не только алгоритм, но и процедура обмена данными между разными участниками, связанная с обменом сообщениями по компьютерным сетям в соответствии с установленными правилами. Таким обра-

зом, существует еще один аспект эффективности протоколов: количество сеансов связи, которые часто называют **раундами** (communication rounds). Обычно сеанс связи считается более дорогим, чем этап локальных вычислений (как правило, этот этап состоит из выполнения компьютерных инструкций, например, умножения двух чисел). Следовательно, было бы желательно минимизировать количество раундов в криптографическом протоколе. В соответствии со стандартным критерием качества алгоритм считается эффективным, если время его выполнения ограничено полиномом невысокой степени, зависящим от размера задачи. Аналогично протокол считается эффективным, если количество раундов связи ограничено полиномом *очень* малой степени: константой (степень равна 0) или линейной функцией (степень равна 1). Протокол, в котором количество раундов превышает линейную функцию, нельзя признать практически эффективным.

В разделе 18.2.3 мы рассмотрим некоторые протоколы с нулевым разглашением, в которых количество раундов оценивается нелинейными полиномами. Следует отметить, что эти протоколы непригодны для практических применений, однако они играют важную роль в теоретической криптографии и теории вычислительной сложности. В главе 18 мы опишем многочисленные исследовательские усилия, направленные на разработку практически эффективных протоколов с нулевым разглашением.

1.2.4 Использование полезных примитивов и услуг

Уровень стойкости, достаточный для одного приложения, может оказаться недостаточно высоким для другого. Вернемся к протоколу “Подбрасывание монеты по телефону”. В разделе 1.2.1 мы пришли к выводу, что с помощью одной из практических однонаправленных функций Алиса и Боб могут успешно разрешить свой спор о свидании. Однако во многих криптографических приложениях такой протокол должен обеспечивать гораздо более высокую степень защиты от мошенничества. В некоторых приложениях понятие строгости используется в буквальном смысле.

Например, в главе 18 рассматриваются протоколы с нулевым разглашением, в которых на вход должна поступать случайная строка битов. Эта строка должна вызывать одобрение у обеих сторон — как доказывающей, так и проверяющей, иначе может возникнуть серьезная опасность. Если две общающиеся стороны не имеют доступа к посреднику, генерирующему случайные числа, или не доверяют ему (чтобы подчеркнуть непрактичность этой услуги, ее обычно называют “случайные числа, взятые с потолка”), они должны самостоятельно генерировать взаимно признаваемые случайные числа бит за битом (т.е. путем честной жеребьевки с помощью подбрасывания монеты), чтобы обеспечить правильность и нулевое разглашение протокола. В этом случае уровень практической стойкости

(например, присущей односторонней хэш-функции из протокола "Подбрасывание монеты по телефону") вероятнее всего окажется недостаточным.

Одна из насущных задач прикладной криптографии — создать высококачественные средства защиты информации на основе *практичных* и *приемлемых* примитивов. Разъясним эту идею на примере протокола "Подбрасывание монеты по телефону". Этот протокол представляет собой вариант протокола "подбрасывания монеты по телефону", предложенного Блюмом (Blum) [43]. Хотя этот протокол основан на использовании *практически стойкой* и *легко реализуемой* "однонаправленной" функции, он достигает *весьма высокой* степени стойкости.

- Во-первых, он позволяет получить количественную оценку сложности вычислений, выполняемых стороной, подбрасывающей монету (т.е. Алисой), в случае мошенничества, т.е. вычислений, необходимых для подготовки коллизии, состоящей из двух чисел, таких что если $x \neq y$, то $f(x) = f(y)$. В данном случае сложность вычислений сравнима со сложностью факторизации большого составного целого числа, которая представляет собой общепризнанную трудноразрешимую задачу.
- Во-вторых, угадывающая сторона *абсолютно не способна* выработать стратегию, вероятность успеха в которой была бы выше 50%. Это условие гарантирует полную безопасность.

Таким образом, протокол "подбрасывания монеты по телефону", предложенный Блюмом, *особенно хорош* для достижения высокой стойкости, обеспечиваемой лишь практичными криптографическими примитивами. Он усиливает и уточняет первый протокол, рассмотренный в книге, и станет последним протоколом, который мы рассмотрим.

Несколько лет назад была изобретена криптография с открытым ключом [97, 98, 246]. Со временем стало очевидным, что некоторые хорошо известные алгоритмы шифрования с открытым ключом (мы называем их "учебными") имеют слабые места: 1) в них существует утечка частичной информации о зашифрованном сообщении; 2) они чрезвычайно уязвимы для активных атак (см. главу 14). Это значит, что "учебная" криптография непригодна для практических целей. Первые попытки полностью устранить эти недостатки и предотвратить активные атаки неизменно сводились к побитовому шифрованию и даже к применению способов доказательства с нулевым разглашением на уровне битов. Кроме того, для их защиты использовались средства аутентификации. Эти результаты были весьма ценными для дальнейшего развития гарантированно стойких алгоритмов шифрования с открытым ключом, однако оставались непригодными для практического применения, поскольку требование нулевого разглашения или применение средств аутентификации являются непрактичными по отношению к алгоритмам шифрования.

С момента появления первой успешной работы, использующей схему случайного заполнения (randomized padding scheme) для усиления алгоритма шифрования с открытым ключом [24], стойкость популярных "учебных" алгоритмов стали повышать, используя широко распространенные примитивы, такие как хэш-функции и датчики псевдослучайных чисел. Усиленные схемы шифрования вполне практичны, поскольку они используют практичные примитивы. Следовательно, их эффективность сравнима с эффективностью "учебных" аналогов. Благодаря этому обстоятельству некоторые алгоритмы, построенные на основе практичных примитивов, стали стандартами шифрования с открытым ключом и цифровой подписью. Некоторые из этих схем будут рассмотрены в главах 15 и 16.

Кроме того, криптографические схемы, протоколы и системы защиты информации, использующие широко распространенные примитивы, становятся все более стойкими, поскольку они привлекают внимание все более широкого круга специалистов.

1.2.5 Ясность

В конце 1960-х годов программное обеспечение стало очень крупным и сложным. Разразился так называемый "кризис программного обеспечения". Сложное программное обеспечение становилось все более подверженным ошибкам, а стоимость отладки программ стала превышать стоимость их разработки. Вскоре специалисты в области компьютерных наук обнаружили несколько причин возникшего кризиса, которые присущи плохому стилю программирования. К этим причинам относятся следующие факторы.

- Неограниченное использование оператора GOTO (переходы из одной точки программы в другую казались программистам очень удобными).
- Избыточное применение глобальных переменных (что приводит к неконтролируемым изменениям их значений, т.е. к неожиданному поведению программ во время выполнения).
- Использование переменных без объявления их типа (например, в языке Fortran переменные можно вводить, не указывая их тип, следовательно, действительное число может округляться до целого без ведома программиста).
- Неструктурированные и неорганизованные порции кода, предназначенные для одновременного решения многих задач (количество строк в таких порциях может достигать нескольких тысяч).
- Недостаточное комментирование программ (ведь комментарии не выполняются!)

Приемы, описанные выше, удобны для программиста, однако они создают трудности при отладке, поддержке и дальнейшем совершенствовании программы. Программное обеспечение, написанное в таком стиле, слишком запутанно

и непонятно. Довольно часто программист не понимал программы, написанной им же несколько месяцев и даже недель назад.

Осознав опасность плохого стиля программирования, специалисты в области компьютерных наук занялись выработкой *методологии проектирования программ* (program design methology), выдвинув на первый план требование *ясности*. Они ограничили использование оператора GOTO и глобальных переменных (порекомендовав не применять их совсем), потребовали включать в программы явное объявление типа любой переменной, что позволило компилятору осуществлять систематическую и автоматическую проверку типов, создали технологию модульного программирования (разделив большие программы на более мелкие части, каждая из которых предназначена для решения отдельной задачи) и настоятельно рекомендовали программистам включать в тексты программ и сопровождающую документацию подробные и как можно более ясные комментарии.

Системы защиты информации (криптографические алгоритмы и протоколы) включают в себя программное и аппаратное обеспечение. В рамках протокола разные части программы выполняются на большом количестве отдельных компьютеров, образующих сеть, причем многие программы выполняются в параллельном и интерактивном режимах. Принцип ясности по умолчанию распространяется и на разработку систем защиты информации (в особенности это касается протоколов). Однако, поскольку предполагается, что система защиты информации функционирует во вражеском окружении, в котором даже законный пользователь может иметь злой умысел, ее разработчик должен иметь в виду многие дополнительные факторы. Перечислим три важных аспекта, которые можно считать основными руководящими принципами разработки и реализации систем защиты. (В книге описаны многочисленные атаки на алгоритмы и протоколы, ставшие возможными вследствие неясностей, присущих структуре или спецификации систем защиты.)

1. **Следует ясно формулировать все необходимые предположения.**

   Система защиты информации взаимодействует с окружением, и, следовательно, это окружение должно удовлетворять определенным условиям. Эти условия называются **предположениями** (или **предпосылками**), обеспечивающими функционирование системы. Нарушение предположений, на основе которых создан протокол, делает его уязвимым для атак и может разрушить систему защиты. Особенно трудно проверить нарушение предположения, которое не было сформулировано явно (скрытое допущение). Таким образом, все предположения, на которых основана система защиты информации, должны быть явно указаны.

   Например, довольно часто протокол использует неявное предположение, что компьютер, входящий в сеть, может генерировать хорошие случайные числа. Однако на практике такому требованию соответствуют лишь неко-

торые компьютеры. Так называемая **атака с малой энтропией** (low-entropy attack) может взломать протокол, использующий плохой датчик случайных чисел. Широко известен пример атаки с малой энтропией на раннюю реализацию протокола SSL (Secure Sockets Layer), который используется для аутентификации в World Wide Web (см. раздел 12.5) [123].

Кроме того, явная идентификация и спецификация предположений могут оказаться полезными для анализа сложных систем. В работах Де Милло с соавторами (DeMillo et al.) (глава 4 книги [91]), а также Де Милло и Меррит (Merritt) [92] предложен двухэтапный подход к разработке и анализу криптографических алгоритмов. Эти этапы, уточненные Муром (Moore) [204, 205], перечислены ниже.

а) Идентифицировать **все** предположения, сделанные в протоколе.

б) Определить влияние, которое оказывает на стойкость защиты нарушение каждого из предположений, сформулированных в пункте 1.

2. **Необходимо явно и точно указывать все предлагаемые услуги по защите информации.**

Криптографический алгоритм или протокол обеспечивают определенные услуги по защите информации. К их числу относятся: обеспечение конфиденциальности (сообщение не должно быть понятным для постороннего), аутентификация (возможность проверить целостность или источник сообщения), невозможность отречения (невозможность отрицать авторство сообщения), доказательство знания (демонстрация осведомленности без раскрытия содержания) и фиксация (например, наш первый криптографический протокол “Подбрасывание монеты по телефону” обязывал Алису фиксировать строку и не изменять ее в дальнейшем).

Проектируя криптографический протокол, разработчик должен очень четко формулировать, для решения каких задач он предназначен. Ясная идентификация и спецификация задач позволит не только правильно выбрать криптографические примитивы, но и корректно реализовать протокол. Довольно часто разработчики недостаточно хорошо формулируют задачи, и поэтому возникает необходимость их уточнить. Вот некоторые способы такого уточнения.

Конфиденциальность	$\Rightarrow$	секретность, анонимность, невидимость, неразличимость.
Аутентификация	$\Rightarrow$	источник данных, целостность данных, равноправный объект.
Невозможность отречения	$\Rightarrow$	отправка сообщения, получение сообщения.
Доказательство знания	$\Rightarrow$	обладание знанием, структура знания.

Неверная идентификация задач, поставленных перед протоколом, может привести к неправильному использованию криптографических примитивов и, как следствие, к снижению стойкости протокола. В главах 2 и 11 мы рассмотрим примеры взлома протоколов аутентификации, ставшего возможным из-за неправильного выбора между конфиденциальностью и аутентификацией.

Существует множество задач, решаемых системами защиты информации, и еще больше их разнообразных названий (например, свежесть сообщения, неподатливость сообщения, секретность пересылки, абсолютно нулевое разглашение, честность, связывание, возможность отречения, свобода получения и т.п.). Эти понятия можно считать производными от понятий, перечисленных выше (производные понятия могут быть и отрицаниями, например, возможность отречения (deniability) является противоположностью невозможности отречения (non-repudiation)). Несмотря на это во избежание недоразумений такие формулировки необходимо часто уточнять.

3. **Необходимо явно выделять частные случаи математических задач.**

   Как указывалось в разделе 1.2.2, некоторые трудноразрешимые задачи в теории вычислительной сложности могут обеспечивать высокий уровень стойкости криптографических алгоритмов и протоколов. Однако существует частный случай трудноразрешимой задачи, который относительно просто решить. Например, хорошо известно, что задача факторизации большого составного целого числа, в принципе, трудноразрешима. Однако факторизация *большого* составного целого числа $N = PQ$, в котором Q является простым числом, следующим за *большим* простым числом P, вовсе не является трудноразрешимой задачей! Ее можно эффективно решить, вычислив величину $\left\lfloor \sqrt{N} \right\rfloor$ (где символ $\lfloor \cdot \rfloor$ означает целую часть числа, т.е. наибольшее целое число, не превышающее заданное) и выполнив несколько делений для определения чисел P и Q.

   Обычные алгебраические структуры, с которыми работают криптографические алгоритмы (такие как группы, кольца и поля, изучаемые в главе 5), имеют специальные варианты, не представляющие никакой вычислительной сложности. В качестве иллюстрации можно указать элементы, имеющие малый мультипликативный порядок (small multiplicative order) в мультипликативной группе или конечном поле (см. главу 5). Например, когда основание степени в протоколе обмена ключами Диффи–Хеллмана (см. раздел 8.3) является единичным элементом в этих алгебраических структурах, возникает экстремальная ситуация. Другим примером такого рода являются вырожденные виды эллиптических кривых — "сверхсингулярные" и "аномальные" кривые. Проблему вычисления дискретного логарифма на сверхсингулярной кривой можно свести к задаче вычисления дискретного лога-

рифма над конечным полем, решение которой известно под названием "атака Менезеса–Окамото–Ванстоуна" ("Menezes–Okamoto–Vanstone attack") [197] (см. раздел 13.3.4.1). **Аномальной** называется кривая, число точек которой равно размеру базисного поля. Это позволяет вычислить дискретный логарифм на этой кривой за полиномиальное время. Решение этой задачи называется атакой Сато–Араки (Satoh–Araki) [252], Семаева (Semaev) [258] и Смарта (Smart) [278].

Если разработчик алгоритма или протокола не знает о существовании особых ситуаций или неточно описал их в своей спецификации, возникает опасность успешной атаки. Таким образом, криптографы обязаны знать математические особенности решаемой задачи и должны явно описывать процедуры, позволяющие избегать этих ситуаций при проектировании алгоритмов и протоколов.

Продолжить описание ситуаций, в которых необходима ясность, совсем нетрудно (например, ясность требуется при формулировке правил в протоколах управления ключами, чтобы разделить ключи, имеющие разное предназначение, правильно их передавать и т.п.). Вследствие специфичной природы этих задач, мы не можем перечислить их полностью. Однако на протяжении всей книги мы будем уделять особое внимание ясности алгоритмов и протоколов, а также их спецификаций. В целом, чем более ясно спроектирован и описан алгоритм или протокол, тем легче его анализировать. Следовательно, вероятность, что он будет правильно реализован, увеличивается, а вероятность успешного взлома уменьшается.

1.2.6 Открытость

С давних пор криптография являлась прерогативой правительств. Военные и дипломатические организации использовали ее для сохранения секретности сообщений. В те времена большинство криптографических разработок осуществлялось за закрытыми дверями, а протоколы и алгоритмы оставались секретными. Разумеется, у правительств были и есть все основания проводить свои криптографические исследования в тайне. Представьте, что произошло бы, если бы некое правительственное агентство опубликовало бы какой-нибудь шифр. Это можно допустить лишь в том случае, если опубликованный шифр имеет гарантированную стойкость. В противном случае его опубликование было бы слишком опасным и могло бы создать трудности для правительства. Однако и в этой ситуации другие правительства могли бы воспользоваться шифром, имеющим гарантированную стойкость (provable secure cipher), и, следовательно, снизить эффективность работы криптоаналитиков, работающих на правительство, опубликовавшее его.

Однако в наши дни криптографические механизмы стали неотъемлемой частью многих гражданских систем (в начале главы приведен их далеко не полный

перечень). Гражданские криптографические исследования должны быть открытыми. Разумеется, криптографические алгоритмы используют секреты, однако секретность должна распространяться лишь на криптографические ключи или материалы, имеющие к ним отношение (например, пароли и PIN-коды). Сами алгоритмы следует делать открытыми. Рассмотрим причины, обусловливающие это требование.

Во многих областях качество исследования зависит от открытого обмена идеями с помощью конференций и публикаций в специализированных журналах. Однако в области криптографических алгоритмов, протоколов и систем защиты информации исследования должны быть открытыми не только потому, что специалистам необходим обмен опытом. Важной функцией открытых исследований является их публичная проверка. Криптографические алгоритмы, протоколы и системы защиты информации печально известны своей уязвимостью и подверженностью ошибкам. Как только результаты криптографических исследований опубликованы, многочисленные эксперты принимаются их испытывать. Вероятность, что в ходе этих проверок обнаружатся ошибки (в проектировании или в анализе стойкости), допущенные разработчиками, резко возрастает. Если же алгоритм сохраняется в секрете, его проверяют лишь несколько экспертов, имеющих разрешение. В результате вероятность выявить ошибки снижается. Хуже всего, если разработчик знает о существовании ошибки и может скрытно воспользоваться этим в своих интересах.

В настоящее время общепризнанно, что криптографические алгоритмы, протоколы и системы защиты информации, имеющие гражданское предназначение, должны быть опубликованы и детально проверены многочисленными экспертами, причем оценку криптографических средств должен проводить враждебно настроенный специалист.

1.3 Резюме

В главе рассмотрен простой пример, иллюстрирующий применение прикладной криптографии. Он преследовал три цели.

1. Продемонстрировать эффективность криптографии при решении задач.
2. Дать представление об основных принципах криптографии.
3. Сделать особый упор на прикладных аспектах защиты информации.

Именно эти темы будут предметами обсуждения на протяжении всей книги.

В главе рассмотрены основы криптографии и перечислены области человеческой деятельности, в которых она применяется. Конечно, это обсуждение не могло быть исчерпывающим. Более подробные сведения о принципах криптографических систем и областях их применения можно найти в работах Шнайера (Schneier)

[254], Голлмана (Gollmann) [129] и Андерсона (Anderson) [14]. Кроме того, Шнайер ежемесячно распространяет электронный журнал "Crypto-Gram Newsletters", который будет полезен читателям. Для того чтобы подписаться на эту рассылку, достаточно обратиться по адресу: `schneier@counterpane.com`.

Упражнения

1.1. В чем заключается разница между протоколом и алгоритмом?

1.2. В протоколе 1.1 Алиса может выбирать орла или решку. В некоторых приложениях эта возможность создает несправедливые преимущества. Модифицируйте протокол так, чтобы Алиса больше не имела этой возможности.

Подсказка: пусть право выбора определяется жребием.

1.3. Пусть функция f отображает пространство 200-битовых целых чисел в пространство 100-битовых целых чисел по следующему правилу.

$$f(x) \stackrel{\text{def}}{=} (\text{старшие 100 бит числа } x) \oplus (\text{младшие 100 бит числа } x),$$

где символ $\oplus$ обозначает поразрядную операцию исключающего ИЛИ (XOR), т.е.

$$a \oplus b = \begin{cases} 0, & \text{если } a = b, \\ 1 & \text{в противном случае.} \end{cases}$$

а) Эффективна ли функция f?

б) Обладает ли функция f "волшебным" свойством I?

в) Обладает ли функция f "волшебным" свойством II?

г) Можно ли применить эту функцию в протоколе 1.1?

1.4. Можно ли утверждать, что еще не взломанный криптографический алгоритм более стоек, чем взломанный? Если нет, то почему?

1.5. Сложные системы подвержены ошибкам. Назовите еще одну причину, по которой сложные системы безопасности более уязвимы.

Глава 2

Борьба между защитой и нападением

2.1 Введение

Одна из причин многочисленности криптографических протоколов заключается в следующем: создать правильный криптографический протокол очень трудно. Попытки разработать правильные протоколы не прекращаются по сей день. Было предложено множество новых протоколов, исправляющих недостатки, обнаруженные у их предшественников. Уязвимое место криптографического протокола всегда можно обнаружить с помощью сценария одной или нескольких атак, преодолевающих средства защиты. В области криптографических протоколов это напоминает непрерывную борьбу между разработчиками и взломщиками. Разработчики предлагают некий протокол, взломщики раскрывают его, затем авторы устраняют недостатки, после этого следует новая атака, новые исправления и т.д.

В этой главе рассматривается ряд примеров, иллюстрирующих борьбу между атакой и защитой. Начнем с искусственного протокола, который имеет умышленный недостаток. С помощью этого протокола мы продемонстрируем процесс под названием “исправление, атака, новое исправление и новая атака”. В результате мы получим два протокола, предназначенных для защиты информации в реальном приложении (все предшествующие протоколы, подвергавшиеся взломам и исправлениям, являются искусственными). Эти два протокола, полученные в результате последовательных атак и исправлений, не только реальны, но и хорошо известны. Они играют очень плодотворную роль как в приложениях, так и при формальном анализе криптографических протоколов.

К сожалению, наши исправления не избавляют эти протоколы от недостатков, обнаруженных через *длительное* время после их опубликования. В одном из протоколов изъян был обнаружен через три года после публикации, а во втором — лишь спустя еще четырнадцать лет! Выявив эти недостатки, мы сделаем последнюю попытку исправить их, однако полное исследование свойств протокола, возникших в результате усовершенствования, будет проведено лишь в последних главах, когда мы овладеем техническими средствами анализа. Оставив

нерешенными проблемы стойкости, мы лишь сделаем первое предупреждение: криптографические алгоритмы, протоколы и системы часто имеют изъяны.

Эта глава содержит технические сведения, позволяющие новичкам в области криптографии овладеть важными понятиями и средствами анализа. К числу этих понятий относятся термины (впервые упоминаемые термины выделяются **полужирным шрифтом**) и имена участников протоколов, которые будут встречаться на протяжении всей книги. Атаки на протоколы, имеющие уязвимые места, позволят нам изучить типичное поведение особого участника нашей игры: врага, для защиты от которого мы создали криптографический протокол.

2.1.1 Краткий обзор

В разделе 2.2 вводится упрощенное понятие о шифровании, которое можно применять только в рамках этой главы. В разделах 2.3–2.5 рассматривается стандартная модель угрозы, а также среда функционирования и цель криптографических протоколов, в частности, протоколов аутентификации. В заключение в разделе 2.6 описывается ряд протоколов аутентификации.

2.2 Шифрование

Все протоколы, разработанные в главе, используют **шифрование** (encryption, or encipherment). Следует заранее предупредить, что во многих случаях применение шифрования является некорректным, и необходимо применять другие криптографические примитивы. Излагая содержание книги, мы постепенно научим читателей правильно выбирать криптографические примитивы, обеспечивающие требуемую стойкость защиты. Однако для начала будем считать, что для достижения нашей цели шифрования вполне достаточно.

Шифрование (которое иногда называют **кодированием**) представляет собой процесс преобразования порции информации в непонятный вид. Исходную информацию называют **открытым текстом** (plaintext, or cleartext), а результат преобразования — **зашифрованным текстом** (ciphertext) или **криптограммой** (cryptogram). Обратный процесс преобразования зашифрованного текста в открытый называется **декодированием** (decryption) или **расшифровкой** (decipherment). Обратите внимание на то, что открытый и зашифрованный текст образуют пару взаимосвязанных понятий: открытый текст поступает на вход алгоритма шифрования, а зашифрованный текст является его результатом. Открытый текст не обязан иметь понятную форму. Например, в случае двойного шифрования зашифрованный текст может быть открытым по отношению к алгоритму повторного кодирования. Во многих местах этой главы мы неоднократно убедимся, что в криптографических протоколах широко используется шифрование случайного числа. Как правило, открытый текст представляет собой сообщение, принад-

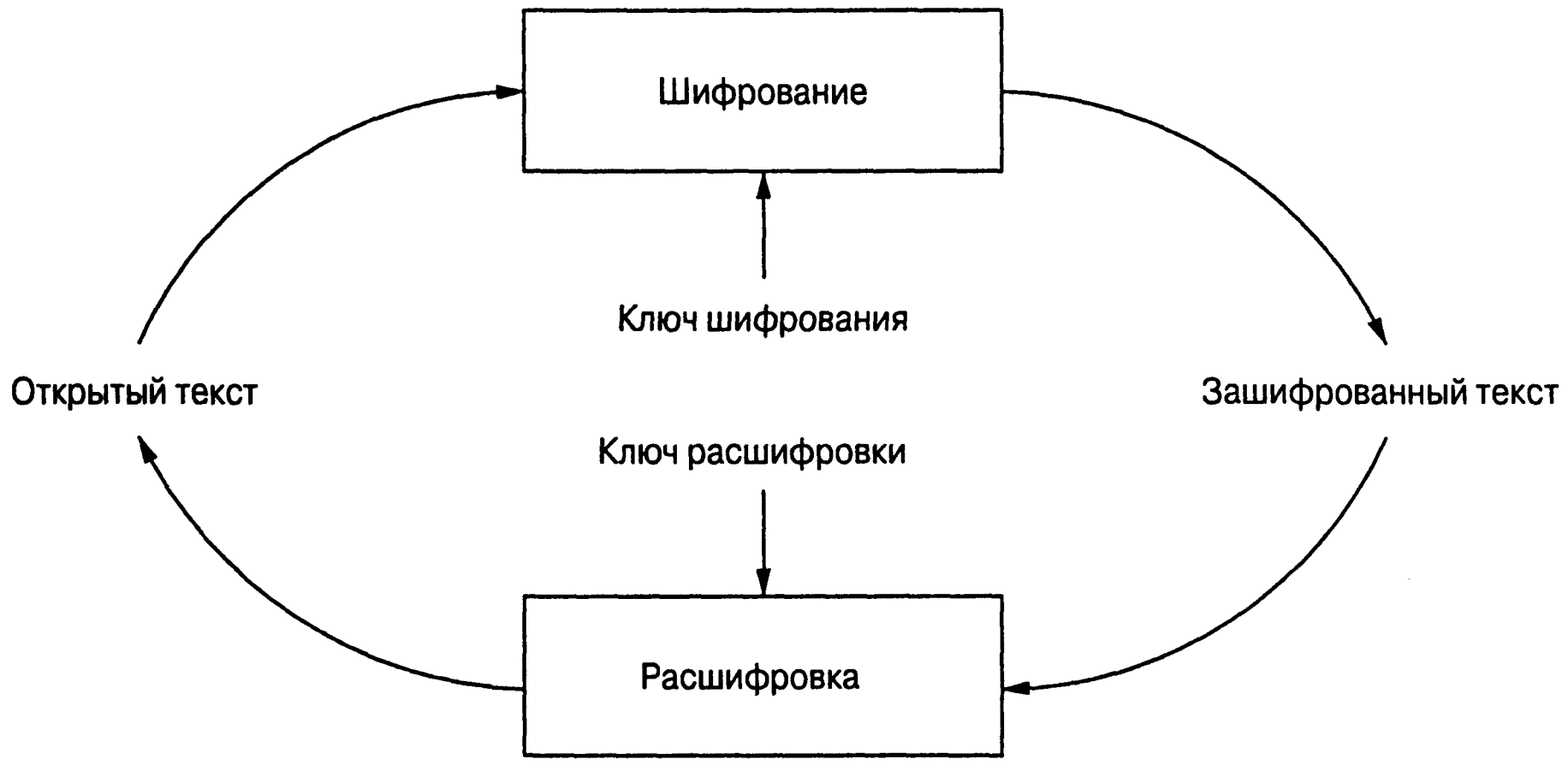


Рис. 2.1. Упрощенная схема криптографической системы

лежащее небольшому подмножеству совокупности всех возможных сообщений, которое имеет определенное распознаваемое распределение. Мы изучим это распределение в разделе 3.7.

Алгоритмы шифрования и расшифровки называются **криптографическими алгоритмами**, **криптографическими системами** или просто **криптосистемами** (cryptographic algorithms, cryptographic systems or cryptosystems). Процессы шифрования и расшифровки управляются криптографическим **ключом** (key) или несколькими ключами. В **симметричной криптосистеме** (symmetric cryprosystem), которую называют также **системой с общим ключом** (shared-key system), используются одинаковые (или практически одинаковые) ключи. В асимметричной криптосистеме (asymmetric cryprosystem), или **криптосистеме с открытым ключом** (public-key cryptosystem), используются два разных ключа: **ключ шифрования** (encryption key) и (соответствующий) **ключ расшифровки** (decryption key). Ключ шифрования можно открыть (поэтому его называют **открытым**) без разглашения соответствующего ключа расшифровки, поэтому ключ расшифровки, используемый в криптосистеме с открытым ключом, иногда называется **секретным** (private). На рис. 2.1 показана упрощенная схема криптографической системы. Более точная структура криптосистемы будет описана в главе 7 (рис. 7.1).

Следует отметить, что в рамках этой главы термины "открытый текст" и "зашифрованный текст", "шифрование" и "расшифровка", "ключ расшифровки" и "ключ дешифрования" образуют пары взаимосвязанных понятий. Обозначив сообщение (которое может быть как открытым, так и зашифрованным текстом) буквой M, криптографический алгоритм (который может выполнять как шифрование, так и расшифровку) — буквой A, а криптографический ключ (который может быть ключом шифрования или расшифровки) — буквой K, мы можем записать

выражение

$$M' = A(K, M),$$

обозначающее **криптографическое преобразование** (cryptographic transformation). Оно выражает функциональные отношения, изображенные в верхней и нижней части рис. 2.1. Таким образом, введя обозначения A' и K', можно записать выражение

$$M = A'(K', M'),$$

т.е.

$$M = A'(K', A(K, M)).$$

Последнее выражение полностью описывает процесс, изображенный на рис. 2.1. В симметричной криптосистеме ключ K' совпадает с ключом K, а в асимметричной — ключ K' соответствует открытому или секретному компоненту ключа K. В данной главе зашифрованный текст в протокольном сообщении удобно обозначить как $\{M\}_K$.

Позднее, после изучения вероятностного распределения сообщений (в разделах 3.7–3.8), мы выясним, что открытые сообщения (точнее, открытые тексты в понятной форме) образуют небольшое подмножество всего пространства сообщений, а шифрованные сообщения имеют более широкое распределение. Это различие между открытыми и зашифрованными текстами является очень существенным.

Следует заметить, что в рамках этой главы обозначение открытого текста всегда подразумевает применение "идеального" криптографического алгоритма, обладающего следующими свойствами.

Свойство 2.1. Идеальное шифрование $\{M\}_K$

I. Без ключа K (в симметричной криптосистеме) или соответствующего секретного ключа K (в асимметричной криптосистеме) зашифрованный текст $\{M\}_K$ не позволяет восстановить исходное сообщение M.

II. Зашифрованный текст $\{M\}_K$, возможно, совместно с известной информацией об открытом сообщении M не позволяет восстановить ключ K (в симметричной криптосистеме) или соответствующий секретный ключ K (в асимметричной криптосистеме).

Идеальное шифрование, обладающее этими двумя свойствами (дополнительные свойства обсуждаются в разделе 2.5.3), представляет собой идеализацию реального алгоритма шифрования. Это удобный инструмент, позволяющий отделить разработку и анализ протокола от разработки и анализа соответствующего криптографического алгоритма и упростить проектирование. Кстати, идеальное шифрование не избавляет протокол от недостатков. Практически ни одна атака на протокол, продемонстрированная в этой главе, не зависит от свойств используемой криптосистемы.

Формальное определение шифрования и большое количество алгоритмов шифрования будут введены в последующих главах (главы 7, 8, 13 и 15). Для целей, преследуемых в этой главе, вполне достаточно абстрактного описания принципов функционирования криптографических систем. Пока можно считать, что алгоритм шифрования является неким замком на шкатулке, в которой заперт открытый текст.

Глоссарий по защите безопасности читатель может найти в [266].

2.3 Уязвимая среда (модель угрозы Долева–Яо)

Как правило, крупные сети, состоящие из компьютеров и других устройств (например, Internet), являются открытыми. Это значит, что **принципал** (principal) (**сущность** (entity), **агент** (agent), **пользователь** (user)), в роли которого может выступать компьютер, устройство, ресурс, провайдер, человек или их совокупность, может присоединяться к сети, чтобы передавать и получать сообщения от других пользователей, входящих в сеть, без разрешения "главного" администратора доступа. Следует иметь в виду, что в открытой среде существуют **плохие парни** (bad guys) (**взломщики** (attackers), **неприятели** (adversaries), **враги** (enemies), **злоумышленники** (intruders), **перехватчики сообщений** (eavesdroppers), **самозванцы** (impostor) и т.д.), находящиеся вдалеке от нас и стремящиеся нанести вред, не только перехватывая сообщения, но и активно вмешиваясь в работу сети (возможно, с помощью неизвестных вычислений или методов), фальсифицируя сообщения, создавая их дубликаты, изменяя их маршрут, удаляя сообщения или создавая новые. Сообщения, вброшенные в сеть, могут быть злонамеренными. Они могут иметь разрушительные последствия для адресата. В литературе по криптографии такие плохие парни называются **активными взломщиками** (active attackers). Взломщика в нашей книге мы будем называть **Злоумышленником** (Malice). Взломщик — это лицо, стремящееся нанести вред или ущерб, скрываясь под чужим именем. Злоумышленник может быть отдельным индивидуумом, коалицией групп взломщиков и, в особых случаях, законным участником протокола (**инсайдером** (insider)).

Обычно предполагается, что Злоумышленник хорошо разбирается в средствах связи, используемых в открытой сети. Его действия непредсказуемы, так как Злоумышленник действует скрытно. Поскольку под маской Злоумышленника может действовать целая коалиция групп взломщиков, он может одновременно контролировать несколько сетевых узлов, географически удаленных друг от друга. Причина, по которой Злоумышленник действительно может осуществлять такой контроль, будет описана в разделе 12.2.

Предвидя существование мощного неприятеля в такой уязвимой среде, как открытая сеть, Долев (Dolev) и Яо (Yao) предложили **модель угрозы** (threat

model), которая широко используется в стандартных криптографических протоколах [101]. В рамках этой модели Злоумышленник обладает следующими характеристиками.

- Может перехватить любое сообщение, передаваемое по сети.
- Является законным пользователем сети и может вступать в контакт с любым другим пользователем.
- Может получать сообщения от любого пользователя.
- Может посылать сообщения любому пользователю, маскируясь под любого другого пользователя.

Таким образом, в **модели угрозы Долева–Яо** (Dolev–Yao threat model) любое сообщение, передаваемое по сети, оказывается в распоряжении Злоумышленника. Следовательно, существует угроза, что любое сообщение, полученное пользователем из сети, сначала было перехвачено Злоумышленником, а затем передано по адресу. Иначе говоря, предполагается, что Злоумышленник имеет полный контроль над всей сетью. Фактически всю открытую сеть можно считать Злоумышленником.

Однако, несмотря на перечисленные выше возможности Злоумышленника, он не является *всемогущим*. Это значит, что есть вещи, которые Злоумышленник не способен сделать, даже если под его маской скрывается коалиция неприятелей, в распоряжении которых находится большое количество компьютеров, выполняющих параллельные вычисления в открытой сети. Ниже мы перечислим те несколько вещей, которые Злоумышленнику не под силу, не уточняя, что значит “не под силу”.

- Злоумышленник не может угадать случайное число, выбранное из достаточно большого пространства.
- Не имея правильного секретного ключа, Злоумышленник не может восстановить открытый текст по его шифрованному варианту и, наоборот, не может правильно зашифровать исходное сообщение с помощью идеального алгоритма шифрования.
- Злоумышленник не способен найти секретный компонент, т.е. секретный ключ, соответствующий заданному открытому ключу.
- Контролируя крупную открытую часть наших вычислений и средства связи, Злоумышленник не имеет доступа ко многим закрытым зонам вычислительной среды, например, к памяти вычислительного устройства автономного пользователя.

Модель угрозы Долева–Яо применима ко всем нашим протоколам.

2.4 Серверы аутентификации

Предположим, что два пользователя, Алиса и Боб, которых мы уже упоминали при описании протокола "Подбрасывание монеты по телефону", хотят посекретничать. Допустим также, что Алиса и Боб раньше никогда не встречались, и, следовательно, у них нет общих секретов, и ни один из них не знает открытого ключа другого. Как же обеспечить секретность их переговоров в полностью открытой сети?

Например, Алиса и Боб могли бы установить общий секретный ключ или обменяться открытыми ключами при личной встрече. Однако в системе, объединяющей N пользователей, желающих сохранить секретность своих сообщений, для этого понадобилось бы $N(N-1)/2$ встреч. К сожалению, пользователи, как правило, живут в разных местах, и поэтому такой способ слишком дорог. Таким образом, непосредственный способ задания секретного ключа в современных системах связи непрактичен.

Однако каждый пользователь, желающий обеспечить секретность своих сообщений, может пройти **аутентификацию** (authentication) и получить доступ к другим участникам с помощью **службы каталогов** (directory service). Нидхем (Needham) и Шредер (Schroeder) предложили, чтобы эту возможность обеспечивал **сервер аутентификации** (authentication server) [213]. Такой сервер напоминает службу регистрации имен. Он поддерживает базу данных, содержащую имена обслуживаемых клиентов, и может идентифицировать пользователя на основе криптографического ключа, заранее разделенного между ним и сервером.

Сервер аутентификации является особым пользователем, которому доверяют все остальные пользователи (**клиенты**). Иначе говоря, сервер аутентификации всегда отвечает на запрос клиента точно в соответствии со спецификацией протокола и не выполняет никаких других действий, которые могли бы непреднамеренно снизить степень безопасности (например, сервер аутентификации никогда не раскрывает третьей стороне секретные ключи, разделенные между ним и его клиентами). Такой пользователь называется **доверенным посредником** (trusted third party — TTP). Мы будем называть его **Трентом**.

Допустим, что Алиса и Боб пользуются услугами своих серверов аутентификации. В крупной сети нецелесообразно иметь центральный сервер аутентификации. Нидхем и Шредер предложили использовать аутентификационные серверы, знающие друг друга. Таким образом, пользователи, обслуживаемые одним и тем же сервером аутентификации, получают имя, имеющее вид "СерверАутентификации.ИмяПользователя". Идея использовать несколько серверов аутентификации была также предложена Диффи и Хеллманом [97].

Для простоты в рамках данной главы будем считать, что Алиса и Боб используют один и тот же сервер аутентификации Трент. Пример, в котором используется несколько серверов аутентификации, обслуживающих разные области сети, будет

рассмотрен в главе 12, где изучается сетевой протокол аутентификации Kerberos, используемый в операционной системе Windows 2000.

Поскольку Алиса и Боб обслуживаются одним и тем же сервером аутентификации, будем предполагать, что каждый из них имеет с Трентом общий криптографический ключ. Обозначим эти ключи как K_{AT} (ключ Алисы и Трента) и K_{BT} (ключ Боба и Трента). Они называются **ключами шифрования ключей** (key-encryption key), поскольку с их помощью шифруются другие криптографические ключи. Вследствие высоких затрат на создание таких ключей они используются долго, поэтому их называют также **долговременными ключами** (long-term key).

2.5 Стойкость создания аутентифицированного ключа

Все протоколы, описанные в этой главе, относятся к одной и той же категории: они обеспечивают создание аутентифицированных ключей (authenticated key-establishment). Точный смысл этой **службы безопасности** (security service) раскрывается ниже.

Обозначим через K общий секретный ключ, который должен быть разделен между Алисой и Бобом. Протокол, обеспечивающий безопасность, должен обладать следующими тремя свойствами.

1. Ключ K должен быть известен только Алисе и Бобу (или, возможно, пользователю, которому они доверяют).
2. Алиса и Боб должны знать, что ключ K известен другому пользователю.
3. Алиса и Боб должны знать, что ключ K был сгенерирован заново.

Первое свойство вытекает из общего смысла аутентификации, которая заключается в идентификации пользователя, вступающего в контакт. Алиса (соответственно Боб) должны быть уверены, что на другом конце линии связи, защищенной ключом K, находится именно Боб (соответственно Алиса). Если создание ключа осуществляется с помощью Трента, он гарантирует, что не будет имитировать ни Алису, ни Боба.

Второе свойство придает аутентификации новое измерение — **аутентификацию сущности** (entity authentication), или **реальность** (liveness), идентифицированного пользователя, вступающего в контакт. Алиса (соответственно Боб) должны быть уверены, что Боб (соответственно Алиса) действительно существует и способен вступить в общение в соответствии с установленным протоколом. В дальнейшем мы убедимся, что это свойство необходимо для того, чтобы разрушить сценарий атаки, основанный на повторе старых сообщений.

Необходимость третьего свойства вытекает из давно известного криптографического принципа **управления ключом** (key management). Этот принцип за-

ключается в том, что секретный ключ нельзя использовать долгое время, если он распределен между несколькими лицами и применяется для шифрования большого количества данных. Такое применение ключа существенно отличается от использования "ключа шифрования ключей", или долговременного ключа, который был рассмотрен в разделе 2.4. Принцип управления ключом основан на двух соображениях. Во-первых, если ключ для шифрования данных распределен между несколькими лицами, то, даже если одна из сторон, скажем, Алиса, пользуется им очень осторожно, никто не может гарантировать, что другая сторона, в данном случае Боб, обращается со своим ключом точно так же. Это снижает безопасность Алисы. Во-вторых, большинство данных в конфиденциальных сообщениях содержит известную или предсказуемую информацию или имеет заранее определенную структуру. Например, фрагмент компьютерной программы содержит большое количество известных слов, таких как "begin", "end", "class", "int", "if", "then", "else", "++" и т.д. Такие данные содержат большое количество **избыточной информации** (redundancy). Определение этого понятия содержится в разделе 3.8. Шифрование подобных данных делает ключ предметом **криптоанализа** (cryptanalysis), целью которого является восстановление ключа или открытого текста. Продолжительное использование ключа для шифрования таких данных облегчает задачу криптоаналитиков. Мы должны предположить, что Злоумышленник имеет неограниченное время для поиска старого ключа и может воспользоваться им при вычислении нового. Таким образом, хорошо известный и общепризнанный принцип управления ключом оговаривает, что общий ключ шифрования должен использоваться на протяжении только *одного* сеанса связи. По этой причине такой ключ называется **сеансовым** (session key) или **кратковременным** (short-term key). Третье свойство процесса создания аутентифицированного ключа гарантирует Алисе и Бобу, что сеансовый ключ K будет сгенерирован заново.

2.6 Протоколы создания аутентифицированного ключа с помощью шифрования

Теперь мы готовы разработать протокол создания аутентифицированного ключа. Первый протокол иллюстрирует простую идею: даже если Алиса и Боб незнакомы, они оба знают Трента и могут разделить с ним соответствующие долговременные ключи. Следовательно, Трент может безопасно передавать сообщения от Алисы к Бобу и наоборот.

2.6.1 Протоколы, обеспечивающие конфиденциальность сообщений

Поскольку среда, в которой передаются сообщения, уязвима для атак, наши протоколы должны использовать шифрование. На первом этапе разработки протокола мы рассмотрим угрозы, целью которых является нарушение секретности сообщений.

2.6.1.1 Протокол "Алиса — Бобу"

Пусть выполнение протокола начинает Алиса. Сначала она генерирует случайный сеансовый ключ и шифрует его с помощью общего ключа, известного ей и Тренту. Затем она посылает Тренту зашифрованный текст, указывая имена: свое и Боба. Получив запрос Алисы на доставку сеансового ключа, Трент находит в своей базе данных два общих долговременных ключа, упомянутых в запросе Алисы. Затем Трент расшифровывает полученный текст, используя ключ Алисы, снова шифрует его с помощью ключа Боба и пересылает Бобу результат. В заключение, получив и расшифровав доставленный сеансовый ключ, Боб подтверждает его получение, отправляя Алисе сообщение, зашифрованное с помощью этого ключа. В данном протоколе Алиса является **инициатором** (initiator), а Боб — **ответчиком** (responder).

В этой главе большинство протоколов и атак описываются с помощью рисунков и спецификаций. Рисунки иллюстрируют потоки информации между участниками протокола, а спецификации описывают детали их действий при отправке и получении сообщений. Рисунки должны помочь читателям, которые лишь начинают изучать криптографию. Поскольку спецификации достаточно точно и полно описывают протоколы, в остальных главах мы не будем больше прибегать к помощи рисунков.

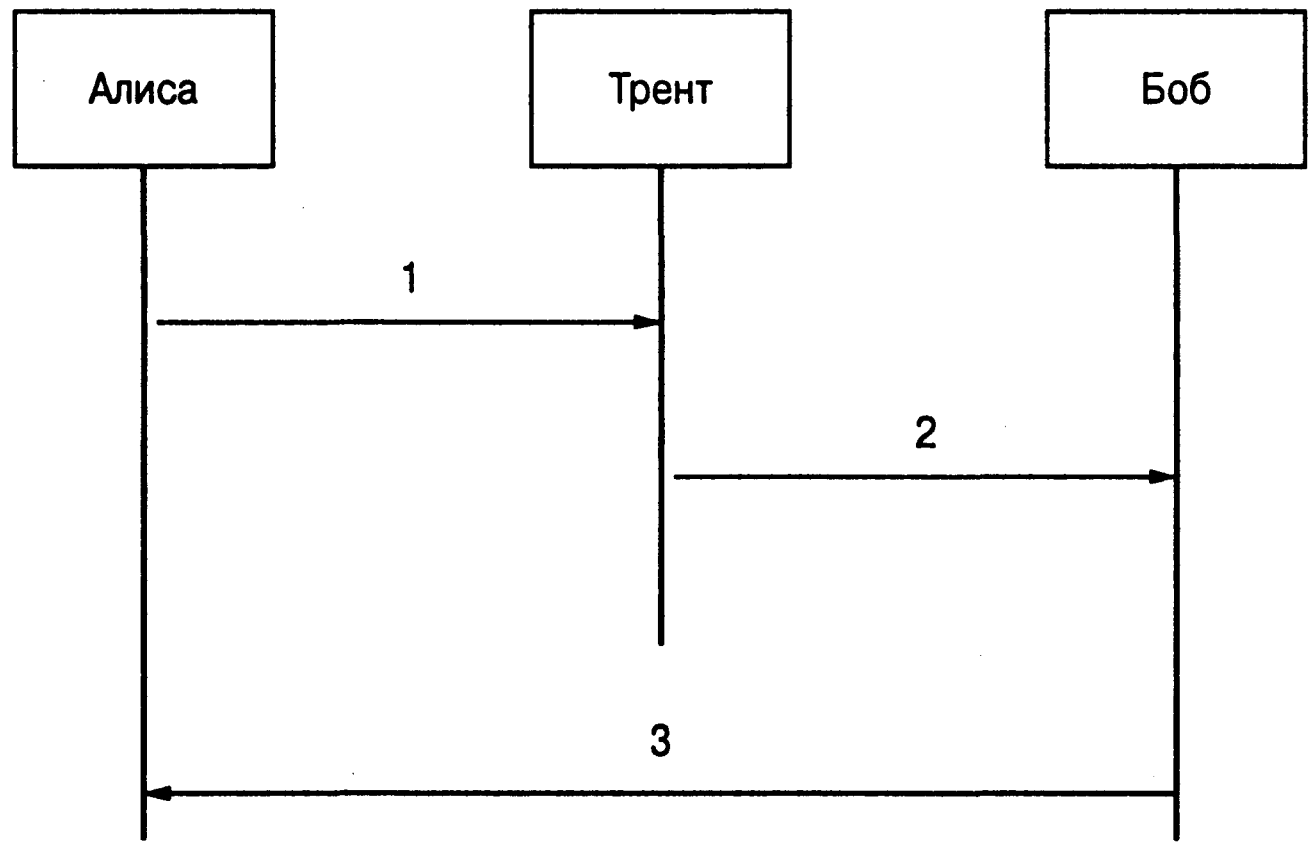
Прежде чем приступить к анализу стойкости протокола "Алиса — Бобу", сделаем несколько замечаний о его особенностях. Протокол позволяет Алисе генерировать сеансовый ключ, который она разделяет с Бобом. Понравится ли это Бобу? Если сеансовый ключ, сгенерированный Алисой, окажется недостаточно случайным (сеансовый ключ должен быть случайным, чтобы предотвратить возможность его угадывания), безопасность Боба будет подвергнута риску — ведь ключ общий! Вдруг Алиса не заботится о том, чтобы сеансовый ключ был стойким, или боится его забыть?! Поскольку Боб может не доверять Алисе (он может даже не быть с ней знаком до начала протокола), использование общего ключа, сгенерированного Алисой, создает для Боба определенные неудобства. Учитывая этот факт, мы сформулируем модифицированный протокол, лишенный этого недостатка, и обсудим стойкость этого протокола.

Протокол 2.1. Алиса — Бобу

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Трент имеют общий ключ K_{AT}; Боб и Трент имеют общий ключ K_{BT}.

ЦЕЛЬ: Алиса и Боб желают создать новый общий секретный ключ K.



1. Алиса генерирует случайный ключ K, создает сообщение $\{K\}_{K_{AT}}$ и посылает Тренту следующую информацию: *Алиса*, *Боб*, $\{K\}_{K_{AT}}$.
2. Трент находит в базе данных ключи K_{AT} и K_{BT}, восстанавливает ключ K, создает сообщение $\{K\}_{K_{BT}}$ и посылает Бобу следующую информацию: *Алиса*, *Боб*, $\{K\}_{K_{BT}}$.
3. Боб расшифровывает сообщение $\{K\}_{K_{BT}}$, восстанавливает ключ K и посылает Алисе сообщение $\{$*Привет, Алиса, я — Боб*!$\}_K$.

2.6.1.2 Протокол “сеансовый ключ от Трента”

Поскольку Тренту доверяют оба клиента, генерацию сеансового ключа можно поручить ему. Таким образом, протокол 2.1 трансформируется в протокол 2.2. Он начинается с того, что Алиса посылает Тренту имена двух пользователей, желающих установить общий сеансовый ключ для безопасного обмена сообщениями, т.е. свое и Боба. Получив запрос от Алисы, Трент находит в своей базе данных соответствующие ключи обоих клиентов, генерирует новый сеансовый ключ, разделяемый между Алисой и Бобом, и шифрует его с помощью ключей, принадлежащих каждому из пользователей. Затем Трент отсылает Алисе сообщение, состоящее из двух частей и содержащее сеансовый ключ, зашифрованный разными ключами. После этого Алиса извлекает из этого сообщения свою часть

и отсылает Бобу часть, предназначенную для него. В заключение Боб выполняет свою часть протокола, который завершается отправкой сообщения, подтверждающего получение сеансового ключа. Будем называть этот протокол "Сеансовый ключ от Трента".

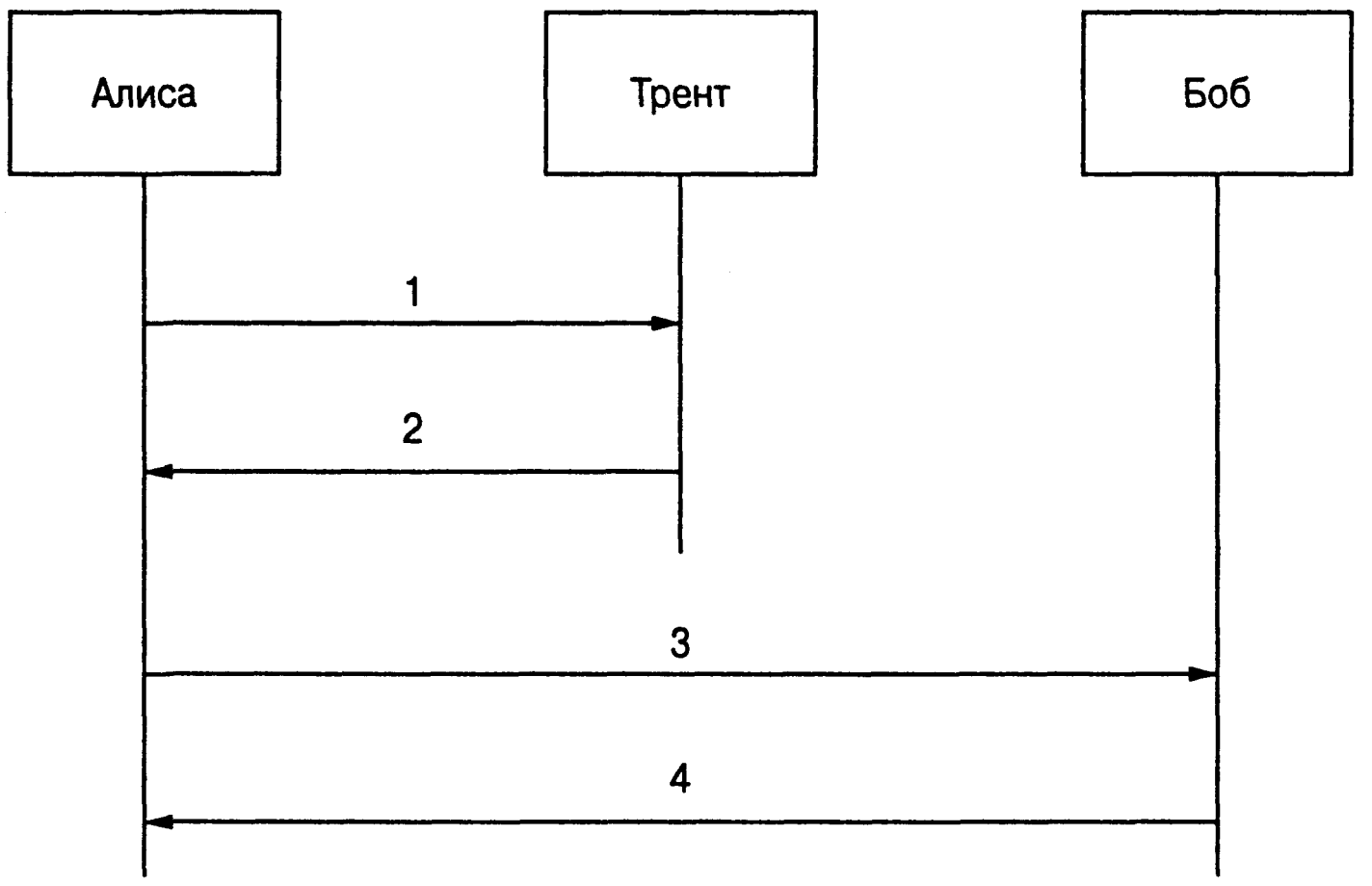
Если сеансовый ключ K зашифрован с помощью идеальной схемы шифрования, пассивный перехватчик сообщений (passive eavesdropper), наблюдающий за протоколом "Сеансовый ключ от Трента" и не знающий ключей K_{AT} и K_{BT}, не сможет определить сеансовый ключ K, поскольку его могут прочесть лишь законные получатели, расшифровав его своим отдельным ключом.

Протокол 2.2. Сеансовый ключ от Трента

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Трент имеют общий ключ K_{AT}; Боб и Трент имеют общий ключ K_{BT}.

ЦЕЛЬ: Алиса и Боб желают установить новый общий секретный ключ K.



1. Алиса посылает Тренту сообщение: *Алиса, Боб.*
2. Трент находит в базе данных ключи K_{AT} и K_{BT}, генерирует случайный ключ K и посылает Алисе следующую информацию: $\{K\}_{K_{AT}}$, $\{K\}_{K_{BT}}$.
3. Алиса расшифровывает сообщение $\{K\}_{K_{AT}}$ и посылает Бобу информацию: *Трент, Алиса,* $\{K\}_{K_{BT}}$.
4. Боб расшифровывает сообщение $\{K\}_{K_{BT}}$, восстанавливает ключ K и посылает Алисе сообщение $\{$*Привет, Алиса, я — Боб*!$\}_K$.

2.6.2 Атака, исправление, атака, исправление...

Проиллюстрируем стандартную схему, которая красной нитью проходит через всю книгу: атака, исправление, атака, исправление...

2.6.2.1 Атака

И все же протокол "Сеансовый ключ от Трента" имеет недостаток: информация о том, *кто именно* должен получить сеансовый ключ, не защищена. Рассмотрим атаку 2.1. В ходе этой атаки Злоумышленник перехватывает некоторые сообщения, передаваемые через сеть, модифицирует их и посылает некоторым пользователям, маскируясь под других пользователей. В атаке 2.1 содержится выражение "Алиса — Злоумышленнику ("Тренту")...". Это значит, что Злоумышленник перехватил сообщение, которое Алиса отправила Тренту. Аналогично выражение "Злоумышленник ("Алиса") — Тренту..." означает, что Злоумышленник посылает сообщение Тренту, маскируясь под Алису. Следует заметить, что в соответствии с моделью угрозы Долева–Яо, описанной в разделе 2.3, Злоумышленник полностью контролирует уязвимую сеть. Таким образом, Злоумышленник способен на враждебные действия. Имя пользователя можно сравнить с маской, которую надевает на себя Злоумышленник, манипулируя протоколом сообщений, передаваемых через сеть. В разделе 12.2 мы проведем технический анализ и покажем, как Злоумышленник может манипулировать сообщениями в открытой сети.

Злоумышленник начинает с перехвата исходного сообщения, отправленного Алисой Тренту. Это сообщение содержит инструкцию Тренту, следуя которой он должен сгенерировать сеансовый ключ, разделяемый им с Алисой и Бобом. Злоумышленник подменяет имя Боба своим и посылает Тренту фальсифицированное сообщение. Трент будет считать, что Алиса хочет связаться со Злоумышленником, поэтому генерирует новый сеансовый ключ K_{AM}, общий для Алисы и Злоумышленника. Затем Трент зашифровывает его с помощью соответствующих ключей, которые он разделяет с обоими клиентами: Алисой и Злоумышленником. Поскольку Алиса не может различить сообщения, посылаемые другим пользователям, она не заметит подмены. Затем Злоумышленник перехватывает сообщение, посланное Алисой Бобу, и Боб не узнает, что Алиса вызывает его на контакт. В результате атаки Алиса будет уверена, что Боб успешно выполнил протокол, в то время как на самом деле Боба заменил Злоумышленник. Поскольку ему известен ключ K_{AM}, вся информация, передаваемая Алисой Бобу, будет раскрыта. Обратите внимание на то, что эта атака будет успешной лишь в том случае, если Злоумышленник является законным пользователем, известным Тренту. Это вполне реалистичное предположение — атаки изнутри встречаются чаще, чем атаки извне.

Как мы видели, описанная выше атака стала возможной благодаря подмене Боба Злоумышленником. Эта подмена стала возможной потому, что имя Боба

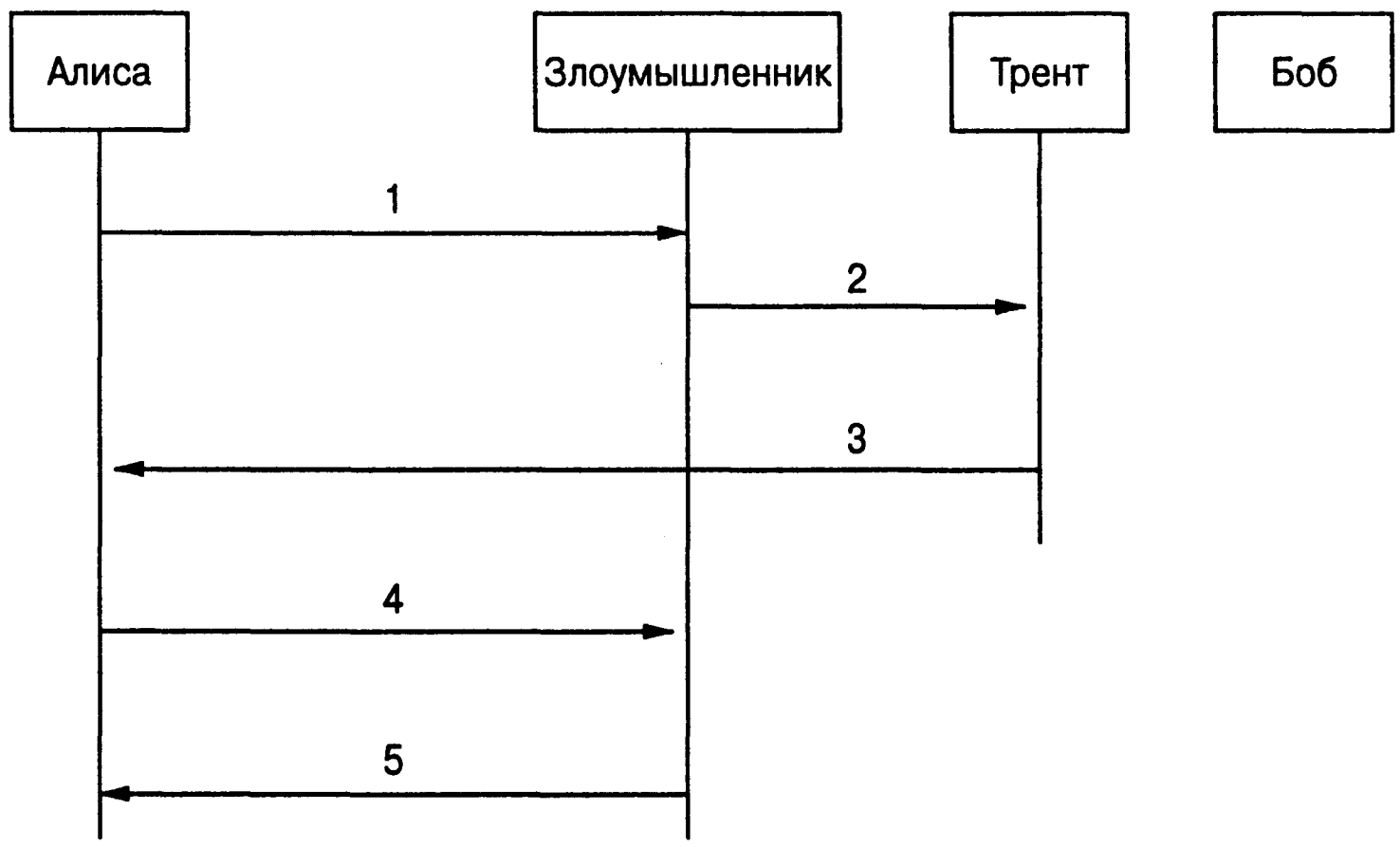
Атака 2.1. Атака на протокол "Сеансовый ключ от Трента"

ИСХОДНЫЕ УСЛОВИЯ:

В дополнение к протоколу "Сеансовый ключ от Трента" Злоумышленник и Трент имеют общий ключ K_{MT}.

РЕЗУЛЬТАТ АТАКИ:

Алиса думает, что имеет общий ключ с Бобом, а на самом деле она делит ключ со Злоумышленником.



1. Алиса — Злоумышленнику ("Тренту"): *Алиса*, *Боб*.
2. Злоумышленник ("Алиса") — Тренту: *Алиса*, *Злоумышленник*.
3. Трент находит в базе данных ключи K_{AT} и K_{MT}, генерирует случайный ключ K_{AM} и посылает Алисе следующую информацию: $\{K_{AM}\}_{K_{AT}}$, $\{K_{AM}\}_{K_{MT}}$.
4. Алиса расшифровывает сообщение $\{K_{AM}\}_{K_{AT}}$ и посылает Бобу информацию: *Трент*, *Алиса*, $\{K_{AM}\}_{K_{MT}}$.
5. Злоумышленник ("Боб") — Алисе: $\{$*Привет, Алиса, я — Боб*!$\}_{K_{AM}}$.

пересылалось открытым текстом. Значит, протокол можно исправить, скрыв имя Боба.

2.6.2.2 Исправление

Атака, в ходе которой происходит подмена Боба Злоумышленником, демонстрирует уязвимое место протокола "Сеансовый ключ от Трента". Например, можно модифицировать протокол, скрыв имя Боба в первом сообщении и зашиф-

ровав его с помощью ключа, общего для Алисы и Трента. Иначе говоря, первое сообщение в протоколе "Сеансовый ключ от Трента" следует сформулировать так.

1. Алиса — Тренту: *Алиса*, $\{Боб\}_{K_{AT}}$.

Обратите внимание на то, что имя Алисы должно по-прежнему передаваться открытым текстом, чтобы Трент мог определить, какой ключ следует применить для дешифровки сообщения.

2.6.2.3 Новая атака

Однако предложенного исправления недостаточно. Например, Злоумышленник может сделать следующее.

1. Злоумышленник ("Алиса") — Тренту: *Алиса*, $\{Злоумышленник\}_{K_{AT}}$.

Остальную часть атаки оставим без изменения. Хотя в начале Злоумышленник и не знает, кого именно Алиса вызывает на связь, ему все же известно, какая часть информации в перехваченном сообщении содержит адрес Боба. Ее местоположение фиксировано, чтобы сообщение было правильно передано через сеть. Обратите внимание на то, что в ходе этой атаки предполагается, что Злоумышленник имеет зашифрованное сообщение $\{Злоумышленник\}_{K_{AT}}$. Это возможно, если Злоумышленник записал его в ходе выполнения предыдущего (корректного) протокола связи между Алисой и Злоумышленником.

2.6.2.4 Еще одна атака

Приведем пример атаки, в которой вообще не используется подмена адресата. Вместо этого Злоумышленник может изменить сообщение, направленное Трентом Алисе (вторая строка протокола "Алиса — Бобу").

2. Злоумышленник ("Трент") — Алисе: $\{K'\}_{K_{AT}}, \ldots$

Здесь K' — это сеансовый ключ, переданный в предыдущем (корректном) протоколе связи между Алисой и Злоумышленником, в ходе которого Злоумышленник записал зашифрованное сообщение $\{K'\}_{K_{AT}}$. Остальная часть атаки аналогична атаке 2.1: Злоумышленник должен перехватить последующее сообщение, направленное Алисой Бобу, и подтвердить его получение, маскируясь под Боба.

Злоумышленник ("Боб") — Алисе: $\{Привет,\ Алиса,\ я — Боб!\}_{K'}$.

Тот факт, что исправленный вариант протокола "Сеансовый ключ от Трента" можно атаковать, не подменяя имени Боба явным образом, демонстрирует, что защита информации, идентифицирующей имя Боба в первом сообщении, недостаточно сильна. Кроме того, в ходе атаки, описанной выше, предполагалось, что Злоумышленник может скрытно изменять протокольные сообщения. Таким образом, протокол должен содержать средства безопасности, защищающие сообщения от тайного вмешательства.

Это приводит нас к следующей идее.

2.6.3 Протокол с аутентификацией сообщений

До сих пор Злоумышленник всегда мог скрытно фальсифицировать протокольные сообщения. Действительно ни один из протоколов, описанных выше, не обеспечивал криптографической защиты сообщений от искажений. Следовательно, для исправления таких протоколов следует предусмотреть средства, позволяющие законным пользователям, владеющими правильными криптографическими ключами, обнаруживать несанкционированное изменение любого протокольного сообщения. Такая защита называется **аутентификацией сообщений** (message authentication). (В некоторых учебниках это свойство называется **целостностью данных** (data integrity), однако в главе 11 мы будем различать эти понятия.)

2.6.3.1 Протокол "Аутентификация сообщений"

Как мы видели, злонамеренное изменение протокольных сообщений приводит к двум последствиям: сеансовый ключ либо разделяется между неправильными клиентами или вообще неверно устанавливается. Таким образом, необходимо предусмотреть аутентификацию сообщений, обеспечивающую связь между создаваемым сеансовым ключом и пользователями, вступающими в контакт. Это порождает новый протокол 2.3, в котором имена Алисы и Боба передаются Тренту в зашифрованном виде. Назовем этот протокол "Аутентификация сообщений".

Обратите особое внимание на часть спецификации, содержащую следующие инструкции.

3. Алиса расшифровывает сообщение $\{Боб, K\}_{K_{AT}}$, **устанавливает личность Боба**...
4. Боб расшифровывает сообщение $\{Алиса, K\}_{K_{BT}}$, **устанавливает личность Алисы**...

Идентификация личности клиента представляет собой чрезвычайно важное различие между данным и предшествующими протоколами (т.е. протоколом "Сеансовый ключ от Трента" и его модификациями). Идентификация личности возможна лишь после правильной расшифровки соответствующего зашифрованного текста с помощью правильных криптографических ключей. Таким образом, криптографическая операция "расшифровка и проверка", выполняемая получателем, позволяет осуществить аутентификацию и убедиться, что сеансовый ключ установлен верно и предназначен для законных пользователей. Правильный результат расшифровки означает, что шифрованное сообщение не было фальсифицировано при передаче по сети. Таким образом, протокол "Аутентификация сообщений" должен предотвратить атаки, описанные выше.

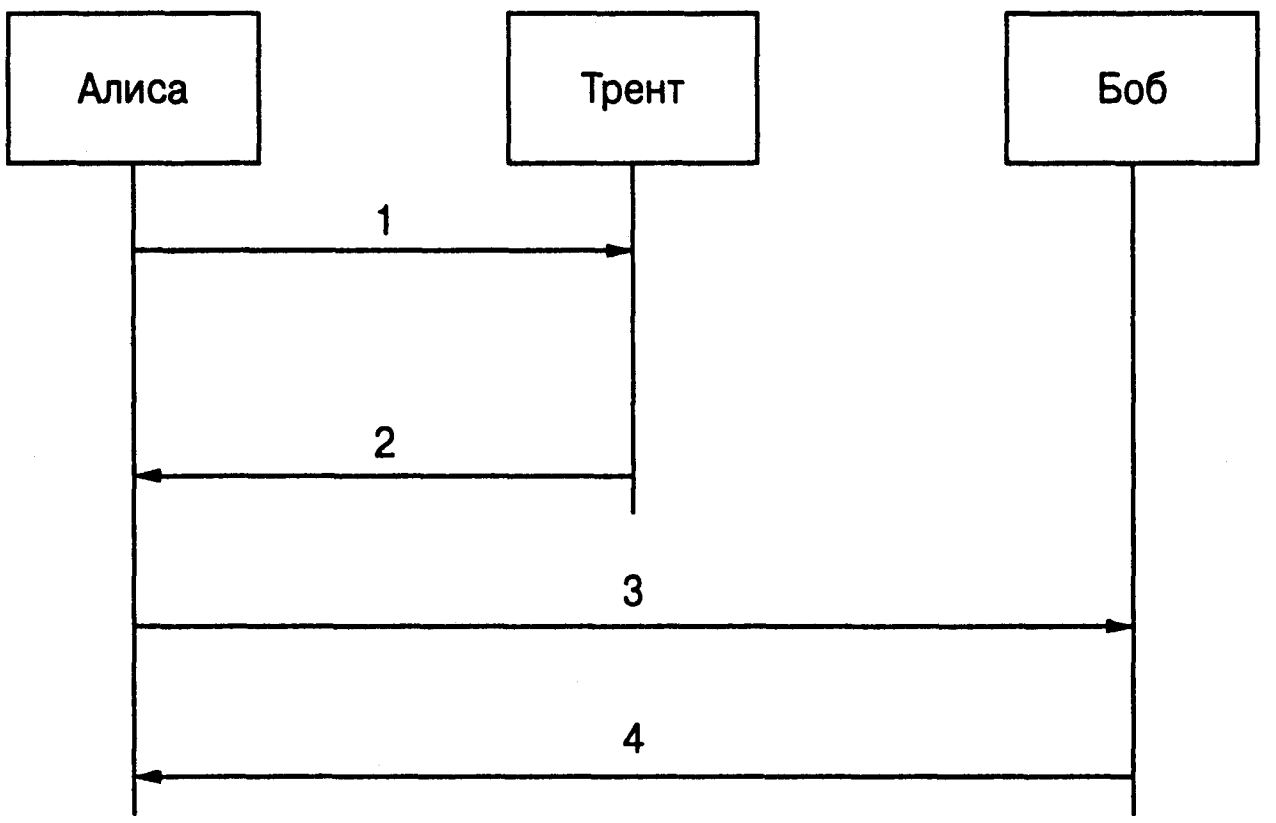
Необходимо отметить, что криптографическая операция "расшифровка и проверка" (выполняемая получателем) описана не совсем точно. В главе 17 мы убедимся, что более точным было бы назвать ее "повторное шифрование и проверка"

Протокол 2.3. Аутентификация сообщений

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Трент имеют общий ключ K_{AT}; Боб и Трент имеют общий ключ K_{BT}.

ЦЕЛЬ: Алиса и Боб желают создать новый общий секретный ключ K.



1. Алиса посылает Тренту сообщение: *Алиса*, *Боб*.
2. Трент находит в базе данных ключи K_{AT} и K_{BT}, генерирует случайный ключ K и посылает Алисе следующую информацию: $\{Боб, K\}_{K_{AT}}$, $\{Алиса, K\}_{K_{BT}}$.
3. Алиса расшифровывает сообщение $\{Боб, K\}_{K_{AT}}$, устанавливает личность Боба и посылает ему информацию: *Трент*, $\{Алиса, K\}_{K_{BT}}$.
4. Боб расшифровывает сообщение $\{Алиса, K\}_{K_{BT}}$, устанавливает личность Алисы и посылает Алисе сообщение $\{Привет, Алиса, я — Боб!\}_K$.

(снова выполняемая получателем). Причина, по которой мы пользуемся не совсем корректным вариантом этой операции, весьма проста: на данном этапе мы предполагаем, что из всех криптографических операций читателю известны лишь "шифровка отправителем" и "расшифровка получателем".

Поскольку в ходе аутентификации сообщения мы пользуемся неправильной операцией, необходимо явно указать, какими дополнительными свойствами должен обладать алгоритм шифрования. Эти свойства перечислены ниже (нумерация продолжает нумерацию пунктов из перечня свойств "Идеальное шифрование $\{M\}_K$" в разделе 2.2).

Свойство 2.2. Идеальное шифрование $\{M\}_K$ (для аутентификации сообщений)

III. Без ключа K, даже имея открытый текст M, невозможно изменить сообщение $\{M\}_K$ так, чтобы получатель не заметил этого при расшифровке.

Для того чтобы продемонстрировать важность данного ограничения, рассмотрим атаку на протокол "Аутентификация сообщений", предполагая, что идеальный алгоритм шифрования не обладает свойством III (т.е., что алгоритм шифрования обладает лишь свойствами идеальной секретности, перечисленными в разделе 2.2). Для простоты представим зашифрованные блоки

$$\{Боб, K\}_{K_{AT}}, \quad \{Алиса, K\}_{K_{BT}}$$

в виде

$$\{Боб\}_{K_{AT}}, \quad \{K\}_{K_{AT}}, \quad \{Алиса\}_{K_{BT}}, \quad \{K\}_{K_{BT}}.$$

Такое представление зашифрованных блоков позволяет разрушить криптографическую связь между пользователями и сеансовым ключом, несмотря на то, что алгоритм шифрования обладает свойствами идеальной секретности. Протокол "Аутентификация сообщений", использующий "идеальную" схему шифрования, должен содержать следующие строки.

2. Трент — Алисе: $\{Боб\}_{K_{AT}}$, $\{K\}_{K_{AT}}$, $\{Алиса\}_{K_{BT}}$, $\{K\}_{K_{BT}}$.
3. Алиса расшифровывает сообщения $\{Боб\}_{K_{AT}}$ и $\{K\}_{K_{AT}}$, устанавливает личность Боба...
4. Боб расшифровывает сообщения $\{Алиса\}_{K_{BT}}$ и $\{K\}_{K_{BT}}$, устанавливает личность Алисы...

Очевидно, что конфиденциальность сообщения ничем не гарантируется. Злоумышленнику достаточно просто наблюдать за обменом сообщениями между отправителями и получателями, чтобы определить, в каком месте зашифрованных сообщений $\{Боб\}_{K_{AT}}$ и $\{Алиса\}_{K_{BT}}$ находится открытый текст. Следовательно, модифицированный протокол, по существу, совпадает с протоколом "Сеансовый ключ от Трента" и, как показано в разделе 2.6.2, может быть уязвим для атаки. Читатель может решить эту задачу в качестве самостоятельного упражнения.

2.6.3.2 Атака на протокол "Аутентификация сообщений"

Даже если алгоритм шифрования обладает возможностью аутентификации сообщений, протокол "Аутентификация сообщений" уязвим для атак. Проблема порождается качественной разницей между долговременными ключами для шифрования ключей, изначально распределенными между Трентом и клиентами, и сеансовыми ключами, генерируемыми для каждого выполнения протокола.

Во-первых, обратим внимание на то, что отношения между Трентом и каждым клиентом носят долговременный характер. Это значит, что ключи, распределенные между ним и его клиентами, являются долговременными. Как правило,

создание ключа для связи между сервером аутентификации и клиентом — более сложный и дорогой процесс, чем генерация сеансового ключа для связи между двумя клиентами. (Он требует строгих процедур безопасности, а иногда и личной встречи.) К счастью, такие ключи обычно применяются в протоколах аутентификации, которые редко используют шифрование небольшого количества сообщений с малой избыточностью, и, следовательно, такие ключи предоставляют мало материала для криптоанализа. Таким образом, секретные ключи, разделенные между сервером аутентификации и его клиентами, можно долго использовать. Именно поэтому их называют долговременными.

С другой стороны, следует помнить о принципе управления ключом, описанном в разделе 2.5, который требует, чтобы сеансовый ключ использовался лишь на протяжении одного сеанса. Следовательно, протокол создания сеансовых ключей не должен генерировать повторяющиеся ключи. Однако к протоколу "Аутентификация сообщений" это не относится. Атака на этот протокол приводит к нарушению принципа управления сеансовым ключом. В ходе этой атаки Злоумышленнику достаточно сделать две вещи. Сначала он должен перехватить запрос Алисы (см. протокол 2.3).

1. Алиса — Злоумышленнику ("Тренту"): ...

Затем нужно заменить сообщение во второй строке.

2. Злоумышленник ("Трент") — Алисе: $\{Боб, K'\}_{K_{AT}}$, $\{Алиса, K'\}_{K_{BT}}$.

Здесь два шифрованных блока, содержащих сообщение K', являются **воспроизведениями** (replay) старых сообщений, записанных Злоумышленником в ходе предыдущих сеансов протокола (нормальных контактов между Алисой и Бобом). Следовательно, эта атака заставит Алису и Боба повторно использовать старый сеансовый ключ K', чего делать не следовало. Обратите внимание на то, что, поскольку ключ K' уже однажды использовался, Злоумышленник может раскрыть его значение (либо вследствие нарушения принципов безопасности, либо в силу уязвимости сеансового ключа, о которой шла речь в разделе 2.5). Следовательно, злоумышленник может либо перехватить секретный сеанс связи между Алисой и Бобом, либо вступить в диалог с Алисой под маской Боба.

Атака такого рода называется **атакой с повторной передачей сообщений** (message replay attack).

2.6.4 Протокол "оклик-отзыв"

Существует несколько механизмов, позволяющих пользователю проверить, что протокольное сообщение не является воспроизведением старого сообщения. Эти механизмы будут рассмотрены в главе 11. Пока мы можем улучшить наш протокол с помощью хорошо известного метода **"оклик-отзыв"** (challenge-response). Иногда его называют также **квитированием установления связи** (handshake).

С помощью этого метода в начале протокола Алиса генерирует новое случайное число N_A и посылает его Тренту вместе с запросом на новый сеансовый ключ. Если в ответ она получит сеансовый ключ, связанный с числом N_A, причем эти два куска криптографически связаны между собой и допускают аутентификацию (т.е. Алиса может верифицировать целостность зашифрованного сообщения, содержащего число N_A), то Алиса может прийти к выводу, что данная криптографическая связь была образована Трентом, получившим число N_A. Более того, поскольку Трент всегда вне подозрений (см. раздел 2.4), Алиса знает, что Трент всегда строго следует протоколу. Значит, Трент действительно создал новый сеансовый ключ *после* того, как получил от Алисы случайное число. Следовательно, сеансовый ключ является новым (или **свежим**, или **текущим**), т.е. не является повторением одного из старых ключей. Случайное число N_A, созданное Алисой для реализации механизма "оклик-отзыв", называется **одноразовым** (nonce — a number **u**sed **once**) [61].

2.6.4.1 Протокол "оклик-отзыв" (Нидхема–Шредера)

Протокол 2.4 использует механизм "оклик-отзыв", с помощью которого Алиса может проверить новизну сеансового ключа. Пока будем называть его протоколом "оклик-отзыв" (вскоре мы изменим это название).

В этом протоколе Боб также создает одноразовое случайное число N_B, однако оно не отсылается Тренту, поскольку в рамках этого протокола Боб с Трентом непосредственно не контактируют. Вместо этого Боб отсылает случайное число N_B Алисе, а затем получает число $N_B - 1$. Если Алису удовлетворяет сеансовый ключ M, она применяет его к числу N_B, чтобы Боб, в свою очередь, убедился в новизне сеансового ключа. Таким образом, устанавливается обоюдная уверенность в качестве сеансового ключа.

Протокол "оклик-отзыв", сформулированный выше, вероятно, является самым знаменитым протоколом аутентификации и установления ключа. Именно в таком виде его сформулировали Нидхем и Шредер в 1978 г. [213]. В дальнейшем мы будем называть его протоколом Нидхема–Шредера для аутентификации с симметричным ключом (Needham–Schroeder Symmetric-key Authentication Protocol). Этот протокол стал основой для целого класса подобных протоколов.

2.6.4.2 Атака на протокол Нидхема–Шредера для аутентификации с симметричным ключом

К сожалению, протокол Нидхема–Шредера уязвим для атаки, изобретенной Деннингом (Denning) и Сакко (Sacco) в 1981 г. [94]. В ходе этой атаки Злоумышленник перехватывает сообщения, которые присылаются Алисе и отсылаются ею в пунктах 3, 4 и 5, и заменяет их своими вариантами.

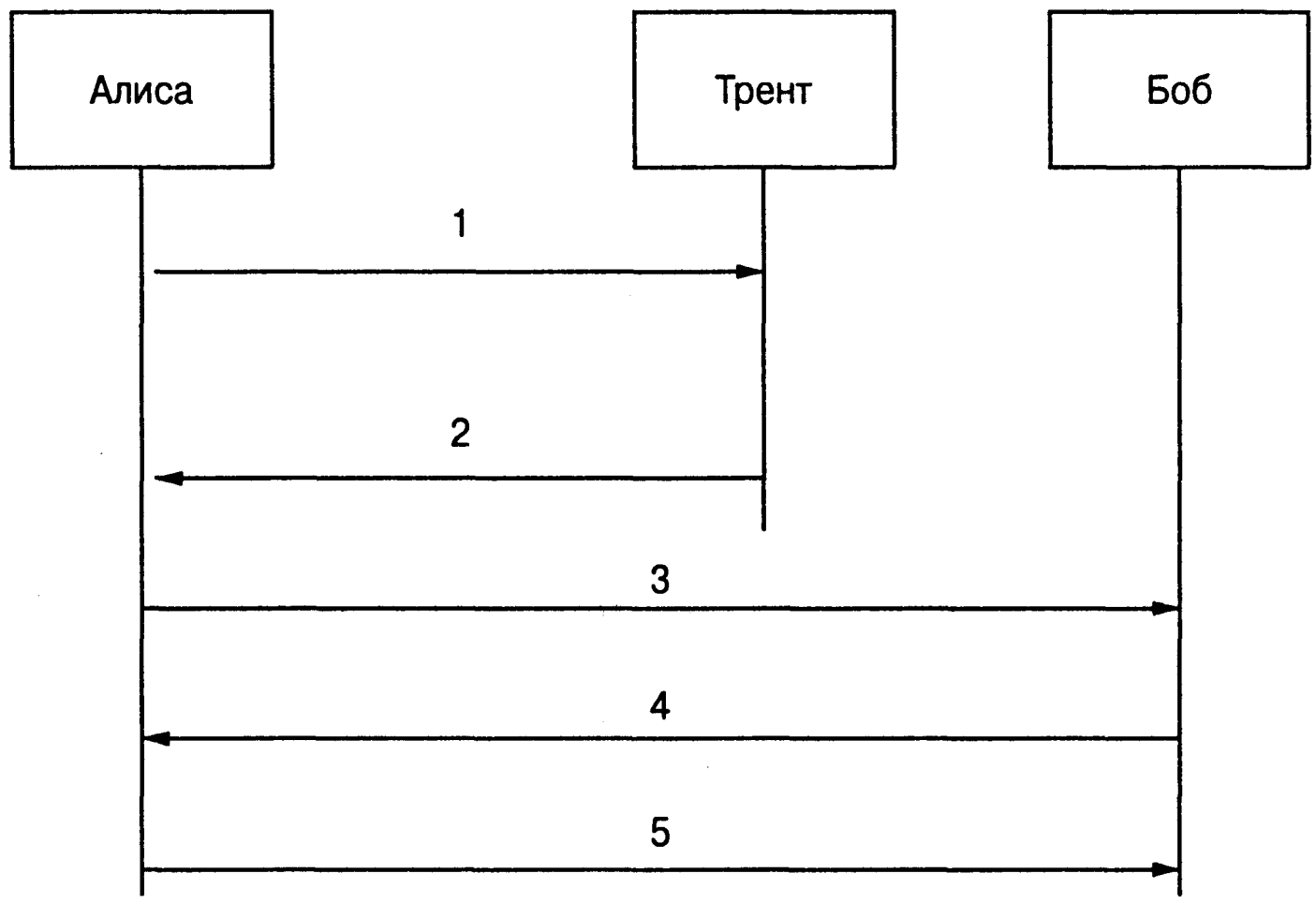
В ходе этой атаки Злоумышленник активизируется на третьем этапе, перехватывая сообщение, направленное Алисой Бобу. Он полностью блокирует канал

Протокол 2.4. Оклик-отзыв

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Трент имеют общий ключ K_{AT}; Боб и Трент имеют общий ключ K_{BT}.

ЦЕЛЬ: Алиса и Боб желают создать новый общий секретный ключ K.



1. Алиса генерирует случайное число N_A и посылает Тренту сообщение: *Алиса*, *Боб*, N_A.
2. Трент генерирует случайный ключ K и посылает Алисе следующую информацию: $\{N_A, K, \textit{Боб}, \{K, \textit{Алиса}\}_{K_{BT}}\}_{K_{AT}}$.
3. Алиса расшифровывает сообщение $\{N_A, K, \textit{Боб}, \{K, \textit{Алиса}\}_{K_{BT}}\}_{K_{AT}}$, проверяет число N_A, устанавливает личность Боба и посылает ему информацию: *Трент*, $\{K, \textit{Алиса}\}_{K_{BT}}$.
4. Боб расшифровывает сообщение, устанавливает личность Алисы и посылает ей сообщение $\{\textit{Привет, Алиса, я — Боб}!, N_B\}_K$.
5. Алиса посылает Бобу сообщение: $\{\textit{Я — Алиса}!\ N_B - 1\}_K$.

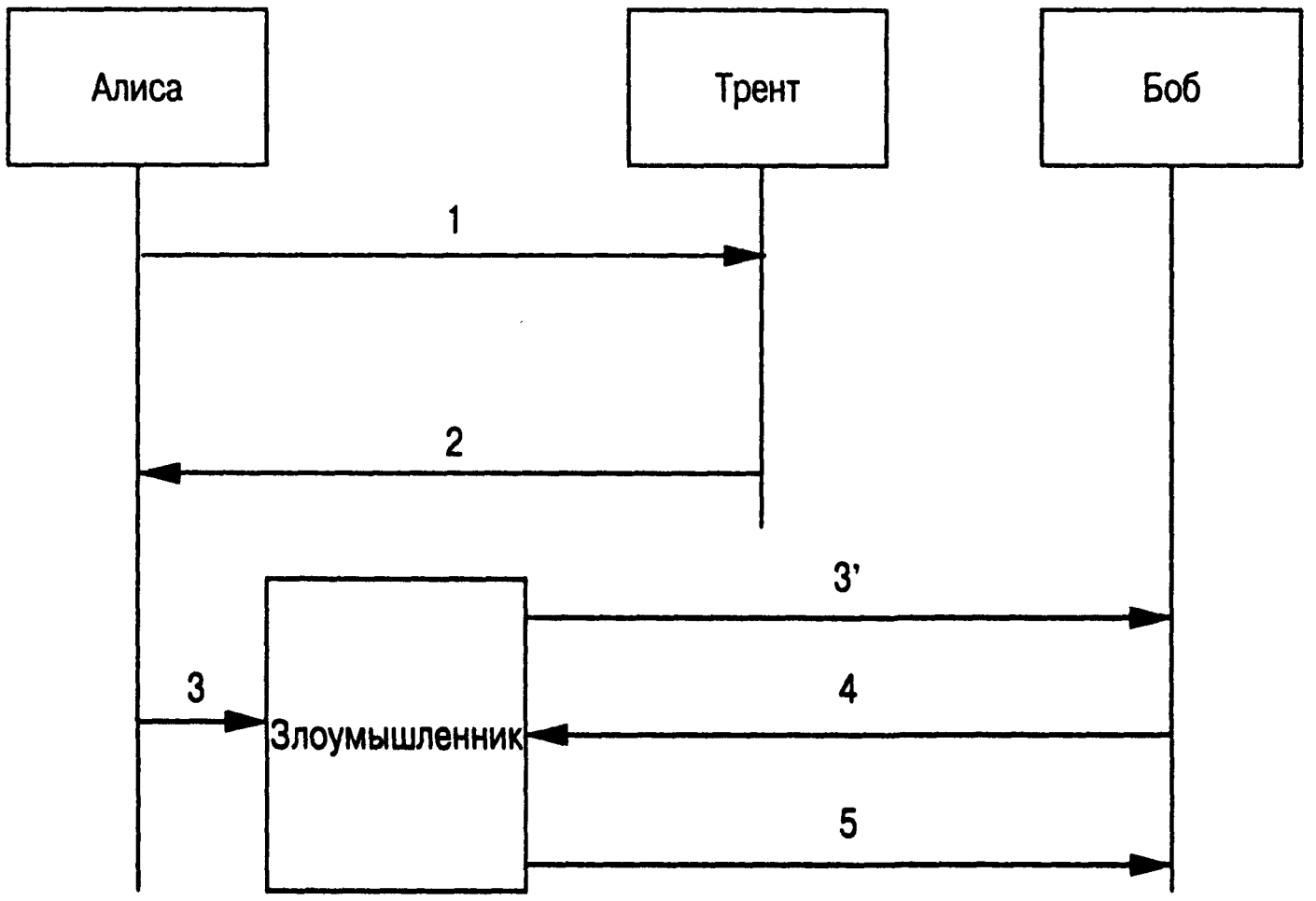
связи Алисы и заменяет его содержание материалом старого сеанса между Алисой и Бобом: $\{K', \textit{Алиса}\}_{K_{BT}}$. Исходя из предположения об уязвимости старого сеансового ключа, Злоумышленник может узнать значение K' и начать атаку, вступив в контакт с Бобом под маской Алисы.

Использование старого сеансового ключа — лишь одна из опасностей, порождаемых этой атакой. Другая угроза заключается в том, что Злоумышленник успешно разрушает один из основных механизмов аутентификации. В чем заклю-

Атака 2.2. Атака на протокол Нидхема–Шредера для аутентификации с симметричным ключом

РЕЗУЛЬТАТ АТАКИ:

Боб думает, что имеет общий с Алисой новый сеансовый ключ, но на самом деле ключ является старым и известен Злоумышленнику.



1. Алиса генерирует случайное число N_A и посылает Тренту сообщение: *Алиса*, *Боб*, N_A.
2. Трент генерирует случайный ключ K и посылает Алисе следующую информацию: $\{N_A, K, Боб, \{K, Алиса\}_{K_{BT}}\}_{K_{AT}}$.
3. Алиса расшифровывает сообщение $\{N_A, K, Боб, \{K, Алиса\}_{K_{BT}}\}_{K_{AT}}$, проверяет число N_A, устанавливает личность Злоумышленника ("Боба") и посылает ему информацию: *Трент*, $\{K, Алиса\}_{K_{BT}}$.

3'. Злоумышленник ("Алиса") отсылает Бобу сообщение $\{K', Алиса\}_{K_{BT}}$.

4. Боб расшифровывает сообщение, устанавливает личность Алисы и посылает ей сообщение $\{Я - Боб!\ N_B\}_{K'}$.
5. Злоумышленник ("Алиса") посылает Бобу сообщение: $\{Я - Алиса!\ N_B - 1\}_{K'}$.

чается этот механизм, мы укажем в разделе 11.2.2, а способ его преодоления будет описан в разделе 11.7.1.

2.6.5 Протокол с аутентификацией сущности

Механизм "оклика-отзыва", используемый в протоколе Нидхема–Шредера (связь между Алисой и Трентом), обеспечивает так называемую **аутентификацию сущности** (entity authentication). Подобно аутентификации сообщения, аутентификация сущности обеспечивается с помощью проверки некоей криптографической операции. Эта проверка выполняется верифицирующим пользователем (verification principal). Разница между этими двумя видами аутентификации заключается в том, что во втором случае демонстрируется существование доказывающего пользователя (proving principal). Это существование считается подтвержденным, если доказывающий пользователь выполнил некую криптографическую операцию *после* события, которое верифицирующий пользователь считает *последним*. На втором этапе протокола Нидхема–Шредера, получив сообщение от Трента, Алиса расшифровывает одноразовое случайное число N_A, сгенерированное ею на первом этапе. Это убеждает ее в том, что Трент выполнил шифрование только *после* получения числа N_A от Алисы (поскольку Алиса и Трент используют общий ключ). Таким образом, Алиса знает, что Трент существовал после этого события. В этом и заключается аутентификация существования Трента по отношению к Алисе.

Однако Боб, участвующий в протоколе Нидхема–Шредера, не имеет никаких доказательств существования Трента.

Как правило, если проблема обнаружена, ее довольно просто разрешить: Трент должен доказать свое существование *обоим* клиентам. Это можно сделать, например, если Боб также пошлет Тренту свое одноразовое случайное число, которое Трент включит в свое ответное сообщение вместе с сеансовым ключом. Это приведет к увеличению количества сообщений, передаваемых в рамках протокола (дополнительное квитирование установления связи между Бобом и Трентом). Для того чтобы избежать возникновения дополнительных информационных потоков, Деннинг и Сакко предложили использовать **метки времени** (timestamps) [94].

2.6.5.1 Метки времени

Обозначим метку времени буквой T. Деннинг и Сакко предложили следующий способ решения проблемы аутентификации существования.

1. Алиса — Тренту: *Алиса*, *Боб*.
2. Трент — Алисе: $\{Боб, K, T, \{Алиса, K, T\}_{K_{BT}}\}_{K_{AT}}$.
3. Алиса — Бобу: $\{Алиса, K, T\}_{K_{BT}}$.
4. ... То же, что и в протоколе Нидхема–Шредера.
5. ... То же, что и в протоколе Нидхема–Шредера.

Получив протокольные сообщения от Трента, Алиса и Боб могут обнаружить, что их послания остались без ответа, проверив неравенство

$$|Время - T| < \Delta t_1 + \Delta t_2.$$

Здесь *Время* означает локальное время получателя, Δt_1 — интервал, представляющий допустимую разницу между временем Трента и локальным временем, Δt_2 — ожидаемая временна́я задержка. Если часы всех клиентов сверены по эталону, то величина Δt_1, равная одной-двум минутам, вполне допустима. Поскольку величина $\Delta t_1 + \Delta t_2$ меньше длины интервала времени, прошедшего с момента последнего протокольного действия, этот метод предотвращает атаку с повторениями (replay attack), т.е. атаку 2.2. Поскольку метка времени T зашифрована с помощью секретных ключей K_{AT} и K_{BT}, имитация Трента в идеальной схеме шифрования невозможна.

Нидхем и Шредер также рассматривали возможность применения меток времени, однако отвергли ее, поскольку для их использования необходим качественный эталон времени [212].

2.6.6 Протокол на основе криптосистем с открытым ключом

Последний протокол, который мы рассмотрим в этой главе, называется протоколом Нидхема–Шредера для аутентификации с открытым ключом (Needham–Schroeder Public-key Authentication Protocol) [213]. Этот протокол интересен по двум причинам, выходящим за рамки рассматриваемых тем. Во-первых, протокол позволяет получить первое представление о криптосистемах с открытым ключом. Во-вторых, он позволяет продемонстрировать изощренную атаку. Несмотря на то что протокол выглядит довольно просто, эта атака была изобретена лишь через семнадцать лет после его опубликования.

2.6.6.1 Криптосистемы с открытым ключом

Обозначим открытый ключ Алисы через K_A, а соответствующий секретный ключ Алисы — через K_A^{-1}. Предполагается, что секретный ключ известен только Алисе. Открытый текст M, идеально зашифрованный с помощью ключа K_A, обозначим символами

$$\{M\}_{K_A}.$$

Предполагается, что расшифровка этого текста возможна только с помощью ключа K_A^{-1}. Поскольку этот ключ известен только Алисе, лишь она может расшифровать исходный текст M. Аналогично символами

$$\{M\}_{K_A^{-1}}$$

обозначается открытый текст M, идеально зашифрованный с помощью секретного ключа K_A^{-1}. Расшифровать его можно только с помощью открытого ключа K_A, принадлежащего Алисе. Это значит, что зашифрованный текст $\{M\}_{K_A^{-1}}$ также создан Алисой, поскольку для его шифрования необходим ключ, которым владеет только она. По этой причине зашифрованный текст $\{M\}_{K_A^{-1}}$ называется **цифровой подписью** (digital signature) сообщения M, а его расшифровка с помощью ключа K_A называется **верификацией** подписи Алисы на сообщении M.

2.6.6.2 Протокол Нидхема–Шредера для аутентификации с открытым ключом

Допустим, что Трент владеет открытыми ключами всех обслуживаемых им клиентов. Кроме того, каждый клиент имеет аутентифицированную копию открытого ключа Трента. Протокол 2.5 представляет собой протокол Нидхема–Шредера для аутентификации с открытым ключом.

Протокол 2.5. Протокол Нидхема–Шредера для аутентификации с открытым ключом

ИСХОДНЫЕ УСЛОВИЯ:

Алиса имеет открытый ключ K_A, Боб имеет открытый ключ K_B, Трент имеет открытый ключ K_T.

ЦЕЛЬ: Алиса и Боб желают создать новый общий секретный ключ.

1. Алиса — Тренту: *Алиса*, *Боб*.
2. Трент — Алисе: $\{K_B, \textit{Боб}\}_{K_T^{-1}}$.
3. Алиса верифицирует подпись Трента на сообщении $\{K_B, \textit{Боб}\}_{K_T^{-1}}$, генерирует одноразовое случайное число N_A и посылает Бобу сообщение: *Трент*, $\{N_A, \textit{Алиса}\}_{K_B}$.
4. Боб расшифровывает сообщение, устанавливает личность Алисы и посылает Тренту сообщение: $\{\textit{Боб}, \textit{Алиса}\}$.
5. Трент — Бобу: $\{K_A, \textit{Алиса}\}_{K_T^{-1}}$.
6. Боб верифицирует подпись Трента на сообщении $\{K_A, \textit{Алиса}\}_{K_T^{-1}}$, генерирует свое одноразовое случайное число N_B и посылает Алисе сообщение: $\{N_A, N_B\}_{K_A}$.
7. Алиса расшифровывает сообщение и посылает Бобу информацию: $\{N_B\}_{K_B}$.

Инициатором протокола является Алиса, желающая установить сеанс связи с Бобом при посредничестве Трента. На первом этапе Алиса посылает сообщение Тренту, запрашивая открытый ключ Боба. На втором этапе Трент отсылает ей

ключ K_B вместе с именем Боба (чтобы предотвратить атаку, описанную в разделе 2.6.2). Ключ и номер зашифрованы с помощью секретного ключа Трента K_T^{-1}. Это сообщение представляет собой цифровую подпись Трента, которая должна убедить Алису, что сообщение, полученное ею на втором этапе, отправлено именно Трентом (Алиса должна верифицировать ее с помощью открытого ключа Трента). Затем Алиса генерирует одноразовое случайное число N_A и посылает его Бобу вместе со своим именем (п. 3), зашифровывая сообщение с помощью открытого ключа Боба. Получив сообщение от Алисы, Боб расшифровывает его и получает число N_A. Затем он запрашивает (п. 4) и получает (п. 5) аутентичную копию открытого ключа Алисы. После этого он возвращает Алисе число N_A и свое собственное одноразовое случайное число N_B, зашифрованные с помощью открытого ключа Алисы (п. 6). Получив сообщение от Боба, Алиса убеждается, что она общается именно с Бобом, поскольку только Боб может расшифровать сообщение на этапе 3, содержащее число N_A. Кроме того, Боб мог расшифровать это сообщение только *после* того, как Алиса его отослала (недавнее действие). Затем Алиса возвращает Бобу число N_B, зашифровав его с помощью открытого ключа Боба. Получив это сообщение от Алисы, Боб также убеждается, что он общается именно с ней, поскольку только Алиса могла расшифровать сообщение на этапе 6, содержащее число N_B (еще одно недавнее действие). Таким образом, в результате успешного выполнения протокола возникают два случайных числа N_A и N_B, известные только Алисе и Бобу, причем, поскольку эти числа каждый раз генерируются заново, свойство новизны гарантируется. Кроме того, каждый клиент должен быть убежден в полной случайности этих чисел, поскольку сам генерирует одно из этих чисел.

Нидхем и Шредер предложили, чтобы числа N_A и N_B, принадлежащие широкому пространству, использовались для инициализации общего секретного ключа ("в качестве основы для последовательных блоков шифрования") [213], обеспечивающего секретную связь между Алисой и Бобом.

Как указали Деннинг и Сакко, этот протокол не гарантирует, что открытые ключи, полученные клиентами, являются новыми, а не повторениями старых, возможно, скомпрометированных ключей [94]. Эту проблему можно разрешить разными способами, например, с помощью включения меток времени в сообщения, содержащие ключи[1]. В дальнейшем будем предполагать, что открытые ключи, полученные ими от Трента, являются качественными и новыми.

---

[1] Этот способ предложен Деннингом и Сакко в работе [94]. Однако он имеет один изъян. Причину этого недостатка и способ его исправления мы рассмотрим в разделе 11.7.7.

2.6.6.3 Атака на протокол Нидхема–Шредера для аутентификации с открытым ключом

Атака на протокол Нидхема–Шредера для аутентификации с открытым ключом была изобретена Лоу (Lowe) [180].

Лоу заметил, что этот протокол можно разделить на две части, логически не связанные между собой: п. 1, 2, 4 и 5 связаны с получением открытого ключа, а п. 3, 6 и 7 относятся к аутентификации Алисы и Боба. Следовательно, можно предположить, что каждый клиент изначально имеет аутентифицированные копии открытых ключей своего визави, и сосредоточиться на следующих этапах (здесь перечисляются только сообщения, упоминаемые в протоколе 2.5).

3. Алиса — Бобу: $\{N_A, Алиса\}_{K_B}$.
6. Боб — Алисе: $\{N_A, N_B\}_{K_A}$.
7. Алиса — Бобу: $\{N_B\}_{K_B}$.

Попробуем разобраться, как Злоумышленник может вмешаться в этот протокол. Будем предполагать, что Злоумышленник является законным пользователем системы, и остальные клиенты могут проводить с ним стандартные сеансы связи. Действительно атака 2.3, описанная ниже, начинается с того, что Алиса выходит на связь со Злоумышленником.

Эта атака использует два одновременных запуска протокола: в ходе первого выполнения протокола (п. 1.3, 1.6 и 1.7) Алиса проводит корректный сеанс связи со Злоумышленником; в ходе второго выполнения (п. 2.3, 2.6 и 2.7) протокола Злоумышленник имитирует Алису и проводит фальсифицированный сеанс связи с Бобом. На этапе 1.3 Алиса начинает устанавливать обычный сеанс связи со Злоумышленником, посылая ему одноразовое случайное число N_A. На этапе 2.3 Злоумышленник выдает себя за Алису, чтобы установить фальсифицированный сеанс связи с Бобом. Для этого он посылает Бобу число N_A, полученное от Алисы. На этапе 2.6 Боб отвечает, генерируя новое одноразовое число N_B и возвращая его Алисе вместе с ее числом N_A. Злоумышленник перехватывает это сообщение, однако не может расшифровать его, так как оно зашифровано с помощью открытого ключа Алисы. Поэтому Злоумышленник на этапе 1.6 пересылает это сообщение Алисе. Обратите внимание на то, что это сообщение не вызывает у Алисы никаких подозрений, поскольку имеет ожидаемый вид. Алиса расшифровывает его, обнаруживает случайное число N_B и возвращает его Злоумышленнику на этапе 1.7 (зашифровав сообщение с помощью открытого ключа Злоумышленника). Теперь Злоумышленник может расшифровать сообщение, узнать число N_B и вернуть его Бобу в п. 2.7. На этом второе выполнение протокола завершается. Следовательно, Боб уверен, что Алиса правильно установила сеанс связи, и секретные числа N_A и N_B известны только им.

Решающий момент атаки состоит в том, что Злоумышленник может заставить Алису расшифровать число N_B, посланное Бобом. Будем говорить, что поль-

зователь выступает **оракулом** (oracle) или предоставляет **услуги оракула** (oracle service), если он невольно выполняет некую криптографическую операцию в интересах взломщика. На протяжении книги мы опишем немало примеров, в которых используются оракулы, и в итоге разработаем способ, позволяющий подписывать криптографические алгоритмы и протоколы так, чтобы они оставались стойкими, даже если их пользователи предоставляют взломщикам услуги оракулов.

Представим себе последствия этой атаки. Злоумышленник может включить общие одноразовые случайные числа в последующее сообщение, предлагая сеансовый ключ, а Боб будет уверен, что автором этого сообщения является Алиса. Если Боб — это банк, то Злоумышленник, имитируя Алису, может послать сообщение, вроде следующего.

Злоумышленник ("Алиса") — Бобу:

$\{N_A, N_B, \text{“}\textit{Переведите}\ 1\ 000\ 000\ \textit{долларов на счет Злоумышленника}\text{”}\}_{K_Б}$.

2.6.6.4 Исправление

Предотвратить описанную выше атаку довольно легко, если на этапе 6 включить в сообщение имя отвечающего.

6. Боб — Алисе: $\{\textit{Боб}, N_A, N_B\}_{K_A}$.

Тогда п. 2.6 будет выглядеть так.

2.6. Боб — Злоумышленнику ("Алисе"): $\{\textit{Боб}, N_A, N_B\}_{K_A}$.

Поскольку Алиса ожидает сообщение, содержащее имя Злоумышленника, он не сможет успешно ответить на него в п. 1.6 и заставить Алису выступить в роли оракула.

Это исправление представляет собой пример использования принципа разработки криптографических протоколов, предложенного Абади (Abadi) и Нидхемом [1]:

> *Если личность пользователя оказывает существенное влияние на смысл сообщения, следует явно указывать его имя.*

Однако следует воздержаться от заявления, что мы разработали стойкий протокол. В разделе 17.2.1 описано еще несколько проблем, связанных с нежелательными свойствами этого протокола. Будем называть это свойство "аутентификацией сообщения с помощью расшифровки и проверки". (Мы уже упоминали о нем в разделе 2.6.3.1.) Это свойство присуще всем протоколам аутентификации, использующим криптографию с секретным или открытым ключом. (Это относится и к нашему "исправлению" протокола Нидхема–Шредера для аутентификации с открытым ключом, поэтому этот вариант нельзя считать корректным.) Методически правильные исправления протоколов аутентификации Нидхема–Шредера будут описаны в разделе 17.2.3.

Уязвимость протоколов аутентификации породила целую теорию разработки корректных протоколов. Эта тема рассматривается в главе 17.

2.7 Резюме

Одни разрабатывают средства защиты, другие стремятся взломать их. В этом нет ничего необычного. Однако в данной главе мы выявили досадные недостатки протоколов аутентификации: они легко компрометируются.

Действительно, все сложные системы содержат ошибки, заложенные в них на этапе разработки. Эти системы можно разделить на две категории: враждебные и дружеские. Например, внимательный пользователь программного обеспечения, кишащего ошибками, может научиться обходить ловушки, чтобы избежать краха всей системы. Однако в системе защиты информации среда и некоторые пользователи всегда являются враждебными: единственная цель их существования — атака на систему. В такой системе ошибки проектирования создают возможности для взлома.

Мы использовали протоколы аутентификации для того, чтобы проиллюстрировать уязвимость систем безопасности. Хотя, как известно, слабость этих протоколов таится в их коммуникационной природе, мы рассмотрели их еще и потому, что они требуют применения относительно простых криптографических средств, доступных пониманию новичков. Следует помнить, что все системы защиты информации функционируют во враждебном окружении, и при их разработке следует проявлять повышенную осторожность.

Позднее мы еще вернемся к протоколам аутентификации для того, чтобы изучить их принципы и структуру, классифицировать атаки (глава 11), описать несколько протоколов, предназначенных для реальных приложений (глава 12) и разработать формальные подходы к проектированию корректных протоколов аутентификации (глава 17).

Упражнения

2.1. Какие действия может предпринимать активный взломщик?

2.2. В модели Долева–Яо Злоумышленник является очень сильным противником, поскольку он контролирует всю открытую сеть. Может ли он шифровать или дешифровать сообщения, не имея правильного ключа? Может ли он восстановить ключ шифрования ключей по зашифрованному сообщению? Может ли он предсказать одноразовое случайное число?

2.3. Какую роль играет Трент в протоколах создания аутентифицированного ключа?

2.4. Что такое долговременный ключ, ключ шифрования ключей, кратковременный ключ и сеансовый ключ?

2.5. Почему, несмотря на идеальное шифрование и идеальную аутентификацию сообщений, протоколы аутентификации остаются уязвимыми?

2.6. Что такое одноразовое случайное число? Что такое метка времени? Какую роль они играют в протоколах аутентификации и протоколах создания аутентифицированного ключа?

2.7. Почему некоторые сообщения, передаваемые по сети в рамках протокола аутентификации или протокола создания аутентифицированного ключа, должны быть новыми?

2.8. Как пользователь может определить новизну сообщения?

2.9. Используя для идеального шифрования обозначение $\{M\}_K$, дайте классификацию следующих свойств: 1) конфиденциальность сообщения; 2) секретность ключа; 3) аутентификация сообщения.

2.10. Разработайте новую атаку на протокол "Сеансовый ключ от Трента" (протокол 2.2).

2.11. Чем аутентификация сообщения отличается от аутентификации сущности?

2.12. Разработайте новую атаку на протокол аутентификации Нидхема–Шредера, в котором Алиса (и Трент) полностью автономны (offline).

2.13. Насколько важна цифровая подпись в протоколе Нидхема–Шредера для аутентификации с открытым ключом?

Подсказка: протокол можно упростить так, чтобы он содержал только этапы 2, 6 и 7.

Часть II

Математические основы

В этой части приводятся основные обозначения, методы, алгебраические операции, в также структурные части алгоритмических процедур для моделирования, выявления, анализа, преобразования и решения различных проблем, рассматриваемых в книге.

Часть состоит из четырех глав: теория вероятностей и информации (глава 3), теория вычислительной сложности (глава 4), алгебраические основы (глава 5) и теория чисел (глава 6). Она представляет собой вполне самостоятельный математический справочник. Рассматривая нетривиальные математические задачи в остальных частях книги, мы будем ссылаться на конкретные разделы этих четырех глав. Это позволит читателям активно овладевать математическими основами современной криптографии.

Мы подробно рассматриваем алгоритмы и теоремы, имеющие большое теоретическое и практическое значение. Особенно важные теоремы приводятся с доказательствами. Иногда для раскрытия темы нам приходится ссылаться на факты из других областей математики (например, линейной алгебры), которые не имеют непосредственного отношения к криптографии. В таких случаях мы упоминаем их без доказательства.

Стандартные обозначения

Ниже приводятся стандартные обозначения, использованные в книге. Определения некоторых из них будут даны непосредственно в тексте перед из первым упоминанием, другие используются без дальнейших уточнений.

Аргумент	Описание
$\emptyset$	Пустое множество
$S \cup T$	Объединение множеств S и T
$S \cap T$	Пересечение множеств S и T
$S \backslash T$	Разность множеств S и T
$S \subseteq T$	Множество S является подмножеством множества T
$\#S$	Количество элементов в множестве S (т.е., $\#\emptyset = 0$)
$x \in S$, $x \notin S$	Элемент x принадлежит (не принадлежит) множеству S
$x \in_U S$	Выборочный элемент x из генеральной совокупности, равномерно распределенной в множестве S
$x \in (a, b)$, $x \in [a, b]$	Элемент x принадлежит интервалу (a, b) (элемент x принадлежит отрезку $[a, b]$)
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$	Множества натуральных, целых, рациональных, действительных и комплексных чисел
$\mathbb{Z}_n$	Множество целых чисел по модулю n
$\mathbb{Z}_n^*$	Мультипликативная группа целых чисел по модулю n
$\mathbb{F}_q$	Конечное поле q элементов
$\mathrm{desc}\,(A)$	Характеристика алгебраической структуры A
$x \leftarrow D$	Присваивание случайного значения из генеральной совокупности, имеющего распределение D

Аргумент	Описание
$x \leftarrow_U S$	Присваивание случайного значения из генеральной совокупности, имеющего равномерное распределение в множестве S
$a(\mathrm{mod}\, b)$	Деление по модулю: вычисление остатка от деления числа a на число b
$x \mid y$, $x \nmid y$	Целое число y делится (не делится) на число x
$\stackrel{\text{def}}{=}$	По определению
$\forall$	Для всех
$\exists$	Существует
$\gcd(x, y)$	Наибольший общий делитель чисел x и y
$\mathrm{lcm}(x, y)$	Наименьший общий множитель чисел x и y
$\log_b x$	Логарифм числа x по основанию b. Если основание не указано, логарифм считается натуральным
$\lfloor x \rfloor$	Максимальное целое число, не превосходящее число x
$\lceil x \rceil$	Минимальное целое число, большее или равное числу x
$\lvert x \rvert$	Длина целого числа x, равная $1 + \lfloor \log_2 x \rfloor$ для $x \geqslant 1$, или модуль числа x
$\phi(n)$	Функция Эйлера от числа n
$\lambda(n)$	Функция Кармайкла от числа n
$\mathrm{ord}(x)$	Порядок элемента группы
$\mathrm{ord}_n(x)$	Порядок числа $x(\mathrm{mod}\, n)$
$\langle g \rangle$	Циклическая группа, порожденная элементом g
$\left(\frac{x}{y}\right)$	Символ Лежандра–Якоби для целого числа x по модулю целого числа y
$J_n(1)$	$\left\{x \mid x \in \mathbb{Z}_n^*, \left(\frac{x}{n}\right) = 1\right\}$
QR_n	Множество квадратичных вычетов по модулю n
QNR_n	Множество квадратичных невычетов по модулю n
$\deg(P)$	Степень полинома P
$\sum_{i=1}^{n} v_i$, $\sum_{i \in S} v_i$	Сумма величин v_i, для $i = 1, 2, \ldots, n$ или для $i \in S$
$\prod_{i=1}^{n} v_i$, $\prod_{i \in S} v_i$	Произведение величин v_i, для $i = 1, 2, \ldots, n$ или для $i \in S$

Аргумент	Описание
$\bar{E}$	Событие, дополнительное по отношению к событию E
$E \cup F$	Сумма событий E и F, т.е. происходит либо событие E, либо событие F
$E \cap F$	Произведение событий E и F, т.е. происходят оба события E и F
$E \subseteq F$	Событие F содержит событие E, т.е. если происходит событие E, то происходит и событие F
$E \backslash F$	Разность событий E и F, равная $E \cap \bar{F}$
$\bigcup_{i=1}^{n} E_i$, $\bigcup_{i \in S} E_i$	Сумма событий E_i, для $i = 1, 2, \ldots, n$ или для $i \in S$
$\bigcap_{i=1}^{n} E_i$, $\bigcap_{i \in S} E_i$	Произведение событий E_i, для $i = 1, 2, \ldots, n$ или для $i \in S$
$\mathrm{Prob}[E]$	Вероятность события E
$\mathrm{Prob}[E\|F]$	Условная вероятность события E, т.е. вероятность того, что событие E произойдет, если произошло событие F
$n!$	Факториал числа n, равный $n(n-1)(n-2)\ldots 1$, причем $0! = 1$
$\binom{n}{k}$	Число сочетаний из n элементов по k, равное $\frac{n!}{k!\,(n-k)!}$
$b(k; n, p)$	Биномиальное распределение k успехов среди n испытаний в схеме Бернулли, если вероятность успеха равна p
$O(f(n))$	Функция $g(n)$, такая что $\|g(n)\| \leqslant c\|f(n)\|$, где $c > 0$ — некая константа, для всех достаточно больших чисел n
$O_B()$	$O()$ в поразрядных вычислениях
$\neg x$	Логическая операция НЕ (x — булева переменная) либо побитовая операция: поразрядное отрицание (x — битовая строка)
$x \wedge y$	Логическая операция И (x, y — булевы переменные) либо побитовая операция: поразрядное И (x, y — битовые строки)
$x \vee y$	Логическая операция ИЛИ (x, y — булевы переменные) либо побитовая операция: поразрядное ИЛИ (x, y — битовые строки)

Аргумент	Описание
$x \oplus y$	Логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ (x, y — булевы переменные) либо побитовая операция: поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ (x, y — битовые строки)
$(* \cdots *)$	Комментарии в алгоритмах или протоколах
$\square$	Конец доказательства, замечания или примера

Глава 3

Теория вероятностей и теория информации

3.1 Введение

Теория вероятностей и теория информации образуют фундамент современных криптографических методов.

Вероятность — основное средство анализа безопасности. Часто возникает необходимость оценить, *насколько вероятно* некое опасное событие при определенных условиях. Например, рассматривая протокол "Подбрасывание монеты по телефону", описанный в главе 1, мы должны оценить вероятность того, что Алиса найдет коллизию однонаправленной функции f (желательно, чтобы эта вероятность была как можно меньше), а также вероятность того, что Боб распознает четность числа x при заданной функции $f(x)$ (желательно, чтобы эта вероятность была как можно ближе к числу $\frac{1}{2}$).

Теория информации тесно связана с теорией вероятностей. Важным аспектом безопасности алгоритмов шифрования является "неопределенность шифров": алгоритм должен создавать текст, имеющий случайное распределение по всему пространству зашифрованных текстов. Шеннон (Shannon) дал количественную оценку неопределенности информации, введя понятие энтропии. С исторической точки зрения стремление достичь высокой энтропии шифров происходит от потребности максимально затруднить расшифровку, которая использует избыточность естественных языков и частое употребление распространенных устойчивых выражений.

В последнее время потребность обеспечить вероятностное поведение современных криптографических систем, в частности, криптосистем с открытым ключом, приняла еще более строгую форму — возникла необходимость обеспечить семантическую стойкость (semantic security). Это свойство можно описать следующим образом. Допустим, Алиса шифрует нуль и единицу с одинаковой вероятностью, используя семантически безопасный алгоритм шифрования, посылает зашифрованный текст Бобу и просит его ответить, чему равно зашифрованное число. Если Боб не имеет правильного ключа расшифровки, то не должен суще-

ствовать алгоритм, который позволил бы ему выбрать один из двух возможных вариантов с вероятностью, выше, чем вероятность простого угадывания. Следует отметить, что многие учебные алгоритмы шифрования не обладают этим свойством.

3.1.1 Структурная схема главы

Основные понятия теории вероятностей, достаточные для понимания всей книги, изложены в разделах 3.2–3.6. Теория информации излагается в разделах 3.7–3.8.

3.2 Основные понятия теории вероятностей

Пусть $\mathbb{S}$ — произвольное, но фиксированное множество точек, называемое **полем вероятностей** (probability space) или **выборочным пространством** (sample space). Любая точка $x \in \mathbb{S}$ называется **выборочным элементом** (sample point) или **исходом** (outcome), **простым событием** (simple event), а также **неразложимым событием** (indecomposable event). (Для краткости мы будем называть этот элемент просто **точкой** (point).) **Событие** (составное, или разложимое) является подмножеством множества $\mathbb{S}$ и обычно обозначается прописной буквой (например, E). **Эксперимент** (experiment), или наблюдение, представляет собой извлечение точки из множества $\mathbb{S}$. Событие E происходит, если в результате эксперимента выясняется, что некоторая точка x из $\mathbb{S}$ принадлежит множеству E.

Пример 3.1. Рассмотрим эксперимент, в ходе которого из "идеальной" колоды извлекается игральная карта (термин "идеальная" означает, что карта извлекается случайным образом). Перечислим некоторые примеры поля вероятностей, точек и событий.

1. $\mathbb{S}_1$: пространство состоит из 52 точек — по одной на каждую карту. Допустим, что событие E_1 обозначает "туз" (т.е. $E_1 = \{T\spadesuit, T\heartsuit, T\diamondsuit, T\clubsuit\}$). Оно происходит, если из колоды извлекается туз любой масти.
2. $\mathbb{S}_2 = \{\text{красная масть, черная масть}\}$. Допустим, что $E_2 = \{\text{красная масть}\}$. Это событие происходит, если из колоды извлекается карта красной масти.
3. $\mathbb{S}_3$: пространство состоит из 13 точек — 2, 3, 4, ..., 10, В, Д, К, Т. Допустим, что $E_3 = \{\text{числа}\}$. Это событие происходит, если из колоды извлекается карта 2, или 3, ..., или 10. □

Определение 3.1 (Классическое определение вероятности). *Допустим, что в ходе эксперимента извлекается одна из $n = \#\mathbb{S}$ равновероятных точек и что*

в результате каждого эксперимента обязательно извлекается одна точка. Обозначим через m количество точек, принадлежащих событию E. Тогда вероятностью события E называется число $\frac{m}{n}$. Эта вероятность обозначается следующим образом:

$$\text{Prob}[E] = \frac{m}{n}.$$

Пример 3.2. В примере 3.1 вероятности таковы.

1. $\text{Prob}[E_1] = \frac{4}{52} = \frac{1}{13}$.
2. $\text{Prob}[E_2] = \frac{1}{2}$.
3. $\text{Prob}[E_3] = \frac{9}{13}$. □

Определение 3.2 (Статистическое определение вероятности). *Допустим, что при одинаковых условиях проводятся n экспериментов, в которых μ раз происходит событие. Если при достаточно больших значениях n величина $\frac{\mu}{n}$ становится и остается устойчивой, то говорят, что событие E происходит с вероятностью*

$$\text{Prob}[E] \approx \frac{\mu}{n}.$$

В разделе 3.5.3 мы увидим, что при некоторых интуитивных предположениях определение 3.2 можно вывести из теоремы, которая является следствием закона больших чисел.

3.3 Свойства

1. Поле вероятностей само по себе является **достоверным событием** (sure event). Например, $\mathbb{S} = \{\text{ОРЕЛ}, \text{РЕШКА}\}$. Тогда

$$\text{Prob}[\mathbb{S}] = 1.$$

2. Обозначим через $\mathbb{O}$ событие, не содержащее ни одной точки (т.е. событие, которое никогда не происходит, например, черные бубны). Это событие называется **невозможным** (impossible event). Вероятность невозможного события равна нулю.

$$\text{Prob}[\mathbb{O}] = 0.$$

3. Вероятность любого события удовлетворяет неравенству

$$0 \leqslant \text{Prob}[\mathbb{O}] \leqslant 1.$$

4. Если $E \subseteq F$, то говорят, что событие E влечет событие F, и

$$\text{Prob}[E] \leqslant \text{Prob}[F].$$

5. Обозначим через $\bar{E} = \mathbb{S} \backslash E$ событие, **дополнительное** (complementary) по отношению к событию E. Тогда

$$\text{Prob}[E] + \text{Prob}[\bar{E}] = 1.$$

3.4 Основные вычисления

Обозначим через $E \cup F$ сумму событий E и F (происходит одно из двух событий), а через $E \cap F$ — произведение событий E и F (происходят оба события).

3.4.1 Правила сложения

1. $\text{Prob}[E \cup F] = \text{Prob}[E] + \text{Prob}[F] - \text{Prob}[E \cap F]$.
2. Если $E \cap F = \mathbb{O}$, говорят, что события E и F являются взаимно исключающими, или несовместными, и

$$\text{Prob}[E \cup F] = \text{Prob}[E] + \text{Prob}[F].$$

3. Если $\bigcup_{i=1}^{n} E_i = \mathbb{S}$ и $E_i \cap E_j = \mathbb{O}\,(i \neq j)$, то

$$\sum_{i=1}^{n} \text{Prob}\,[E_i] = 1.$$

Пример 3.3. Покажем, что

$$\text{Prob}[E \cup F] = \text{Prob}[E] + \text{Prob}[F \cap \bar{E}]. \tag{3.4.1}$$

Поскольку $E \cup F = E \cup \left(F \cap \bar{E}\right)$, где события E и $F \cap \bar{E}$ являются взаимно исключающими, равенство (3.4.1) является следствием правила 2. □

Определение 3.3 (Условная вероятность). *Пусть E и F — два события, причем событие E имеет ненулевую вероятность. Вероятность события F при условии, что произошло событие E, называется условной вероятностью события F при условии события E и вычисляется по формуле*

$$\text{Prob}\,[F|E] = \frac{\text{Prob}\,[E \cap F]}{\text{Prob}\,[E]}.$$

Пример 3.4. Рассмотрим семьи с двумя детьми. Обозначим буквами g (girl) и b (boy) пол ребенка (девочка и мальчик соответственно), причем первая буква обозначает пол старшего ребенка. Существует четыре возможности gg, gb, bg и bb. Эти точки образуют поле вероятностей $\mathbb{S}$. Вероятность извлечь каждую из этих точек из поля вероятностей равна $\frac{1}{4}$. Обозначим через E событие "в семье есть девочка", а через F — событие "в семье есть две девочки". Чему равна вероятность события F при условии, что произошло событие E (т.е. $\mathrm{Prob}[F|E]$)?

Событие $E \cap F$ представляет собой точку gg, поэтому $\mathrm{Prob}[E \cap F] = \frac{1}{4}$. Поскольку событие E состоит из точек gg, gb или bg, $\mathrm{Prob}[E \cap F] = \frac{3}{4}$. Следовательно, по определению 3.3 $\mathrm{Prob}[F|E] = \frac{1}{3}$. Действительно, в одной трети случаев в семьях, имеющих двух детей, одна из которых — девочка, оба ребенка являются девочками. □

Определение 3.4 (Независимые события). *События E и F называются независимыми, тогда и только тогда, когда*

$$\mathrm{Prob}[F|E] = \mathrm{Prob}[F].$$

3.4.2 Правила умножения

1. $\mathrm{Prob}[E \cap F] = \mathrm{Prob}[F|E] \times \mathrm{Prob}[E] = \mathrm{Prob}[E|F] \times \mathrm{Prob}[F]$.
2. Если события E и F являются независимыми, то

$$\mathrm{Prob}[E \cap F] = \mathrm{Prob}[E] \times \mathrm{Prob}[F].$$

Пример 3.5. Вернемся к примеру 3.1. Предположим, что события E_1 и E_2 являются независимыми. Их вероятности равны $\frac{1}{13}$ и $\frac{1}{2}$ соответственно (пример 3.2). Поскольку эти события независимы, применяя второе правило умножения, получаем, что вероятность их одновременной реализации (из колоды извлекается туз красной масти) равна $\frac{1}{26}$. □

3.4.3 Закон полной вероятности

Закон полной вероятности (law of total probability) выражается следующей теоремой.

Теорема 3.1. *Если* $\bigcup_{i=1}^{n} E_i = \mathbb{S}$ *и* $E_i \cap E_j = \mathbb{O}\ (i \neq j)$, *то для любого события* E

$$\mathrm{Prob}\,[A] = \sum_{i=1}^{n} \mathrm{Prob}\,[A \mid E_i]\,\mathrm{Prob}\,[E_i].$$

Доказательство. Поскольку

$$A = A \cap \mathbb{S} = \bigcup_{i=1}^{n} (A \cap E_i),$$

где $A \cap E_i$ и $A \cap E_j$ $(i \neq j)$ являются взаимно исключающими, вероятность суммы событий в правой части равенства можно вычислить по второму правилу сложения вероятностей, причем вероятность каждого слагаемого вычисляется по первому правилу умножения. □

Закон полной вероятности весьма полезен. Мы будем часто применять его при вычислении (или оценке) условной вероятности события A при взаимоисключающих событиях (как правило, при событиях E и $\bar{E}$). Полезность этой формулы объясняется тем, что часто легче оценить условные вероятности $\mathrm{Prob}\,[A|E_i]$, чем непосредственно вычислять вероятность $\mathrm{Prob}[A]$.

Пример 3.6. (В этом примере используются некоторые элементарные факты из теории чисел. Читатель, которому этот пример покажется слишком сложным, может вернуться к нему после изучения главы 6.)

Пусть $p = 2q + 1$, причем числа p и q являются простыми. Рассмотрим простой случайный выбор двух чисел g и h из множества $S = \{1, 2, \ldots, p-1\}$ (с возвращением). Пусть событие A заключается в том, что "число h порождается числом g", иначе говоря, $h \equiv g^x (\mathrm{mod}\, p)$ для некоторого числа $x < p$ (т.е. существует число $\log_g h (\mathrm{mod}(p-1))$). Чему равна вероятность события A для случайных чисел g и h?

Можно непосредственно оценить вероятность $\mathrm{Prob}[A]$. Однако намного легче сначала оценить несколько условных вероятностей, а затем применить теорему о полной вероятности.

Обозначим через $\mathrm{ord}_p(g)$ мультипликативный порядок числа $g (\mathrm{mod}\, p)$, т.е. наименьшее натуральное число i, такое что $g^i \equiv 1 (\mathrm{mod}\, p)$. Величина $\mathrm{Prob}[A]$ зависит от вероятностей четырех взаимоисключающих событий.

1. $E_1 : \mathrm{ord}_p(g) = p-1 = 2q$. Кроме того, нам известна вероятность $\mathrm{Prob}[E_1] = \frac{\phi(2q)}{p-1} = \frac{q-1}{p-1}$ (здесь ϕ — функция Эйлера, а в множестве S существует точно $\phi(2q) = q-1$ элементов порядка $2q$). В данном случае любое число h, удовлетворяющее условию $h < p$, должно порождаться числом g (число g является порождающим, или образующим элементом множества S). Следовательно, $\mathrm{Prob}[A|E_1] = 1$.
2. $E_2 : \mathrm{ord}_p(g) = q$ и (аналогично варианту 1) известна вероятность $\mathrm{Prob}[E_2] = \frac{q-1}{p-1}$. В данном случае число h может порождаться числом g тогда и только тогда, когда $\mathrm{ord}_p(h) | q$. Поскольку в множестве S существует ровно q элементов, порядки которых являются делителем числа q, получаем $\mathrm{Prob}[A|E_2] = \frac{q}{p-1} = \frac{1}{2}$.

3. E_3 : $\mathrm{ord}_p(g) = 2$. Поскольку существует только один элемент порядка 2, а именно, число $p-1$, получаем $\mathrm{Prob}[E_3] = \frac{1}{p-1}$. Число $p-1$ может порождать только числа 1 и $p-1$, поэтому $\mathrm{Prob}[A|E_3] = \frac{2}{p-1}$.
4. E_4 : $\mathrm{ord}_p(g) = 1$. Только число 1 имеет порядок 1, поэтому $\mathrm{Prob}[E_4] = \frac{1}{p-1}$. Кроме того, число 1 может порождать только число 1, поэтому $\mathrm{Prob}[A|E_4] = \frac{1}{p-1}$.

События, перечисленные выше, не являются взаимоисключающими, однако они исчерпывают все варианты порядка, который может иметь число g. Следовательно, мы можем применить теорему о полной вероятности и вычислить вероятность $\mathrm{Prob}[A]$:

$$\mathrm{Prob}[A] = \frac{q-1}{p-1} + \frac{q-1}{2(p-1)} + \frac{2}{(p-1)^2} + \frac{1}{(p-1)^2}.$$

□

3.5 Случайные величины и распределения вероятностей

В криптографии, как правило, изучаются функции, определенные в дискретном пространстве (например, в качестве пространств криптографических ключей используются интервалы целых чисел либо конечные алгебраические структуры, такие как конечные группы или поля). Допустим, что дискретное пространство S содержит конечное или счетное количество изолированных точек $x_1, x_2, \ldots, x_{\#S}$. Рассмотрим общий случай, когда пространство S содержит счетное количество точек, т.е. $\#S = \infty$. Это позволит провести анализ асимптотической вычислительной сложности наших алгоритмов и протоколов (см. раздел 4.6).

Определение 3.5 (Дискретная случайная величина и ее функция распределения).

1. *Дискретная случайная величина является числовым результатом эксперимента. Она представляет собой функцию, определенную на дискретном выборочном пространстве.*
2. *Пусть $\mathbb{S}$ — дискретное пространство вероятностей, а ξ — случайная величина. Функция распределения дискретной случайной величины ξ представляет собой отображение $\mathbb{S} \mapsto \mathbb{R}$, заданное перечислением вероятностей*

$$\mathrm{Prob}[\xi = x_i] = p_i (i = 1, 2, \ldots, \#\mathbb{S}),$$

удовлетворяющих следующим условиям:

а) $p_i \geqslant 0$;

б) $\sum_{i=1}^{\#\mathbb{S}} p_i = 1.$

Рассмотрим теперь два дискретных распределения, которые часто используются в криптографии. В дальнейшем мы будем пропускать слово "дискретный", поскольку все изучаемые нами распределения являются дискретными.

3.5.1 Равномерное распределение

Наиболее часто в криптографии применяются случайные величины, имеющие **равномерное распределение** (uniform distribution):

$$\text{Prob}[\xi = x_i] = \frac{1}{\#\mathbb{S}} (i = 1, 2, \ldots, \#\mathbb{S}).$$

Пример 3.7. Пусть S — множество неотрицательных чисел, состоящих из не более чем k бит (бинарных цифр). Выберем из множества S случайную точку x, придерживаясь равномерного распределения. Покажем, что вероятность извлечь число, состоящее из k бит, равна $\frac{1}{2}$.

Множество $S = \{0, 1, 2, \ldots, 2^k - 1\}$ можно разбить на два непересекающихся подмножества: $S_1 = \{0, 1, 2, \ldots, 2k - 1 - 1\}$ и $S_2 = \{2^{k-1}, 2^{k-1} + 1, \ldots, 2^k - 1\}$, где множество S_2 состоит из всех k-разрядных чисел, $\#S_1 = \#S_2 = \frac{\#S}{2}$. Применяя второе правило сложения вероятностей, получаем следующее.

$$\begin{aligned}
\text{Prob}[x \in S_2] &= \text{Prob} \bigcup_{i=2^{k-1}}^{2^k-1} \{x = i\} = \\
&= \sum_{i=2^{k-1}}^{2^k-1} \text{Prob}[x = i] = \\
&= \sum_{i=2^{k-1}}^{2^k-1} \frac{1}{\#S} = \\
&= \frac{\#S_2}{\#S} = \\
&= \frac{1}{2}.
\end{aligned}$$

□

Выражение "выберем из множества S случайную точку x, придерживаясь равномерного распределения" часто встречается в криптографии. Поскольку оно слишком многословно, мы будем использовать более короткое выражение "выберем из S равномерно распределенную величину p" или просто обозначение $p \in_U S$.

3.5.2 Биномиальное распределение

Допустим, что эксперимент имеет только два исхода — успех или неудача (например, "ОРЕЛ" или "РЕШКА" при подбрасывании монеты), причем их вероятности постоянны. Повторяющиеся независимые эксперименты, удовлетворяющие этим условиям, называются **испытаниями Бернулли** (Bernoulli trials). Предположим, что в отдельном испытании

$$\text{Prob}[\text{"успех"}] = p, \text{Prob}[\text{"неудача"}] = 1 - p.$$

Тогда

$$\text{Prob}[\text{"}k\text{ успехов в }n\text{ испытаниях"}] = \binom{n}{k} p^k (1-p)^{n-k}, \qquad (3.5.1)$$

где $\binom{n}{k}$ обозначает число сочетаний из n элементов по k.

Докажем справедливость равенства (3.5.1). Во-первых, количество вариантов, в которых может произойти событие "среди n испытаний зафиксировано k "успехов" и $n-k$ "неудач", равно числу сочетаний из n элементов по k, т.е. $\binom{n}{k}$. Во-вторых, каждой точке поля вероятностей соответствует k "успехов" и $n-k$ "неудач", поэтому вероятность извлечь такую точку равна $p^k(1-p)^{n-k}$.

Если случайная величина ξ_n принимает значения $0, 1, \ldots, n$, и для величины $p \in (0,1)$ выполняется условие

$$\text{Prob}[\xi_n = k] = \binom{n}{k} p^k (1-p)^{n-k} \, (k = 0, 1, \ldots, n),$$

то говорят, что величина ξ_n имеет **биномиальное распределение** (binomial distribution). Сравнивая это выражение с формулой (3.5.1), приходим к выводу, что испытания Бернулли подчиняются биномиальному распределению. Обозначим через $b(k; n, p)$ **биномиальный член** (binomial term), соответствующий значениям $k = 0, 1, \ldots, n$ и $0 < p < 1$.

Пример 3.8. Рассмотрим следующие задачи.

1. Предположим, что в ходе эксперимента идеальная монета подбрасывается 10 раз. Чему равна вероятность того, что орел не выпадет ни разу или орел выпадет сколько-то раз (один, два, ... или 10)?
2. Чему равна вероятность события "орел выпадет пять раз"?
3. Чему равна вероятность события "орел выпадет не более пяти раз"?

Поскольку событие, описанное в задаче 1, происходит всегда, его вероятность равна единице. Действительно, применяя второе правило сложения вероятностей,

имеем следующее.

$$\text{Prob}\bigcup_{i=1}^{10}\{\text{“Орел выпадает } i \text{ раз”}\} = \sum_{i=0}^{10}\text{Prob}\left[\text{“Орел выпадает } i \text{ раз”}\right] =$$

$$= \sum_{i=0}^{10}\binom{10}{i}\left(\frac{1}{2}\right)^{i}\left(\frac{1}{2}\right)^{10-i} =$$

$$= (1+1)^{10}\left(\frac{1}{2}\right)^{10} =$$

$$= 1.$$

Решая задачу 2, получаем такой результат.

$$\text{Prob}[\text{“Орел выпадает пять раз из 10”}] = \binom{10}{5}\left(\frac{1}{2}\right)^{10} = \frac{252}{1024} \approx 0{,}246.$$

При решении задачи 3 мы должны просуммировать вероятности всех событий, в которых орел выпадает не более пяти раз.

$$\text{Prob}\bigcup_{i=1}^{5}\{\text{“Орел выпадает } i \text{ раз из 10”}\} = \left(\frac{1}{2}\right)^{10}\sum_{i=0}^{5}\binom{10}{i} \approx 0{,}623.$$

□

На рис. 3.1 изображен график биномиального распределения для $p = 0{,}5$ и $n = 10$, использованного при анализе примера 3.8.

Следует обратить особое внимание на разницу между задачами 3.8.2 и 3.8.3. Вероятность события, указанного в задаче 3.8.2, равна площади среднего прямоугольника, изображенного на рис. 3.1, а вероятность события, описанного в задаче 3.8.3, равна площади первых шести прямоугольников.

В приложениях биномиальных распределений (см. разделы 4.4.1, 4.4.5.1 и 18.5.1) вероятность обнаружить ровно r “успехов” (как в примере 3.8.2) вызывает меньше интереса, чем вероятность обнаружить не менее r “успехов” (как в примере 3.8.3). Более того, сумма нескольких членов может быть более значимой, чем суммы других членов. Перейдем теперь к исследованию “значимых” и “незначимых” сумм в биномиальном распределении.

3.5.2.1 Центральный член и хвосты

Рассматривая последовательные биномиальные члены, приходим к следующей формуле.

$$\frac{b\left(k;n,p\right)}{b(k-1;n,p)} = \frac{(n-k+1)p}{k(1-p)} = 1 + \frac{(n+1)p-k}{k(1-p)}. \tag{3.5.2}$$

Второе слагаемое в правой части является положительным при $k < (n+1)p$ и отрицательным при $k > (n+1)p$. Итак, дробь (3.5.2) больше единицы при

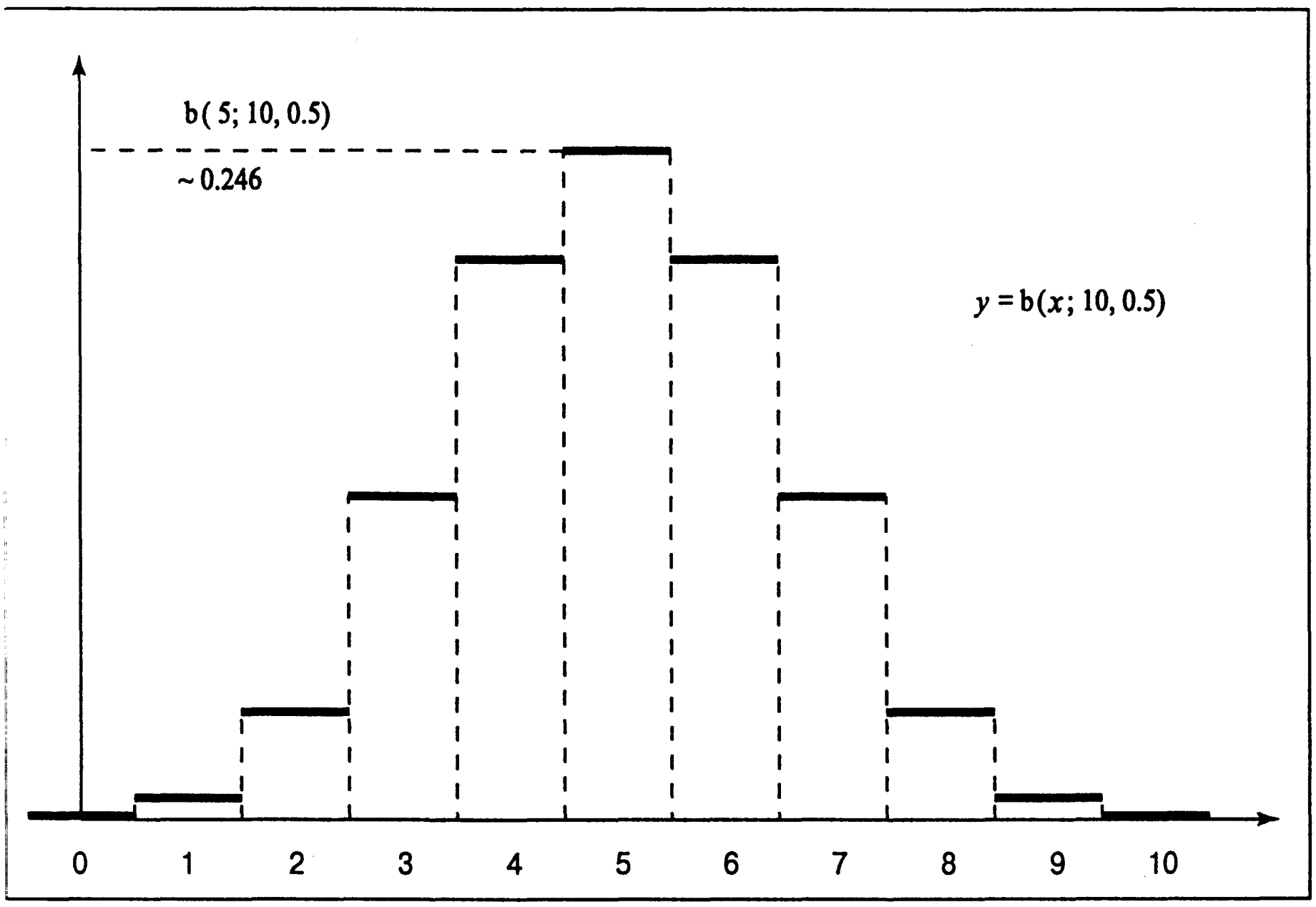


Рис. 3.1. Биномиальное распределение

$< (n+1)p$ и меньше единицы при $k > (n+1)p$. Следовательно, с увеличением числа k величина $b(k;n,p)$ возрастает, если $k < (n+1)p$, и уменьшается, если $k > (n+1)p$. Значит, биномиальный член $b(k;n,p)$ достигает максимума в точке $k = \lfloor (n+1)p \rfloor$. Биномиальный член

$$b(\lfloor (n+1)p \rfloor; n, p) \tag{3.5.3}$$

называется **центральным** (central term). Поскольку точка $\lfloor (n+1)p \rfloor$ является точкой максимума биномиального члена, она представляет собой наиболее вероятное количество успехов. Обратите внимание на то, что, если число $(n+1)p$ является целым, дробь (3.5.2) равна единице, и, следовательно, в данном случае существуют два центральных биномиальных члена $b((n+1)p-1;n,p)$ и $b((n+1)p;n,p)$.

Допустим, что $r > (n+1)p$, т.е. точка r расположена правее точки, соответствующей наиболее вероятному количеству успехов. Как известно, величина $b(k;n,p)$ убывает при всех $k \geqslant r$. Можно оценить скорость убывания этой величины, заменив число k числом r в правой части равенства (3.5.2). Это приводит нас к следующему неравенству.

$$b(k;n,p) < b(k-1;n,p)s, \text{ где } s = \frac{(n+1-r)p}{r(1-p)} < 1. \tag{3.5.4}$$

В частности,

$$b(k;n,p) < b(r;n,p)s.$$

Обратите внимание на то, что оценка (3.5.4) выполняется для всех чисел $k = r + 1, r + 2, \ldots, n$. Следовательно,

$$b(r+i;n,p) < b(r;n,p)s^i \text{ для } i = 1,2,\ldots \tag{3.5.5}$$

Допустим теперь, что $r > np$, и вычислим верхнюю границу вероятности обнаружить не менее r успехов.

$$\text{Prob}[\xi_n \geqslant r] = \sum_{k=r}^{n} b(k;n,p) = \sum_{i=0}^{n-r} b(r+i;n,p). \tag{3.5.6}$$

Из (3.5.5) следует, что

$$\text{Prob}[\xi_n \geqslant r] < b(r;n,p)\sum_{i=0}^{n-r} s^i < b(r;n,p)\sum_{i=0}^{\infty} s^i = b(r;n,p)\frac{1}{1-s}.$$

Заменяя параметр s выражением $\frac{(n+1-r)p}{r(1-p)}$, получаем

$$\text{Prob}[\xi_n \geqslant r] < b(r;n,p)\frac{r(1-p)}{r-(n+1)p}.$$

Учитывая, что между центральным биномиальным членом и величиной $b(r;n,p)$ расположены только $r-(n+1)p$ биномиальных членов, превышающих число $b(r;n,p)$, причем их сумма не превышает единицы, получаем неравенство

$$b(r;n,p) < (r-(n+1)p)^{-1}.$$

В итоге имеем следующую оценку.

$$\text{Prob}[\xi_n \geqslant r] \leqslant \frac{r(1-p)}{(r-(n+1)p)^2} \text{ для } r > (n+1)p. \tag{3.5.7}$$

Левая часть неравенства (3.5.7) называется **правым хвостом** (right tail) биномиального распределения. Легко убедиться, что если величина r *немного* отличается от центральной точки $(n+1)p$, то знаменатель дроби в соотношении (3.5.7) не равен нулю, и весь правый хвост ограничен величиной порядка $(np)^{-1}$. Следовательно, правый хвост является малой величиной и стремится к нулю при увеличении числа n.

Аналогично можно вывести оценку для **левого хвоста** (left tail):

$$\text{Prob}[\xi_n \leqslant r] \leqslant \frac{(n+1-r)p}{((n+1)p-r)^2} \text{ для } r < (n+1)p. \tag{3.5.8}$$

Читатель может самостоятельно выполнить это задание (упражнение 3.7).

На первый взгляд, из неравенств (3.5.7) и (3.5.8) следует, что два хвоста биномиального распределения ограничены величинами, имеющими порядок $\frac{1}{n}$. Однако следует отметить, что из неравенств (3.5.7) и (3.5.8) можно получить лишь верхние оценки. На самом деле хвост стремится к нулю намного быстрее, чем $\frac{1}{n}$. Этот факт иллюстрируется следующим примером (см. также доказательство стойкости и полноты протокола 18.4 в разделе 18.5.1.1).

Пример 3.9. Пусть $p = 0{,}5$. Вычислим левые хвосты биномиального распределения, ограниченные точкой $r = n(p - 0{,}01)$, для разных чисел n.

1. Для $n = 1\,000$ соответствующий левый хвост равен

$$\text{Prob}[\xi < 490] \approx 0{,}25333.$$

2. Для $n = 10\,000$ соответствующий левый хвост равен

$$\text{Prob}[\xi < 4\,900] \approx 0{,}02221.$$

3. Для $n = 100\,000$ соответствующий левый хвост равен

$$\text{Prob}[\xi < 49\,000] \approx 1{,}24241 \times 10^{-10}.$$

Сравнивая эти результаты, легко убедиться, что хвост уменьшается до нуля гораздо быстрее, чем $\frac{1}{n}$.

Поскольку $p = 0{,}5$, распределение плотности вероятности является симметричным (см. рис. 3.1). У симметричного распределения правый и левый хвосты совпадают, если они содержат одинаковое количество членов. Следовательно, в задаче 3.9.3 сумма двух хвостов, состоящих из 98 000 членов (т.е. 98% всех членов), практически равна нулю, а сумма членов, соответствующих наиболее вероятному количеству успехов (т.е. 2% всех членов в окрестности центрального члена), практически равна единице. □

3.5.3 Закон больших чисел

Напомним определение 3.2. Оно утверждает, что если в результате n одинаковых испытаний событие E происходит μ раз, причем эта частота является устойчивой, а число n достаточно велико, то величина $\frac{\mu}{n}$ называется вероятностью события E.

Допустим, что в схеме испытаний Бернулли вероятность успеха равна p, а случайная величина ξ_n представляет собой количество "успехов" среди n испытаний. Тогда $\frac{\xi_n}{n}$ равна среднему количеству "успехов" среди n испытаний. В соответствии с определением 3.2 величина $\frac{\xi_n}{n}$ должна быть близкой к числу p.

Рассмотрим, например, вероятность того, что величина $\frac{\xi_n}{n}$ превосходит число $p+\alpha$ для любого $\alpha > 0$ (т.е. число α является произвольно малым, но фиксированным). Очевидно, что эта вероятность равна следующей величине.

$$\text{Prob}[\xi_n > n(p+\alpha)] = \sum_{i=n(p+\alpha)+1}^{n} b(i;n,p).$$

С учетом оценки (3.5.7) получаем, что

$$\text{Prob}[\xi_n > n(p+\alpha)] < \frac{1}{n\alpha}. \tag{3.5.9}$$

Таким образом,

$$\text{Prob}[\xi_n > n(p+\alpha)] \to 0 (n \to \infty). \tag{3.5.10}$$

Аналогично можно доказать, что

$$\text{Prob}[\xi_n < n(p-\alpha)] \to 0 (n \to \infty).$$

Отсюда следует **закон больших чисел** (law of large numbers):

$$\lim_{n\to\infty} \text{Prob}\left[\left|\frac{\xi_n}{n} - p\right| < \alpha\right] = 1.$$

Эта форма закона больших чисел называется также **теоремой Бернулли** (Bernoulli's theorem). Теперь совершенно ясно, что определение 3.2 является следствием закона больших чисел. Однако мы сформулировали его в виде определения, поскольку этот закон имеет интуитивный характер.

3.6 Парадокс дней рождений

Рассмотрим следующую задачу для произвольной функции $f : X \mapsto Y$, где Y — множество, состоящее из n элементов.

> Найти величину k для оценки вероятности ε (т.е. $0 < \varepsilon < 1$), такую что для k попарно разных элементов $x_1, x_2, \ldots, x_k \in_U X$ набор, состоящий из k значений функции $f(x_1), f(x_2), \ldots, f(x_k)$, удовлетворяет неравенство
>
> $$\text{Prob}[f(x_i) = f(x_j)] \geqslant \varepsilon \text{ для некоторых } i \neq j.$$
>
> Иначе говоря, при k вычислениях функции коллизия возникает с вероятностью не меньше ε.

В этой задаче требуется найти число k, которое удовлетворяет некоторой оценке вероятности снизу для *любой* функции. Предполагается лишь, что функция обладает так называемым свойством случайности: такие функции отображают равномерно распределенные числа из множества X в равномерно распределенные числа из множества Y. Очевидно, что только такая функция позволяет увеличить число k при заданной оценке вероятности так, чтобы эти условия выполнялись для *любой* функции при той же оценке вероятности. Следовательно, необходимо, чтобы $\#X > \#Y$. В противном случае могут существовать функции, у которых коллизии вообще не возникают.

Итак, можно предположить, что вычисление функции в этой задаче порождает n разных и равновероятных точек. Эти вычисления можно отождествить с извлечением шара из урны, содержащей n разноцветных шаров, после чего цвет запи-

сывается, и шар возвращается в урну. Следовательно, задача заключается в поиске такого числа k, при котором какой-нибудь цвет повторялся бы с вероятностью ε.

На цвет первого шара не налагаются никакие ограничения. Пусть y_i — цвет шара, извлеченного при i-й попытке. Цвет второго шара не должен совпадать с цветом первого шара, поэтому вероятность того, что $y_2 \neq y_1$, равна $1 - \frac{1}{n}$, вероятность того, что $y_3 \neq y_1$ и $y_3 \neq y_2$, равна $1 - \frac{2}{n}$ и т.д. При извлечении k-го шара вероятность того, что до этого не возникнет коллизия, равна

$$\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{k-1}{n}\right).$$

При достаточно большом числе n и относительно малом числе x выполняется следующее соотношение.

$$\left(1 + \frac{x}{n}\right)^n \approx e^x,$$

или

$$1 + \frac{x}{n} \approx e^{x/n}.$$

Итак,

$$\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{k-1}{n}\right) = \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right) \approx \prod_{i=1}^{k-1} e^{-i/n} = e^{-\frac{k(k-1)}{2n}}.$$

Полученное число представляет собой вероятность извлечь k шаров без коллизии. Следовательно, вероятность возникновения по крайней мере одной коллизии равна

$$1 - e^{-\frac{k(k-1)}{2n}}.$$

Приравнивая эту величину к числу ε, получаем

$$e^{-\frac{k(k-1)}{2n}} \approx 1 - \varepsilon,$$

или

$$k^2 - k \approx 2n \log \frac{1}{1-\varepsilon}.$$

Иначе говоря,

$$k \approx \sqrt{2n \log \frac{1}{1-\varepsilon}}. \tag{3.6.1}$$

Итак, для случайной функции, отображающей множество X на множество Y, необходимо выполнить по крайней мере $k \approx \sqrt{2n \log \frac{1}{1-\varepsilon}}$ вычислений, чтобы обнаружить коллизию с заданной вероятностью ε. Из соотношения (3.6.1) вытекает, что даже если число ε достаточно велико (т.е. очень близко к единице), величина $\log \frac{1}{1-\varepsilon}$ остается весьма малой, а значит, в принципе, число k прямо пропорционально $\sqrt{n}$.

Если $\varepsilon = \frac{1}{2}$, то

$$k \approx 1{,}1774\sqrt{n}. \tag{3.6.2}$$

Зависимость числа k от величины n, продемонстрированная формулами (3.6.1) и (3.6.2), означает, что для случайной функции, пространство исходов которой состоит из n точек, чтобы обнаружить коллизию со значимой вероятностью, необходимо выполнить всего $\sqrt{n}$ вычислений.

Этот факт оказал значительное влияние на разработку криптосистем и криптографических протоколов. Например, если квадратный корень, извлеченный из размера фрагмента данных (скажем, криптографического ключа или сообщения), скрытого в качестве прообраза криптографической функции (которая, как правило, является случайной), не является достаточно большой величиной, данные можно расшифровать с помощью случайных вычислений значений функции. Такая атака получила название **атака по методу квадратного корня** (square-root attack), или **атака на основе "парадокса дней рождений"** (birthday attack). Второе название возникло из-за внешне парадоксального явления: положив $n = 365$ в формуле (3.6.5), мы получим $k \approx 22{,}49$. Иначе говоря, для того, чтобы с вероятностью более 50% обнаружить двух людей, родившихся в один и тот же день и находящихся в комнате, заполненной случайными людьми, достаточно, чтобы в комнате было всего лишь 23 человека. Эта величина кажется слишком маленькой по сравнению с интуитивно ожидаемой.

3.6.1 Применение парадокса дней рождений: алгоритм кенгуру Полларда для индексных вычислений

Пусть p — простое число. При определенных условиях (которые станут очевидными в главе 5), **функция возведения в степень по модулю** (modulo exponentiation) $f(x) = g^x \pmod p$ является, по существу, случайной. Иначе говоря, для чисел $x = 1, 2, \ldots, p-1$ значения $f(x)$ случайным образом разбросаны по всему

интервалу $[1, p-1]$. Эта функция широко применяется в криптографии, поскольку она является однонаправленной: вычислить $y = f(x)$ очень легко (используя алгоритм 4.3), однако найти обратную функцию, т.е. вычислить $x = f^{-1}(y)$, чрезвычайно трудно для практически всех значений $y \in [1, p-1]$.

Иногда для значения $y = f(x)$ известно, что $x \in [a, b]$ при некоторых значениях a и b. Очевидно, что, вычисляя значения $f(a), f(a+1)$, можно найти число x не более чем за $b-a$ шагов. Если число $b-a$ слишком велико, то такой метод перебора является непрактичным. Однако, если число $\sqrt{b-a}$ не слишком велико (например, если $b-a \approx 2^{100}$, то $\sqrt{b-a} \approx 2^{50}$, что является вполне разумной величиной), то при обращении функции $f(x)$ за $b-a$ шагов проявляется парадокс дней рождений. Используя этот факт, Поллард изобрел метод индексных вычислений [238], получивший название **λ-метод** или **метод кенгуру** (kangaroo method). Смысл этих названий станет ясен позднее.

Поллард описал свой алгоритм, используя в качестве персонажей двух кенгуру — домашнего, T, и дикого, W. Задача вычисления индекса x с помощью функции $y = g^x (\mathrm{mod}\, p)$ сводилась к ловле кенгуру W с помощью кенгуру T. Эту задачу можно решить, позволив обоим кенгуру прыгать вокруг по определенным траекториям. Пусть S — множество целых чисел, состоящее из J элементов ($J = \lfloor \log_2(b-a) \rfloor$, а значит, является достаточно малым числом):

$$\mathcal{S} = \{s(0), s(1), s(2), \ldots, s(J-1)\} = \{2^0, 2^1, \ldots, 2^{J-1}\}.$$

Каждый раз один из кенгуру прыгает на расстояние, которое случайным образом извлекается из множества $\mathcal{S}$, причем общее расстояние, пройденное каждым из кенгуру, регистрируется с помощью счетчика.

Кенгуру T начинает свой путь из известной точки $t_0 = g^b (\mathrm{mod}\, p)$. Поскольку кенгуру T является домашним, в качестве его дома можно рассматривать точку b. Его путь описывается следующей формулой.

$$t(i+1) = t(i) g^{s(t(i)(\mathrm{mod}\, J))} (\mathrm{mod}\, p) \text{ для } i = 0, 1, 2, \ldots \qquad (3.6.3)$$

Допустим, что кенгуру T делает n прыжков, а потом останавливается. Требуется определить, сколько прыжков должен сделать кенгуру T, чтобы поймать кенгуру W. После n-го прыжка счетчик пути, пройденного кенгуру T, показывает величину

$$d(n) = \sum_{i=0}^{n} s(t(i)(\mathrm{mod}\, J)).$$

Используя это значение, мы можем переписать выражение (3.6.3) для следов кенгуру T в следующем виде.

$$t(n) = g^{b+d(n-1)} (\mathrm{mod}\, p).$$

Кенгуру W начинает свой путь из неизвестной точки $w_0 = g^x (\mathrm{mod}\, p)$. Этой неизвестной точкой является число x, и именно поэтому кенгуру W считается диким. Его путь описывается следующей формулой.

$$w(j+1) = w(j) g^{s(w(j)(\mathrm{mod}\, J))} (\mathrm{mod}\, p) \text{ для } j = 0, 1, 2, \ldots \qquad (3.6.4)$$

Счетчик пути, пройденного кенгуру W, регистрирует величину

$$D(j) = \sum_{k=0}^{j} s(w_k (\mathrm{mod}\, J)).$$

Аналогично выражению (3.6.3) выражение (3.6.4) для следов кенгуру W можно переписать в следующем виде.

$$w(i) = g^{x+D(i-1)} (\mathrm{mod}\, p).$$

Очевидно, что следы обоих кенгуру, $t(i)$ и $w(j)$, представляют собой случайные функции. Первая из этих функций задается в точках i, а вторая — в точках j. Согласно парадоксу дней рождений после $n \approx \sqrt{b-a}$ прыжков, сделанных кенгуру T и W соответственно, при некоторых значениях $\xi \leqslant n$ и $\eta \leqslant n$ должна произойти коллизия $t(\xi) = w(\eta)$. Именно в этот момент кенгуру T и W встретятся в одной точке, т.е. кенгуру W попадет в капкан, установленный кенгуру T. Итак, кенгуру W пойман. Если количество случайных прыжков, сделанных обоими кенгуру, превышает $\sqrt{b-a}$, вероятность коллизии быстро стремится к единице.

В соответствии с формулами (3.6.3) и (3.6.4), когда возникает коллизия $t(\xi) = w(\eta)$, выполняются равенства $t(\xi+1) = w(\eta+1), t(\xi+2) = w(\eta+2), \ldots$ и т.д. Иначе говоря, при некоторых числах $n \approx m$ в конце концов будет выполнено равенство $w(m) = t(n)$. Поскольку после встречи оба кенгуру продолжают прыгать по одному и тому же пути, ведущему к равенству $w(m) = t(n)$, коллизию $t(\xi) = w(\eta)$ можно интерпретировать как точку, в которой пересекаются две палочки греческой буквы λ (напомним, что кенгуру T выполняет фиксированное количество прыжков n). По этой причине алгоритм получил название "λ-метод".

После возникновения коллизии выполняется равенство

$$g^x = g^{b+d(n-1)-D(m-1)} (\mathrm{mod}\, p).$$

Отсюда следует, что

$$x = b + d(n-1) - D(m-1).$$

Поскольку оба счетчика показывают величины $d(m-1)$ и $D(n-1)$ соответственно, величину x можно вычислить, используя длину пути, пройденного каждым кенгуру. При этом возможна ситуация, когда оба кенгуру пройдут слишком длинный путь до встречи, и поэтому искомая величина индекса может оказаться

равной $x + o$, где число o удовлетворяет равенству $g^o \bmod p = 1$. В этом случае можно считать, что, что ответом задачи является величина $x + o$.

Следует отметить, что описанный алгоритм является **вероятностным** (probabilistic), т.е. он может не обнаружить коллизию (иначе говоря, вычислить искомую величину индекса). Несмотря на это, благодаря высокой вероятности обнаружить коллизию вероятность отказа можно сделать достаточно малой. Смещая стартовую точку кенгуру W на известную величину δ и повторяя алгоритм, после нескольких итераций мы обязательно обнаружим коллизию.

Условием, обеспечивающим практическую целесообразность λ-алгоритма, является относительная малость величины $\sqrt{b-a}$. Следовательно, при $n = \sqrt{b-a}$ (где n — количество прыжков, сделанных кенгуру T) алгоритм выполняется за время, которое необходимо для вычислений $\sqrt{b-a}$ операций возведения в степень по модулю. Требования к памяти компьютера тривиальны: необходимо хранить лишь $J = \lfloor \log(b-a) \rfloor$ элементов. Временно́е ограничение, равное $\sqrt{b-a}$, означает, что данный алгоритм нельзя применять для вычисления больших индексов. Как образно сказал Поллард, кенгуру не может проскакать через весь континент.

3.7 Теория информации

В соответствии с определением, данным Шенноном (Shannon) в работах [262, 263], **энтропия** (entropy) источника сообщения представляет собой меру объема информации, содержащейся в источнике. Эта мера принимает вид функции распределения вероятностей во множестве сообщений, которые может порождать источник.

Пусть $L = \{a_1, a_2, \ldots, a_n\}$ — язык, состоящий из n разных символов. Предположим, что источник S может порождать эти символы с независимыми вероятностями

$$\text{Prob}[a_1], \text{Prob}[a_2], \ldots, \text{Prob}[a_n],$$

причем эти вероятности удовлетворяют условию

$$\sum_{i=1}^{n} \text{Prob}[a_i] = 1. \tag{3.7.1}$$

Энтропия источника S равна

$$H(S) = \sum_{i=1}^{n} \text{Prob}[a_i] \log_2 \left(\frac{1}{\text{Prob}[a_i]} \right). \tag{3.7.2}$$

Функция энтропии $H(S)$, определенная формулой (3.7.2), выражает величину, которую можно назвать “средним количеством бит в сообщении, порожденном источником”.

Понятие энтропии можно проиллюстрировать следующим примером. Представим себе, что источник S не имеет памяти, поэтому мы должны записывать все символы, которые он порождает. Естественным решением было бы просто записывать все символы, порождаемые источником S. Однако в соответствии с формулой (3.7.1) источник S может порождать символы $a_1, a_2, \ldots, a_n$ с известными вероятностями. Было бы неинтересно и неэффективно просто записывать все подряд. Итак, вопрос заключается в следующем: как *эффективно* записать сообщение источника S, *представляющее какой-либо интерес*?

Пусть источник S порождает сообщения в виде последовательностей, состоящих из k символов. Иначе говоря, сообщение представляет собой слово, состоящее из k символов.

$$a_{i_1} a_{i_2} \ldots a_{i_k} \text{ для } 1 \leqslant i_k \leqslant n.$$

Пусть L_k обозначает минимальное ожидаемое количество бит, которое требуется для записи k-символьного слова, порожденного источником S. Для оценки величины L_k используется следующая теорема.

Теорема 3.2 (Шеннон [262, 263]).

$$\lim_{k \to \infty} \frac{L_k}{k} = H(S).$$

Доказательство. При всех $k > 0$ выполняется следующая двусторонняя оценка.

$$kH(S) \leqslant L_k \leqslant kH(S) + 1.$$

Поделив эти неравенства на число k и перейдя к пределу, получаем искомую оценку. □

Иначе говоря, минимальное среднее количество бит, необходимое для записи сообщений, порождаемых источником S, равно $H(S)$.

3.7.1 Свойства энтропии

Функция $H(S)$ принимает минимальное значение, равное нулю, если источник S порождает некий символ, например a_1, с вероятностью 1. Это следует из следующего выражения.

$$H(S) = \text{Prob}[a_1] \log_2 \left(\frac{1}{\text{Prob}[a_1]} \right) = \log_2 1 = 0.$$

Иначе говоря, если нам известно, что источник S порождает один-единственный символ a_1, зачем нам тратить бит на его запись?

Функция $H(S)$ достигает максимума, равного $\log_2 n$, если источник S порождает каждый из n символов с одинаковой вероятностью, равной $\frac{1}{n}$, т.е. источник

S является случайным генератором равномерного распределения. Это следует из такого выражения.

$$H(S) = \frac{1}{n}\sum_{i=1}^{n} \log_2 n = \log_2 n.$$

Этот факт можно интерпретировать следующим образом: поскольку источник S может порождать один из n символов с одинаковой вероятностью, для записи каждого из этих n чисел следует зарезервировать $\log_2 n$ бит.

Функцию $H(S)$ можно рассматривать как объем *неопределенности*, или *информации*, содержащейся в каждом сообщении, порожденном источником S.

Пример 3.10. Рассмотрим протокол 1.1 ("Подбрасывание монеты по телефону"). Независимо от его реализации — с помощью телефона или компьютерной сети, — этот протокол сводится к жеребьевке с помощью случайного бита. Алиса извлекает большое целое число $x \in_U \mathbb{N}$, затем передает Бобу значение однонаправленной функции $f(x)$ и в заключение раскрывает Бобу значение x после его попытки угадать случайное число. С точки зрения Боба, число x не должно рассматриваться как новая информация, поскольку, даже еще не получив значение $f(x)$, он знает, что x — натуральное число. Боб использует только интересующую его часть сообщения Алисы: критерием жеребьевки является четность числа x. Итак, мы получаем следующее выражение.

$$\begin{aligned} H(\text{Алиса}) = \text{Prob}[x - \text{нечетное}] \log_2 \left(\frac{1}{\text{Prob}[x - \text{нечетное}]}\right) + \\ + \text{Prob}[x - \text{четное}] \log_2 \left(\frac{1}{\text{Prob}[x - \text{четное}]}\right) = \\ = \frac{1}{2}\log_2 2 + \frac{1}{2}\log_2 2 = 1. \end{aligned}$$

Иначе говоря, для записи сообщений Алисы достаточно одного бита, хотя она генерирует большое целое число. □

Повторив протокол 1.1 n раз, Алиса и Боб получат строку, состоящую из n бит: если Боб угадал правильно, в строку записывается единица, если нет — нуль. В этой ситуации и Алиса, и Боб рассматриваются как источники случайных однобитовых протокольных сообщений. Строка битов признается обеими сторонами случайной, поскольку и Алиса, и Боб порождают случайные сообщения и знают, что противоположная сторона не может их контролировать.

3.8 Избыточность естественных языков

Пусть источник $S(L)$ порождает слова естественного языка L. Допустим, что в среднем каждое слово в языке L состоит из k символов. По теореме Шеннона

(теорема 3.2) величина $H(S(L))$ представляет собой минимальное среднее число бит в сообщении, порожденном источником $S(L)$ (напомним, что сообщение источника S представляет собой слово, состоящее из k символов). Следовательно, величина

$$r(L) = \frac{H(S(L))}{k}$$

является минимальным средним количеством битов, необходимых для записи символов языка L. Величина $r(L)$ называется **энтропией языка** L (rate of language L). Пусть L — английский язык. Шеннон показал, что энтропия английского языка $r(English)$ лежит в интервале от 1,0 до 1,5 бит/буква [265].

Пусть $\Sigma = \{a, b, \ldots, z\}$. Тогда $r(\Sigma) = \log_2 26 \approx 4{,}7$ бит/буква. Величина $r(\Sigma)$ называется **абсолютной энтропией языка** (absolute rate of language) с алфавитом Σ. Сравнивая величины $r(\Sigma)$ и $r(English)$, можно убедиться, что фактическая энтропия английского языка значительно меньше абсолютной.

Избыточностью языка L (redundancy of language) с алфавитом Σ называется величина

$$r(\Sigma) - r(L)(\text{бит/символ}).$$

Учитывая консервативную оценку $r(English) = 1{,}5$, получаем, что избыточность английского языка равна $4{,}7 - 1{,}5 = 3{,}2$ бит/буква. В процентном выражении избыточность равна $3{,}2/4{,}7 \approx 68\%$. Иными словами, около 68% букв в английском слове излишни. Это означает, что статью, написанную на английском языке, можно сократить до 32% от первоначального объема без потери информации.

Избыточность естественного языка возникает вследствие наличия некоторых хорошо известных и повторяющихся форм. Например, в английском языке за буквой q практически всегда следует буква u. Кроме того, часто встречаются артикль “the” и окончания слов “ing” и “ed”. Избыточность естественных языков представляет собой весьма ценное свойство для **криптоанализа** (cryptanalysis), позволяющее восстанавливать исходные сообщения или определять криптографический ключ по зашифрованному тексту.

Пример 3.11. Как указано в главе 1, в книге рассматривается множество атак на криптографические алгоритмы и протоколы. В главе 14 мы опишем и изучим четыре разновидности атак на алгоритмы шифрования. Эти атаки имеют довольно длинные названия.

- Пассивная атака на основе неразличимых исходных текстов (passive plaintext indistinguishable attack).
- Активная атака на основе неразличимых подобранных исходных текстов (active plaintext indistinguishable attack in the chosen-plaintext mode).
- Активная атака на основе неразличимых подобранных зашифрованных текстов (active plaintext indistinguishable attack in the non-adaptive chosen-cipthertext mode).

- Активная атака на основе неразличимых адаптивно подобранных зашифрованных текстов (active plaintext indistinguishable attack in the adaptive chosen-ciphertext mode).

Полное описание этих атак будет дано в главе 14, а пока мы отметим два важных обстоятельства.

1. Использование длинных названий вполне обоснованно, поскольку они содержат большое количество информации.
2. В главе 14 будут рассмотрены только четыре атаки, перечисленные выше.

Поскольку в главе 14 мы опишем только эти четыре атаки, фактическая энтропия их названий равна как минимум 2 бит. Однако, так как в главе 14 будут использованы числа 0, 1, 2 и 3 и некоторые другие символы (например, буква "a", индексы "i" и "j", параметр k и т.п.), для однозначной идентификации атак нам придется использовать больше двух бит.

Обратите внимание на то, что в главе 14 мы не используем строки, имеющие вид a0, a1, a2 и a3, хотя название каждой из атак можно было бы сократить именно так, не теряя однозначного соответствия. Следовательно, в рамках главы 14 энтропию названия атак можно было бы снизить до $4{,}7 + 2 = 6{,}7$ бит на одно название. Здесь число 4,7 относится к букве "a", а два бит — к цифрам 0, 1, 2 и 3.

С другой стороны, путем простого подсчета читатели могут убедиться, что средняя длина четырех длинных названий перечисленных выше атак равна 62,75 буквы (в английском варианте. — *Прим. ред.*). Следовательно, среднее количество бит на букву в этих именах равно $6{,}7/67{,}75 = 0{,}107$. Используя этот результат, мы можем вычислить избыточность длинных названий атак, изученных в главе 14.

$$\frac{6{,}7 - 0{,}107}{6{,}7} > 98\%.$$

□

Итак, указанные выше длинные названия атак чрезвычайно избыточны!

Однако отрасль науки, изучающей криптографические системы с доказуемой сильной стойкостью, намного шире, чем тема, рассмотренная в главе 14. Следовательно, имена a0, a1, a2 и a3, использованные в примере 3.11, являются слишком короткими (их использование может привести в неоднозначному пониманию сущности атаки и создать неудобства). На самом деле сокращенные названия последних трех атак, перечисленных в примере 3.11, выглядят так: IND-CPA, IND-CCA и IND-CCA2 соответственно. Именно эти названия используются в главе 14.

В заключение отметим, что сокращения приняты только для последних трех атак, поскольку именно эти атаки изучаются чаще других. Первая атака известна под своим полным названием "passive (plaintext indistinguishable) attack", поскольку ее легко предотвратить и она рассматривается довольно редко.

3.9 Резюме

В главе рассмотрены элементарные основы теории вероятностей и теории информации. Однако изложенного материала вполне достаточно для понимания остальной части книги.

Применяя теорию вероятностей, необходимо владеть основными понятиями, а также знать свойства и правила вычисления вероятностей. Следует подчеркнуть, что достичь хорошего знания теории вероятностей совсем не сложно — закон полной вероятности, закон больших чисел и парадокс дней рождений можно доказать на основе интуитивно понятных элементарных свойств и правил.

В остальной части книги мы будем широко использовать условную вероятность, закон полной вероятности, биномиальные распределения и парадокс дней рождений (в качестве яркого примера его применения мы описали λ-алгоритм Полларда).

В главе изложены основы теории информации. Теперь читатели знают, что энтропия источника сообщения представляет собой меру объема информации, содержащейся в сообщениях, порождаемых этим источником, или степени случайности (непредсказуемости) этих сообщений.

Упражнения

3.1. Предположим, игральные кубики выкидываются один за другим. Определите вероятность следующих событий.

а) Сумма очков равна 7, 1 или не превосходит 12.

б) Количество очков на первом игральном кубике меньше, чем количество очков на втором игральном кубике.

в) По крайней мере на одном из игральных кубиков выпало шесть очков.

г) На обоих игральных кубиках выпало по шесть очков.

3.2. Вычислите вероятность того, что в предыдущей задаче на первом игральном кубике выпало три очка, если сумма очков на обоих кубиках не меньше восьми.

3.3. Допустим, что 4,5% населения и 0,6% женщин являются дальтониками. Каков процент дальтоников среди мужчин, доля которых равна 49,9%?

Подсказка: примените закон полной вероятности.

3.4. Предположим, что случайная величина θ равномерно распределена на интервале $[-\pi/2, \pi/2]$. Найдите вероятность того, что $\theta \leqslant 1/2$, и того, что $|\sin\theta| \leqslant 1/2$.

3.5. Четверть чисел в некотором множестве являются квадратами целых чисел. Извлечем из этого множества пять случайных чисел. Определите вероятность того, что большинство из них окажется квадратами целых чисел.

Подсказка: аналогично примеру 3.8.3 просуммируйте количество вариантов, в которых количество квадратов целых чисел превышает три.

3.6. Дайте определение левого и правого хвостов биномиального распределения.

3.7. Докажите верхнюю оценку (3.5.8) для левого хвоста биномиального распределения.

3.8. Почему определение 3.2 считается теоремой, которую можно доказать с помощью закона больших чисел?

3.9. Пусть $n = pq$, где p и q — разные простые числа приблизительно одинаковой длины. Известно, что для любого $a < n$ и $\gcd(a, n) = 1$ выполняется равенство $a^{p+q} = a^{n+1} (\mathrm{mod}\, n)$. Докажите, что число n можно факторизовать за $n^{1/4}$ шагов.

3.10. В протоколе "Подбрасывание монеты по телефону" Алиса извлекает из равномерно распределенной совокупности большое целое число. Чему равна энтропия Алисы, измеренная со стороны Алисы и Боба соответственно?

3.11. В примере 3.11 мы измерили избыточность четырех очень длинных названий атак, описанных в главе 14, по сравнению с сокращениями a0, a1, a2 и a3. Используя сведения, изложенные в главе. оцените избыточность следующих четырех сокращенных названий атак.

- Passive IND-attack
- IND-CPA
- IND-CCA
- IND-CCA2

Глава 4

Вычислительная сложность

4.1 Введение

Если случайная величина имеет равномерное распределение и не зависит от *какой-либо* известной информации, то *не существует* способа связать эту величину с какой-либо другой информацией с помощью "вычислений". Именно это обстоятельство лежит в основе *безусловно стойкой* (или *теоретико-информационной*) схемы шифрования: одноразового блокнота (one-time pad). Суть этой схемы шифрования заключается в побитовом смешивании строки, состоящей из равномерно распределенных символов (так называемой *ключевой строки*), и строки сообщения (см. раздел 7.3.3). Для обеспечения независимости между ключевой строкой и строкой сообщения необходимо, чтобы они имели одинаковую длину. К сожалению, на практике эти ограничения почти невыполнимы.

Несмотря на это, ситуация не является безнадежной. Во время записи сообщения шифровальщики широко используют не слишком мощные вычислительные устройства и методы. (Заметим, что эти методы доступны и взломщикам шифров.) На сегодняшний день еще не существует удачных вычислительных средств, позволяющих установить связь между двумя обрывками информации, если один из них "выглядит случайным", несмотря на то, что оба текста полностью зависят друг от друга (например, открытый и зашифрованный тексты в большинстве криптосистем). В результате безопасность современных криптографических систем основывается на так называемой *теории вычислительной сложности* (complexity-theoretic model). Эта безопасность *зависит* от предположения, что некоторые задачи являются трудноразрешимыми (intractable problems). Здесь термин "трудноразрешимые задачи" означает, что широкодоступные вычислительные методы не в состоянии эффективно их решать.

Следует заметить, что описанная выше ситуация может оказаться временной. В настоящее время возникли новые, более мощные модели вычислений — *квантовые* (quantum information processing — QIP). В рамках этих моделей гигантское количество вычислений можно выполнять параллельно, манипулируя так называемыми "суперпозициями" квантовых состояний. В результате многие трудноразрешимые задачи, обеспечивающие стойкость криптосистем, будут решены и, следовательно, станут бесполезными. Например, с помощью квантового

компьютера факторизация и умножение целых чисел одинакового размера будут выполняться за одинаковое время, и, следовательно, такие известные криптосистемы с открытым ключом, как система RSA (Rivest, Shamir and Adelman [246]), описанная в разделе 8.5, сойдут со сцены. Однако пока квантовые вычисления остаются лишь на бумаге. Текущий рекорд факторизации составного числа равен 15 [300]. Это наименьшее нечетное составное целое число, не являющееся квадратом.

Итак, пока еще можно не слишком бояться применения квантовых вычислений. В остальной части главы мы будем использовать обычную вычислительную модель и подходы, основанные на применении теории вычислительной сложности.

4.1.1 Структурная схема главы

В разделе 4.2 описывается модель вычислений Тьюринга (Turing). В разделе 4.3 рассматриваются классы детерминированных полиномиальных алгоритмов и способы измерения сложности. В разделах 4.4 и 4.5 изучаются два подкласса недетерминированных полиномиальных задач (NP-задачи). Первый подкласс (раздел 4.4) состоит из вероятностных полиномиальных задач, которые в свою очередь разделяются на четыре подкласса эффективно решаемых задач (разделы 4.4.2–4.4.5). Второй подкласс (раздел 4.5) включает задачи, которые эффективно решаются только, если известна дополнительная информация. Эти задачи играют важную роль в современной криптографии, основанной на теории вычислительной сложности. В разделе 4.6 вводится понятие сложности, не связанное с полиномами. Раздел 4.7 посвящен неполиномиальным задачам, в частности, решению проблемы неразличимости при полиномиальной оценке временны́х затрат работы алгоритма (polynomial-time indistinguishability). В заключительном разделе 4.8 обсуждается связь между теорией вычислительной сложности и современной криптографией.

4.2 Машины Тьюринга

Для того чтобы дать точное определение эффективной процедуры (т.е. алгоритма), Тьюринг изобрел воображаемое вычислительное устройство, называемое **машиной Тьюринга** (Turing machine) и представляющее собой элементарную, но достаточно общую модель вычислений. Она позволяет описать вопросы, связанные с вычислительной сложностью. Ниже мы рассмотрим вариант машины Тьюринга, вполне достаточный для раскрытия темы. Общее описание машины Тьюринга дано в разделе 1.6 работы [9].

В нашем варианте машина Тьюринга (рис. 4.1) состоит из *управляющего устройства с конечным числом состояний* (finite-state control unit), $k(\geqslant 1)$ *лент*

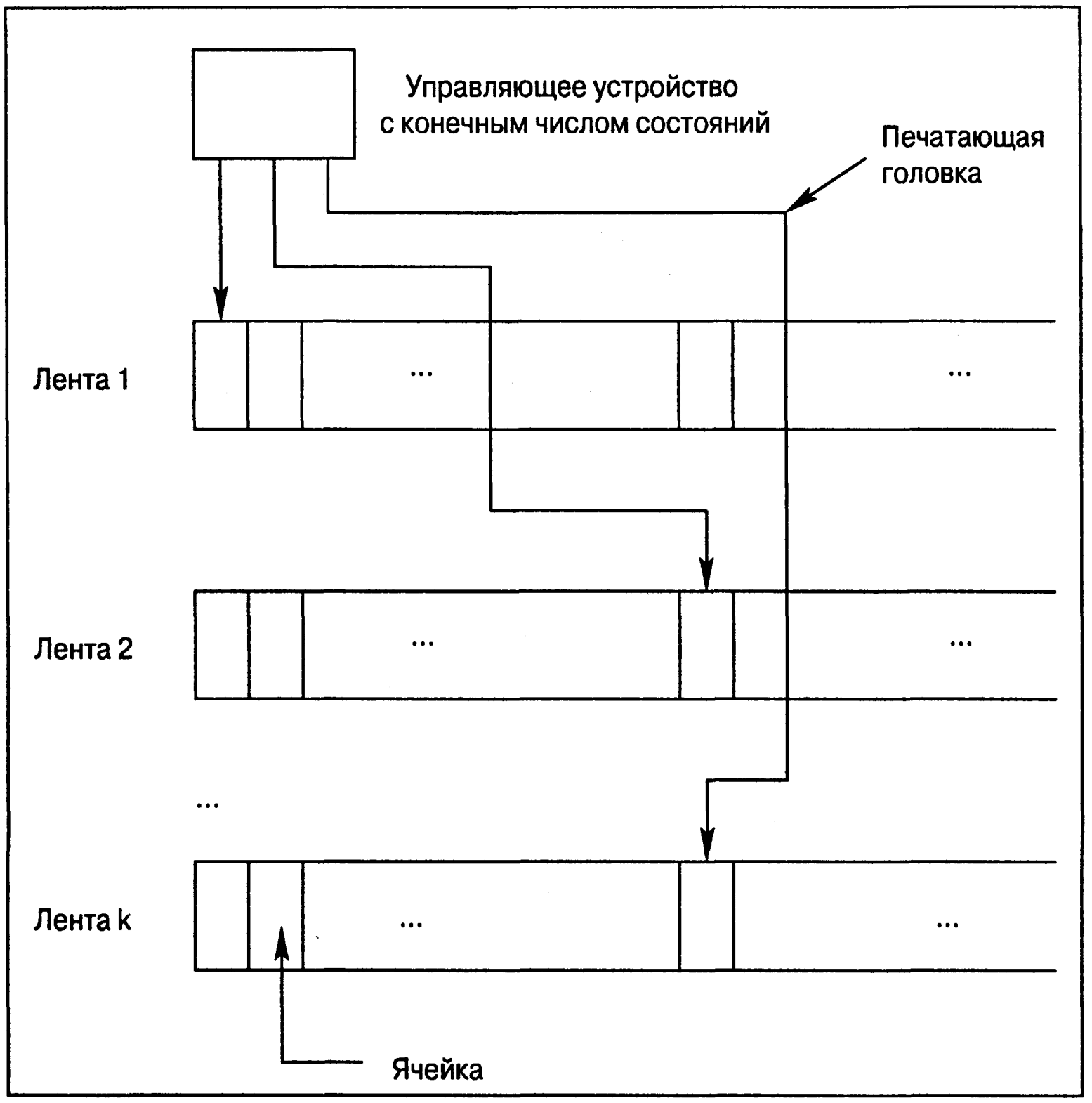


Рис. 4.1. Машина Тьюринга

(tapes) и такого же количества *головок* (tapeheads). Управляющее устройство контролирует операции, выполняемые головками, которые считывают информацию с лент или записывают ее на ленту. Каждая головка имеет доступ к *своей* ленте и может перемещаться вдоль нее влево и вправо. Каждая лента разделена на бесконечное количество *ячеек* (cells). Машина решает задачу, перемещая головку вдоль строки, состоящей из конечного количества символов, расположенных последовательно, начиная с крайней левой ячейки. Каждый символ занимает одну ячейку, а оставшиеся ячейки ленты, расположенные справа, остаются *пустыми* (blank). Описанная выше строка называется *исходными данными задачи* (input). Сканирование начинается с крайней слева ячейки ленты, содержащей исходные данные, когда машина находится в предписанном *начальном состоянии* (initial state). В каждый момент времени только одна из головок имеет доступ к своей ленте. Операция доступа головки к своей ленте называется *тактом* (legal move). Если машина начинает работу с начального состояния, последовательно выполняет такты, сканирует исходную строку и завершает работу, достигая *заключительного состояния* (termination condition), говорят, что машина *распознает*

(recognize) исходные данные. В противном случае в некоторый момент машина не может выполнить очередной такт и останавливается, не распознав исходные данные. Исходные данные, распознаваемые машиной Тьюринга, называются *предложением* (instance) распознаваемого *языка* (language).

Для каждой конкретной задачи машину Тьюринга можно полностью определить с помощью функции, описывающей работу управляющего устройства с конечным числом состояний. Такая функция может иметь вид таблицы, в которой для каждого состояния указана операция, выполняемая на следующем такте. В свое время мы рассмотрим пример и описание машины Тьюринга (пример 4.1).

Количество тактов T_M, которые машина Тьюринга M должна выполнить при распознавании исходной строки, называется продолжительностью работы или *временно́й сложностью* (time complexity) машины M. Очевидно, что величину T_M можно представить в виде функции $T_M(n) : \mathbb{N} \mapsto \mathbb{N}$, где n — *длина* (length), или *размер* (size), исходного предложения, т.е. количество символов, из которых состоит исходная строка, когда машина M пребывает в начальном состоянии. Легко видеть, что $T_M(n) \geqslant n$. Кроме временны́х ограничений, машина M имеет ограничения памяти S_M, представляющие собой количество ячеек, которые доступны для записи. Величину S_M также можно представить в виде функции $S_M(n) : \mathbb{N} \mapsto \mathbb{N}$, которая называется *пространственной сложностью* (space complexity) машины M.

Конкретная машина Тьюринга будет рассмотрена в следующем разделе.

4.3 Детерминированное полиномиальное время

Начнем с рассмотрения класса языков, распознаваемых детерминированными машинами Тьюринга за **полиномиальное время** (polynomial time). Функция $p(n)$ является полиномиальной по целому аргументу n, если она имеет вид

$$p(n) = c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0, \tag{4.3.1}$$

где числа k и $c_i (i = 0, 1, 2, \ldots, k)$ — целые константы, причем $c_k \neq 0$. Число $k > 0$ называется **степенью** (degree) полинома и обозначается как $\deg(p(n))$, а числа c_i называются **коэффициентами** (coefficients) полинома $p(n)$.

Определение 4.1 (Класс $\mathcal{P}$). *Символом $\mathcal{P}$ обозначается класс языков, имеющих следующие характеристики. Язык L принадлежит классу $\mathcal{P}$, если существует машина Тьюринга M и полином $p(n)$, такие что машина M распознает любое предложение $I \in L$ за время $T_M(n)$, где $T_M(n) \leqslant p(n)$ для всех неотрицательных целых чисел n, а n — параметр, задающий длину предложения I. В этом случае говорят, что язык L распознается за полиномиальное время.*

Проще говоря, языки, распознаваемые за полиномиальное время, считаются "*всегда* простыми", а полиномиальные машины Тьюринга — "*всегда* эффективными" (смысл терминов "простой язык" и "эффективная машина" будет раскрыт в разделе 4.4.6). Рассмотрим смысл слова "*всегда*". Все машины Тьюринга, распознающие язык L из класса $\mathcal{P}$, называются **детерминированными** (deterministic). Детерминированная машина Тьюринга порождает результат, который полностью предопределен исходными данными и начальным состоянием машины. Иначе говоря, повторный запуск машины Тьюринга с теми же исходными данными и начальным состоянием приводит к идентичным результатам.

Следует отметить, что в определении 4.1 нечеткие ограничения "*любое предложение* $I \in L$" и "*для всех неотрицательных целых чисел* n" имеют большое значение. В работах, посвященных теории вычислительной сложности, задача считается решенной, только если любой экземпляр данной задачи можно решить с помощью одной и той же машины Тьюринга (т.е. с помощью одного и того же метода). Только в этом случае алгоритм считается достаточно общим и может называться методом. Для иллюстрации рассмотрим следующий пример.

Пример 4.1 (Язык DIV3). Пусть DIV3 — множество неотрицательных целых чисел, кратных трем. Покажем, что DIV3 $\in \mathcal{P}$.

Для того чтобы распознать язык DIV3 за полиномиальное время, построим машину Тьюринга с одной лентой.

Сначала заметим, что если записать исходные данные в виде троичных чисел (в позиционной системе счисления по основанию 3), т.е. в виде строки символов из множества $\{0, 1, 2\}$, то задача распознавания становится тривиальной: исходная строка x принадлежит языку DIV3 тогда и только тогда, когда последняя цифра в строке x равна нулю. Следовательно, создаваемая машина должна просто перемещать головку вправо, пока не обнаружит пустой символ. Машина должна остановиться и выдать ответ "ДА", если и только если последний непустой символ был равен нулю. Очевидно, что данная машина может распознавать любое предложение, состоящее из цифр, причем количество шагов алгоритма равно длине исходной строки. Следовательно, DIV3 $\in \mathcal{P}$.

Однако требуется еще доказать, что факт DIV3 $\in \mathcal{P}$ не зависит от способа представления исходной строки (т.е. основания позиционной системы счисления). Для этого достаточно рассмотреть случай, когда исходная строка представлена в двоичном виде. Назовем создаваемую машину Div3. Конечное управляющее устройство машины Div3 выполняет следующий такт, руководствуясь функцией, описанной на рис. 4.2.

Покажем, что машина Div3, определенная функцией, представленной на рис. 4.2, является достаточно общей для распознавания всех предложений в языке DIV3.

Текущее состояние	Символ на ленте	Следующий такт	Новое состояние
q_0 (начальное состояние)	0 1 пустой	вправо вправо "Звонок" & Стоп	q_0 q_1
q_1	0 1	вправо вправо	q_2 q_0
q_2	0 1	вправо вправо	q_1 q_2

Рис. 4.2. Операции машины Div3

Во-первых, для распознавания того факта, что двоичная строка x принадлежит языку DIV3, машине Div3 достаточно трех состояний, соответствующих ситуациям, когда ее головка сканирует строки $3k$, $3k+1$ и $3k+2$ (для $k \geqslant 0$) соответственно. Наименьшая исходная строка равна нулю. Следовательно, после сканирования нулевой строки машина Div3 должна остаться в начальном состоянии. Без ограничения общности будем обозначать начальное состояние через q_0, состояние, в которое переходит машина Div3, сканируя исходную строку, состоящую из единицы, — через q_1, а состояние, в которое переходит машина Div3, сканируя исходную строку, состоящую из двойки $(= (10)_2)$[1], — через q_2.

Приписав справа к неотрицательному двоичному целому числу a нуль (или единицу), мы получим число $2a$ (или $2a+1$ соответственно). Следовательно, после сканирования числа $a = 3k$ машина Div3 должна остаться в начальном состоянии, если следующий сканируемый символ является нулем, поскольку в этом случае $2a = 6k = 3k'$. Если следующий сканируемый символ равен единице, машина должна перейти в состояние q_1, поскольку $2a+1 = 6k+1 = 3k'+1$. Аналогично после сканирования числа $a = 3k+1$ машина Div3 из состояния q_1 должна перейти в состояние q_2, если $2a = 6k+2 = 3k'+2$, или в состояние q_0, если $2a+1 = 6k+3 = 3k'$. Остальные два варианта относятся к ситуации, когда $a = 3k+2$: если $2a = 6k+4 = 3k'+1$ машина Div3 переходит из состояния q_2 в состояние q_1, а если $2a+1 = 6k+5 = 3k'+2$, машина Div3 остается в состоянии q_2.

[1] Мы используем символы $(a_1 a_2 \ldots a_n)_b$ при $a_i < b$ и $i = 1, 2, \ldots, n$ для обозначения числа, записанного в позиционной системе счисления по основанию b. Когда $b = 10$ и $b = 2$, основание системы счисления часто не указывается, если это не вызывает недоразумений.

Итак, машина Div3 завершает сканирование строк $3k$, $3k+1$ и $3k+2$ (для $k \geqslant 0$), пребывая в состоянии q_0, q_1 или q_2 соответственно. Теперь, если головка обнаружит пустой символ, машина остановится и выдаст ответ "ДА", только если она пребывает в состоянии q_0. В остальных ситуациях машина Div3 прекращает работу, не завершив распознавания.

Легко показать, что $T_{\text{Div3}(n)} = n$. Следовательно, машина Div3 распознает язык DIV3 за полиномиальное время. □

Пример 4.2.

1. Битовая строка $10101 (= (21)_{10})$ распознается. Машина Div3 распознает ее за $T_{\text{Div3}(|10101|)} = |10101| = 5$ тактов.
2. Битовая строка $11100001 (= (225)_{10})$ распознается. Машина Div3 распознает ее за $T_{\text{Div3}(|11100001|)} = |11100001| = 8$ тактов.
3. Битовая строка $10 (= (2)_{10})$ не распознается. Машина Div3 находит ответ за два такта. □

4.3.1 Полиномиальные вычислительные задачи

По определению класс $\mathcal{P}$ является классом языков, распознаваемых за полиномиальное время. Задача распознавания языка является **задачей принятия решений** (decisional problem). При любых исходных данных результатом решения такой задачи является ответ "ДА" или "НЕТ". Однако класс $\mathcal{P}$ является более широким и содержит **полиномиальные вычислительные задачи** (polynomial-time computational problems). При любых исходных данных результатом решения таких задач является более общий ответ, чем "ДА" и "НЕТ". Поскольку машина Тьюринга может записывать символы на ленту, она позволяет решать такие задачи.

Например, можно создать машину Тьюринга, которая не только распознает любое предложение x из языка DIV3, но и выводит на ленту число $x/3$. Назовем эту машину Div3-Comp. Для простоты предположим, что исходная строка записана в троичном виде. Тогда исходная строка принадлежит языку DIV3, если и только если ее последняя цифра равна нулю, а результатом деления $x/3$ является содержимое ленты, полученное после стирания последней цифры, равной нулю, если эта цифра не является единственным символом на ленте. Если предположить, что машина Div3-Comp должна вводить и выводить только двоичные числа, то ее реализация может выглядеть следующим образом. Сначала переведем исходную строку x из двоичного вида в троичный, получим результат деления $x/3$, а затем переведем ответ в двоичный вид. Очевидно, что эти преобразования можно *автоматически* выполнить цифра за цифрой за $c|x|$ тактов, где c — константа.

Следовательно,

$$T_{\text{Div3-Comp}}(|x|) \leqslant C|x|,$$

где C — константа. Очевидно, что класс $\mathcal{P}$ должен содержать задачу, которую можно решить с помощью машины Div3-Comp.

В пользу того, что класс $\mathcal{P}$ содержит полиномиальные вычислительные задачи, говорят следующие рассуждения общего характера. Вычислительное устройство, имеющее неймановскую архитектуру (иначе говоря, всем известную современную компьютерную архитектуру [227]), состоит из счетчика, памяти и центрального процессора (central processor unit — CPU), поочередно выполняющего элементарные команды, называемые *микрокомандами*.

Load (Загрузить)	Загрузить содержимое из памяти в регистр центрального процессора.
Store (Сохранить)	Сохранить в памяти содержимое регистра.
Add (Сложить)	Сложить содержимое двух регистров.
Comp (Дополнение)	Вычислить дополнение к содержимому регистра (для вычитания с помощью операции Add).
Jump (Перейти)	Присвоить счетчику новое значение.
JumpZ (Условный переход)	Выполнить команду Jump, если содержимое регистра равно нулю.
Stop (Остановиться)	Прекратить работу.

Хорошо известно (см. раздел 1.4 в работе [9]), что перечисленных выше микрокоманд достаточно для создания алгоритмов, решающих любые арифметические задачи на неймановском компьютере. (Однако следует заметить, что выражение "любые арифметические задачи" не означает "предложения произвольного размера".) Можно доказать (см. теорему 1.3 в работе [9]), что каждую микрокоманду из указанного выше набора, можно имитировать на машине Тьюринга за полиномиальное время. Следовательно, задачу, которую можно решить за полиномиальное время на неймановском компьютере (т.е. количество микроинструкций, используемых в алгоритме, представляет собой значение полинома, зависящего от размера исходных данных), можно решить за полиномиальное время и с помощью машины Тьюринга. Это возможно благодаря тому, что для любых полиномов $p(n)$ и $q(n)$ произвольные арифметические комбинации $p(n)$, $q(n)$, $p(q(n))$ и $q(p(n))$ также являются полиномами, зависящими от аргумента n. Отметим, что из множества микрокоманд произвольно исключены операции умножения и деления. Умножение двух чисел размера n выполняется с помощью n сложений. Следовательно, затраты, связанные с операцией умножения, вычисляются по формуле $n \times$ Затраты(Add). Затраты на деление равны затратам на умножение, поскольку

деление можно свести к повторному выполнению операции вычитания, которая представляет собой сложение дополнений.

Следует упомянуть о некоторых несущественных различиях между неймановским компьютером и машиной Тьюринга. По определению 4.1 задача считается разрешимой с помощью машины Тьюринга тогда и только тогда, когда *любой* вариант задачи можно решить с помощью *одной и той же* машины ("одна машина решает все варианты задачи"). Затраты, связанные с решением задачи с помощью машины Тьюринга, *равномерно* зависят от размера задачи. Предварительные ограничения на размеры задачи необязательны. Машина Div3 в примере 4.1 наглядно иллюстрирует это утверждение. Благодаря этому свойству оценки затрат, используемые для измерения сложности вычислений на машине Тьюринга, являются **равномерными** (uniform cost measure). В противоположность этому регистровые и логические схемы, представляющие собой базовые строительные блоки неймановского компьютера, имеют фиксированный размер. Следовательно, задачи, разрешимые с помощью неймановского компьютера, должны иметь заранее фиксированный размер: чем больше вариант одной и той же задачи, тем больший компьютер требуется для ее решения. В общем, машины разных размеров не соответствуют равномерным оценкам затрат, связанных с ее решением. Следовательно, оценки затрат в неймановской модели вычислений носят **неравномерный характер**. Однако до сих пор разница между оценками равномерных и неравномерных затрат не создала ни одного нового класса сложности и не разрушила ни одного существующего класса. По этой причине мы утверждаем, что эта разница несущественна.

В оставшейся части главы мы будем часто пренебрегать различиями между задачами принятия решений и вычислительными задачами, а также между вычислениями с помощью машины Тьюринга, современного компьютера, процедуры или алгоритма. Задачи принятия решений и вычислительные задачи будут называться просто задачами, а машины, компьютеры, процедуры и алгоритмы будут называться просто методами или алгоритмами. Иногда мы будем упоминать задачу распознавания языка и только в этом случае будем использовать машину Тьюринга в качестве основного инструмента вычислений.

4.3.2 Алгоритмы и оценки вычислительной сложности

Рассмотрим три весьма полезных алгоритма с полиномиальным временем выполнения. По ходу изложения мы ознакомимся с языком программирования, который будем использовать в книге для описания алгоритмов и протоколов, примем некоторые обозначения, касающиеся оценки сложности алгоритмов и протоколов, и оценим временну́ю сложность большого количества арифметических операций, которые наиболее часто применяются в криптографии.

Как упоминалось выше, машина Тьюринга представляет собой общую модель вычислений и позволяет формализовать понятие вычислительной сложности процедур. Однако, как правило, алгоритмы не описываются ни в терминах операции такой примитивной машины, ни в терминах микрокоманд современного компьютера (см. раздел 4.3.1). Для точного описания алгоритмов и математических выражений мы будем использовать псевдокод, который очень похож на большинство популярных языков программирования высокого уровня, таких как Pascal или C, и понятен всем читателям.

4.3.2.1 Наибольший общий делитель

Первый алгоритм, который мы изучим, — знаменитый алгоритм Евклида, предназначенный для вычисления наибольшего общего делителя (алгоритм 4.1). Обозначим через $\gcd(a, b)$ наибольший общий делитель целых чисел a и b, т.е. наибольшее целое число, на которое делятся оба эти числа.

Алгоритм 4.1. Алгоритм Евклида для вычисления наибольшего общего делителя

ВВОД: Целые числа $a > b \geqslant 0$.

ВЫВОД: $\gcd(a, b)$.

1. if $b = 0$ return (a).
2. return$(\gcd(b, a \bmod b))$.

В алгоритме 4.1 выражение "$a \bmod b$" обозначает остаток от деления числа a на число b. (В разделе 4.3.2.5 мы дадим формальное определение операции деления по модулю и приведем некоторые полезные сведения по модулярной арифметике.) Условие $a > b \geqslant 0$ приведено исключительно для простоты изложения. В конкретной реализации, если $|a| < |b|$, можно заменить числа a и b их абсолютными величинами и вызвать функцию $\gcd(|b|, |a|)$.

Перейдем к анализу алгоритма 4.1. Для положительных чисел $a \geqslant b$ всегда существует целое число $q \neq 0$ (частное от деления числа a на число b) и число $0 \leqslant r < b$ (остаток от деления числа a на число b), так что

$$a = bq + r. \tag{4.3.2}$$

По определению на число $\gcd(a, b)$ делятся и число a, и число b. Из формулы (4.3.2) следует, что число r также должно делиться на $\gcd(a, b)$. Следовательно, число $\gcd(a, b)$ равно $\gcd(b, r)$. Поскольку число r (остаток от деления числа a на число b) обозначается как $a \bmod b$, получаем, что

$$\gcd(a, b) = \gcd(b, a \bmod b).$$

Именно этот факт используется в алгоритме 4.1, в котором число $\gcd(a, b)$ рекурсивно определяется через число $\gcd(b, a \bmod b)$. Последовательность рекурсивных

вызовов функции gcd приводит к решению следующего ряда уравнений, каждое из которых имеет вид (4.3.2) и образовано путем деления двух входных чисел:

$$\begin{aligned} a &= bq_1 + r_1, \\ b &= r_1 q_2 + r_2, \\ r_1 &= r_2 q_3 + r_3, \\ &\vdots \\ r_{k-3} &= r_{k-2} q_{k-1} + r_{k-1}, \\ r_{k-2} &= r_{k-1} q_k + r_k, \end{aligned} \tag{4.3.3}$$

где $r_k = 0$ (условие окончания алгоритма), а $q_1, q_2, \ldots, q_k, r_1, r_2, \ldots, r_{k-1}$ — ненулевые целые числа. Условие $r_k = 0$ в последнем уравнении означает, что число r_{k-1} является делителем чисел $r_{k-2}, r_{k-3}, \ldots, r_1$ и в конце концов — чисел a и b. Ни один из других остатков в других уравнениях не обладает этим свойством (потому они и называются остатками, а не делителями). Итак, единственным делителем является число r_{k-1}. Следовательно, число r_{k-1} является наибольшим общим делителем чисел a и b, т.е. $r_{k-1} = \gcd(a, b)$.

Например, вызов функции $\gcd(108, 42)$ приведет к следующим рекурсивным вызовам:

$$\gcd(108, 42) = \gcd(42, 24) = \gcd(24, 18) = \gcd(18, 6) = \gcd(6, 0) = 6.$$

4.3.2.2 Обобщенный алгоритм Евклида

Алгоритм 4.1 не сохраняет промежуточные частные. Если бы он накапливал их в ходе вычислений наибольшего общего делителя, мы смогли бы вычислить кое-что еще. Попробуем определить, что же именно мы могли бы вычислить.

Первое уравнение в последовательности (4.3.3) можно переписать в виде

$$a + b(-q_1) = r_1.$$

Умножая обе части уравнения на q_2, получаем

$$aq_2 + b(-q_1 q_2) = r_1 q_2.$$

Используя это уравнение и второе уравнение из последовательности (4.3.3), имеем

$$a(-q_2) + b(1 + q_1 q_2) = r_2. \tag{4.3.4}$$

Продолжая аналогичные вычисления для $i = 1, 2, \ldots k$, приходим к уравнению

$$a\lambda_i + b\mu_i = r_i, \tag{4.3.5}$$

где числа λ_i и μ_i являются целыми и зависят от промежуточных остатков. Как мы убедились в разделе 4.3.2.1, продолжая вычисления, мы достигнем ситуации, когда $r_k = 0$. Следовательно,

$$a\lambda_{k-1} + b\mu_{k-1} = r_{k-1} = \gcd(a, b). \tag{4.3.6}$$

Алгоритм, на вход которого поступают числа a и b, а на выходе вычисляются числа λ_{k-1} и μ_{k-1}, удовлетворяющие условию (4.3.6), называется **обобщенным алгоритмом Евклида** (extended Euclid algorithm). Он будет широко использоваться в остальной части книги для деления целых чисел по модулю. А теперь уточним этот алгоритм, т.е. опишем общий метод накопления промежуточных частных.

Обратите внимание на уравнения (4.3.3) и на тот факт, что $r_{-1} = a, r_0 = b, \lambda_{-1} = 1, \mu_{-1} = 0, \lambda_0 = 0, \mu_0 = 1$. Для $i = 1, 2, \ldots, k-1$ из i-го уравнения в последовательности (4.3.3) следует, что

$$r_{i+1} = r_{i-1} - r_i q_{i+1}. \tag{4.3.7}$$

Заменяя числа r_{i-1} и r_i в правой части уравнения (4.3.7) с помощью уравнения (4.3.5), получаем следующее:

$$r_{i+1} = a(\lambda_{i-1} - q_{i+1}\lambda_i) + b(\mu_{i-1} - q_{i+1}\mu_i). \tag{4.3.8}$$

Сравнивая уравнения (4.3.8) и (4.3.5), получаем для $i = 0, 1, \ldots, k-1$ следующие соотношения.

$$\begin{aligned} \lambda_{i+1} &= \lambda_{i-1} - q_{i+1}\lambda_i, \\ \mu_{i+1} &= \mu_{i-1} - q_{i+1}\mu_i. \end{aligned} \tag{4.3.9}$$

Эти два уравнения лежат в основе общего метода, позволяющего накапливать промежуточные частные в ходе вычисления наибольшего общего делителя (алгоритм 4.2).

Примечание 4.1. *Для упрощения описания алгоритмов 4.1 и 4.2 мы пожертвовали эффективностью. В следующих двух разделах (4.3.2.3 и 4.3.2.4) мы проанализируем их временну́ю сложность и сравним полученные результаты с наилучшими оценками, достигнутыми при вычислении наибольшего общего делителя.* □

4.3.2.3 Временна́я сложность алгоритмов Евклида

Оценим временну́ю сложность обоих алгоритмов Евклида. Очевидно, что количество рекурсивных вызовов в алгоритме 4.1 равно количеству циклов в алгоритме 4.2, которое в свою очередь равно числу k — т.е. числу уравнений в последовательности (4.3.3).

Алгоритм 4.2. Обобщенный алгоритм Евклида

ВВОД: Целые числа $a > b \geqslant 0$.

ВЫВОД: Целые числа λ и μ, удовлетворяющие условию $a\lambda + b\mu = \gcd(a, b)$.

1. (*Инициализация*)
   $i \leftarrow 0$; $r_{-1} \leftarrow a$; $r_0 \leftarrow b$; $\lambda_{-1} \leftarrow 1$; $\mu_{-1} \leftarrow 0$; $\lambda_0 \leftarrow 0$; $\mu_0 \leftarrow 1$;.
2. (* Пока выполняется условие $a\lambda_i + b\mu_i = r_i$ *)
   while $(r_i = a\lambda_i + b\mu_i \neq 0)$ do.
   а) (* Символ $\div$ обозначает деление целых чисел *)
      $q \leftarrow r_{i-1} \div r_i$;
   б) (* Суммирование частных *)
      $\lambda_{i+1} \leftarrow \lambda_{i-1} - q\lambda_i$; $\mu_{i+1} \leftarrow \mu_{i-1} - q\mu_i$;
   в) $i \leftarrow i + 1$.
3. return $((\lambda_{i-1}, \mu_{i-1}))$.

Рассмотрим случай, когда $a > b$ и проанализируем уравнение (4.3.7) при $i = = 0, 1, \ldots k - 1$. Возможен один из двух вариантов:

$$|r_i| < |r_{i-1}|, \tag{4.3.10}$$

или

$$|r_{i+1}| < |r_{i-1}|. \tag{4.3.11}$$

Учитывая, что $r_{i+1} < r_i$, приходим к выводу, что условие (4.3.11) является следствием условия (4.3.10). Следовательно, условие (4.3.11) является инвариантным. Это означает, что максимальное значение k ограничено величиной $2|a|$. Если операция модулярной арифметики является элементарной и на нее затрачивается единица времени, то временна́я сложность алгоритма 4.1 ограничена величиной $2|a|$. Это выражение является линейной функцией, зависящей от аргумента a.

Теорема 4.1. *Наибольший общий делитель* $\gcd(a, b)$ *можно вычислить с помощью не более чем* $2\max(|a|, |b|)$ *операций модулярной арифметики. Следовательно, алгоритмы 4.1 и 4.2 завершатся по меньшей мере через* $2\max(|a|, |b|)$ *итераций.* □

Первую половину теоремы 4.1 впервые доказал Ж. Ламе (1795–1870) (G. Lamé). Ее можно считать первой теоремой в теории вычислительной сложности.

Последовательность уравнений (4.3.3) свидетельствует о *линейной* природе алгоритма Евклида. За все время его существования еще никому не удалось существенно сократить количество операций, необходимых для вычисления наибольшего общего делителя.

4.3.2.4 Две оценки вычислительной сложности

При оценке вычислительной сложности алгоритма часто невозможно или не обязательно уточнять постоянный коэффициент, входящий в выражение, ограничивающее меру сложности. O-**символика** (order notation) облегчает задачу оценки сложности.

Определение 4.2 (O-символика). *Символом* $O(f(n))$ *обозначается функция* $g(n)$, *для которой существует константа* $c > 0$ *и натуральное число* N, *такие что* $|g(n)| \leqslant c|f(n)|$ *для всех* $n \geqslant N$.

Используя O-символику, мы можем выразить временну́ю сложность алгоритмов 4.1 и 4.2 как $O(\log a)$. Обратите внимание на то, что в этом выражении мы заменили величину $|a|$ числом $\log a$, не указывая явно основание логарифма (поскольку по умолчанию предполагается, что основание натурального логарифма e не указывается). Читатель может самостоятельно убедиться, что в оценке сложности можно использовать любое основание $b > 1$ (упражнение 4.10).

До сих пор мы предполагали, что выполнение одной операции модулярной арифметики выполняется за одну единицу времени, т.е. ее временна́я сложность имеет порядок $O(1)$. На самом деле операция "$a \bmod b$" в общем случае использует деление $a \div b$, с помощью которой в алгоритме 4.2 вычисляются частные. Следовательно, временна́я сложность операции модульной арифметики должна зависеть от размеров двух операндов. С практической точки зрения (смысл которой описан в разделе 4.4.6) оценка сложности $O(1)$ для операции деления является слишком грубой.

Для оценки сложности **поразрядных** (bitwise) арифметических операций используется модифицированная O-символика. В поразрядных вычислениях все переменные принимают значения нуль или единица, а операции носят не арифметический, а логический характер: $\wedge$ (для операции AND), $\vee$ (для операции OR), $\ominus$ (для операции XOR, т.е. "исключающего или") и $\neg$ (для операции NOT).

Определение 4.3 (O-символика для поразрядных вычислений.). *При оценке сложности поразрядных вычислений вместо символа* $O()$ *используется символ* $O_B()$.

В рамках модели поразрядных вычислений на сложение и вычитание двух целых чисел i и j затрачиваются $\max(|i|, |j|)$ побитовых операций, т.е. порядок временно́й сложности равен $O(\max(|i|, |j|))$. Интуиция подсказывает, что на умножение и деление двух целых чисел i и j затрачиваются $|i| \cdot |j|$ побитовых операций, т.е. порядок их временно́й сложности равен $O(\log i \times \log j)$. Следует отметить, что при использовании метода быстрого преобразования Фурье (БПФ) порядок сложности умножения (и деления) можно снизить до $O_B(\log(i+j) \times \log\log(i+j))$. Однако эта оценка является лишь асимптотической и связана с намного

бóльшей константой (отображающей затраты на быстрое преобразование Фурье). При относительно небольших размерах операндов (которые часто используются в современной криптографии) быстрое преобразование Фурье может даже ухудшить исходную оценку сложности. По этой причине в книге при умножении и делении быстрое преобразование Фурье не применяется. Итак, в дальнейшем при оценке сложности умножения и деления мы будем использовать интуитивные рассуждения.

Оценим временну́ю сложность алгоритмов 4.1 и 4.2, используя более точные побитовые оценки $O_B()$. По теореме 4.1 при $a > b$ наибольший общий делитель $\gcd(a, b)$ можно вычислить за $O(\log a)$ единиц времени. Если оба входных числа ограничены величиной a, то операция деления по модулю имеет сложность $O_B((\log a)^2)$, а временна́я сложность алгоритмов 4.1 и 4.2 имеет порядок $O_B((\log a)^3)$.

Напомним примечание 4.1: мы описали наиболее простые, но не самые эффективные варианты алгоритмов. Как видим, наше упрощение оказалось слишком грубым!

Более тщательная реализация алгоритмов Евклида должна учитывать следующие обстоятельства.

1. Временна́я сложность операции деления, выполняемой при вычислении чисел $a = bq + r$, имеет порядок $O_B(\log a \times \log b)$.
2. Частные $q_1, q_2, \ldots, q_k$ в уравнениях (4.3.3) удовлетворяют условию

$$\sum_{i=1}^{k} \log q_i = \log \prod_{i=1}^{k} q_i \leqslant \log a. \tag{4.3.12}$$

Следовательно, более точная оценка временно́й сложности алгоритма вычисления наибольшего общего делителя имеет следующий вид:

$$\sum_{i=1}^{k} O_B\left(\log a \times \log q_i\right) \leqslant O_B\left((\log a)^2\right).$$

Эффективные реализации алгоритмов Евклида приведены в главе 1 книги [79].

В остальной части нашей книги при оценке сложности вычисления наибольшего общего делителя с помощью обоих алгоритмов Евклида мы будем использовать наиболее широко известную оценку $O_B((\log a)^2)$.

4.3.2.5 Модулярная арифметика

Одним из наиболее важных детерминированных полиномиальных алгоритмов, которые мы рассмотрим, является возведение в степень по модулю. Эта операция часто используется в криптографии с открытым ключом. Для начала кратко изложим основы модулярной арифметики (читатели, уже знакомые с модулярной арифметикой, могут пропустить этот раздел).

Определение 4.4 (Деление по модулю). *При заданных целых числах x и $n > 1$ операция "$x(\mathrm{mod}\, n)$" возвращает остаток от деления числа x на число n, т.е. неотрицательное целое число $r \in [0, n-1]$, удовлетворяющее условию*

$$x = kn + r$$

для некоторого целого числа k.

Теорема 4.2 (Свойства деления по модулю). *Пусть $x, y, n \neq 0$ — целые числа с $\gcd(y, n) = 1$. Операция деленця по модулю имеет следующие свойства:*

1. $(x + y)(\mathrm{mod}\, n) = [(x(\mathrm{mod}\, n)) + (y(\mathrm{mod}\, n))](\mathrm{mod}\, n)$;
2. $(-x)(\mathrm{mod}\, n) = (n - x)(\mathrm{mod}\, n) = n - (x(\mathrm{mod}\, n))$;
3. $(x \cdot y)(\mathrm{mod}\, n) = [(x(\mathrm{mod}\, n)) \times (y(\mathrm{mod}\, n))](\mathrm{mod}\, n)$;
4. *Обозначим через $y^{-1}(\mathrm{mod}\, n)$* ***величину, обратную к y по модулю n*** *(multiplicative inverse of y modulo n). Она является единственным числом из отрезка $[1, n-1]$, удовлетворяющим условию*

$$(y \cdot y^{-1})(\mathrm{mod}\, n) = 1.$$

Доказательство. Мы докажем только свойства 1 и 4. (Свойства 2 и 3 читатели могут доказать самостоятельно (упражнение 4.4).)

Запишем $x = kn + r, y = \ell n + s$, где $0 \leqslant r, s \leqslant n - 1$.

Для свойства 1 имеем следующее.

$$\begin{aligned}(x + y)(\mathrm{mod}\, n) &= [(kn + r) + (\ell n + s)](\mathrm{mod}\, n) = \\ &= [(k + \ell)n + (r + s)](\mathrm{mod}\, n) = \\ &= (r + s)(\mathrm{mod}\, n) = \\ &= [(x(\mathrm{mod}\, n)) + (y(\mathrm{mod}\, n))](\mathrm{mod}\, n).\end{aligned}$$

Для свойства 4, учитывая, что $\gcd(y, n) = 1$, и применяя обобщенный алгоритм Евклида 4.2 к исходным данным y и n, получаем целые числа λ и μ, удовлетворяющие условию

$$y\lambda + n\mu = 1. \tag{4.3.13}$$

Без потери общности можно утверждать, что $\lambda < n$, поскольку в противном случае можно просто заменить число λ величиной $\lambda(\mathrm{mod}\, n)$, а число μ — числом $yk + \mu$ при некотором k, не нарушая справедливости соотношения (4.3.13). По определению 4.4 $y\lambda \bmod n = 1$. Следовательно, $y^{-1} = \lambda < n$ является величиной, обратной к числу y по модулю n. Докажем единственность числа y^{-1} на отрезке $[1, n-1]$. Предположим, что существует еще одна величина, обратная к числу y по модулю n. Обозначим ее через $\lambda' \in [1, n-1], \lambda' \neq \lambda$. Тогда

$$y(\lambda' - \lambda)(\mathrm{mod}\, n) = 0,$$

т.е.

$$y(\lambda' - \lambda) = an, \tag{4.3.14}$$

где a — некоторое целое число. Известно, что $y = \ell n + 1$ при некотором целом числе ℓ. Следовательно, уравнение (4.3.14) принимает вид

$$(\ell n + 1)(\lambda' - \lambda) = an,$$

или

$$\lambda' - \lambda = bn,$$

где b — некоторое целое число. Это противоречит нашему предположению, что λ и λ' принадлежат отрезку $[1, n-1]$. Значит, $\lambda' \neq \lambda$. □

Как и при делении рациональных чисел, деление чисел по модулю n определяется как умножение на обратное число. Разумеется, для этого необходимо, чтобы обратное число существовало. Итак, для любого числа y, удовлетворяющего условию $\gcd(y, n) = 1$ выражение $xy^{-1} (\mathrm{mod}\, n)$ можно переписать в виде $x/y (\mathrm{mod}\, n)$.

Поскольку для вычисления величины y^{-1} применяется обобщенный алгоритм Евклида, его сложность имеет порядок $O_B((\log n)^2)$. Следовательно, временна́я сложность деления по модулю имеет порядок $O_B((\log n)^2)$.

Из теоремы 4.2 следует, что модулярная арифметика очень похожа на целочисленную. Легко показать, что операции сложения и умножения подчиняются следующим законам коммутативности и ассоциативности (символ "$\circ$" обозначает либо сложение, либо умножение).

$$a \circ b \bmod n = b \circ a \bmod n \qquad \text{(коммутативность)},$$

$$a \circ (b \circ c) \bmod n = (a \circ b) \circ c \bmod n \qquad \text{(ассоциативность)}.$$

В заключение заметим, что в определении операции деления по модулю $x \bmod n$ (см. определение 4.4) величина k (частное от деления x на n) не важна. Следовательно, в тождестве

$$x \bmod n = y \bmod n, \tag{4.3.15}$$

числа x и y могут отличаться на величину, кратную n. Следовательно, уравнение (4.3.15) можно переписать в виде

$$x \equiv y (\mathrm{mod}\, n)$$

или

$$x (\mathrm{mod}\, n) \equiv y.$$

Эта операция называется **сравнением** по модулю n (congruence modulo n), а числа x и y называются сравнимыми по модулю n.

4.3.2.6 Возведение в степень по модулю

При $x, y < n$ определение **числа x в степени y по модулю n** (modular exponentiation) $x^y (\mathrm{mod}\, n)$ совпадает с определением обычной степени целого числа, т.е. произведению числа x самого на себя y раз, но только по модулю n.

$$x^y \stackrel{\text{def}}{=} \underbrace{xx\ldots x}_{y} (\mathrm{mod}\, n).$$

Обозначим через $y \div 2$ частное от деления числа y на 2 с округлением до целого числа.

$$y \div 2 = \begin{cases} \frac{y}{2}, & \text{если } y \text{ — четное число,} \\ \frac{y-1}{2}, & \text{если } y \text{ — нечетное число.} \end{cases}$$

Применяя к умножению по модулю закон ассоциативности, получаем

$$x^y = \begin{cases} \left(x^2\right)^{y \div 2}, & \text{если } y \text{ — четное число,} \\ \left(x^2\right)^{y \div 2} x, & \text{если } y \text{ — нечетное число.} \end{cases}$$

Эта формула описывает популярный алгоритм возведения в степень по модулю, известный под названием "метода повторяющихся возведения в квадрат и умножения". Этот алгоритм сводится к следующему повторяющемуся процессу: разделить степень на 2, возвести в квадрат, и выполнить дополнительное умножение, если степень является нечетной. Рекурсивная версия этого алгоритма описана ниже.

Алгоритм 4.3. Возведение в степень по модулю

ВВОД: x, y, n: целые числа, $x > 0, y \geqslant 0, n > 1$.

ВЫВОД: $x^y (\mathrm{mod}\, n)$

mod_exp(x, y, n)

1. if $y = 0$ return(1);
2. if $y (\mathrm{mod}\, 2) = 0$ return(mod_exp $(x^2 (\mathrm{mod}\, n), y \div 2, n)$);
3. return$(x \cdot$ mod_exp $(x^2 (\mathrm{mod}\, n), y \div 2, n) (\mathrm{mod}\, n))$.

Обратите внимание на третий шаг алгоритма 4.3, являющийся результатом рекурсивной реализации: выполнение операции "return" означает, что дальнейшие шаги алгоритма выполнить невозможно, поскольку операция "return(значение)" возвращает процесс вычислений в точку вызова функции mod_exp. Следовательно, если выполняется шаг 2, то шаг 3 не будет реализован.

В качестве иллюстрации начнем алгоритм с вызова функции mod_exp(2, 21, 23).

$$\begin{aligned}
\text{mod_exp}(2,21,23) &= \\
&= 2 \cdot \text{mod_exp}(4(\equiv 2^2 (\bmod 23)), 10, 23) && \text{(шаг 3)} \\
&= 2 \cdot \text{mod_exp}(16(\equiv 4^2 (\bmod 23)), 5, 23) && \text{(шаг 2)} \\
&= 2 \cdot 16 \cdot \text{mod_exp}(3(\equiv 16^2 (\bmod 23)), 2, 23) && \text{(шаг 3)} \\
&= 2 \cdot 16 \cdot \text{mod_exp}(9(\equiv 3^2 (\bmod 23)), 1, 23) && \text{(шаг 2)} \\
&= 2 \cdot 16 \cdot 9 \cdot \text{mod_exp}(12(\equiv 9^2 (\bmod 23)), 0, 23) && \text{(шаг 3)} \\
&= 2 \cdot 16 \cdot 9 \cdot 1 && \text{(шаг 1)}
\end{aligned}$$

Обратите внимание на то, что эти шесть строк содержат пять рекурсивных вызовов функции mod_exp. Последняя строка "mod_exp(12, 0, 23)" означает "вернуть значение 1" и не является рекурсивным вызовом. Последнее значение, возвращаемое функцией mod_exp(2, 21, 23), равно 12 и может быть представлено в следующем виде.

$$12 \equiv 2 \cdot 16 \cdot 9 \equiv 2^1 \cdot \left(2^2\right)^2 \cdot \left(\left(\left(2^2\right)^2\right)^2\right)^2 (\bmod 23).$$

Оценим временну́ю сложность функции mod_exp, реализующей алгоритм 4.3. Поскольку при $y > 0$ операция "поделить на два" выполняется за $\lfloor \log_2 y \rfloor + 1$ единиц времени, прежде чем вычислит частное, равное нулю, вызов функции $\text{mod\_exp}(x, y, n)$ порождает ровно $\lfloor \log_2 y \rfloor + 1$ рекурсивных вызовов, прежде чем достигнет завершения алгоритма. Каждый рекурсивный вызов содержит возведение в квадрат или возведение в квадрат с последующим умножением, порядок сложности которого равен $O_B((\log x)^2)$. Если предположить, что x и y меньше n, временна́я сложность алгоритма 4.3 оценивается величиной порядка $O_B((\log n)^3)$.

Как и вычисление наибольшего общего делителя, возведение в степень по модулю носит последовательный характер. Это кажется очевидным вследствие повторяющихся операций возведения в квадрат: x^4 можно вычислить, только вычислив значение x^2. За все время существования этого алгоритма никому не удалось улучшить порядок сложности $O_B((\log n)^3)$ (без применения быстрого преобразования Фурье).

Оценки сложности основных операций в модулярной арифметике приведены на рис. 4.3. Следует отметить, что операция деления по модулю не сводится к обычному делению, поскольку $0 \leqslant a, b < n$, а значит, $-n < a \pm b < 2n$. Следовательно,

$$a \pm b (\bmod n) = \begin{cases} a \pm b, & \text{если } 0 \leqslant a \pm b < n, \\ a \pm b - n, & \text{если } a \pm b \geqslant n, \\ n + (a \pm b), & \text{если } a \pm b < 0. \end{cases}$$

Операция над $a, b \in_U [1, n)$	Сложность
$a \pm b \pmod n$	$O_B(\log n)$
$a \cdot b \pmod n$	$O_B((\log n)^2)$
$b^{-1} \pmod n$	$O_B((\log n)^2)$
$a/b \pmod n$	$O_B((\log n)^2)$
$a^b \pmod n$	$O_B((\log n)^3)$

Рис. 4.3. Поразрядные оценки сложности основных операций в модулярной арифметике

4.4 Вероятностное полиномиальное время

Общепризнанно, что если язык не принадлежит классу $\mathcal{P}$, то не существует машины Тьюринга, которая распознавала бы его *и всегда* была эффективной[2]. Однако существует класс языков, обладающих следующим свойством: их принадлежность к классу $\mathcal{P}$ не доказана, но они *всегда* эффективно распознаются определенной машиной Тьюринга, которая *иногда* делает ошибки.

Причина, по которой машина Тьюринга может иногда делать ошибки, заключается в том, что некоторые такты являются **случайными** (random move). Иногда случайные такты приводят к правильным результатам, а иногда — нет. Такая модель называется **недетерминированной машиной Тьюринга** (non-deterministic Turing machine). Подкласс задач принятия решений, который мы рассмотрим, обладает свойством, ограничивающим вероятность ошибки.

> Вероятность того, что недетерминированная машина Тьюринга сделает ошибку, решая задачу принятия решений, ограничена константой (вероятностным пространством является случайная лента машины).

Недетерминированную машину Тьюринга с ограниченной ошибкой мы будем называть **вероятностной машиной Тьюринга** (probabilistic Turing machine). По этой причине название “недетерминированная машина Тьюринга” на самом

[2]Точный смысл выражения “эффективная машина” будет определен в разделе 4.4.6. Пока будем считать, что эффективность означает быстродействие.

деле означает другой класс задач принятия решений, который будет рассмотрен в разделе 4.5.

Вероятностная машина Тьюринга имеет несколько лент. Одна из этих лент называется **случайной** (random tape) и содержит равномерно распределенные случайные символы. Сканируя исходное предложение I, машина взаимодействует со случайной лентой, считывает случайный символ, а затем работает как детерминированная машина Тьюринга. Случайная строка называется **случайным вводом** (random input) вероятностной машины Тьюринга. Из-за наличия случайного ввода распознавание исходного предложения I вероятностной машиной Тьюринга уже не является детерминированной функцией, а зависит от случайной величины. Иначе говоря, распознавание представляет собой функцию, зависящую от случайного машинного ввода. Эта случайная переменная определяет некую **вероятность ошибки** (error probability) при распознавании предложения I.

Класс языков, которые распознаются вероятностными машинами Тьюринга, называется **классом вероятностных полиномиальных языков** (PPT languages — probabilistic polynomial-time languages), и обозначается символами $\mathcal{PP}$.

Определение 4.5 (Класс $\mathcal{PP}$). *Символами $\mathcal{PP}$ обозначается класс, обладающий следующими свойствами. Говорят, что язык L принадлежит классу $\mathcal{PP}$, если существует вероятностная машина Тьюринга PM и полином $p(n)$, такие что машина PM распознает любое предложение $I \in L$ с определенной вероятностью ошибки, представляющей собой случайную переменную, зависящую от случайного такта машины PM, за время $T_{PM}(n) \leqslant p(n)$ для всех неотрицательных целых чисел n, где n — длина предложения I.*

В определении 4.5 остался один неясный момент: “машина PM распознает любое предложение $I \in L$ с определенной вероятностью ошибки”. Выражение “определенная вероятность ошибки” должно быть уточнено с помощью двух выражений, содержащих оценки условной вероятности.

$$\text{Prob}[PM \text{ распознает } I \in L | I \in L] \geqslant \varepsilon, \tag{4.4.1}$$

$$\text{Prob}[PM \text{ распознает } I \in L | I \notin L] \leqslant \delta, \tag{4.4.2}$$

где числа ε и δ — константы, удовлетворяющие условиям

$$\varepsilon \in \left(\frac{1}{2}, 1\right], \quad \delta \in \left[0, \frac{1}{2}\right). \tag{4.4.3}$$

Вероятностным пространством является случайная лента машины PM.

Выражение (4.4.1) представляет собой оценку вероятности правильного распознавания предложения. Она называется **оценкой полноты** (completeness probability bound).[3] Термин “полнота” означает, что любое предложение языка в конце

[3] Величину иногда называют чувствительностью алгоритма распознавания. Соответственно, величина $1 - \varepsilon$ представляет собой вероятность ошибки первого рода. — *Прим. ред.*

концов оказывается распознанным. Оценка этой вероятности снизу необходима для того, чтобы ограничить возможность ошибочно отклонить предложение. Более точное представление выражения (4.4.1) выглядит следующим образом.

$$\text{Prob}[PM \text{ приходит к выводу, что } I \notin L | I \in L] < 1 - \varepsilon. \qquad (4.4.4)$$

Число $1 - \varepsilon$ является вероятностью ошибочного отклонения предложения I. В таких случаях говорят, что полнота машины PM ограничена вероятностью ошибочно отклонить предложение I.

Выражение (4.4.2) представляет собой вероятность ошибочного распознавания предложения, не принадлежащего языку. Это выражение называется **оценкой непротиворечивости** (soundness probability). Термин "непротиворечивость" означает, что предложение, не принадлежащее языку, никогда не распознается. Необходимость ограничить вероятность такой ошибки очевидна. В таких случаях говорят, что непротиворечивость машины PM ограничена вероятностью ложного распознавания.[4]

4.4.1 Вероятность ошибки

Вероятность ошибки для машины PM ограничена двумя константами, ε и δ, принадлежащими интервалам, указанным в выражении (4.4.3). Покажем, что, хотя точное значение данных констант не определено, это не создает никаких проблем.

4.4.1.1 Характеристики полиномиального времени

Если n раз применить к исходному предложению I вероятностную машину Тьюринга PM, полнота которой ограничена величиной $\varepsilon \in \left(\frac{1}{2}, 1\right]$, а непротиворечивость — числом $\delta \in \left[0, \frac{1}{2}\right)$, то это повторение, обозначенное как $PM'(I, n)$, также окажется вероятностной машиной Тьюринга. В качестве критерия для распознавания или отклонения предложения I машиной $PM'(I, n)$ можно использовать "голосование большинством" ("majority election criterion"). Иначе говоря, если $\left\lfloor \frac{n}{2} \right\rfloor + 1$ раз запуск машины $PM(I)$ завершался распознаванием (отклонением), то машина $PM'(I, n)$ распознает (отклоняет) предложение I. Очевидно, что полнота и непротиворечивость машины $PM'(I, n)$ зависит от числа n. Покажем, что алгоритм $PM'(I, n)$ по-прежнему имеет полиномиальное время выполнения, зависящее от размера предложения I.

Поскольку случайные такты, выполняемые в ходе n запусков машины $PM(I)$, являются независимыми, каждый запуск машины $PM(I)$ может считаться испытанием Бернулли, у которого вероятность успеха равна ε(δ — для стойкости), а вероятность неудачи — $1 - \varepsilon$($1 - \delta$ — для непротиворечивости). Применяя биномиальное распределение (раздел 3.5.2), приходим к выводу, что при "голосовании

[4] Величина δ представляет собой вероятность ошибки второго рода. — *Прим. ред.*

большинством" вероятность ошибки для машины $PM'(I,n)$ равна вероятности того, что в n испытаниях Бернулли количество успехов равно или больше $\left\lfloor \frac{n}{2} \right\rfloor + 1$. Для полноты эта сумма равна

$$\varepsilon(n) = \text{Prob}\left[\xi_n \geqslant \left\lfloor \frac{n}{2} \right\rfloor + 1\right] = \sum_{i=\left\lfloor \frac{n}{2}+1 \right\rfloor + 1}^{n} b(i; n, \varepsilon). \tag{4.4.5}$$

Для непротиворечивости оценка имеет следующий вид:

$$\delta(n) = \text{Prob}\left[\eta_n \geqslant \left\lfloor \frac{n}{2} \right\rfloor + 1\right] = \sum_{j=\left\lfloor \frac{n}{2}+1 \right\rfloor + 1}^{n} b(j; n, \delta). \tag{4.4.6}$$

Эти два выражения представляют собой суммы соответствующих биномиальных вероятностей. Поскольку $\varepsilon > \frac{1}{2}$ и $\delta < \frac{1}{2}$, центральный член (см. раздел 3.5.2.1) первого распределения расположен в точке $(n+1)\varepsilon > \left\lfloor \frac{n}{2} \right\rfloor + 1$. (В этой точке биномиальный член достигает максимума.) Центральный член второго распределения находится в точке $(n+1)\delta > \left\lfloor \frac{n}{2} \right\rfloor + 1$.

Поведение этих сумм было рассмотрено в разделе 3.5.2.1. Сумма в выражении (4.4.6) представляет собой правый хвост биномиального распределения, поскольку $\left\lfloor \frac{n}{2} \right\rfloor + 1 > (n+1)\delta$. Применяя формулу (3.5.7) и полагая $r = \left\lfloor \frac{n+1}{2} \right\rfloor$ и $p = \delta$, получаем, что

$$\delta(n) < \frac{2(1-\delta)}{(1-2\delta)^2} \cdot \frac{1}{n+1}.$$

Поскольку число δ — константа,

$$\delta(n) \to 0 \text{ при } n \to \infty.$$

Читатели могут самостоятельно получить аналогичный результат для полноты:

$$\varepsilon(n) > 1 - \frac{c}{n},$$

где c — некая константа (упражнение 4.7).

Поскольку хвосты стремятся к нулю быстрее[5], чем $\frac{1}{n}$, положив $n = |I|$, приходим к выводу, что машина $PM'(I,n)$ работает $|I| \cdot poly(|I|)$ единиц времени, где $poly(|I|)$ — время работы машины PM, на вход которой поступает предложение I. Следовательно, машина PM' по-прежнему имеет полиномиальное время выполнения.

[5]Оценки (3.5.7) и (3.5.8) являются лишь оценками сверху. Реальная скорость убывания хвостов к нулю намного превышает $\frac{1}{n}$. Числовые иллюстрации приведены в примере 3.9. Это свойство будет в дальнейшем подтверждено изучением полноты и непротиворечивости протокола 18.4 в разделе 18.5.1.1.

4.4.1.2 Почему оценки не равны $\frac{1}{2}$?

Если бы выполнялось условие $\varepsilon = \delta = \frac{1}{2}$, то центральный член обоих распределений (4.4.5) и (4.4.6) находился бы в точке $\lfloor \frac{1}{2} \rfloor$. Легко проверить, что при нечетном числе n

$$\varepsilon(n) = \delta(n) = \frac{1}{2},$$

а при четном

$$\varepsilon(n) \approx \delta(n) \approx \frac{1}{2}.$$

Иначе говоря, величина $\varepsilon(n)$ никогда не увеличивается, а величина $\delta(n)$ — никогда не уменьшается — независимо от количества запусков машины $PM(I)$, они равны (или почти равны) $\frac{1}{2}$. Итак, машина $PM'(I)$, представляющая собой n независимых запусков машины $PM(I)$, не может прийти ни к какому решению, поскольку и полнота, и непротиворечивость этой машины равны $\frac{1}{2}$, т.е. в половине случаев машина распознает предложение, а половине случаев — нет. Если количество запусков не ограничено и машина $PM(I)$ не принимает никаких решений, машина $PM'(I)$ никогда не завершит работу и, следовательно, не может быть полиномиальным алгоритмом.

Итак, поскольку класс $\mathcal{PP}$ представляет собой класс языков, предложения которых распознаются за полиномиальное время с определенной вероятностью, необходимо потребовать, чтобы величины ε и δ отличались от $\frac{1}{2}$.

Однако следует отметить, что это требование является необходимым лишь для наиболее *общих вариантов* задач распознавания языков, принадлежащих классу $\mathcal{PP}$, который содержит подкласс задач с "ошибками, имеющими двусторонние оценки" (раздел 4.4.5). Если ошибка ограничена только с одной стороны (например, $\varepsilon = 1$ или $\delta = 0$, как в разделах 4.3 и 4.4.4), условие $\varepsilon \neq \frac{1}{2}$ и $\delta \neq \frac{1}{2}$ не обязательно, поскольку в односторонних алгоритмах не используется критерий "голосования большинством". Вместо него можно применить критерий "голосования меньшинством" ("minority election criterion"). Например, можно использовать критерий "единогласного голосования" ("unanimous election criterion"), в котором машина $PM'(I)$ распознает (отклоняет) предложение I, только если все n запусков машины $PM(I)$ завершились одинаково. В таких случаях $\varepsilon(n) \to 1$ или $\delta(n) \to 0$ при $n \to \infty$ с экспоненциальной скоростью при любых $\varepsilon, \delta \in (0, 1)$.

На практике возможны ситуации, когда $\varepsilon \leqslant \frac{1}{2}$ или $\delta \geqslant \frac{1}{2}$ (однако, как мы уже указывали, эти неравенства не обязаны выполняться одновременно). Для таких задач изменение критерия голосования (например, на голосование меньшинством) может намного увеличить или уменьшить ошибку распознавания. В разделе 18.5.1 мы рассмотрим протокол, имеющий вероятность распознавания, равную $\varepsilon = \frac{1}{2}$, и увеличим вероятность полноты, повторив его с применением критерия "голосования меньшинством".

Подклассы класса $\mathcal{PP}$

Класс $\mathcal{PP}$ содержит несколько подклассов, характеризующихся разной степенью полноты и непротиворечивости. Рассмотрим их поближе. Каждый из этих классов можно упростить с помощью некоего алгоритма. Аналогично детерминированной машине Тьюринга, моделирующей полиномиальный алгоритм, вероятностная машина Тьюринга имитирует **рандомизированный полиномиальный алгоритм** (randomized polynomial-time algorithm). Следовательно, примеры алгоритмов, рассмотренные во введении, не ограничены распознаванием языков.

4.4.2 Подкласс "всегда высокочувствительных и всегда точных" алгоритмов

Подкласс класса $\mathcal{PP}$ называется подклассом $\mathcal{ZPP}$ (сокращение от Zero-sided Probabilistic Polynomial Time — **полиномиальное вероятностное время с нулевой ошибкой**), если оценки вероятностей (4.4.1) и (4.4.2) обладают следующими свойствами. Для любого языка $L \in \mathcal{ZPP}$ существует рандомизированный алгоритм A, при котором для любого предложения I выполняются условия

$$\text{Prob}[A \text{ распознает } I \in L \mid I \in L] = 1,$$
$$\text{Prob}[A \text{ распознает } I \in L \mid I \notin L] = 0.$$

Это означает, что случайная операция в рандомизированном алгоритме никогда не приводит к ошибкам. На первый взгляд класс $\mathcal{ZPP}$ ничем не отличается от класса $\mathcal{P}$. Однако существует класс задач, которые можно решить с помощью как детерминированных, так и рандомизированных алгоритмов *за полиномиальное время*. Причем, хотя рандомизированный алгоритм не делает никаких ошибок, его быстродействие намного выше, чем у детерминированного алгоритма. В свое время мы рассмотрим такой пример.

4.4.2.1 Примеры алгоритмов с нулевой ошибкой

Некоторые рандомизированные алгоритмы настолько естественны, что их детерминированные аналоги никогда не применяются. Например, чтобы взвесить предмет с помощью весов, пользователь должен передвигать гирьки на рычаге противовеса случайным образом. В этом случае он определит вес намного быстрее, чем детерминированным способом. Другим примером такого алгоритма является поиск телефонного номера в записной книжке.

Очевидно, что случайная операция, предусмотренная в алгоритме 4.4, не порождает ошибок. Следовательно, это действительно рандомизированный алгоритм с нулевой ошибкой. Если записная книжка состоит из N страниц, алгоритм 4.4 выполнит поиск за $O(\log N)$ шагов. Следует отметить, что детерминированный алгоритм найдет телефонный номер в записной книжке за $O(N)$ шагов.

Алгоритм 4.4. Поиск телефонного номера в записной книжке ($\mathcal{ZPP}$-алгоритм)

ВВОД: *Name*: имя человека.

Book: записная книжка.

ВЫВОД: Искомый телефонный номер.

1. Повторять следующие операции, пока в записной книжке *Book* остается хотя бы одна страница.

   {

   а) Открыть записную книжку *Book* на произвольной странице.

   б) Если искомое имя *Name* расположено до этой страницы, *Book* ← Предыдущие_страницы(*Book*).

   в) Иначе *Book* ← Следующие страницы.

   }

2. return(искомый телефонный номер).

Причина, объясняющая такое высокое быстродействие алгоритма 4.4, заключается в том, что все имена в записной книжке упорядочены по алфавиту. Заметим, что сама по себе сортировка также является $\mathcal{ZPP}$-алгоритмом: например, алгоритм быстрой сортировки [9] является рандомизированным. Он может упорядочить N элементов за $N \log N$ шагов, причем случайные операции не порождают никаких ошибок. В противоположность ему алгоритм сортировки методом пузырька является детерминированным. Он упорядочивает N элементов за N^2 шагов [9].

Можно сказать, что подкласс $\mathcal{ZPP}$ содержит языки, которые можно распознать с помощью "всегда высокочувствительных и всегда точных" рандомизированных алгоритмов ("always fast and always correct" randomized algorithms).

4.4.3 Подкласс "всегда высокочувствительных и, вероятно, точных" алгоритмов

Подкласс класса $\mathcal{PP}$ под названием $\mathcal{PP}$(Монте-Карло) обладает следующими характеристиками. Для любого языка $L \in \mathcal{PP}$(Монте-Карло) существует рандомизированный алгоритм A, при котором для любого предложения I

$$\text{Prob}[A \text{ распознает } I \in L \mid I \in L] = 1,$$
$$\text{Prob}[A \text{ распознает } I \in L \mid I \notin L] \leqslant \delta,$$

где δ — произвольная константа из интервала $\left(0, \frac{1}{2}\right)$. (Здесь термин "Монте-Карло" используется в качестве синонима слова "рандомизированный".) Однако, как ука-

зано в разделе 4.4.1.2, поскольку алгоритмы с односторонними оценками ошибок не используют критерий "голосования большинством" для уменьшения вероятности ложного распознавания, число δ на самом деле является произвольной константой из интервала $(0, 1)$.

Обратите внимание на то, что $\delta \neq 0$. В противном случае рассматриваемый подкласс вырождается в частный случай подкласса $\mathcal{ZPP}$. Рандомизированные алгоритмы с ограниченными вероятностями ошибок характеризуются **односторонней оценкой** вероятности ложного распознавания (one-sided error). Иначе говоря, такой алгоритм может распознать предложение, не относящееся к языку. Однако, если исходное предложение действительно принадлежит языку, оно всегда распознается. Этот подкласс алгоритмов называется **алгоритмами Монте-Карло** (Monte Carlo algorithms).

Как указано в разделе 4.4.1, повторяя метод Монте-Карло, можно сделать вероятность ошибки сколь угодно малой, причем итерационный алгоритм остается полиномиальным. Таким образом, можно утверждать, что алгоритм Монте-Карло всегда является высокочувствительным и, вероятно, точным ("always fast and probably correct").

Покажем теперь, что язык PRIMES (множество всех простых чисел) принадлежит подклассу $\mathcal{PP}$(Монте-Карло).

4.4.3.1 Пример алгоритма Монте-Карло

Со времен Ферма известно, что если число p является простым, а число x — взаимно простым с числом p, то $x^{p-1} \equiv 1(\mathrm{mod}\, p)$. Этот факт образует основу следующего метода Монте-Карло для проверки простоты числа [282]. Этот метод сводится к извлечению числа $x \in_U (1, p-1]$ с $\gcd(x, p) = 1$ и проверке

$$x^{(p-1)/2} \stackrel{?}{\equiv} \pm 1(\mathrm{mod}\, p). \tag{4.4.7}$$

Проверка повторяется $k = \log_2 p$ раз и завершается, когда $x^{(p-1)/2}$ равно -1.

В соответствии с **малой теоремой Ферма** (теорема 6.10 в разделе 6.4), если число p является простым, то для всех чисел $x < p$ выполняется следующее тождество:

$$x^{p-1} \equiv 1(\mathrm{mod}\, p). \tag{4.4.8}$$

Итак, если число p является простым, то функция Prime_Test(p) всегда возвращает ответ "ДА", т.е. выполняется равенство

$$\mathrm{Prob}\left[x^{(p-1)/2} \equiv \pm 1(\mathrm{mod}\, p) \mid p - \text{простое число}\right] = 1.$$

С другой стороны, если число p является составным, то равенство (4.4.7), как правило, не будет выполняться. Как известно из теории групп (пример 5.2.3 и теорема 5.1 в разделе 5.2.1), если равенство (4.4.7) нарушается для какого-

Алгоритм 4.5. Вероятностная проверка простоты (алгоритм Монте-Карло)

ВВОД: p: положительное целое число.

ВЫВОД: ДА, если число p является простым, и НЕТ — в противном случае.

Prime_Test(p)

1. Повторять $\log_2 p$ раз.

   а) $x \in_U (1, p-1]$;

   б) if $\gcd(x,p) > 1$ или $x^{(p-1)/2} \not\equiv \pm 1 (\mathrm{mod}\, p)$ return(НЕТ);

   end_of_repeat;

2. if (число в пункте 1.2 никогда не равно -1) return(ДА).

3. return(ДА).

нибудь числа $x < p$, для которого $\gcd(x,p) = 1$, то оно нарушается по крайней мере для половины чисел, удовлетворяющих неравенству $x < p$. Следовательно, для $x \in_U (1, p-1]$, такого что $\gcd(x,p) = 1$, выполняется неравенство:

$$\mathrm{Prob}\left[x^{(p-1)/2} \equiv \pm 1 (\mathrm{mod}\, p) \parallel p - \text{составное число}\right] \leqslant 1/2. \qquad (4.4.9)$$

Таким образом, если из интервала $(1, p-1]$ k раз случайным образом извлекаются числа x, которые успешно проходят проверку (напомним, что в результате проверки хотя бы один раз вычисляется число -1), то вероятность того, что число p не является простым, не превышает 2^{-k}. Здесь используется критерий "единогласного голосования": число p отклоняется, если среди $\log_2 p$ проверок хотя бы одна оказалась неудачной. Обратите внимание на то, что критерий "единогласного голосования" отличается от критерия "голосования большинством", изученного нами в разделе 4.4.1 для задач с двусторонними оценками ошибок. Критерий "единогласного голосования" позволяет сделать вероятность противоречивости произвольно малой намного быстрее, чем критерий "голосования большинством".

Поскольку $k = \log_2 p$, для любого входного числа p выполняется неравенство

$$\mathrm{Prob}\,[\mathrm{Prime\_Test}(p) = \text{ДА} \parallel p - \text{составное число}] \leqslant 2^{-\log_2 p}.$$

В разделе 4.3 показано, что оценка сложности возведения в степень и вычисления наибольшего общего делителя для входного числа, состоящего из $\log_2 p$ бит, имеет порядок $O_B((\log_2 p)^3)$. Следовательно, временна́я сложность алгоритма Prime_Test(p) имеет порядок $O_B((\log p)^4)$.

Итак, мы показали, что язык PRIMES (множество всех простых чисел) принадлежит классу $\mathcal{PP}$(Монте-Карло).

Несмотря на то что этот факт не был опровергнут, в августе 2002 года три индийских специалиста в области компьютерных наук, Аггравал (Agrawal), Кайал

(Kayal) и Саена (Saena), изобрели *детерминированный* полиномальный алгоритм для проверки простоты числа [8]. Следовательно, фактически язык PRIMES принадлежит классу $\mathcal{P}$.

4.4.4 Подкласс "вероятно, высокочувствительных и всегда точных" алгоритмов

Подкласс $\mathcal{PP}$(Лас-Вегас) класса $\mathcal{PP}$ обладает следующими свойствами. Для любого языка $L \in \mathcal{PP}$(Лас-Вегас) существует рандомизированный алгоритм A, такой что

$$\text{Prob}[A \text{ распознает } I \in L \parallel I \in L] \geqslant \varepsilon,$$
$$\text{Prob}[A \text{ распознает } I \in L \parallel I \notin L] = 0,$$

где ε — произвольная константа из интервала $\left(\frac{1}{2}, 1\right)$. На самом деле константа ε может принадлежать всему интервалу $(0, 1)$, поскольку, как и в случае односторонних оценок стойкости (раздел 4.4.3), в алгоритмах из класса $\mathcal{PP}$(Лас-Вегас) нет необходимости применять критерий "голосования большинством".

Следует также отметить, что $\varepsilon \neq 1$. В противном случае подкласс $\mathcal{PP}$(Лас-Вегас) вырождается в частный случай подкласса $\mathcal{ZPP}$. Такие рандомизированные алгоритмы имеют одностороннюю оценку полноты. Иными словами, алгоритмы подкласса $\mathcal{PP}$(Лас-Вегас) могут ошибочно отклонить предложение, которое на самом деле принадлежит языку. Однако если предложение распознано, ошибка исключена. Такие алгоритмы называются **алгоритмами Лас-Вегаса** (Las Vegas algorithms). Термин "Лас-Вегас" был впервые предложен в работе [16] и относится к алгоритмам, которые либо дают правильный ответ, либо не дают ответа вовсе.

Анализ, проведенный в разделе 4.4.1.1, показывает, что вероятность положительного ответа при использовании алгоритма Лас-Вегаса можно сделать сколь угодно близкой к единице, повторяя его применение. При этом алгоритм сохраняет полиномиальное время выполнения, а повторения должны быть независимыми друг от друга. В то время как алгоритмы Монте-Карло являются всегда высокочувствительными и не всегда точными, алгоритмы Лас-Вегаса являются, вероятно, высокочувствительными и всегда безошибочными ("probably fast and always correct").

С точки зрения вероятностей ошибок классы $\mathcal{ZPP}$, $\mathcal{PP}$(Монте-Карло) и $\mathcal{PP}$(Лас-Вегас) связаны между собой следующим соотношением.

$$\mathcal{ZPP} = \mathcal{PP}(\text{Монте-Карло}) \cap \mathcal{PP}(\text{Лас-Вегас}).$$

4.4.4.1 Пример алгоритма Лас-Вегаса

Пусть p — нечетное положительное целое число, а $p - 1 = q_1 q_2 \ldots q_k$ — полное разложение на простые множители числа $p - 1$ (некоторые простые множители

могут повторяться). В главе 5 будет доказан следующий факт: (теорема 5.12 из раздела 5.4.4): число p является простым тогда и только тогда, когда существует положительное целое число $g \in [2, p-1]$, такое что

$$\begin{aligned} g^{p-1} &\equiv 1 \bmod p, \\ g^{(p-1)/q_i} &\not\equiv 1 \bmod p \text{ для } i = 1, 2, \ldots, k. \end{aligned} \tag{4.4.10}$$

Этот факт позволяет сформулировать алгоритм доказательства простоты числа. Введем нечетное число p, разложим число $p-1$ на простые множители, а затем попытаемся найти число g, удовлетворяющее условию (4.4.10). Если такое число найдено, алгоритм выводит сообщение "ДА", и число p является простым. В противном случае алгоритм не принимает никакого решения, т.е. неизвестно, является ли число p простым или нет.

Алгоритм 4.6. Доказательство простоты числа (алгоритм Лас-Вегаса)

ВВОД: p: нечетное положительное целое число;
$q_1, q_2, \ldots, q_k$: все простые множители числа $p-1$.

ВЫВОД: ДА, если число p является простым,
и НЕПРИНЯТИЕ_РЕШЕНИЯ — в противном случае.

1. Извлекаем число $g \in_U [2, p-1]$;
2. for $(i = 1, i++, k)$ do
   if $g^{(p-1)/q_i} \equiv 1 \bmod p$ output НЕПРИНЯТИЕ_РЕШЕНИЯ и terminate;
3. if $g^{p-1} \not\equiv 1 (\bmod p)$ output НЕТ и terminate;
4. output ДА и terminate.

Поскольку $k \leqslant \log_2(p-1)$, алгоритм 4.6 имеет полиномиальное время выполнения, зависящее от размера числа p.

Из теоремы 5.12 (раздел 5.4.4) следует, что если алгоритм 4.6 выводит результат "ДА", то входное целое число p обязательно является простым — ошибка исключена. Если же алгоритм выводит ответ "НЕТ", ответ также является правильным, поскольку в противном случае не выполнялась бы малая теорема Ферма (4.4.8). Эти два свойства означают безошибочный алгоритм. Именно по этой причине он называется не "проверкой", а "доказательством" простоты числа.

Однако, если алгоритм 4.6 выводит ответ "НЕПРИНЯТИЕ_РЕШЕНИЯ", остается неизвестным, является ли число p простым или нет. Если на самом деле число p является простым, значит, алгоритм извлек неверное случайное число g. После изучения теоремы 5.12 из раздела 5.4.4 мы узнаем, что неверное число g не является первообразным корнем (primitive root).

Итак, мы показали, что полнота алгоритма 4.6 имеет одностороннюю оценку, т.е. он относится к подклассу алгоритмов Лас-Вегаса. Этот алгоритм можно модифицировать так, чтобы он не отказывался принимать решения, а искал другое случайное число g. Модифицированный алгоритм остается алгоритмом Лас-Вегаса и становится "вероятно, высокочувствительным", поскольку не исключено, что он будет постоянно извлекать тестовое число, не являющееся первообразным корнем. К счастью, для любого нечетного простого числа p мультипликативная группа по модулю p (определенная в главе 5) содержит большое количество первообразных корней, так что существует ненулевая вероятность, что при простом случайном выборе алгоритм извлечет из группы по модулю p искомое тестовое число. (В главе 5 будет дана оценка доли первообразных корней в мультипликативной группе по простому модулю.)

Алгоритмы Лас-Вегаса и Монте-Карло называются **рандомизированными алгоритмами с односторонней ошибкой** (randomized algorithms with one-sided error). Алгоритмы этого класса (напомним, что он включает в себя класс $\mathcal{ZPP}$) действительно являются эффективными. Несмотря на то что они являются недетерминированными, их временна́я сложность аналогична сложности алгоритмов из класса $\mathcal{P}$.

4.4.4.2 Еще один пример алгоритма Лас-Вегаса: квантовая факторизация

Квантовый компьютер может разложить целое число на простые множители за время, которое полиномиально зависит от размера входного числа (т.е. FACTORIZATION $\in \mathcal{QP}$). Этот алгоритм был предложен Шором (Shor) в работе [267] (см. также [300]). Покажем, что квантовая факторизация Шора является алгоритмом Лас-Вегаса.

Для того чтобы разложить на простые множители целое число N, из генеральной совокупности извлекается случайное целое число a. Используя идею Саймона (Simon) об определении периода квантового состояния путем случайного выбора преобразования Фурье [276], квантовый алгоритм может найти период функции $f(x) = a^x (\mathrm{mod}\, N)$, т.е. наименьшее положительное целое число r, удовлетворяющее условию $f(r) = 1$. В главе 6 мы покажем, что для составного числа N довольно большое количество целых чисел a, удовлетворяющих условию $gcd(a, N) = 1$, имеет четный период (называемый мультипликативным порядком элемента a), т.е. r является четным числом.

Если период r оказался четным, то из неравенства $a^{r/2} \neq \pm 1 (\mathrm{mod}\, N)$ следует, что число $a^{r/2} (\mathrm{mod}\, N)$ является нетривиальным квадратным корнем единицы по модулю N. В разделе 6.6.2 (теорема 6.17) мы покажем, что число $\gcd(a^{r/2} \pm 1, N)$ является нетривиальным множителем числа N, т.е. алгоритм успешно разложил число N на простые множители.

Если число r оказался нечетным или $a^{r/2} = \pm 1 (\mathrm{mod}\, N)$, то $\gcd(a^{r/2} \pm 1, N)$ является тривиальным множителем числа N, т.е. единицей или числом N. Следовательно, алгоритм не дает ответа на поставленный вопрос. Однако для случайно выбранного целого числа $a < N$ вероятность обнаружить неравенство $a^{r/2} \neq \neq \pm 1 (\mathrm{mod}\, N)$ ограничена снизу константой $\varepsilon > 1/2$. Значит, процедуру можно повторить, используя другое случайное число a. Анализируя алгоритм Шора (Shor), как показано в разделе 4.4.1.1, приходим к выводу, что он имеет полиномиальное время выполнения.

4.4.5 Подкласс "вероятно, высокочувствительных и, вероятно, безошибочных" алгоритмов

Алгоритм относится к подклассу $\mathcal{BPP}$ (сокращение от названия "**B**ounded error probability **P**robabilistic **P**olinomial Time" — "вероятностные полиномиальные алгоритмы с ограниченной вероятностью ошибки") класса $\mathcal{PP}$, если оценки (4.4.1) и (4.4.2) обладают следующими свойствами:

$$\varepsilon \in \left[\frac{1}{2} + \alpha, 1\right) \quad \text{и} \quad \delta \in \left(0, \frac{1}{2} - \beta\right], \tag{4.4.11}$$

где $\alpha > 0$ и $\beta > 0$. Следует обратить внимание на две особенности этих ограничений.

1. $\varepsilon \neq 1$ и $\delta \neq 0$. В противном случае подкласс $\mathcal{BPP}$ вырождается в один из трех частных случаев: $\mathcal{ZPP}$, $\mathcal{PP}$(Монте-Карло) и $\mathcal{PP}$(Лас-Вегас). Условия $\varepsilon \neq 1$ и $\delta \neq 0$ гарантируют, что алгоритмы из подкласса $\mathcal{BPP}$ имеют **двусторонние ошибки** (two-sided errors): могут ошибочно отвергать предложение (неполнота) или ошибочно распознавать предложение (противоречивость).
2. $\alpha > 0$ и/или $\beta > 0$. Эти условия означают, что вероятности ошибок алгоритмов из подкласса $\mathcal{BPP}$ *четко* отделены от числа $\frac{1}{2}$. В разделе 4.4.1 было показано, что если $\varepsilon \neq \frac{1}{2}$ и $\delta \neq \frac{1}{2}$, то повторное применение алгоритма с использованием критерия "голосования большинством" может увеличить полноту (и уменьшить непротиворечивость). Если $\varepsilon = \frac{1}{2}$ или $\delta = \frac{1}{2}$, то критерий "голосования большинством" не работает, поскольку в этом случае большинства голосов за распознавание или отклонение не существует. Однако в этих ситуациях можно по-прежнему применять критерий "единогласного голосования" (пример такого критерия будет рассмотрен в разделе 18.5.1). В заключение заметим, что если $\varepsilon = \frac{1}{2}$ и $\delta = \frac{1}{2}$, то ни один критерий голосования не работает, и задача не принадлежит классу $\mathcal{PP}$ (т.е. не может быть решена недетерминированной машиной Тьюринга независимо от времени работы машины).

Поскольку, кроме Монте-Карло и Лас-Вегаса, в мире существует еще один знаменитый центр азартных игр — Атлантик-Сити, соблазняющий людей увеличивать количество ставок, чтобы увеличить вероятность выигрыша, рандомизированные алгоритмы с двусторонними ошибками называются **алгоритмами Атлантик-Сити** (Atlantic City algorithms). Рассмотрим пример такого алгоритма.

4.4.5.1 Пример алгоритма Атлантик-Сити

В **квантовой криптографии** (quantum cryptography) существует знаменитый протокол, называемый **протоколом распределения квантовых ключей** (quantum key distribution protocol) — протокол QKD [31]. Этот протокол позволяет согласовать битовую строку между двумя общающимися сторонами, не встречаясь друг с другом, причем обе стороны могут быть глубоко уверены в том, что согласованная битовая строка известна только им. Протокол QKD является рандомизированным алгоритмом с двусторонней ошибкой. Рассмотрим этот алгоритм и проанализируем его ошибки.

Сначала опишем физические принципы, на которых основан протокол QKD. Распределение секретной битовой строки в рамках протокола QKD достигается за счет того, что отправитель (Алиса) передает строку фотонов, поляризованных в четырех направлениях. Каждый из этих фотонов пребывает в определенном состоянии, которое обозначается одним из четырех символов:

$$-, \quad |, \quad /, \quad \backslash.$$

Первые два состояния фотона излучаются поляризатором, имеющим прямолинейную ориентацию, а последние два — поляризатором, имеющим диагональную ориентацию. Обозначим эти два вида ориентации символами $+$ и $\times$ соответственно. Теперь мы можем шифровать информацию с помощью указанных четырех состояний фотонов. Схема кодирования приведена ниже.

$$+(0) = -, \quad +(1) = |, \quad \times(0) = /, \quad \times(1) = \backslash. \tag{4.4.12}$$

Эта схема кодирования является открытой. Если Алиса хочет передать бит 0 (соответственно 1), она может выбрать ориентацию $+$ и переслать бит по квантовому каналу — (соответственно $|$) или выбрать ориентацию $\times$ и переслать бит по квантовому каналу $/$ (соответственно $\backslash$). Выбор ориентации поляризатора для каждого последовательного бита, передаваемого в рамках протокола QKD, должен осуществляться случайным образом.

Чтобы распознать состояние фотона, получатель (адресат Боб или перехватчик Ева) должен использовать устройство, называемое *приемником фотонов* (photon observer), которое имеет либо прямолинейную, либо диагональную ориентацию. Эти виды ориентации также обозначаются символами $+$ и $\times$ соответственно. Обозначим через $\overset{+}{\Rightarrow}$ и $\overset{\times}{\Rightarrow}$ два приемника фотонов, имеющих прямолинейную или

диагональную ориентацию, и будем считать, что фотоны передаются слева направо. Приемник фотонов подчиняется следующим правилам.

Правильные наблюдения (состояния не изменяются)

$$- \overset{+}{\Rightarrow} -, \quad | \overset{+}{\Rightarrow} |, \quad / \overset{+}{\Rightarrow} /, \quad \backslash \overset{+}{\Rightarrow} \backslash$$

Неправильные наблюдения (состояния изменяются)

$$/ \overset{+}{\Rightarrow} \begin{cases} - & \text{вероятность } \frac{1}{2}, \\ | & \text{вероятность } \frac{1}{2}, \end{cases} \qquad \backslash \overset{\times}{\Rightarrow} \begin{cases} - & \text{вероятность } \frac{1}{2}, \\ | & \text{вероятность } \frac{1}{2}, \end{cases}$$

$$- \overset{\times}{\Rightarrow} \begin{cases} / & \text{вероятность } \frac{1}{2}, \\ \backslash & \text{вероятность } \frac{1}{2}, \end{cases} \qquad | \overset{\times}{\Rightarrow} \begin{cases} / & \text{вероятность } \frac{1}{2}, \\ \backslash & \text{вероятность } \frac{1}{2}. \end{cases}$$

Эти правила имеют следующую интерпретацию. Прямолинейно ориентированные состояния правильно распознаются приемником, имеющим прямолинейную ориентацию. Аналогично диагонально ориентированные состояния правильно распознаются приемником, имеющим диагональную ориентацию. Однако, если диагонально (прямоугольно) ориентированный приемник получает фотон, пребывающий в прямоугольно (диагонально) ориентированном состоянии, то с вероятностью $\frac{1}{2}$ происходит переориентация фотона на $\pm 45^\circ$. Эти наблюдения считаются неправильными и представляют собой неизбежный результат принципа неопределенности Гейзенберга, лежащего в основе протокола QKD.

Итак, если ориентация приемника совпадает с ориентацией поляризатора Алисы, то состояние фотона будет распознано правильно. Открытая схема кодирования битов (4.4.12) представляет собой взаимно-однозначное соответствие между множеством битов и множеством состояний фотонов. Следовательно, бит, посланный Алисой, будет расшифрован правильно. С другой стороны, если ориентации приемника и поляризатора не совпадают, состояние фотона будет изменено, хотя получатель не имеет никакой информации о том, что на самом деле посланный фотон был уничтожен.

Теперь мы готовы дать формальное описание протокола QKD.

Этапы 1 и 2 довольно просты. Алиса посылает Бобу m случайных состояний фотонов, используя случайные настройки $p_1, p_2, \ldots, p_m \in_U \{+, \times\}$ (п. 1), а Боб должен принять их, используя случайные настройки приемника $o_1, o_2, \ldots, o_m \in_U \{+, \times\}$ (п. 2). Алиса зашифровывает и передает m последовательных битов $a_1, a_2, \ldots, a_m$, а Боб получает и расшифровывает биты $b_1, b_2, \ldots, b_m$.

На третьем этапе Алиса и Боб обмениваются информацией через открытый канал связи для того, чтобы сравнить $k = m/10$ элементов из m пар $\{(p_i, o_i)\}_{i=1}^m$. Если среди этого множества существуют k пар, состоящих из одинаковых элементов, процесс можно продолжать. В противном случае протокол прекращается

Протокол 4.1. Распределение квантовых ключей (алгоритм Атлантик-Сити)

Схематичное описание протокола

Квантовый канал. Алиса посылает Бобу m фотонов, имеющих один из следующих видов ориентации $\{-, \quad |, \quad /, \quad \backslash\}$.

Условный канал, открытые согласования. Выбираются $k = m/10$ "просеянных битов", переданных с поляризатора Алисы на приемник Боба. Затем выполняется сравнение ℓ $(\ell < k)$ случайных тестовых битов, выбранных из k просеянных битов, чтобы обнаружить перехват. Если перехват не обнаружен, принимаются остальные $k - \ell$ секретных битов.

1. Алиса генерирует m случайных битов $a_1, a_2, \ldots, a_m \in_U \{0, 1\}$, устанавливает m случайно ориентированных поляризаторов $p_1, p_2, \ldots, p_m \in_U \{+, \times\}$ и посылает Бобу m фотонных состояний $p_1(a_1), p_2(a_2), \ldots, p_m(a_m)$ в соответствии со схемой кодировки битов (4.4.12).

2. Боб устанавливает m случайно ориентированных приемников фотонов $o_1, o_2, \ldots, o_m \in_U \{+, \times\}$ и применяет их для получения m фотонных состояний. Затем, используя схему кодировки битов (4.4.12), Боб расшифровывает последовательные биты $b_1, b_2, \ldots, b_m$, и сообщает Алисе: "Все биты получены!".

3. Алиса и Боб в открытом режиме сравнивают свои установки $(p_1, o_1), (p_2, o_2), \ldots, (p_m, o_m)$. Если больше $k = m/10$ пар установок согласованы между собой, выполняется условие

$$p_i = o_i \quad \text{для } 1 \leqslant i \leqslant k.$$

Алиса и Боб выполняют следующие шаги, в противном случае выполнение протокола прекращается. (* Ошибка вследствие неполноты *).

4. (* Теперь множество $\{(a_i, b_i)\}_{i=1}^{k}$ содержит k пар просеянных битов, распределенных в соответствии с согласованными установками поляризатора и приемника. *) Алиса и Боб открыто сравнивают ℓ пар, принадлежащих множеству $\{(a_i, b_i)\}_{i=1}^{k}$. Сравниваемые биты называются *тестовыми*. Если хотя бы одна пара тестовых битов не совпадает, выдается сообщение "Выявлен перехват!", и выполнение алгоритма прекращается.

5. Оставшиеся $k - \ell$ битов выводятся в виде распределенного секретного ключа. Протокол успешно завершен. (* Правда, может возникнуть ошибка, связанная с противоречивостью протокола. *)

вследствие неполноты. В свое время мы покажем, как оценивается вероятность неполноты протокола.

Допустим, что ошибка, связанная с неполнотой протокола, не обнаружена. В этом случае обе стороны переходят к выполнению четвертого этапа. Теперь у них есть k просеянных битов, распределенных по k устройствам, согласованным между собой. Не ограничивая общности, можно считать, что биты, переданные Алисой, обозначены как $a_1, a_2, \ldots, a_k$, а биты, принятые Бобом, — как $b_1, b_2, \ldots, b_k$. Теперь Алиса и Боб переходят к открытому обсуждению по условленному каналу, сравнивая ℓ случайных пар просеянных битов. Любое несовпадение свидетельствует о присутствии перехватчика Евы. Если признаки присутствия Евы не обнаружены, протокол успешно завершается на пятом этапе. Теперь Алиса и Боб обмениваются оставшимися $k - \ell$ битами, которые они считают неперехваченными. Однако в результате ошибки, связанной со стойкостью, оставшиеся биты могут быть не распознаны. Попробуем оценить вероятность этой ошибки.

Вероятность ошибки, связанной с противоречивостью

Допустим, что Ева прослушивает квантовый канал. Для того чтобы получить фотоны, посланные Алисой, Ева должна настроить свои приемники точно так же, как Боб. Для этого Ева случайным образом выбирает ориентацию m своих приемников, а затем пересылает m состояний фотонов Бобу. Вследствие принципа неопределенности Гейзенберга неверные настройки приемника Евы уничтожат фотоны, посланные Алисой. Поскольку Ева не знает правильных настроек, ей неизвестно, что именно она переслала Бобу. Одна из стратегий перехвата заключается в пересылке совершенно нового набора из m состояний, сгенерированных случайным образом (как это делает Алиса), в надежде на то, что Боб не сможет отличить биты, посланные Алисой, от битов, посланных Евой. Другая стратегия перехвата состоит в пересылке всех полученных от Алисы фотонов в надежде, что их состояние не изменилось. На самом деле эти стратегии никак не влияют на вероятность непротиворечивости протокола.

Рассмотрим вторую стратегию (первая стратегия приводит к тем же самым оценками вероятности непротиворечивости — см. упражнение 4.9). Если Ева правильно настроила свой приемник e_i, т.е. $e_i = p_i$, то она правильно распознает состояние $p_i(a_i)$ и, следовательно, бит a_i. Это означает, что Боб также получит правильный бит. Таким образом, в этом случае присутствие Евы останется незамеченным. Поскольку вероятность того, что Ева правильно настроит i-й приемник, равна $1/2$, вероятность не обнаружить перехват в i-й позиции также равна $1/2$.

Если Ева настроила i-й приемник неправильно, то i-е состояние фотонов окажется неверным, и она перешлет Бобу неправильный бит. Несмотря на это, приемник Боба изменит неверное состояние на $\pm 45^\circ$, причем вероятность каждого направления равна $1/2$. Следовательно, вероятность того, что Боб получит правильный или неправильный бит, в обоих случаях равна $1/2$. Если Боб правильно распознал бит, Ева останется незамеченной. Обратите внимание на то, что эта ситуация представляет собой частный случай, когда Ева передала неверный бит,

но, тем не менее, осталась нераскрытой. Вероятность последнего события также равна 1/2. Следовательно, поскольку приемники Евы и Боба настраиваются независимо друг от друга, вероятность того, что Ева останется незамеченной после передачи неверного бита, равна $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$.

Суммируя вероятности, вычисленные выше, приходим к выводу, что вероятность не заметить перехват на i-й позиции равна $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$. Поскольку для того, чтобы завладеть распределенным ключом, Ева должна прослушивать все просеянные состояния, Алиса и Боб сравнивают ℓ случайных тестовых битов и любое несовпадение считают сигналом опасности (это правило представляет собой критерий "единогласного голосования", в котором не допускается ни одного несовпадения). Вероятность не заметить перехват на всех позициях равна $\left(\frac{3}{4}\right)^{\ell}$. Эта величина является вероятностью ошибки, связанной со стойкостью. Она очень быстро стремится к нулю.

Вероятность ошибки, связанной с неполнотой

Оценим вероятность ошибки, связанной с неполнотой протокола. Пусть настройки m поляризаторов Алисы задаются случайным двоичным вектором $V = (v_1, v_2, \ldots, v_m)$, а настройки приемников Боба — вектором $W = (w_1, w_2, \ldots, w_m)$. Ошибка, связанная с неполнотой протокола, возникает, когда вектор

$$V \oplus W = (v_1 \oplus w_1, v_2 \oplus w_2, \ldots, v_m \oplus w_m)$$

содержит меньше $m/10$ нулей. Поскольку настройки Алисы и Боба независимы и равновероятны, вектор $V \oplus W$ должен быть случайным равномерно распределенным вектором, состоящим из m бит. Вероятность того, что вектор $V \oplus W$ содержит i нулей, имеет биномиальное распределение для m испытаний и i успехов, где вероятность успеха равна 0,5. Очевидно, что наиболее вероятное количество нулей в векторе $V \oplus W$ равно $m/2$. Иначе говоря, центральный член биномиального распределения находится в точке $\left\lfloor \frac{m+1}{2} \right\rfloor$. Следовательно, точка $\frac{m}{10}$ расположена далеко слева от точки $\left\lfloor \frac{m+1}{2} \right\rfloor$. Итак, вероятность ошибки, связанной с неполнотой протокола, равна

$$\text{Prob}\left[\text{zeroes\_in}(V \oplus W) < \frac{m}{10}\right]$$

и представляет собой площадь фигуры, ограниченной левым хвостом биномиального распределения. Учитывая оценки левого хвоста биномиального распределения (3.5.8), получим оценку вероятности ошибки, связанной с неполнотой протокола.

$$\text{Prob}\left[\text{zeroes\_in}(V \oplus W) < \frac{m}{10}\right] < \frac{0{,}5\left(m + 1 - \frac{m}{10}\right)}{\left(0{,}5\left(m+1\right) - \frac{m}{10}\right)^2} < \frac{3}{m} \text{ для } m \geqslant 2.$$

Следовательно, вероятность того, что Алиса и Боб не завершат протокол на третьем этапе, равна $1 - \frac{3}{m}$.

Вероятности двусторонних ошибок

Итак, вероятности двусторонних ошибок протокола 4.1 оцениваются следующим образом. Вероятность ошибки, связанной с неполнотой протокола, удовлетворяет следующему неравенству.

$$\text{Prob}\left[\text{Количество просеянных битов} \geqslant \frac{m}{10} \mid \text{Среди } m \text{ распределенных состояний фотонов}\right] > 1 - \frac{3}{m}.$$

Вероятность ошибки, связанной с противоречивостью протокола, удовлетворяет следующему неравенству.

$$\text{Prob}\left[\text{Ева не обнаружена} \mid \text{Алиса и Боб проверили } \ell \text{ тестовых битов}\right] \leqslant \left(\frac{3}{4}\right)^{\ell}.$$

Следует отметить, что оценка левого хвоста величиной $\frac{3}{m}$, полученная по неравенству (3.5.8), является слишком грубой. Левый хвост стремится к нулю быстрее, чем $\frac{3}{m}$ (см. упражнение 3.9).

Оценки вероятностей ошибок протокола QKD показывают, что он вполне приемлем для практики. В реальных приложениях условленный канал, по которому Алиса и Боб ведут открытый обмен информацией, должен обеспечивать аутентификацию. Это необходимо для того, чтобы секретный ключ был распределен среди полномочных партнеров. Проблемы аутентификации рассматриваются в части IV.

Появление коммерческих QKD протоколов ожидалось в 2004 году [268].

4.4.6 Эффективные алгоритмы

Изучая классы полиномиальных алгоритмов и подклассы вероятностных полиномиальных алгоритмов (PPT), мы установили следующие включения.

$$\mathcal{P} \subseteq \mathcal{ZPP} \subseteq \begin{matrix}\mathcal{PP}(\text{Монте-Карло}) \\ \mathcal{PP}(\text{Лас-Вегас})\end{matrix} \subseteq \mathcal{BPP} \subseteq \mathcal{PP}.$$

Алгоритмы, способные решать задачи из этих классов, называются *эффективными*.

Определение 4.6 (Эффективный алгоритм). *Эффективным называется детерминированный или рандомизированный алгоритм, время выполнения которого полиномиально зависит от размера исходных данных.*

Это определение тесно связано с понятием **разрешимости** (tractability): можно ли решить задачу с помощью детерминированного или рандомизированного

полиномиального алгоритма при разумных ограничениях времени и памяти, даже если размер исходных данных очень велик? Если ответ отрицателен, задача считается **трудноразрешимой** (intractable).

Однако, поскольку полиномы, характеризующие время выполнения алгоритмов, принадлежащих классам $\mathcal{P}$ или $\mathcal{PP}$, могут иметь совершенно разные степени, временна́я сложность задач также сильно варьируется. Таким образом, эффективный алгоритм решения разрешимой задачи на практике может оказаться непригодным. В последующих главах мы рассмотрим несколько протоколов, имеющих полиномиальную временну́ю сложность. Следовательно, по определению 4.6 такие протоколы являются эффективными. Однако для практических приложений они не представляют никакой ценности, поскольку степень полиномов, ограничивающих их временну́ю сложность, слишком велика. И наоборот, существуют практичные алгоритмы, не являющиеся полиномиальными (см. раздел 4.6), которые вполне пригодны для эффективного решения теоретически трудноразрешимых задач с небольшим объемом входных данных (например, метод кенгуру Полларда для индексных вычислений, рассмотренный в разделе 3.6.1).

Мы будем называть **практически эффективными** (practically efficient) полиномиальные алгоритмы с невысокой степенью полиномов, ограничивающих временну́ю сложность. Например, машина Тьюринга Div3, алгоритмы gcd, mod_exp и Prime_Test, а также протокол QKD являются практически эффективными. Рассмотрим еще один пример практически эффективного алгоритма, широко применяемого в современной криптографии.

4.4.6.1 Пример эффективного алгоритма

Идею вероятностной проверки простоты числа можно непосредственно воплотить в виде алгоритма, генерирующего случайное **вероятностно простое** (probabilistic prime) число заданного размера. Число n называется вероятностно простым, если алгоритм Prime_Test(n) возвращает ответ “ДА”.

Алгоритм 4.7. Генерация случайного k-битового вероятностно простого числа

ВВОД: k: положительное целое число;
(* размер числа k должен быть равен k битов *).

ВЫВОД: k-битовое случайное вероятностно простое число

Prime_Gen(k).

1. $p \in_U (2^{k-1}, 2^{k-1} - 1]$, где p — нечетное.
2. if Prime_Test(p) = НЕТ return(Prime_Gen(k)).
3. return(p).

Предположим, что алгоритм Prime_Gen(k) завершил свою работу. Это означает, что алгоритм нашел число p, удовлетворяющее условию Prime_Test(p) = ДА (п. 2). Из оценки вероятности ошибки алгоритма Prime_Test следует, что вероятность события "число p — не простое" не превосходит 2^{-k}, где $k = \log_2 p$.

Возникает естественный вопрос: завершится ли алгоритм Prime_Gen(k) вообще?

В теории чисел хорошо известна теорема [170], утверждающая, что количество простых чисел, не превышающих числа X, ограничено снизу числом $\frac{X}{\log X}$. Так, количество простых чисел, состоящих из k бит, приблизительно равно

$$\frac{2^k}{k} - \frac{2^{k-1}}{k-1} \approx \frac{2^k}{2k}.$$

Следовательно, можно *ожидать*, что алгоритм Prime_Gen(k) на этапе 2 рекурсивно вызовет себя $2k$ раз, найдет вероятностно простое число и завершит работу.

Временна́я сложность алгоритма Prime_Test(p) имеет порядок $O_B((\log p)^4) = O_B(k^4)$. Следовательно, при 2^k вызовах алгоритма Prime_Test временна́я сложность алгоритма Prime_Gen(k) имеет порядок $O_B(k^5)$.

Второй вопрос, на который необходимо ответить: означает ли оценка $O_B(k^5)$ существование полинома, зависящего от аргумента k, т.е. может ли эта величина полиномиально зависеть от *размера* исходных данных алгоритма Prime_Gen(k)?

Если число n записано в позиционной системе счисления с основанием $b > 1$, его размер равен $\log_b n$ и всегда меньше n. Для того чтобы время выполнения алгоритма Prime_Gen(k) полиномиально зависело от размера исходных данных, необходимо явно указать, что число k должно состоять из k битов.

Определение 4.7 (Унарное представление числа). *Унарное представление положительного целого числа n имеет вид*

$$1^n \equiv \underbrace{11\ldots1}_{n}.$$

С этого момента для того, чтобы подчеркнуть размер входного числа, мы будем использовать его унарное представление.

4.5 Недетерминированное полиномиальное время

Рассмотрим следующую задачу принятия решений.

Задача SQUARE-FREENESS

ВВОД: N: положительное и нечетное составное целое число.

ВОПРОС: Является ли число N свободным от квадратов?
"ДА", если не существует простого числа p, такого что $p^2 \mid N$.

Задача SQUARE-FREENESS очень сложна. До сих пор не существует алгоритма (детерминированного или вероятностного), временна́я сложность которого полиномиально зависела бы от размера числа N. Разумеется, существуют алгоритмы, позволяющие ответить на поставленный вопрос. Вот один из них: вводим число N, выполняем деление на квадраты всех нечетных простых чисел, не превышающих $\lfloor N \rfloor$, и, если число N не делится ни на один из квадратов, выводим ответ "ДА". Однако, если число N произвольно, этот метод выполняется за $O\left(\left\lfloor\sqrt{N}\right\rfloor\right) = O\left(e^{\frac{\log N}{2}}\right)$ единиц времени, т.е. временна́я сложность экспоненциально зависит от половины размера числа N.

Несмотря на это, задача SQUARE-FREENESS не является трудноразрешимой. Если нам известна некая дополнительная информация о задаче — **свидетельство** (witness) (или **сертификат** (sertificate), или **вспомогательные исходные данные** (auxiliary input)) — ее можно решить за время, полиномиально зависящее от размера исходного числа. Например, если на ввод поступает число N, в качестве свидетельства, гарантирующего эффективность алгоритма верификации целых чисел, свободных от квадратов, можно использовать число $\phi(N)$ — количество всех положительных чисел, не превосходящих N и взаимно простых с числом N (определение 5.11 в разделе 5.2.3), вычисленное с помощью **функции Эйлера** ϕ (Euler function phi).

Алгоритм 4.8. Square-Free$(N, \phi(N))$

1. $d \leftarrow \gcd(N, \phi(N))$;
2. if $d = 1$ или $d^2 \nmid N$ ответ "ДА" else ответ "НЕТ".

Читатели, знакомые с функцией Эйлера, могут убедиться в правильности алгоритма 4.8 (упражнение 4.13). Верификация алгоритма достигается благодаря факту из теории чисел, который будет описан в главе 6 (раздел 6.3). Анализ временно́й сложности алгоритма поиска наибольшего общего делителя (раздел 4.3.2.3) показывает, что она полиномиально зависит от размера числа N.

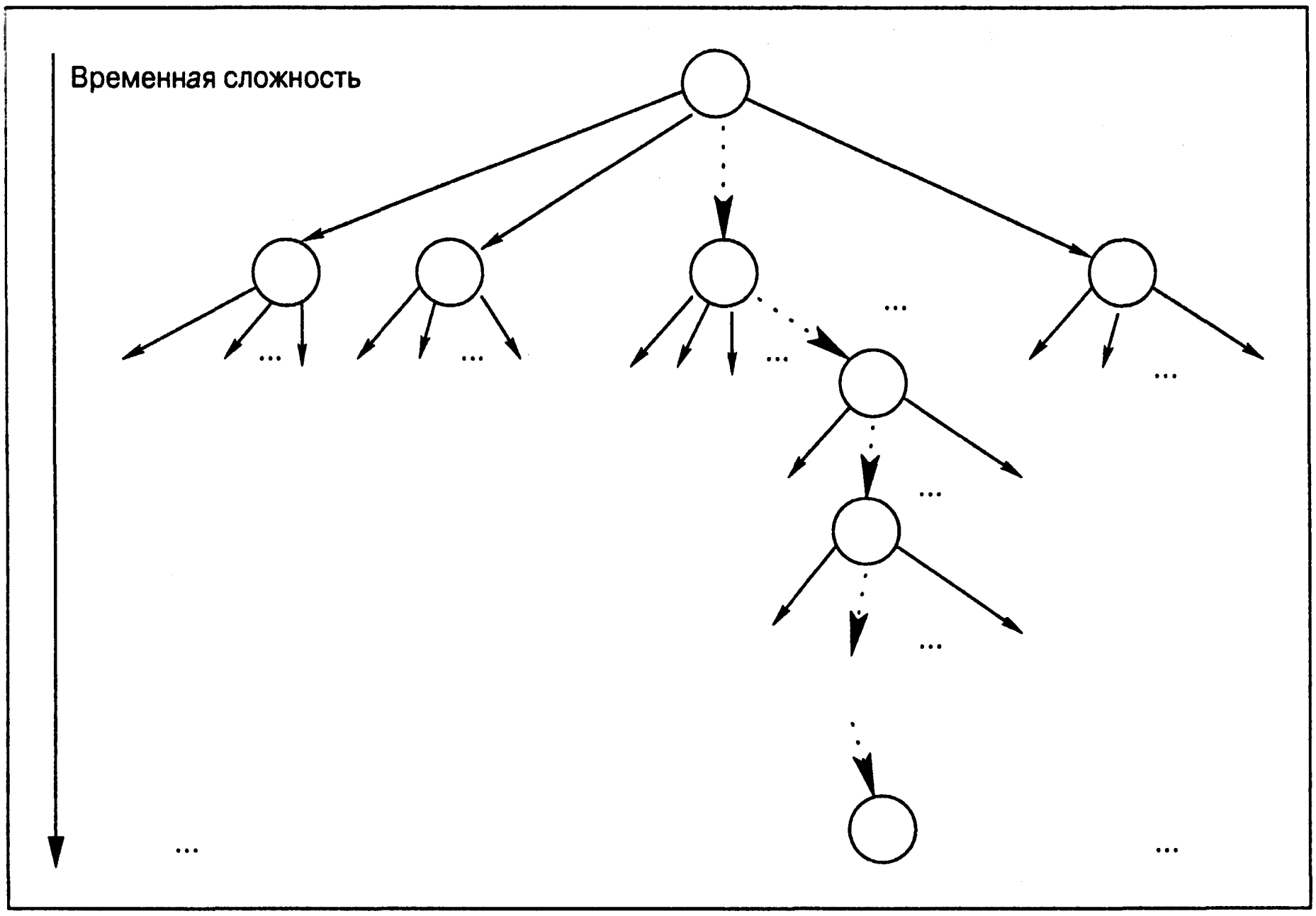


Рис. 4.4. Все возможные такты недетерминированной машины Тьюринга при решении задачи распознавания

Опишем некое вычислительное устройство, моделирующее метод решения задач, принадлежащих тому же классу сложности, что и задача SQUARE-FREENESS. Ход вычислений описывается на рис. 4.4.

Это устройство называется **недетерминированной машиной Тьюринга** (nondeterministic Turing machine). На каждом такте этой машины существует конечное количество вариантов следующего такта. Входная строка считается распознанной, если существует хотя бы одна последовательность разрешенных тактов, начинающаяся считыванием первого символа строки и завершающаяся считыванием последнего символа. Такая последовательность тактов называется *последовательностью распознавания* (recognition sequence).

Работу недетерминированной машины Тьюринга можно представить в виде серии догадок. В этом случае последовательность распознавания представляет собой серию правильных догадок. Таким образом, все возможные такты образуют дерево, называемое **вычислительным деревом** (computational tree) недетерминированной машины Тьюринга. Совершенно очевидно, что размер (количество узлов) этого дерева экспоненциально зависит от размера входа. Однако, поскольку количество тактов в последовательности распознавания входной строки равно глубине дерева d, получаем, что $d = O(\log(\text{количество узлов дерева}))$ и количество тактов в последовательности распознавания полиномиально зависит от размера

входной строки. Итак, временна́я сложность распознавания строки с помощью серии правильных догадок полиномиально зависит от размера исходных данных.

Определение 4.8 (Класс $\mathcal{NP}$). *Язык принадлежит классу $\mathcal{NP}$, если он распознается недетерминированной машиной Тьюринга за полиномиальное время.*

Очевидно, что

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Точнее говоря, каждый язык (задача принятия решений), принадлежащий классу $\mathcal{P}$, тривиально распознается недетерминированной машиной Тьюринга. Точно так же легко убедиться, что

$$\mathcal{ZPP}, \mathcal{PP}(\text{Монте-Карло}), \mathcal{PP}(\text{Лас-Вегас}) \subseteq \mathcal{NP}.$$

Действительно, классы $\mathcal{ZPP}$, $\mathcal{PP}$(Монте-Карло) и $\mathcal{PP}$(Лас-Вегас), по существу, являются подклассами класса $\mathcal{NP}$, поскольку содержат задачи, для решения которых требуются *недетерминированные полиномиальные* алгоритмы[6]. Единственная причина, по которой задачи, принадлежащие этим подклассам, допускают эффективное решение, заключается в том, то эти NP-задачи содержат много дополнительной информации, позволяющей делать правильные догадки. По общепринятому соглашению класс $\mathcal{NP}$ считается классом недетерминированных полиномиальных задач принятия решений, имеющих **немногочисленные свидетельства** (sparse witnesses). Смысл выражения "немногочисленные свидетельства" заключается в следующем: для вычислительного дерева решения NP-задачи дробь

$$\frac{\text{количество последовательностей распознавания}}{\text{количество всех последовательностей}}$$

является пренебрежимо малой (определение 4.13).

В разделе 18.2.3 будет показано, что

$$\mathcal{NP} \subseteq \mathcal{PP}. \tag{4.5.1}$$

Если NP-задача обладает немногочисленными свидетельствами, то недетерминированная машина Тьюринга, использующая случайные догадки, не гарантирует эффективного алгоритма для ее решения. Этим NP-задачи с немногочисленными свидетельствами отличаются от NP-задач с большим количество свидетельств. Для NP-задач с фрагментарными свидетельствами недетерминированные машины Тьюринга просто моделируют класс задач принятия решений, обладающих следующими свойствами.

---

[6]Вспомните причину, по которой в начале раздела 4.4 мы назвали подкласс недетерминированных полиномиальных машин Тьюринга "вероятностными машинами Тьюринга".

При наличии свидетельств ответ на задачу принятия решения можно получить за полиномиальное время.

Свидетельство для NP-задачи моделируется последовательностью распознавания в вычислительном дереве недетерминированной машины Тьюринга (пунктирные линии на рис. 4.4).

Возникает вопрос: какова временна́я сложность решения произвольной NP-задачи без наличия свидетельств? Ответ *неизвестен*. Все существующие алгоритмы решения *произвольных* NP-задач *без использования свидетельств* имеют полиномиально неограниченную временну́ю сложность. Еще никому не удалось доказать, что $\mathcal{P} \neq \mathcal{NP}$. Кроме того, противоположный факт, т.е. $\mathcal{P} = \mathcal{NP}$, также не доказан. Вопрос

$$\mathcal{P} = \mathcal{NP}\,?$$

является хорошо известной нерешенной задачей в области теории компьютерных наук.

Определение 4.9 (Нижние и верхние оценки сложности). *Величина B называется нижней оценкой сложности задачи P, если любой алгоритм A решения задачи P имеет сложность $C(A) \geqslant B$. Величина U называется верхней оценкой сложности задачи P, если любой алгоритм A решения задачи P имеет сложность $C(A) \leqslant U$.*

Как правило, нижние оценки сложности любой задачи из класса $\mathcal{P}$ определяются довольно просто. Кроме того, для этих задач можно точно определить полиномиальную оценку количества шагов, *необходимых* для их решения. В качестве примера рассмотрим машину Div3 (пример 4.1). Она распознает n-битовую строку ровно за n шагов, т.е. именно за столько шагов, сколько требуется для записи входного предложения.

Однако для задач из класса $\mathcal{NP}$ нижнюю оценку сложности получить очень трудно. Все известные оценки сложности NP-задач являются верхними. Например, как показано выше, для решения задачи SQUARE_FREENESS с помощью тривиального деления верхняя оценка равна $\left\lfloor \sqrt{N} \right\rfloor$, где N — размер входного числа. По существу, верхняя оценка означает следующее: "для решения задачи требуется именно столько шагов", хотя при этом умалчивается, что задачу можно решить и за меньшее количество шагов. Фактически задачу SQUARE_FREENESS можно решить методом числового решета (Number Field Sieve) за меньшее количество шагов, однако число $\left\lfloor \sqrt{N} \right\rfloor$ остается верхней оценкой.

Не следует путать точную "нижнюю границу" и просто "нижнюю границу". Последнее выражение часто встречается в литературе (например, в знаменитой статье Кука (Cook) [80], в которой доказано, что "задача выполнимости" ("Satisfiability Problem") является "NP-полной"). Оно означает вновь полученную оценку

сложности, которая оказалась меньше, чем ранее существующие. Даже определение обычной (не точной) нижней границы должно сопровождаться доказательством. В свою очередь, определение *точной* нижней границы сложности NP-задачи является основным камнем преткновения в теории вычислительной сложности.

Трудность определения точной нижней неполиномиальной оценки сложности NP-задачи порождает серьезные последствия для современной криптографии, в которой вопросы безопасности рассматриваются через призму вычислительной сложности. Эти вопросы будут изучены в разделе 4.8.

4.5.1 Недетерминированные полиномиально полные задачи

Хотя до сих пор неизвестно, выполняется ли равенство $\mathcal{P} = \mathcal{NP}$, сложность решения некоторых задач из класса $\mathcal{NP}$ эквивалентна сложности решения произвольной задачи из класса $\mathcal{NP}$. Иначе говоря, если существует эффективный алгоритм решения одной из таких задач, то с его помощью можно решить любую задачу из класса $\mathcal{NP}$. Такие задачи называются **недетерминированными полными задачами с полиномиальным временем решения** (non-deterministic polynomial-time complete problem) или **NP-полными** (сокращенно NPC).

Определение 4.10 (Полиномиально приводимый язык). *Язык L называется полиномиально приводимым (polynomially reducible) к другому языку L_0, если существует детерминированная машина Тьюринга M с полиномиальным временем работы, которая переводит любое предложение $I \in L$ в предложение $I_0 \in L_0$, так что $I \in L$ тогда и только тогда, когда $I_0 \in L_0$.*

Определение 4.11 (NP-полный язык). *Язык $L_0 \in \mathcal{NP}$ называется недетерминированным полным языком с полиномиальным временем распознавания (NP-полным), если любой язык $L \in \mathcal{NP}$ приводится к языку L_0 за полиномиальное время.*

К числу хорошо известных NP-полных задач относится так называемая задача выполнимости (SATISFIABILITY), поставленная Куком в работе [80]. Это — первая NP-полная задача [227]. Обозначим через $E(x_1, x_2, \ldots, x_n)$ булевское выражение, построенное из n булевских переменных $x_1, x_2, \ldots, x_n$ и булевских операторов $\wedge$, $\vee$ и $\neg$.

Задача SATISFIABILITY

ВВОД: $X = (x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n)$;
$E(x_1, x_2, \ldots, x_n)$.

Истинностной подстановкой (truth assignement) для выражения $E(x_1, x_2, \ldots, x_n)$ является подсписок X' списка X, такой что для $1 \leqslant i \leqslant n$ подсписок X' содержит либо x_i, либо $\neg x_i$, но не оба одновременно, причем $E(X') = \texttt{True}$.

ВОПРОС: Является ли выражение $E(x_1, x_2, \ldots, x_n)$ выполнимым?
Иначе говоря, существует ли истинностная подстановка?
Если выражение $E(x_1, x_2, \ldots, x_n)$ является выполнимым,
алгоритм выдает ответ "ДА".

Если истинностная подстановка существует, ответ ДА можно проверить за время, ограниченное полиномом степени n. Следовательно, по определению 4.8 SATISFIABILITY $\in \mathcal{NP}$. Следует обратить внимание на то, что существует 2^n возможных истинностных подстановок, и до сих пор не известен ни один детерминированный полиномиальный алгоритм, который определил бы, существует ли истинностная подстановка.

Доказательство того, что задача SATISFIABILITY является NP-полной (Кук [80]), дано в главе 10 книги [9]. Это доказательство является конструктивным и основано на преобразовании произвольной недетерминированной машины Тьюринга с полиномиальным временем выполнения в машину, которая решает задачу SATISFIABILITY.

Большой список NP-полных задач приведен в книге [118].

Любую новую оценку сложности NP-полной задачи, не превышающую верхнюю оценку, за полиномиальное время можно преобразовать в новый результат для всего класса NP-задач. Следовательно, как указано в работе [98], желательно, чтобы стойкость криптографических алгоритмов была обеспечена NP-полной задачей. Последовательная атака на подобную криптосистему должна привести к решению всего класса таких задач, что крайне маловероятно. Однако это вполне разумное требование не очень практично ни с точки зрения стойкости криптосистем, ни с точки зрения решения всех NP-задач с помощью атаки на такие криптосистемы. Это странное явление будет изучено в разделе 4.8.2.

4.6 Неполиномиальные оценки

Существует громадное множество функций, растущих быстрее полиномов.

Определение 4.12 (Неполиномиально ограниченная функция). *Функция* $f(n) : \mathbb{N} \mapsto \mathbb{R}$ *называется не ограниченной никаким полиномом степени* n*, если для любого полинома* $p(n)$ *существует натуральное число* n_0*, такое что для всех* $n > n_0$ *выполняется условие* $f(n) > p(n)$.

Функция $f(n)$ называется **полиномиально ограниченной** (polynomially bounded), если она не является неполиномиально ограниченной.

Пример 4.3. Покажем, что для любого числа $a > 1, 0 < \varepsilon < 1$ функции

$$f_1(n) = a^{n^{\varepsilon}(\log n)^{1-\varepsilon}}, \quad f_2(n) = a^{(\log\log\log n)^{\varepsilon}}$$

не ограничены никаким полиномом степени n.

Пусть $p(n)$ — произвольный полином. Обозначим через d степень этого полинома, а через c — его наибольший коэффициент. Следовательно, $p(n) \leqslant cn^d$. Во-первых, пусть $n_0 = \max\left(c, \left\lfloor \left(\frac{d+1}{\log a}\right)^{\frac{2}{\varepsilon}} \right\rfloor\right)$, тогда $f_1(n) > p(n)$ для всех $n > n_0$. Во-вторых, пусть $n_0 = \max\left(c, \left\lfloor \exp\left(\exp\left(\exp\left(d+1\right)^{\frac{1}{\varepsilon}}\right)\right) \right\rfloor\right)$, тогда $f_2(n) > p(n)$ для всех $n > n_0$. □

В отличие от задач с полиномиальным временем решения (детерминированным или рандомизированным), задачи с неполиномиальным временем решения считаются трудноразрешимыми или не имеющими решения. Это связано с ограниченностью ресурсов, необходимых для решения таких задач, поскольку при увеличении размера исходных данных затраты растут настолько быстро, что задача становится практически неразрешимой. Например, пусть N — составное целое число, имеющее размер n (т.е. $n = \log N$). Тогда функция $f_1(\log N)$ в примере 4.3, где $a \approx \exp(1{,}9229994\cdots + o(1))$, $o(1) \approx \frac{1}{\log N}$, $\varepsilon = \frac{1}{3}$, характеризует временну́ю сложность факторизации числа N с помощью *решета числового поля* (Number Field Sieve — NFS) [70]:

$$\exp\left((1{,}9229994\cdots + o(1))(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}\right). \tag{4.6.1}$$

Выражение (4.6.1) является **субэкспоненциальным** (sub-exponential) по N. Если заменить число $\frac{1}{3}$ числом 1, выражение станет экспоненциальным. Субэкспоненциальная функция растет намного медленнее, чем экспоненциальная, но намного быстрее, чем полиномиальная. Если число N состоит из 1024 бит, выражение (4.6.1) превышает 2^{86}. Это чрезвычайно большое число операций, которое невозможно выполнить даже с помощью крупной сети параллельных компьютеров. Формула субэкспоненциальной временно́й сложности справедлива также для алгоритма дискретного логарифмирования в конечном поле (см. определение 8.2 в разделе 8.4).

Следует, однако, отметить **асимптотический** характер сравнения функций, примененный в определении 4.12 (функция $f(n)$ в определении 4.12 также асимптотически больше любого полинома, т.е. превышает любой полином при достаточно большом числе n). Даже если функция $f(n)$ не ограничена ни одним полиномом степени n, зачастую при достаточно большом числе n_0 функция $f(n)$ оказывается меньше, чем некоторый полином $p(n)$ при $n \leqslant n_0$. Например, функция $f_2(n)$ в примере 4.3 при $\varepsilon = 0{,}5$ остается меньше, чем квадратичная функция

n^2 при всех $n \leqslant 2^{74276224545492773674354 1}$, хотя функция $f_2(n)$ асимптотически больше, чем n^d при любых $d \geqslant 1$. По этой причине на практике некоторые алгоритмы с неполиномиальной временно́й сложностью остаются эффективными при решении задач для небольших размеров входных данных. Примером такого алгоритма является λ-метод Полларда (раздел 3.6.1), позволяющий вычислять небольшие значения дискретного логарифма.

Используя O-символику (см. определение 4.2 в разделе 4.3.2.4), мы преднамеренно игнорируем любые постоянные коэффициенты в выражении для оценки сложности. Однако следует иметь в виду, что в выражениях для неполиномиальной сложности постоянные коэффициенты, стоящие в степени числа, приобретают большую важность (например, число $1{,}9229994\cdots + o(1)$ в выражении (4.6.1)). Например, если усечь число 1,9229994 в выражении (4.6.1) до единицы, получится новый алгоритм разложения целого числа на простые множители. Его сложность для целого числа, состоящего из 1024 бит, уменьшается с 2^{86} до 2^{45}. Такое количество операций уже вполне доступно для современных технологий. Для метода решета числового поля основные усилия исследователей направлены на ускорение метода путем уменьшения степени (правда, такое сокращение может привести к росту затрат памяти).

Выше мы сформулировали определение неполиномиальной оценки для больших чисел. Теперь мы можем сформулировать это понятие для малых величин.

Определение 4.13 (Пренебрежимо малые величины). *Функция* $\varepsilon(n) : \mathbb{N} \mapsto \mathbb{R}$ *имеет пренебрежимо малые значения по сравнению с числом* n*, если функция* $\frac{1}{\varepsilon(n)}$ *является неполиномиально ограниченной.*

Например, для любого полинома p величина $\frac{p(n)}{2^n}$ является пренебрежимо малой. По этой причине иногда говорят, что подмножество, состоящее из $p(n)$ точек множества $\{1, 2, 3, \ldots, 2^n\}$, является пренебрежимо малым по сравнению с размером всего множества, или что подмножество, состоящее из $p(n)$ точек множества $\{1, 2, 3, \ldots, 2^n\}$, является разреженным.

Если ε — пренебрежимо малая величина, то число $1 - \varepsilon$ называется **огромным** (overwhelming quantity). (Как правило, этот эпитет относится к вероятности. — *Прим. ред.*) Следовательно, **не разреженному** (non-sparse) (т.е. **плотному** (dense)) подмножеству множества $\{1, 2, 3, \ldots, 2^n\}$ принадлежит огромное количество точек этого множества.

Пренебрежимо малая функция стремится к нулю быстрее, чем $\frac{1}{p(n)}$, где $p(n)$ — произвольный полином. Если неполиномиально ограниченная величина считается недостижимой (например, при выделении ресурсов), то можно совершенно безопасно пренебречь любой величиной, обратной к ней.

Рассмотрим несколько примеров.

Вероятность

$$\text{Prob}[\text{“Prime\_Gen}(1^k) \text{ — не простое число”}]$$

пренебрежимо мала, а вероятность

$$1 - \text{Prob}[\text{“Prime\_Gen}(1^k) \text{ — не простое число”}] = \\ = \text{Prob}[\text{“Prime\_Gen}(1^k) \text{ — простое число”}]$$

является огромной.

Вернемся к примеру 3.6. Если число p является простым и состоит из k бит (число $q = \frac{p-1}{2}$ также является простым), можно пренебречь величиной $\frac{1}{p-1}$ и получить приближенную оценку $\text{Prob}[A] \approx \frac{3}{4}$.

В заключение заметим, что если величина не является пренебрежимо малой, то ее часто называют **значимой** (significant quantity). Например, в ряде примеров, посвященных задачам принятия решений, принадлежность которых классу $\mathcal{PP}$ можно эффективно проверить, существует значимая вероятность того, что при простом случайном выборе из пространства вычислительных деревьев (рис. 4.4) мы найдем свидетельство, позволяющее подтвердить ее принадлежность классу $\mathcal{PP}$.

4.7 Полиномиальная неразличимость

В предыдущем разделе мы показали, что пренебрежимо малые величины можно игнорировать. Однако в некоторых ситуациях мы, наоборот, *вынуждены* отказаться от попыток игнорировать определенные величины.

В качестве примера рассмотрим два эксперимента в пространстве больших нечетных составных чисел фиксированной длины. Обозначим первый эксперимент как E_{2_Prime}, а второй — E_{3_Prime}. В ходе этих экспериментов генерируются большие случайные целые числа одинакового размера: в эксперименте E_{2_Prime} порождаются произведения двух разных больших простых множителей, а в эксперименте E_{3_Prime} — трех и больше. Затем испытуемому предъявляется некое целое число N, порожденное одним из этих экспериментов. Можно ли уверенно определить, в результате которого из двух экспериментов появилось число N? (Напомним, что в ходе эксперимента E_{2_Prime} и E_{3_Prime} генерируются целые числа одинакового размера.)

По определению 3.5 (раздел 3.5) результатом таких экспериментов являются случайные величины, зависящие от случайных операций. Случайные величины, порожденные в ходе экспериментов E_{2_Prime} и E_{3_Prime}, имеют совершенно разные распределения: эксперимент E_{2_Prime} генерирует произведение двух простых

множителей с вероятностью 1, а эксперимент E_{3_Prime} — нет. Однако различить целые числа, возникшие в результате этих экспериментов, очень трудно.

Дадим точное определение **неразличимых ансамблей** (indistinguishable ensembles), или **неразличимых экспериментов** (indistinguishable experiments).

Определение 4.14 (Классификатор ансамблей). *Пусть* $E = \{e_1, e_2, \ldots\}$ *и* $E' = \{e'_1, e'_2, \ldots\}$ *— два множества ансамблей, в которых* e_i *и* e'_i *— случайные величины из конечного выборочного пространства* $\mathbb{S}$. *Введем обозначение* $k = \log_2 \#\mathbb{S}$. *Пусть* $a = (a_1, a_2, \ldots, a_\ell)$ *— вектор, состоящий из случайных переменных, которые все порождены в результате либо эксперимента* E, *либо эксперимента* E', *где* ℓ *— величина, ограниченная полиномом, зависящим от аргумента* k.

Классификатор (distinguisher) $\mathcal{D}$ *пары* (E, E') *представляет собой вероятностный алгоритм, завершающий работу за время, ограниченное полиномом, зависящим от аргумента* k. *Результат работы алгоритма* $\mathcal{D}$ *принадлежит множеству* $\{0, 1\}$ *и удовлетворяет следующим условиям: 1)* $\mathcal{D}(a, E) = 1$ *тогда и только тогда, когда вектор* a *принадлежит множеству* E; *и 2)* $\mathcal{D}(a, E') = 1$ *тогда и только тогда, когда вектор* a *принадлежит множеству* E'.

Будем говорить, что классификатор $\mathcal{D}$ *различает пару* (E, E') *с преимуществом (advantage)* $\text{Adv} > 0$, *где*

$$\text{Adv}(\mathcal{D}) = \mid \text{Prob}[\mathcal{D}(a, E) = 1] - \text{Prob}[\mathcal{D}(a, E') = 1] \mid .$$

Следует подчеркнуть, что в определении преимущества классификатора $\mathcal{D}$ используются распределения вероятностей: классификатор является вероятностным алгоритмом с полиномиальным временем выполнения, а его ввод имеет полиномиально ограниченный размер.

Многие случайные переменные легко поддаются распознаванию. Рассмотрим пример.

Пример 4.4. Пусть $E = \{k\text{-битовые простые числа}\}$ и $E' = \{k\text{-битовые составные числа}\}$. По определению $\mathcal{D}(a, E) = 1$ тогда и только тогда, когда $\text{Prime\_Test}(a) \rightarrow$ ДА, и $\mathcal{D}(a, E') = 1$ тогда и только тогда, когда $\text{Prime\_Test}(a) \rightarrow$ НЕТ (см. алгоритм 4.5). Если $a \in E$, то $\text{Prob}[\mathcal{D}(a, E) = 1] = 1$ и $\text{Prob}[\mathcal{D}(a, E') = 1] = 0$. Если $a \in E'$, то $\text{Prob}[\mathcal{D}(a, E) = 1] = 2^{-k}$ и $\text{Prob}[\mathcal{D}(a, E') = 1] = 1 - 2^{-k}$. Следовательно, $\text{Adv}(\mathcal{D}) \geqslant 1 - 2^{-(k-1)}$. □

Определение 4.15 (Полиномиальная неразличимость). *Пусть ансамбли* E, E' *и параметр безопасности* k *удовлетворяют условиям, указанным в определении 4.14. Ансамбли* E *и* E' *называются полиномиально неразличимыми, если для пары* (E, E') *не существует ни одного классификатора с преимуществом* $\text{Adv} > 0$, *которое при всех достаточно больших значениях* k *не было бы пренебрежимо малым.*

В теории вычислительной сложности приняты следующие вполне правдоподобные предположения.

Предположение 4.1 (Предположение об общей неразличимости). *Существуют полиномиально неразличимые ансамбли.*

Ансамбли E_{2_Prime} и E_{3_Prime} считаются полиномиально неразличимыми. Иначе говоря, если некто предъявит нам множество целых чисел, которые все сгенерированы в ходе либо эксперимента E_{2_Prime}, либо эксперимента E_{3_Prime}, а их количество ограничено полиномом, и мы применим наилучший из классификаторов, то мы не сможем их распознать за разумное время.

Поскольку мы можем факторизовать число N и правильно ответить на вопрос, преимущество классификатора не должно превосходить функцию $\exp\left((1{,}9229994\cdots + o(1))(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}\right)^{-1}$. Однако эта величина слишком мала и должна быть проигнорирована. Мы не сможем распознать два ансамбля, потому что преимущество наилучшего классификатора, который мы могли бы применить, имеет ничтожно малое преимущество, зависящее от размера целых чисел, входящих в ансамбли. Преимущество такого классификатора представляет собой медленно растущую функцию (slow-growing function), зависящую от объема вычислительных ресурсов. Здесь термин "медленно растущая" означает, что даже если бы мы обладали огромными вычислительными ресурсами, преимущество увеличилось бы крайне мало, и ситуация осталась бы безнадежной.

Полиномиальная неразличимость представляет собой важный критерий стойкости многих криптографических систем и протоколов. Существует много практических способов создать полиномиально неразличимые ансамбли, полезные для современной криптографии. Например, важным компонентом криптографии является генератор **псевдослучайных чисел** (pseudo-random number generator). Псевдослучайные числа, порождаемые такими генераторами, имеют полностью детерминированное распределение, зависящее от начального числа (seed). Хороший датчик генерирует псевдослучайные числа, полиномиально неотличимые от истинно случайных чисел, т.е. распределение псведослучайных величин, порожденных таким генератором, и равномерное распределение строк, имеющих такую же длину, являются полиномиально неразличимыми. Исходя из этих соображений, можно уточнить предположение 4.1 следующим образом.

Предположение 4.2 (Неразличимость псевдослучайности и истинной случайности). *Существуют псевдослучайные функции, которые невозможно отличить от истинно случайных функций за полиномиальное время.*

В главе 8 мы рассмотрим псевдослучайную функцию (генератор псевдослучайных чисел), которая является полиномиально неотличимой от равномерного распределения. В главе 14 мы продолжим изучение **криптосистемы с от-**

крытым ключом, широко известной как **криптосистема Голдвассера–Микали** Goldwasser–Micali cryptosystem). Ее стойкость основана на использовании полиномиально неразличимых ансамблей, связанных с экспериментами E_{2_Prime} и E_{3_Prime} (см. раздел 6.5.1). В качестве еще одного примера полиномиально неразличимых ансамблей можно назвать **кортеж Диффи–Хеллмана** (Diffie–Hellman tuple) (определение 13.1 в разделе 13.3.4.3), состоящий из четырех элементов определенной **абелевой группы** (abelian group), и случайную четверку, состоящую из элементов той же группы. Они обеспечивают стойкость **криптосистемы Эль-Гамаля** (ElGamal cryprosystem) и **многих протоколов с нулевым разглашением** (zero-knowledge proof protocol). В дальнейших главах мы неоднократно будем использовать понятие полиномиальной неразличимости.

4.8 Теория вычислительной сложности и современная криптография

В заключение обсудим связь между вычислительной сложностью и современной криптографией.

4.8.1 Необходимое условие

С одной стороны, современная криптография, основанная на теории вычислительной сложности, в качестве необходимого условия использует предположение, что $\mathcal{P} \neq \mathcal{NP}$. Назовем это условие гипотезой $\mathcal{P} \neq \mathcal{NP}$[7].

С другой стороны, алгоритм шифрования должен предоставлять пользователю, обладающему правильными ключами, эффективный алгоритм шифрования и/или дешифровки. Кроме того, он должен представлять собой неразрешимую задачу для тех (перехватчиков или криптоаналитиков), кто попытается восстановить по зашифрованному тексту открытый текст или создать корректный зашифрованный текст без применения правильных ключей. Следовательно, в криптосистеме, основанной на NP-задаче, криптографические ключи играют роль свидетельств или вспомогательных исходных данных (более точное название).

Необходимое условие криптографии, основанной на теории сложности, может вызывать возражение. Это возражение основано на предположении, что существует криптосистема, которая могла быть основана на асимметричной задаче из класса $\mathcal{P}$: шифрование может осуществляться с помощью алгоритма, имеющего сложность порядка $O(n)$, а наилучший алгоритм взлома должен был бы иметь порядок сложности $O(n^{100})$. Действительно, даже в самом простом случае, когда $n = 10$, величина $O(n^{100})$ представляет собой число, приблизительно равное

[7]Недавний опрос показал, что большинство специалистов в области компьютерных наук считают, что $\mathcal{P} \neq \mathcal{NP}$.

2^{332}. Такое количество операций выходит за все разумные пределы. Следовательно, если бы такая полиномиальная криптосистема существовала, мы были бы защищены, даже если бы оказалось, что $\mathcal{P} = \mathcal{NP}$. Однако, хотя класс $\mathcal{P}$ содержит $O(n^k)$ задач для *любого* целого числа k, он не содержит ни одной задачи, обладающей свойством *асимметричной* сложности. Если вариант любой задачи из класса $\mathcal{P}$, имеющий размер n, можно решить за время n^k, то из-за *детерминированного* характера алгоритма для его решения совершенно не требуется время $n^{k+\alpha}$ для любого $\alpha > 0$.

Кроме того, гипотеза $\mathcal{P} \neq \mathcal{NP}$ является необходимым условием существования **однонаправленной функции** (one-way function). В начале книги (раздел 1.1.1) было сформулировано предположение, что однонаправленная функция $f(x)$ должна обладать "волшебным" свойством 1.1: значение $f(x)$ для всех целых чисел x вычисляется легко, а восстановить число x для большинства значений $f(x)$ (за исключением пренебрежимо малого количества случаев) чрезвычайно трудно. Теперь мы знаем, что это условие выполняется для алгоритмов из класса $\mathcal{NP}$. Например, задача SATISFIABILITY определяет однонаправленную функцию, отображающую n-мерное пространство булевских кортежей в множество $\{\texttt{True}, \texttt{False}\}$.

В свою очередь, существование однонаправленной функции представляет собой необходимое условие для применения **цифровых подписей** (digital signatures), которые легко распознать, но трудно подделать.

Более того, понятие полиномиальной неразличимости, изученное нами в разделе 4.7, также основано на гипотезе $\mathcal{P} \neq \mathcal{NP}$. Задача полиномиальной неразличимости представляет собой задачу принятия решений из класса $\mathcal{NP}$. В главах 14, 15 и 17 мы рассмотрим роль, которую играет полиномиальная неразличимость в современной криптографии при доказательстве корректности криптографических алгоритмов и протоколов.

Необходимо упомянуть о том, насколько важна гипотеза $\mathcal{P} \neq \mathcal{NP}$ для **протоколов с нулевым разглашением** [126] и для интерактивных систем доказательства.

Протокол с нулевым разглашением — это интерактивная процедура, выполняемая двумя участниками, называемыми **доказывающим** (prover) и **верификатором** (verifier), причем последний обладает полиномиально ограниченными вычислительными мощностями. Протокол позволяет доказывающему лицу убедить верификатора, что доказывающий знает ответ "ДА" для NP-задачи (например, ответ "ДА" для задачи SQUARE-FREENESS или ответ "ДА" на вопрос "Порождено ли число N в ходе эксперимента E_{2_Prime}?"), поскольку доказывающий владеет вспомогательной информацией, но не раскрывает проверяющему ее суть. Следовательно, проверяющий ничего не знает о доказательстве, которым владеет доказывающий. Такое доказательство можно смоделировать с помощью недетерминированной машины Тьюринга с дополнительной случайной лентой. Доказыва-

ющий может использовать вспомогательную информацию, поэтому машина под управлением доказывающего всегда может выполнить правильный такт вдоль последовательности распознавания (т.е. продемонстрировать, что доказывающий действительно знает ответ "ДА"). Следовательно, временна́я сложность доказательства полиномиально зависит от размера входной информации. Верификатор должен послать доказывающему запрос и попросить его провести машину Тьюринга вдоль последовательности распознавания или вдоль какой-либо иной последовательности, причем запрос должен быть равномерно распределенной случайной величиной. Итак, с точки зрения верификатора, система доказательств ведет себя как рандомизированная машина Тьюринга (раздел 4.4). Независимо повторяя запуски машины Тьюринга, ошибку рандомизированной машины Тьюринга можно уменьшить до ничтожно малой величины (раздел 4.4.1.1). Это убеждает верификатора, что доказывающий действительно знает ответ "ДА" для поставленной задачи.

Гипотеза $\mathcal{P} \neq \mathcal{NP}$ имеет для протоколов с нулевым разглашением двойное значение: 1) вспомогательная информация для NP-задачи позволяет доказывающему провести эффективное доказательство и 2) трудность задачи означает, что верификатор самостоятельно не в состоянии проверить утверждение доказывающего лица.

4.8.2 Недостаточное условие

Гипотеза $\mathcal{P} \neq \mathcal{NP}$ не является достаточным условием стойкости криптосистем, даже если криптосистема основана на решении NP-полной задачи. Примером нестойкой NP-полной задачи является задача об укладке ранца [200].

Изучив основы теории вычислительной сложности, перейдем к разбору причин, по которым криптосистемы, основанные на решении NP-задачи, часто оказываются нестойкими.

Во-первых, как мы уже указывали (определение 4.1), в рамках теории вычислительной сложности язык L (задача) включается в класс сложности с помощью квантора всеобщности: "любое предложение $I \in L$". Это ограничение приводит на **анализу худшего случая** (worst-case complexity): задача считается трудноразрешимой, даже если у нее есть лишь несколько сложных вариантов. В противоположность этому криптоанализ считается успешным, если с его помощью удается решить значительное количество вариантов. Именно по этой причине взлом криптосистемы, основанной на решении NP-задачи, не означает решения самой задачи. Очевидно, что анализ худших вариантов бесполезен для измерения стойкости практических криптосистем.

Вторая причина заключается в *изначальной* трудности определения новой, более низкой верхней границы сложности для NP-задачи (см. раздел 4.5). Стойкость криптосистем, основанных на решении NP-задачи, даже если задача является

трудноразрешимой, в лучшем случае остается недоказанной. Как правило, остается недоказанной даже сама трудная разрешимость задач, лежащих в основе стойкости таких криптосистем.

Еще одна причина недостаточной стойкости современных криптографических систем, основанных на вычислительной сложности, заключается в особенностях прикладной криптографии. Реальные криптографические системы являются результатом компромиссов, которые будут рассмотрены в остальной части книги.

Позитивный подход к разработке и анализу безопасных криптосистем, получивший широкое признание, заключается в формальном доказательстве их стойкости (**доказуемая стойкость** (provable security)) с помощью методов полиномиальной редукции (определение 4.10): *любая* эффективная атака на криптосистему путем эффективного преобразования сводится к решению некоторого варианта одной из известных NP-задач. Такая редукция называется **сведением к противоречию** (reduction to contradictory), поскольку считается, что все известные NP-задачи не имеют эффективного решения. Доказательство, полученное таким способом, создает сильную уверенность в стойкости исследуемой криптосистемы. Эта методология рассматривается в главах 14 и 15.

4.9 Резюме

Вычислительная сложность — наиболее важная часть современной криптографии. По этой причине главу, посвященную теории вычислительной сложности, следует рассматривать как самостоятельный и самодостаточный справочник.

Мы начали с понятия вычислимости по Тьюрингу и рассмотрели класс задач, для решения которых применяется машина Тьюринга. Некоторые задачи из этого класса являются разрешимыми (эффективно разрешимыми за полиномиальное время). Они могут быть как детерминированными (класс $\mathcal{P}$), так и недетерминированными (вероятностные полиномиальные задачи из некоторых подклассов класса $\mathcal{PP}$). Остальные задачи являются трудноразрешимыми (в разделе 18.2.3 будет показано, что класс $\mathcal{NP}$ является подклассом класса $\mathcal{PP}$). Задачи из класса $\mathcal{NP}$ не решаются ни детерминированными, ни вероятностными алгоритмами, хотя их принадлежность классу эффективно верифицируется с помощью вспомогательной информации.

Мы ввели понятия вычислительной сложности и описали их приложения в современной криптографии. К ним относятся эффективные алгоритмы (некоторые важные алгоритмы имеют точные оценки временно́й сложности), O-символика, полиномиальная редукция, пренебрежимо малые величины, нижние, верхние и неполиномиальные оценки, а также неразличимость. Эти понятия будут часто использоваться в остальной части книги.

В заключительной части главы рассмотрена фундаментальная роль, которую играют NP-задачи в современной криптографии.

Упражнения

4.1. Постройте машину Тьюринга для распознавания четных целых чисел. Затем постройте машину для распознавания целых чисел, кратных 6.

Подсказка: вторая машина может использовать таблицу операций, сопряженную с первой таблицей и таблицей операций машины Div3, изображенной на рис. 4.2.

4.2. Почему при оценке вычислительной сложности алгоритма более предпочтительной является побитовая сложность, основанная на подсчете битовых операций, а не сложность, основанная на подсчетах количества операций умножения целых чисел?

Подсказка: рассмотрите задачу, варианты которой имеют разные размеры.

4.3. По теореме 4.1 для вычисления $\gcd(x, y)$, где $x > y$, необходимо выполнить $\log x$ операций деления по модулю. Учитывая, что побитовая сложность операции деления по модулю имеет порядок $O_B((\log x)^2)$, для вычисления $\gcd(x, y)$ необходимо выполнить $O_B((\log x)^3)$ поразрядных операций. Однако, как правило, в учебниках указывается, что затраты на вычисление $\gcd(x, y)$ имеют порядок $O_B((\log x)^2)$. В чем заключается ошибка?

Подсказка: обратите внимание на неравенство (4.3.12).

4.4. Докажите утверждения 2 и 3 в теореме 4.2.

4.5. Докажите, что классы $\mathcal{PP}$(Монте-Карло) и $\mathcal{PP}$(Лас-Вегас) являются взаимно дополняющими друг друга (т.е. $\mathcal{PP}$(Монте-Карло) = **co**$\mathcal{PP}$(Лас-Вегас)). Иначе говоря, докажите, что алгоритм Монте-Карло для распознавания предложения $I \in L$ является алгоритмом Лас-Вегаса для распознавания $I \in \bar{L}$, и наоборот. Используя тот же прием, докажите, что $\mathcal{BPP} = \mathbf{co}\mathcal{BPP}$.

4.6. В литературе, посвященной теории вычислительной сложности, часто можно прочитать утверждение, что в определении класса $\mathcal{BPP}$ в неравенстве (4.4.1) $\varepsilon = \frac{2}{3}$, а в неравенстве (4.4.2) $\beta = \frac{1}{3}$. В нашей книге дано более широкое определение: $\varepsilon \in \left[\frac{1}{2} + \alpha, 1\right)$, $\delta \in \left(0, \frac{1}{2} - \beta\right]$, где $\alpha > 0$ и $\beta > 0$. Насколько существенной является эта разница?

4.7. Покажите, что в выражении (4.4.5) $\varepsilon(k) \to 1$ при $k \to \infty$.

Подсказка: докажите, что $1 - \varepsilon(k) \to 0$.

4.8. Объясните, почему вероятности ошибок в классе $\mathcal{BPP}$ не должны равняться $\frac{1}{2}$, т.е. числа α и β в выражении (4.4.11) не должны равняться нулю.

Подсказка: рассмотрите "несимметричную" монету, у которой вероятность выпадения одной стороны на ничтожно малую величину превышает вероятность выпадения другой. Можно ли определить наиболее вероятный исход, подбрасывая монету и применяя критерий "голосования большинством"?

4.9. Оценивая непротиворечивость протокола QKD, мы указали, что Ева может применять две стратегии: посылать Бобу совершенно новые состояния m фотонов или пересылать ему перехваченные состояния без изменения. Мы оценили непротиворечивость протокола, только если Ева придерживается второй стратегии. Оцените непротиворечивость протокола, если Ева применяет первую стратегию.

4.10. Для положительного целого числа n величина $|n| = \log_2 n$ представляет собой оценку его длины, т.е. количества битов в двоичном представлении числа n. Однако в большинстве случаев размер числа n можно записать в виде $\log n$ (т.е. по основанию натурального логарифма). Покажите, что для любого основания $b > 1$ величина $\log_b n$ правильно оценивает размер числа n, т.е. выражение "полиномиальный размер числа n" остается инвариантным для любого основания $b > 1$.

4.11. Иногда положительное число записывают в унарном виде, т.е. число n представляют в виде 1^n. Зачем это нужно?

4.12. Что такое эффективный алгоритм? Что такое практически эффективный алгоритм?

4.13. Докажите корректность алгоритма 4.8, используя свойства функции Эйлера $\phi(N)$ (см. раздел 6.3).

4.14. Приведите два примера неразличимых ансамблей.

4.15. Почему криптосистема, основанная на решении NP-полной задачи, не обязательно является стойкой?

4.16. Дайте определения и раскройте взаимосвязи между следующими понятиями.

а) Вычислимость по Тьюрингу.

б) Трудноразрешимая задача.

в) Разрешимая задача.

г) Детерминированное полиномиальное время.

д) Практическая эффективность.

Глава 5

Алгебраические основы

5.1 Введение

Криптографические алгоритмы и протоколы обрабатывают сообщения в виде чисел или элементов конечного пространства. При этом операции кодирования (шифрования) и декодирования (расшифровки), переводящие сообщения из одного вида в другой, должны обладать свойством *замкнутости* в конечном пространстве (closure property). Однако обычные арифметические операции над числами — сложение, вычитание, умножение и деление — не обладают этим свойством. По этой причине криптографические алгоритмы, действующие в конечном пространстве сообщений, как правило, не сводятся лишь к обычным арифметическим операциям над числами. Как правило, они работают в пространствах, обладающих определенной алгебраической структурой и обеспечивающих реализацию свойства замкнутости.

В главе рассматриваются три алгебраические структуры, которые не только представляют собой основные понятия абстрактной алгебры, но и являются фундаментом современной криптографии и криптографических протоколов. К этим структурам относятся группа, кольцо и поле.

5.1.1 Структурная схема главы

В разделе 5.2 изучаются группы, в разделе 5.3 — кольца и поля, а в разделе 5.4 — структура конечных полей. В заключение, в разделе 5.5 рассматривается реализация конечной группы с помощью точек на эллиптической кривой.

5.2 Группы

Кратко говоря, группа — это множество, для каждой пары элементов которого определена некая операция. Эта структура вполне естественна. Например, в старину каждый вечер пастухи должны были пересчитывать свое стадо овец. Несмотря на то что пастух мог не знать арифметики, он обязан был выполнить определенную операцию. Он выбирал камешки из мешка и каждому камешку ставил

в соответствие отдельную овцу. Если количество камешков совпадало с количеством овец, пастух спокойно отправлялся спать. Пастух, не зная этого, генерировал группу, используя операцию "добавить единицу". При этом вид объектов, используемых для сопоставления, не имел никакого значения: главное — чтобы была определена операция над множеством объектов, и результат этой операции оставался внутри множества.

Определение 5.1 (Группа). *Группой* $(G, \circ)$ *называется пара, состоящая из множества* G *и операции* $\circ$*, удовлетворяющей следующим условиям.*

1. $\forall a, b \in G : a \circ b \in G$ (аксиома замкнутости).
2. $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$ (аксиома ассоциативности).
3. $\exists$ *единственный элемент* $e \in G : \forall a \in G : a \circ e = e \circ a = a$ (аксиома тождества).

   Элемент e *называется* ***нейтральным*** *(identity element).*
4. $\forall a \in G : \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$ (аксиома инверсии).

В обозначении группы $(G, \circ)$ операция $\circ$ часто пропускается, а группа обозначается символом G.

Определение 5.2 (Конечные и бесконечные группы). *Группа* G *называется конечной, если множество* G *состоит из конечного количества элементов, в противном случае группа* G *называется бесконечной.*

Определение 5.3 (Абелева группа). *Группа* G *называется абелевой, если для любых* $a, b \in G : a \circ b = b \circ a$.

Иначе говоря, абелева группа является **коммутативной** (commutative). В книге не рассматриваются некоммутативные группы, поэтому мы часто будем опускать прилагательное "абелева".

Пример 5.1 (Группы).

1. Множество целых чисел $\mathbb{Z}$ является группой относительно операции $+$, т.е. пара $(\mathbb{Z}, +)$ — это группа, где $e = 0$ и $a^{-1} = -a$. Эта группа является **аддитивной** (additive), бесконечной и абелевой. Множество рациональных чисел $\mathbb{Q}$, множество действительных чисел $\mathbb{R}$ и множество комплексных чисел $\mathbb{C}$ также являются аддитивными и бесконечными группами, в которых нейтральный и обратный элементы определены по тем же правилам.
2. Ненулевые элементы множеств $\mathbb{Q}$, $\mathbb{R}$ и $\mathbb{C}$ по отношению к умножению $\cdot$ являются группами, в которых $e = 1$ и a^{-1} является мультипликативной инверсией, определенной по обычным правилам. Обозначим эти группы через

$\mathbb{Q}^*$, $\mathbb{R}^*$ и $\mathbb{C}^*$ соответственно. Итак, полные обозначения этих групп выглядят следующим образом: $(\mathbb{Q}^*, \cdot)$, $(\mathbb{R}^*, \cdot)$ и $(\mathbb{C}^*, \cdot)$. Эти группы называются **мультипликативными** (multiplicative) и они бесконечны.

3. Для любого целого числа $n \geqslant 1$ множество целых чисел по модулю n образует конечную аддитивную группу, состоящую из n элементов. Нейтральным элементом этой группы является число 0, причем для любого элемента a выполняется условие $a^{-1} = n - a$ (свойство 2 теоремы 4.2 из раздела 4.3.2.5). Эта группа обозначается как $\mathbb{Z}_n$. Итак, полное обозначение этой группы выглядит следующим образом: $(\mathbb{Z}_n, +(\mathrm{mod}\, n))$. (Обратите внимание на то, что $\mathbb{Z}_n$ — это сокращение стандартного обозначения $\mathbb{Z}/n\mathbb{Z}$. Причина, по которой вводится такое обозначение, будет указана в примере 5.5.)

4. Числа, обозначающие часы на циферблате, образуют группу $\mathbb{Z}_{12}$ по отношению к сложению по модулю 12. Будем называть группу $(\mathbb{Z}_{12}, +(\mathrm{mod}\, 12))$ "*часовой группой*" ("clock group").

5. Подмножество множества $\mathbb{Z}_n$, содержащее элементы, взаимно простые с числом n (т.е. $\gcd(a, n) = 1$), образует конечную мультипликативную группу. Операция умножения выполняется по модулю n, нейтральный элемент $e = 1$, и для любого элемента a из этой группы обратный элемент a^{-1} можно вычислить с помощью алгоритма Евклида (алгоритм 4.2). Эта группа обозначается как $\mathbb{Z}_n^*$. Например, $(\mathbb{Z}_n^*, \cdot(\mathrm{mod}\, 15)) = (\{1, 2, 4, 7, 8, 11, 13, 14\}, \cdot (\mathrm{mod}\, 15))$.

6. Для пары множеств $B = \{F, T\}$ обозначим через $\circ$ операцию $\oplus$ (логическое "исключающее или"): $F \oplus F = F$, $F \oplus T = T \oplus F = T$, $T \oplus T = F$. Тогда пара множеств B относительно операции $\oplus$ является конечной группой, в которой $e = F$ и $T^{-1} = T$.

7. Корни уравнения $x^3 - 1 = 0$ образуют конечную группу относительно умножения, в которой $e = 1$ (очевидно, что единица является корнем этого уравнения). Обозначим эту группу как $\mathrm{Roots}(x^3 - 1)$. Найдем другие элементы группы $\mathrm{Roots}(x^3 - 1)$, а также обратные к ним элементы. Поскольку $x^3 - 1$ является полиномом третьей степени, он имеет только три корня. Обозначим неизвестные нам корни буквами α и β. Из соотношения $x^3 - 1 = (x - 1)(x^2 + x + 1)$ следует, что числа α и β должны быть корнями уравнения $x^2 + x + 1 = 0$. Используя зависимость между корнями и коэффициентами квадратного уравнения, получаем, что $\alpha\beta = 1$. Следовательно, $\alpha^{-1} = \beta$ и $\beta^{-1} = \alpha$. Читатели могут самостоятельно убедиться, что аксиома замкнутости выполняется (т.е. α^2 и β^2 также являются корнями уравнения $x^3 - 1 = 0$). □

Определение 5.4 (Сокращенное представление повторяющейся операции группы). *Пусть G — группа с операцией $\circ$. Для любого элемента $a \in G$ и любого неотрицательного целого числа $i \in \mathbb{N}$ обозначим через $a^i \in G$ элемент*

$$\underbrace{a \circ a \circ \cdots \circ a}_{i}.$$

Примечание 5.1.

1. *Выражение $a^i \in G$ используется только в качестве сокращенного представления повторяющейся операции $\underbrace{a \circ a \circ \cdots \circ a}_{i}$. Следует помнить, что эта "операция" **не является** операцией группы.*
2. *Некоторые группы для удобства записываются в аддитивном виде, например $(\mathbb{Z}_n, +(\bmod n))$. В этом случае элемент a^i означает $i \cdot a$. Однако и в этой ситуации следует иметь в виду, что операция $\cdot$ **не является** операцией группы, а элемент i, как правило, **не является** элементом группы (например, в группе $(\mathbb{Z}_n, +(\bmod n))$ при $i > n$).* □

Определение 5.5 (Подгруппа). *Подгруппой группы G называется непустое подмножество H множества G, которое само является группой относительно той же операции, что и группа G. Для того чтобы подчеркнуть, что H является подгруппой группы G, используется запись $H \subseteq G$. Если H является собственной подгруппой группы G, используется запись $H \subset G$ (т.е. $H \neq G$).*

Пример 5.2 (Группы).

1. Относительно сложения $\mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R} \subseteq \mathbb{C}$.
2. Относительно сложения множество четных чисел и нуль образуют подгруппу каждой из групп, перечисленных выше.
3. "Часовая группа" $(\mathbb{Z}_{12}, +(\bmod 12))$ имеет следующие подгруппы: $(\{0\}, +)$, $(\{0, 6\}, +)$, $(\{0, 4, 8\}, +)$, $(\{0, 3, 6, 9\}, +)$, $(\{0, 2, 4, 6, 8, 10\}, +)$, $(\mathbb{Z}_{12}, +)$.
4. Относительно умножения $\mathbb{Q}^* \subseteq \mathbb{R}^* \subseteq \mathbb{C}^*$.
5. Пусть n — нечетное положительное целое число, а $\mathrm{Fermat}(n)$ — подмножество множества $\mathbb{Z}_n^*$ такое, что любой элемент $a \in \mathrm{Fermat}(n)$ удовлетворяет условию $a^{\frac{n-1}{2}} \equiv \pm 1 \bmod n$. Тогда

$$\mathrm{Fermat}(n) \subseteq \mathbb{Z}_n^*.$$

Более того, если число n — целое, то по малой теореме Ферма (теорема 6.10 в разделе 6.4) $\mathrm{Fermat}(n) = \mathbb{Z}_n^*$, в противном случае $\mathrm{Fermat}(n)$ является собственной подгруппой группы $\mathbb{Z}_n^*$.

6. В примере 5.1.6 множество $\{F\}$ является собственной подгруппой группы B. Однако множество $\{T\}$ не является подгруппой группы B, поскольку не содержит нейтрального элемента (т.е. нарушается аксиома тождества).
7. (См. пример 4.1.) Полиномиальный язык DIV3 является подгруппой группы $\mathbb{Z}$.
8. Множество $\{e\}$ является подгруппой любой группы. □

Определение 5.6 (Порядок группы). *Порядком $\#G$ конечной группы G называется количество ее элементов.*

Пример 5.3 (Группы).

1. $\#\mathbb{Z}_n = n$.
2. В примере 5.1.6 $\#B = 2$.
3. В примере 5.1.7 $\#\,\mathrm{Roots}(x^3 - 1) = 3$. □

5.2.1 Теорема Лагранжа

Рассмотрим одну из замечательных теорем теории групп.

Определение 5.7 (Смежный класс). *Пусть G — абелева группа и $H \subseteq G$. При $a \in G$ множество $a \circ H \stackrel{\text{def}}{=} \{a \circ h \mid h \in H\}$ называется левым смежным классом (coset) подгруппы H.*

Теорема 5.1 (Теорема Лагранжа). *Если H — подгруппа группы G, то $\#H \mid \#G$, т.е. число $\#H$ является делителем числа $\#G$.*

Доказательство. Если $H = G$ условие $\#H \mid \#G$ выполняется тривиальным образом. Рассмотрим вариант, когда $H \neq G$.

По аксиоме замкнутости для любого элемента $a \in G \setminus H$ смежный класс $a \circ H$ является подмножеством группы G. Докажем два факта.

1. Если для любого элемента $a \neq a'$ выполняется условие $a \notin a' \circ H$, то $(a \circ H) \cap (a' \circ H) = \emptyset$.
2. $\#(a \circ H) = \#H$.

Для доказательства первого факта предположим, что $\exists b \in (a \circ H) \cap (a' \circ H)$. Тогда $\exists c, c' \in H : a \circ c = b = a' \circ c'$. Применяя аксиомы инверсии, тождества, замкнутости и ассоциативности, получаем

$$a = a \circ e = a \circ (c \circ c^{-1}) = b \circ c^{-1} = (a' \circ c') \circ c^{-1} = a' \circ (c' \circ c^{-1}) \in a' \circ H.$$

Это противоречит предположению, что $a \notin a' \circ H$. Если же $a \notin H = e \circ H$, то $H \cap (a \circ H) = \emptyset$.

Для доказательства второго факта отметим, что условие $\#(a \circ H) \leqslant \#H$ тривиальным образом вытекает из определения смежного класса. Допустим, что неравенство является строгим. Это возможно лишь тогда, когда для некоторых $b \neq c$, где $b, c \in H$, выполняется условие $a \circ b = a \circ c$. Применяя к группе G аксиому инверсии, получаем, что $b = c$, что противоречит условию $b \neq c$.

Итак, группа G разбивается на подгруппу H и семейство ее попарно непересекающихся смежных классов, каждый из которых имеет размер $\#H$. Следовательно, $\#H \mid \#G$. (Разбиение множества означает его разделение на непересекающиеся подмножества.) □

Пример 5.4.

1. Проверим пример 5.2.3: условие $\#H \mid \#\mathbb{Z}_{12}$ выполняется для каждой подгруппы H "часовой группы" $\mathbb{Z}_{12}$.
2. Конкретизируем пример 5.2.5 для случая, когда $n = 21$. Тогда подгруппа $\mathrm{Fermat}(21) = \{1, 8, 13, 20\}$ удовлетворяет условию $\#\,\mathrm{Fermat}(21) = 4 \mid 12 = \#\mathbb{Z}_{21}^*$. □

Теорема Лагранжа имеет очень большое значение для приложений. Вернемся к вероятностному алгоритму проверки простоты числа Prime_Test, рассмотренному в разделе 4.4.3.1. Этот алгоритм проверяет, является ли нечетное целое число n простым, выполняя сравнение

$$x^{(n-1)/2} \equiv \pm 1 \bmod n,$$

где $x \in_U \mathbb{Z}_n^*$ — случайное число. Анализируя пример 5.2.5, мы убедились, что множество $\mathrm{Fermat}(n)$ является подгруппой группы $\mathbb{Z}_n^*$, определенной этим сравнением, причем она является собственной подгруппой группы $\mathbb{Z}_n^*$ тогда и только тогда, когда число n не является простым. Следовательно, по теореме Лагранжа $\#\,\mathrm{Fermat}(n) \mid \#\mathbb{Z}_n^*$. Значит, если число n — не простое, величина $\#\,\mathrm{Fermat}(n)$ не превышает половины числа $\#\mathbb{Z}_n^*$. Итак, вероятность ошибки на каждом шаге проверки не превышает $\frac{1}{2}$. Это гарантирует работоспособность алгоритма Prime_Test, вероятностным пространством которого является группа $\mathbb{Z}_n^*$.

Другое важное применение теоремы Лагранжа в криптографии с открытым ключом будет рассмотрено в разделе 5.2.2.

Определение 5.8 (Фактор-группа). *Пусть G — абелева группа и $H \subseteq G$. Фактор-группой G/H группы G по модулю H называется множество всех смежных классов $a \circ H$, где элемент a пробегает все множество G с групповой операцией $*$, определенной по правилу $(a \circ H) * (b \circ H) = (a \circ b) \circ H$, и нейтральным элементом $e \circ H$.*

Пример 5.5. Пусть $n > 0$ — целое число. Множество $n\mathbb{Z} = \{0, \pm n, \pm 2n, \dots\}$ является подгруппой группы $\mathbb{Z}$ относительно целочисленного сложения. Факторгруппа

$$\mathbb{Z}/n\mathbb{Z} = \{x + n\mathbb{Z} \mid x \in \mathbb{Z}\}$$

может состоять только из конечного количества элементов, поскольку $n + n\mathbb{Z} = 0 + n\mathbb{Z}, n + 1 + n\mathbb{Z} = 1 + n\mathbb{Z}$ и т.д. Следовательно,

$$\mathbb{Z}/(n\mathbb{Z}) = \{0 + n\mathbb{Z}, 1 + n\mathbb{Z}, 2 + n\mathbb{Z}, \dots, n - 1 + n\mathbb{Z}\}.$$

Учитывая, что множество $n\mathbb{Z}$ может содержать только нуль по модулю n, приходим к выводу, что

$$\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}_n.$$

□

Обозначение $\mathbb{Z}/n\mathbb{Z}$ является стандартным для группы $\mathbb{Z}_n$. Однако для удобства в книге используется более короткое обозначение $\mathbb{Z}_n$.

Следствие 5.1. *Пусть G — конечная абелева группа и $H \subseteq G$. Тогда*

$$\#(G/H) = \frac{\#G}{\#H}.$$

□

Пример 5.6. Пусть m и n — положительные целые числа, удовлетворяющие условию $m \mid n$. Из примера 5.5 вытекают следующие утверждения.

1. Множество $m\mathbb{Z}_n = \{0, m, 2m, \dots, \lfloor \frac{n-1}{m} \rfloor \cdot m\}$ является подгруппой группы $(\mathbb{Z}_n, +)$ и состоит из n/m элементов.
2. $\mathbb{Z}_n/m\mathbb{Z}_n = \mathbb{Z}_m$.
3. $\#(\mathbb{Z}_n/m\mathbb{Z}_n) = \#\mathbb{Z}_m = m = \frac{n}{n/m} = \frac{\#\mathbb{Z}_n}{\#(m\mathbb{Z}_n)}$.

В качестве примера рассмотрим "часовую группу" $\mathbb{Z}_{12}$ (т.е. $n = 12$) и ее подгруппу $3\mathbb{Z}_{12} = \{0, 3, 6, 9\}$ (т.е. $m = 3$). Читатель может повторить анализ примера 5.5 и убедиться, что $\mathbb{Z}_{12}/3\mathbb{Z}_{12} = \mathbb{Z}_3$. Следовательно, $\#(\mathbb{Z}_{12}/3\mathbb{Z}_{12}) = \#\mathbb{Z}_3 = 3 = 12/4 = \frac{\#\mathbb{Z}_{12}}{\#(3\mathbb{Z}_{12})}$. Аналогично доказываются все остальные варианты для $m \mid 12$. □

5.2.2 Порядок элемента группы

Не только нейтральный, но и другие элементы группы могут иметь особые свойства. Одним из таких свойств является "расстояние" от нейтрального элемента.

Определение 5.9 (Порядок элемента группы). *Пусть G — группа и $a \in G$. Порядком элемента a называется наименьшее положительное число $i \in \mathbb{N}$, удовлетворяющее условию $a^i = e$. Это число обозначается* $\mathrm{ord}(a)$. *Если такое число i не существует, то элемент a называется элементом бесконечного порядка.*

Напомним, что обозначение a^i представляет собой сокращение, введенное в определении 5.4 и проиллюстрированное в замечании 5.1.

Пример 5.7.

1. В "часовой группе" $\mathbb{Z}_{12}\,\mathrm{ord}(1) = 12$, поскольку 12 — это наименьшее положительное число, удовлетворяющее условию $12 \cdot 1 \equiv 0 (\mathrm{mod}\, 12)$. Читатель может самостоятельно проверить, что $\mathrm{ord}(2) = 6, \mathrm{ord}(3) = 4, \mathrm{ord}(4) = 3, \mathrm{ord}(5) = 12$. Попробуйте определить порядки остальных элементов.
2. В группе B из примера 5.1.6 $\mathrm{ord}(F) = 1, a\,\mathrm{ord}(T) = 2$.
3. В группе $\mathrm{Roots}(x^3 - 1)$ из примера 5.1.7 $\mathrm{ord}(\alpha) = \mathrm{ord}(\beta) = 3, \mathrm{ord}(1) = 1$.
4. В группе $\mathbb{Z}\,\mathrm{ord}(1) = \infty$. □

Следствие 5.2. *Пусть G — конечная группа и $a \in G$ — произвольный элемент. Тогда* $\mathrm{ord}(a) \mid \#G$.

Доказательство. Для любого $a \in G$, если $a = e$, то $\mathrm{ord}(a) = 1$ и условие $\mathrm{ord}(a) \mid \#G$ является тривиальным. Элементы

$$a, a^2, \ldots, a^{\mathrm{ord}(a)} = e \tag{5.2.1}$$

должны быть разными. Допустим, что это не так, и $a^r = a^s$ при некоторых неотрицательных целых числах r и s, удовлетворяющих условию $1 \leqslant r < s \leqslant \mathrm{ord}(a)$. Применяя аксиому инверсии к обоим частям равенства, получаем, что $a^{s-r} = e$ при $0 < s - r < \mathrm{ord}(a)$. Это противоречит определению порядка $\mathrm{ord}(a)$, который должен быть наименьшим положительным целым числом, удовлетворяющим условию $a^{\mathrm{ord}(a)} = e$.

Легко проверить, что $\mathrm{ord}(a)$ элементов из соотношения (5.2.1) образуют подгруппу группы G. По теореме Лагранжа $\mathrm{ord}(a) \mid \#G$. ⊏

Следствие 5.2, вытекающее из теоремы Лагранжа, описывает связь между порядком группы и порядками элементов этой группы. Эта взаимосвязь имеет важное приложение в криптографии с открытым ключом: знаменитая криптосистема Ривеста, Шамира и Адлемана (RSA) [246] функционирует на основе группы, порядок которой является секретным и известен только владельцу ключа. Зашифрованный текст можно рассматривать как случайный элемент некоей группы. Зная порядок группы, владелец ключа может использовать зависимость

между порядком элемента и порядком группы для преобразования зашифрованного текста в открытый текст (т.е. расшифровать его). Криптосистема RSA изучается в разделе 8.5.

5.2.3 Циклические группы

Пример 5.1.4 означает, что группу $\mathbb{Z}_n$ можно представить в виде n точек, лежащих на окружности. Эта окружность (или n точек, лежащих на ней) создается путем повторного применения n операций $a^1, a^2, \ldots, a^n$ к некоторому элементу $a \in \mathbb{Z}$. Такая конструкция называется *циклическим представлением* группы $\mathbb{Z}_n$. Для операции по модулю n циклическое представление группы $\mathbb{Z}_n$ можно получить с помощью элемента $a = 1$. Читатель может проверить это утверждение, проанализировав пример 5.1.4 для ситуации, когда $n = 12$. Элементы 5, 7 и 11 также позволяют создать циклическое представление группы $\mathbb{Z}_{12}$.

Неформально говоря, если группа имеет циклическое представление, то она называется **циклической** (cyclic group). Такие группы обладают весьма полезными свойствами. Они получили широкое применение в криптографии.

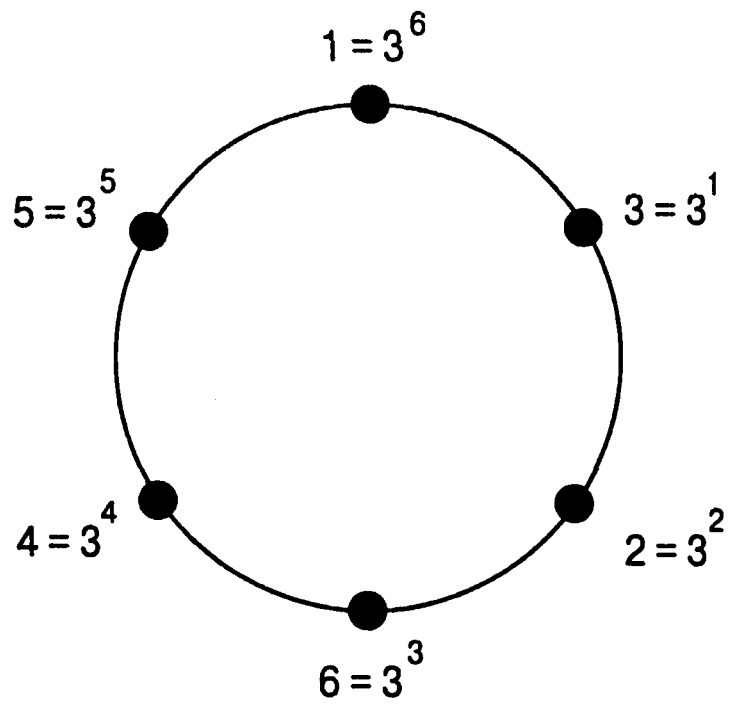
Определение 5.10 (Циклическая группа, порождающий элемент группы).
Группа G называется циклической, если существует элемент $a \in G$, такой что для любого элемента $b \in G$ существует целое число $i \geqslant 0$, удовлетворяющее условию $b = a^i$. Элемент a называется порождающим элементом группы G, а группа G называется группой, порожденной элементом a.
Если группа G порождается элементом a, используется запись $G = \langle a \rangle$.

Порождающий элемент (generator) циклической группы также называется **первообразным корнем** (primitive root) нейтрального элемента группы. Смысл этого названия станет ясен в разделе 5.4.3 (теорема 5.11).

Пример 5.8.

1. Для $n \geqslant 1$ аддитивная группа $\mathbb{Z}_n$ является циклической, поскольку совершенно очевидно, что ее порождающим элементом является единица.
2. В примере 5.1.6 группа B является циклической и порождается элементом T.
3. В примере 5.1.7 группа $\mathrm{Roots}(x^3 - 1)$ является циклической. Она порождается элементами α и β.
4. Пусть p — простое число. В этом случае мультипликативная группа $\mathbb{Z}_p^*$ является циклической, поскольку она содержит элемент $p - 1 = \#\mathbb{Z}_p^*$, и, следовательно, этот элемент порождает всю группу. В алгоритме 4.6 мы осуществили эмпирическую проверку того, что группа $\mathbb{Z}_p^*$ содержит порождающий элемент. Формальное доказательство этого факта содержится в теореме 5.12.

5. В группе $\mathbb{Z}_7^*$ порождающим элементом является число 3. Этот элемент создает циклическое представление группы $\mathbb{Z}_7^*$ (групповой операцией является умножение по модулю 7).



□

Определение 5.11 (Функция Эйлера). *Для $n \in \mathbb{N}$, где $n \geqslant 1$, функция Эйлера $\phi(n)$ равна количеству целых чисел k, таких что $0 \leqslant k < n$ и $\gcd(k, n) = 1$.*

Для циклической группы доказано много полезных результатов.

Теорема 5.2.

1. *Любая подгруппа циклической группы является циклической.*
2. *При каждом положительном делителе d числа $\#\langle a\rangle$ группа $\langle a\rangle$ содержит только одну подгруппу, имеющую порядок d.*
3. *Если $\#\langle a\rangle = m$, то $\#\langle a^k\rangle = \operatorname{ord}(a^k) = m/\gcd(k, m)$.*
4. *При каждом положительном делителе d числа $\#\langle a\rangle$ группа $\langle a\rangle$ содержит $\phi(n)$ элементов, имеющих порядок d.*
5. *Пусть $\#\langle a\rangle = m$. Тогда группа $\langle a\rangle$ содержит $\phi(m)$ порождающих элементов. Они являются элементами a^r, такими что $\gcd(r, m) = 1$.*

Доказательство.

1. Пусть $H \subseteq \langle a\rangle$. Если $H = \langle e\rangle$ или $H = \langle a\rangle$, то совершенно очевидно, что группа H является циклической. Рассмотрим другие варианты. Пусть $d > 1$ — наименьшее целое число, такое что $a^d \in H$, и $a^s \in H$ при некотором $s > d$. Разделим число s на число d: $s = dq + r$, где $0 \leqslant r < d$. Поскольку $a^{dq} \in H$, то $a^r = a^{s-dq} \in H$. Из минимальности числа d и неравенства $H \neq \langle a\rangle$ следует, что $r = 0$. Следовательно, число s кратно числу d. Итак, группа H содержит только степени a^d и является циклической.

2. Пусть $d > 1$ и $d \mid m = \#\langle a \rangle$. Тогда $\left\langle a^{\frac{m}{d}} \right\rangle$ — подгруппа группы $\langle a \rangle$, имеющая порядок d, поскольку d — наименьшее целое число, удовлетворяющее условию $\left\langle a^{\frac{m}{d}} \right\rangle^d = e$. Предположим, что существует другая подгруппа группы $\langle a \rangle$, имеющая порядок d и не совпадающая с подгруппой $\left\langle a^{\frac{m}{d}} \right\rangle$. Из пункта 1 следует, что такая подгруппа должна быть циклической и, следовательно, имеет вид $\langle a^k \rangle$ при некотором $k > 1$. Из равенства $a^{kd} = e$ и минимальности числа m следует, что $m \mid kd$, т.е. $\frac{m}{d} \mid k$. Таким образом, $a^k \in \left\langle a^{\frac{m}{d}} \right\rangle$, т.е. $\langle a^k \rangle \subseteq \left\langle a^{\frac{m}{d}} \right\rangle$. Поскольку обе группы имеют одинаковый порядок, $\langle a^k \rangle = \left\langle a^{\frac{m}{d}} \right\rangle$. Это противоречит нашему предположению $\langle a^k \rangle \neq \left\langle a^{\frac{m}{d}} \right\rangle$.

3. Пусть $d = \gcd(k, m)$. Тогда согласно п. 2 существует единственная подгруппа группы $\langle a \rangle$, имеющая порядок d. Допустим, что ею является подгруппа $\langle a^\ell \rangle$ при некотором $\ell > 1$, где ℓ — наименьшее целое число, удовлетворяющее условию $a^{dl} = e$. Из условия минимальности числа m следует, что $m \mid dl$, т.е. $\frac{m}{d} \mid \ell$. Наименьшее значение числа ℓ достигается, когда $d = \gcd(\ell, m)$, т.е. $\ell = k$.

4. Пусть $d \mid m = \#\langle a \rangle$ и a^k — произвольный элемент группы $\langle a \rangle$ при $0 \leqslant k < m$. Из п. 3 следует, что элемент a^k имеет порядок $\frac{m}{d}$ тогда и только тогда, когда $\frac{m}{d} = \gcd(k, m)$. Представим число k в виде $k = c\frac{m}{d}$, где $0 \leqslant c < d$. Тогда условие $\gcd(k, m) = \frac{m}{d}$ означает, что $\gcd(c, d) = 1$. По определению 5.11 существует $\phi(d)$ таких чисел c.

5. В силу п. 4 из условия $m = \#\langle a \rangle$ следует, что группа $\langle a \rangle$ содержит $\phi(m)$ элементов порядка m, порождающих группу $\langle a \rangle$. Согласно п. 3 этими порождающими элементами являются числа a^r, удовлетворяющие условию $\gcd(r, m) = 1$. □

Следствие 5.3. *Группа, имеющая простой порядок, циклична, и любой ее элемент, не совпадающий с нейтральным, является порождающим.*

Доказательство. Пусть G — группа с простым порядком p. Пусть $a \in G$ — произвольный элемент, не являющийся нейтральным. Из следствия 5.2 вытекает, что $\mathrm{ord}(a) \mid \#G = p$. Поскольку $a \neq e, \mathrm{ord}(a) \neq 1$. Тогда $\mathrm{ord}(a) = p$. Следовательно, $\langle a \rangle = G$, т.е. число a порождает группу G. □

Пример 5.9. Рассмотрим "часовую группу" $\mathbb{Z}_{12}$, которая является циклической.

1. Поскольку $1 \mid 12$, она содержит подгруппу $\{0\}$, имеющую порядок 1. Так как $\phi(1) = 1$, единственным элементом, имеющим порядок 1, является число 0.
2. Поскольку $2 \mid 12$, она содержит подгруппу $\{0, 6\}$, имеющую порядок 2. Так как $\phi(2) = 1$, единственным элементом, имеющим порядок 2, является число 6.
3. Поскольку $3 \mid 12$, она содержит подгруппу $\{0, 4, 8\}$, имеющую порядок 3. В группе существует $\phi(3) = 2$ элемента, имеющих порядок 3: ими являются числа 4 и 8.
4. Поскольку $4 \mid 12$, она содержит подгруппу $\{0, 3, 6, 9\}$, имеющую порядок 4. В группе существует $\phi(4) = 2$ элемента, имеющих порядок 4: ими являются числа 3 и 9.
5. Поскольку $6 \mid 12$, она содержит подгруппу $\{0, 2, 4, 6, 8, 10\}$, имеющую порядок 6. В группе существует $\phi(6) = 2$ элемента, имеющих порядок 6: ими являются числа 2 и 10.
6. Поскольку $12 \mid 12$, она содержит подгруппу, имеющую порядок 12. В группе существует $\phi(12) = 4$ элемента, имеющих порядок 12: ими являются числа 1, 5, 7 и 11.

Мультипликативная группа $\mathbb{Z}_7^*$ анализируется аналогично. □

5.2.4 Мультипликативная группа $\mathbb{Z}_n^*$

Пусть $n = pq$, где p и q — разные нечетные простые числа. Мультипликативная группа $\mathbb{Z}_n^*$ играет очень важную роль в современной криптографии. Рассмотрим ее структуру. На протяжении всего раздела будем считать, что число n является составным.

Элементами группы $\mathbb{Z}_n^*$ являются положительные целые числа, которые меньше числа n и являются взаимно простыми с ним. По определению 5.11 эта группа состоит из $\phi(n) = (p-1)(q-1)$ элементов (см. лемму 6.1).

Теорема 5.3. *Порядок любого элемента в группе $\mathbb{Z}_n^*$ является делителем числа* $\operatorname{lcm}(p-1, q-1)$.

Доказательство. Пусть $a \in \mathbb{Z}_n^*$. Из малой теоремы Ферма (теорема 6.10 из раздела 6.4) следует, что

$$a^{(p-1)} \equiv 1 (\operatorname{mod} p).$$

Обозначая $\lambda = \operatorname{lcm}(p-1, q-1)$, получаем, что

$$a^{\lambda} \equiv 1 (\operatorname{mod} p).$$

Аналогично

$$a^{\lambda} \equiv 1 (\mathrm{mod}\, q).$$

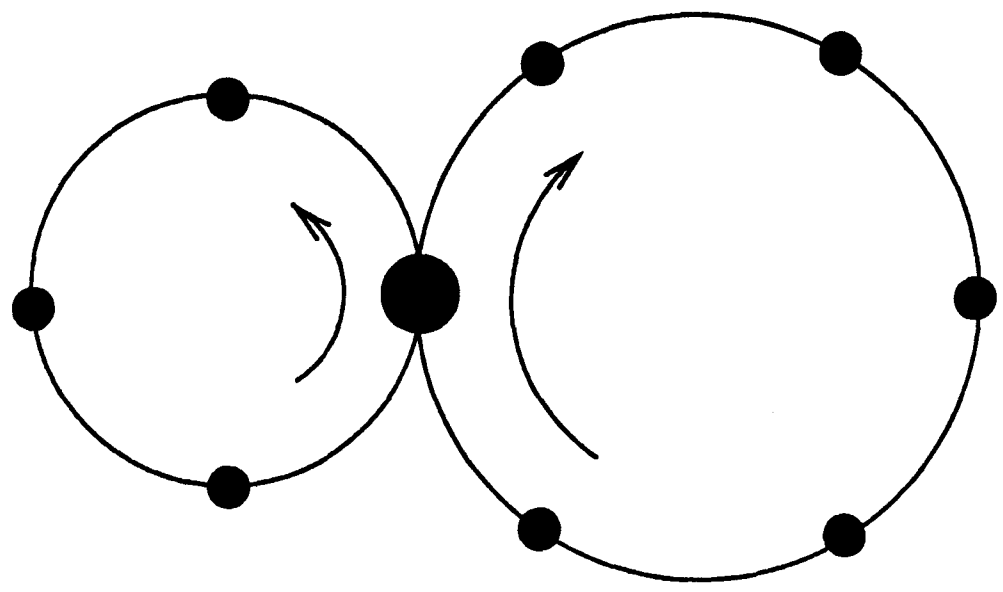
Эти сравнения означают, что число $a^{\lambda} - 1$ кратно числам p и q. Поскольку числа p и q являются простыми и не равны друг другу, число $a^{\lambda} - 1$ должно кратным числу $n = pq$. Следовательно,

$$a^{\lambda} \equiv 1 (\mathrm{mod}\, n).$$

Итак, число λ кратно порядку числа a по модулю n. □

Обратите внимание на то, что числа $p - 1$ и $q - 1$ являются четными. Отсюда следует, что $\lambda = \mathrm{lcm}(p - 1, q - 1) < (p - 1)(q - 1) = \phi(n)$. Из теоремы 5.3 следует, что в группе $\mathbb{Z}_n^*$ не существует элемента, имеющего порядок $\phi(n)$. Иначе говоря, группа $\mathbb{Z}_n^*$ не содержит порождающий элемент. Следовательно, по определению 5.10 группа $\mathbb{Z}_n^*$ не является циклической. Величина $\lambda(n)$ называется **числом Кармайкла** (Carmichael number).

Пример 5.10. Пусть $n = 5 \times 7 = 35$, а элемент $a \in \mathbb{Z}_{35}^*$ удовлетворяет следующим условиям: 1) число $a(\mathrm{mod}\, 5) \in \mathbb{Z}_5^*$, и его максимальный порядок равен 4, т.е. это число порождает циклическое представление группы $\mathbb{Z}_5^*$ (левая окружность с периодом 4); 2) число $a(\mathrm{mod}\, 7) \in \mathbb{Z}_7^*$, и его максимальный порядок равен 6, т.е. это число порождает циклическое представление группы $\mathbb{Z}_7^*$ (правая окружность с периодом 6).



Таким образом, порядок элемента $a \in \mathbb{Z}_{35}^*$ можно представить в виде периода вращения двух сцепленных шестеренок. Одна из шестеренок имеет четыре зубца, а другая — шесть. Точка сцепления изображена на рисунке более крупной. Представим себе, что шестеренки начали вращаться. Тогда крупная точка разделится на две разные точки, принадлежащие разным окружностям. Эти точки встретятся снова после того, как шестеренка с четырьмя зубьями сделает три оборота, а шестеренка с шестью зубьями — два. Следовательно, порядок (период) элемента $a \in \mathbb{Z}_{35}^*$ равен расстоянию, пройденному точками между моментом разделения и моментом воссоединения: $3 \times 4 = 2 \times 6 = \mathrm{lcm}((5 - 1), (7 - 1))$. □

Обозначим через $\mathrm{ord}_x(a)$ порядок элемента по положительному модулю n. Любой элемент $a \in \mathbb{Z}_n^*$ имеет порядок $\mathrm{ord}_n(a)$, который связан с числами $\mathrm{ord}_p(a)$ и $\mathrm{ord}_q(a)$ соотношением

$$\mathrm{ord}_n(a) = \mathrm{lcm}(\mathrm{ord}_p(a), \mathrm{ord}_q(a)). \tag{5.2.2}$$

Поскольку группы $\mathbb{Z}_p^*$ и $\mathbb{Z}_q^*$ являются циклическими, максимальный порядок их элементов равен $p-1$ и $q-1$ соответственно. Следовательно, максимальный порядок элементов в группе $\mathbb{Z}_n^*$ равен $\mathrm{lcm}(p-1, q-1)$. С другой стороны, некий элемент $a \in \mathbb{Z}_n^*$, имеющий максимальный порядок, может удовлетворять условиям $\mathrm{ord}_p(a) < p-1$ и/или $\mathrm{ord}_q(a) < q-1$. Например, поскольку $\mathrm{lcm}(4,3) = \mathrm{lcm}(4,6)$, а группа $\mathbb{Z}_7^*$ содержит элемент, имеющий порядок 3, максимальный порядок элементов в группе $\mathbb{Z}_{35}^*$ равен 12. Этот порядок можно представить с помощью двух сцепленных шестеренок, имеющих четыре и три зубца.

В следующей главе мы введем взаимно-однозначное соответствие между элементами группы $\mathbb{Z}_n^*$ и парами элементов, принадлежащих множеству $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$. Это отображение является вычислимым и позволяет сконструировать элементы группы $\mathbb{Z}_n^*$, не входящие в циклические группы $\mathbb{Z}_p^*$ и $\mathbb{Z}_q^*$. Благодаря некоторым особенностям циклических групп $\mathbb{Z}_p^*$ и $\mathbb{Z}_q^*$ это задание оказывается довольно легким. Например, из-за того, что квадратные корни в группах $\mathbb{Z}_p^*$ и $\mathbb{Z}_q^*$ вычисляются просто, это отображение можно применять для создания квадратных корней в группе $\mathbb{Z}_n^*$.

5.3 Кольца и поля

В один прекрасный день наш древний пастух перестал кочевать и стал земледельцем. Теперь ему понадобилось размежевать свои и соседские участки земли. И тут бывший пастух понял, что одной операцией он не обойдется: придется выполнять не только сложение, но и умножение. С этого момента появилась необходимость выполнять *две* операции над множеством объектов.

Определение 5.12 (Кольцо). *Кольцом R называется множество с двумя операциями: (сложение) $+$ и (умножение) $\cdot$, обладающими следующими свойствами.*

1. *Относительно операции сложения $+$ кольцо R является абелевой группой. Нейтральный элемент относительно сложения (**нуль**) обозначается символом* $\mathbf{0}$.
2. *Относительно операции умножения $\cdot$ кольцо R удовлетворяет аксиомам замкнутости, ассоциативности и аксиоме тождества. Нейтральный элемент относительно умножения (**единица**) обозначается символом* $\mathbf{1}$, *причем* $\mathbf{1} \neq \mathbf{0}$.

3. $\forall a, b \in R : a \cdot b = b \cdot a$ (аксиома коммутативности).
4. $\forall a, b, c \in R : a \cdot (b + c) = a \cdot b + a \cdot c$ (аксиома дистрибутивности).

Следует иметь в виду, что нуль и единица — абстрактные элементы, и не обязательно являются числами (см. пример 5.11.3).

Кольца, описанные в определении 5.12, называются **коммутативными** (commutative ring). Именно такие кольца рассматриваются в книге. Следует иметь в виду, что операции $+$ и $\cdot$ являются абстрактными: они не обязательно означают сложение и умножение целых чисел. Если это не вызывает недоразумений, операцию $a \cdot b$ мы будем записывать как ab. Обозначение $\cdot$ используется только в тех ситуациях, когда операнды не указаны.

Пример 5.11 (Кольца).

1. Множества $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ и $\mathbb{C}$ являются кольцами относительно обычных операций сложения и умножения, причем $\mathbf{0} = 0$ и $\mathbf{1} = 1$.
2. Для любых $n > 0$ группа $\mathbb{Z}_n$ является кольцом относительно сложения и умножения по модулю n, причем $\mathbf{0} = 0$ и $\mathbf{1} = 1$.
3. Пусть B — аддитивная группа, определенная в примере 5.1.6 с нулевым элементом F. Введем операцию умножения $\wedge$ (логическая операция "И"): $F \wedge F = F$, $F \wedge T = T \wedge F = F$, $T \wedge T = T$. Тогда группа B становится кольцом с единицей T. □

На первый взгляд определение 5.12 вводит умножение только для ненулевых элементов. Однако правило умножения нулевого элемента на ненулевой элемент следует из аксиомы дистрибутивности. Например, $\mathbf{0}a = (b + (-b))a = ba + (-b)a = ba - ba = \mathbf{0}$. Более того, кольцо может содержать **нулевой делитель** (zero-divisor), т.е. могут существовать элементы a и b, удовлетворяющие условию $ab = \mathbf{0}$ с $a \neq \mathbf{0}$ и $b \neq \mathbf{0}$. Например, для нетривиальной факторизации числа $n = k\ell$ числа k и ℓ являются ненулевыми элементами кольца $\mathbb{Z}_n$, а произведение $k\ell = n = 0 (\mathrm{mod}\, n)$ — нулем.

Определение 5.13 (Поле). *Если ненулевые элементы кольца образуют группу относительно умножения, то кольцо называется полем.*

Из аксиомы замкнутости для мультипликативной группы (т.е. ненулевых элементов) следует, что поле F не может содержать нулевой делитель, т.е. для любых $a, b \in F$ из условия $ab = \mathbf{0}$ следует, что либо $a = \mathbf{0}$, либо $b = \mathbf{0}$.

Пример 5.12 (Поля).

1. Множества $\mathbb{Q}$, $\mathbb{R}$ и $\mathbb{C}$ являются полями относительно обычных операций сложения и умножения, причем $\mathbf{0} = 0$ и $\mathbf{1} = 1$.

2. Кольцо B, определенное в примере 5.11.3, является полем.
3. Для простого числа p группа $\mathbb{Z}_p$ является полем относительно сложения и умножения по модулю p, причем $\mathbf{0} = 0$ и $\mathbf{1} = 1$. □

В свое время мы рассмотрим и другие примеры полей.

Следует отметить, что группа $\mathbb{Z}$ относительно целочисленного сложения и умножения не является полем, поскольку в группе $\mathbb{Z}$ не для каждого ненулевого элемента существует обратный элемент относительно умножения (нарушается аксиома инверсии). Кроме того, если число n является составным, то группа $\mathbb{Z}_n$ также не является полем, поскольку, как показано выше, группа $\mathbb{Z}_n$ может содержать нулевые делители (нарушается аксиома замкнутости).

Иногда нет необходимости различать группы, кольца и поля. В этих ситуациях они называются **алгебраическими структурами** (algebraic structure).

Понятия конечной группы, подгруппы, фактор-группы и порядка группы легко распространяются на кольца и поля.

Определение 5.14. *Алгебраическая структура называется* ***конечной****, если она состоит из конечного количества элементов. Количество элементов конечной структуры называется ее* ***порядком.***
Подструктурой *алгебраической структуры A называется непустое подмножество S, которое само является алгебраической структурой относительно операций, определенных в структуре A. Если $S \neq A$, то подструктура S называется собственной подструктурой структуры A.*
Пусть A — алгебраическая структура, а $B \subseteq A$ — подструктура структуры A. Фактор-структурой A по модулю B, обозначаемой A/B, называется множество всех смежных классов $a \circ B$, где элемент a пробегает по всей структуре A, относительно операции $\star$, определенной по правилу $(a \circ B) \star (b \circ B) = (a \circ b) \circ B$, с нейтральными элементами $\mathbf{0} \circ B$ и $\mathbf{1} \circ B$.

Из определения 5.14 следует, что кольцо (соответственно поле) может содержать не только подкольцо (соответственно подполе), но и подгруппу (соответственно подкольцо и подгруппу). Соответствующие примеры будут приведены в разделе 5.4.

5.4 Структура конечных полей

Конечные поля широко применяются в криптографии и криптографических протоколах. В частности, основополагающая работа Диффи (Diffie) и Хеллмана (Hellman) по криптографии с открытым ключом, в которой был предложен протокол обмена ключами Диффи–Хеллмана [98] (раздел 8.3), была посвящена конечным полям конкретного вида. С тех пор появилось много криптосистем и прото-

колов, основанных на применении конечных полей: криптосистемы Эль-Гамаля ElGamal) [102], протокол идентификации и схема цифровой подписи Шнорра Schnorr) [257], алгоритм неоспоримых подписей Чаума (Chaum) и протоколы доказательства знания с нулевым разглашением Чаума и Педерсена (Pedersen) 73] и другие. Некоторые современные криптосистемы, такие как современный стандарт шифрования данных (Advanced Encryption Standard) [219] (раздел 7.7) и криптосистема XTR [175], используют конечные поля, имеющие более общий вид. Кроме того, конечные поля лежат в основе эллиптических кривых, которые в свою очередь образуют фундамент целого класса криптосистем [166].

5.4.1 Конечные поля, содержащие простое число элементов

Наиболее простую структуру имеют конечные поля, порядок которых (т.е. количество элементов) является простым числом. Такие поля получили наиболее широкое применение в криптографии.

Определение 5.15 (Простое поле). *Поле, не содержащее собственного подполя, называется простым.*

Например, поле $\mathbb{Q}$ — простое, а множество $\mathbb{R}$ — нет, поскольку множество $\mathbb{Q}$ является собственным подполем поля $\mathbb{R}$. Однако поле $\mathbb{Q}$ является бесконечным. Как мы вскоре убедимся, конечное простое поле содержит простое число элементов, т.е. его порядок является простым числом.

Определение 5.16 (Гомоморфизм и изоморфизм). *Пусть A и B — алгебраические структуры. Отображение $f : A \mapsto B$ называется гомоморфизмом из структуры A в структуру B, если отображение f сохраняет операции, определенные в структуре A. Иначе говоря, если $\circ$ — операция, определенная в структуре A, а $\star$ — операция, определенная в структуре B, то $\forall x, y \in A f(x \circ y) = f(x) \star f(y)$. Взаимно-однозначный гомеоморфизм f из структуры A на структуру B называется изоморфизмом, а структуры A и B — изоморфными.*

Если $f : A \mapsto B$ является гомоморфизмом, а e — нейтральный элемент структуры A (относительно сложения или умножения), то

$$f(e) \star f(e) = f(e \circ e) = f(e).$$

Следовательно, элемент $f(e)$ является нейтральным элементом структуры B. Кроме того, для любого $a \in A$

$$f(a) \star f(a^{-1}) = f(a \circ a^{-1}) = f(e),$$

поэтому $f(a^{-1}) = f(a)^{-1}$ для всех $a \in A$. Более того, если структуры A и B являются изоморфными, они содержат одинаковое количество элементов. Таким образом, две изоморфные структуры устроены одинаково.

Пример 5.13 (Изоморфные алгебраические структуры).

1. Обозначим через F_2 множество $\{0, 1\}$ с операциями $+$ и $\cdot$ целочисленного сложения по модулю 2 и целочисленного умножения соответственно. Структура $\mathbb{F}_2$ является полем, поскольку оно изоморфно полю B, определенному в примере 5.12.2. Легко проверить, что отображение $f(0) = F$, $f(1) = T$ является изоморфизмом.
2. Для любого простого числа p аддитивная группа $\mathbb{Z}_{p-1}$ изоморфна мультипликативной группе $\mathbb{Z}_p^*$. Легко проверить, что функция $f(x) = g^x (\mathrm{mod}\, p)$ является изоморфизмом между этими двумя множествами. □

Очевидно, все поля, состоящие из двух элементов, изоморфны друг другу и полю F_2. Поле, состоящее из двух элементов, является простейшим: в нем есть нуль и единица и больше ничего. Благодаря изоморфизму можно не различать эти поля и рассматривать поле F_2 в качестве единственного поля порядка 2.

Пример 5.14 (Конечное поле простого порядка). Пусть p — произвольное простое число. Тогда множество $\mathbb{Z}_p$, состоящее из целых чисел по модулю p, является конечным полем порядка p (т.е. состоит из p элементов) относительно сложения и умножения по модулю p. В примере 5.11.2 показано, что множество $\mathbb{Z}_p$ является аддитивным кольцом, а в примере 5.1.5 — что множество ненулевых элементов множества $\mathbb{Z}_p$, обозначаемое как $\mathbb{Z}_p^*$, образует мультипликативную группу. □

Определение 5.17 (Поле $\mathbb{F}_p$). *Пусть p — простое число. Символом $\mathbb{F}_p$ обозначается конечное поле $\mathbb{Z}_p$.*

Пусть F — произвольное конечное поле простого порядка p. Поскольку существует изоморфизм из F на $\mathbb{F}_p$, любое конечное поле порядка p изоморфно полю $\mathbb{F}_p$. Так как изоморфные поля отождествляются, любое конечное поле порядка p можно называть полем $\mathbb{F}_p$.

Пусть A — конечная алгебраическая структура с операцией сложения $+$, а символом a обозначен произвольный ненулевой элемент A. Рассмотрим следующую последовательность.

$$a, 2a = a + a, 3a = a + a + a, \ldots \tag{5.4.1}$$

Поскольку A — конечная структура, элемент a имеет конечный порядок, и, следовательно, в последовательности (5.4.1) должна существовать пара чисел (ia, ja) при $i < j$, где i и j — целые числа, такие что $ja - ia = (j - i)a = \mathbf{0}$.

Определение 5.18 (Характеристика алгебраической структуры). *Характеристика алгебраической структуры A, обозначаемая* $\mathrm{char}(A)$, *— это наименьшее положительное целое число n, такое что $na = \mathbf{0}$ для любого элемента $a \in A$. Если такого целого числа не существует, характеристика структуры A равна нулю.*

Теорема 5.4. *Характеристика любого конечного поля является простым числом.*

Доказательство. Пусть F — конечное поле и $a \in A$ — произвольный ненулевой элемент. Для последовательности (5.4.1) выполняются соотношения $(j-i)a = \mathbf{0}$ и $j > i$. Следовательно, поле F имеет положительную характеристику. Обозначим ее через n. Поскольку поле F содержит как минимум два элемента (нуль и единицу), $n \geqslant 2$. Если число $n > 2$ — не простое, его можно представить в виде $n = kl$, где $k, \ell \in \mathbb{Z}, 1 < k, \ell < n$. Следовательно,

$$\mathbf{0} = n\mathbf{1} = (k\ell)\mathbf{1} = (k\ell)\mathbf{1}\mathbf{1} = (k\mathbf{1})\ell\mathbf{1}.$$

Отсюда следует, что либо $k\mathbf{1} = \mathbf{0}$, либо $\ell\mathbf{1} = \mathbf{0}$, поскольку ненулевые элементы поля F образуют мультипликативную группу, не содержащую нулевого элемента $\mathbf{0}$. Значит, либо $ka\mathbf{1} = (k\mathbf{1})a = \mathbf{0}$ для всех $a \in F$, либо $\ell a\mathbf{1} = (\ell\mathbf{1})a = \mathbf{0}$ для всех $a \in F$. Это противоречит определению характеристики n. □

5.4.2 Конечные поля неприводимых полиномов

Порядок простого конечного поля равен характеристике этого поля. Однако простое поле нельзя назвать типичным. Более общий вид конечных полей можно создать с помощью полиномов.

5.4.2.1 Полиномы над алгебраической структурой

В главе 4 мы использовали полиномы над множеством целых чисел. Рассмотрим теперь полиномы над абстрактной алгебраической структурой.

Определение 5.19 (Полиномы над алгебраической структурой). *Пусть A — алгебраическая структура с операциями сложения и умножения. Полиномом над структурой A называется выражение*

$$f(x) = \sum_{i=0}^{n} a_i x^i,$$

где n — неотрицательное целое число, коэффициенты $a_i, 0 \leqslant i \leqslant n$ — элементы структуры A, а x — символ, не принадлежащий структуре A. Коэффициент a_n называется старшим и не является нулевым элементом структуры A при $n \neq 0$. Целое число n называется степенью полинома $f(x)$ и обозначается как

$n = \deg(f(x)) = \deg(f)$. *Если старший коэффициент равен* a_0, *полином* f *называется константой. Если старший элемент равен* $a_0 = 0$, *полином* f *называется нулевым и обозначается как* $f = 0$. *Множество всех полиномов над алгебраической структурой* A *обозначается через* $A[x]$.

Если $f, g \in A[x]$, где

$$f(x) = \sum_{i=0}^{n} a_i x^i \quad \text{и} \quad g(x) = \sum_{i=0}^{m} b_i x^i,$$

то

$$f(x) + g(x) = \sum_{i=0}^{\max(n,m)} c_i x_i, \quad \text{где } c_i = \begin{cases} a_i + b_i, & i = 0, 1, \ldots, \min(n, m), \\ a_i, & i = m+1, \ldots, n, \\ b_i, & i = n+1, \ldots, m, \end{cases} \tag{5.4.2}$$

и

$$f(x)g(x) = \sum_{k=0}^{n+m} c_k x^k, \quad \text{где } c_k = \sum_{\substack{i+j=k \\ 0 \leqslant i \leqslant n \\ 0 \leqslant j \leqslant m}} a_i b_i. \tag{5.4.3}$$

Очевидно, что если A — кольцо, то $A[x]$ — тоже кольцо, причем структура A является подкольцом кольца $A[x]$. Сложение и умножение полиномов над кольцом имеют следующие свойства.

$$\deg(f + g) \leqslant \max(\deg(f), \deg(g)),$$
$$\deg(fg) \leqslant \deg(f) + \deg(g).$$

Если A — поле, то, поскольку оно не содержит нулевых делителей, $c_{n+m} = a_n b_m \neq 0$ при $a_n \neq 0$ и $b_m \neq 0$. Следовательно, в этом случае

$$\deg(f + g) = \deg(f) + \deg(g).$$

Пусть $f, g \in A[x]$, причем $g \neq 0$. Аналогично делению целых чисел (раздел 4.3.2.1) можно записать

$$f = gq + r, \quad \text{где } q, r \in A[x] \text{ и } \deg(r) < \deg(g). \tag{5.4.4}$$

Пример 5.15. Рассмотрим полиномы $f(x) = x^5+x^4+x^3+x^2+x+1 \in \mathbb{F}_2[x]$, $g(x) = x^3+x+1 \in \mathbb{F}_2[x]$. Вычислим полиномы $q, r \in \mathbb{F}_2[x]$ с помощью деления столбиком.

$$
\begin{array}{rl|llllllllll}
 & & x^2 & + & x & & & & & & & \\
\hline
x^3+x+1 \mid & x^5 & + & x^4 & + & x^3 & + & x^2 & + & x & + & 1 \\
 & x^5 & & & + & x^3 & + & x^2 & & & & \\
\hline
 & & & x^4 & & & & & + & x & + & 1 \\
 & & & x^4 & & & + & x^2 & + & x & & \\
\hline
 & & & & & & & x^2 & & & + & 1
\end{array}
$$

Следовательно, $q = x^2 + x$ и $r = x^2 + 1$. □

Определение 5.20 (Неприводимый полином). *Пусть A — алгебраическая структура. Полином $f \in A[x]$ называется неприводимым над структурой A (или неприводимым в множестве $A[x]$, или простым в множестве $A[x]$), если он имеет положительную степень и из равенства $f = gh$, где $g, h \in A[x]$, следует, что либо полином g, либо полином h является константой. Полином называется приводимым над структурой A, если он не является неприводимым над ней.*

Приводимость полинома зависит от алгебраической структуры, на которой он определен. Полином может оказаться приводимым над одной структурой и неприводимым над другой.

Пример 5.16. Рассмотрим квадратичный полином $f(x) = x^2 - 2x + 2$.

1. Является ли он приводимым над обычными бесконечными алгебраическими структурами?
2. Является ли он приводимым над конечными полями $\mathbb{F}_p$, где p — нечетное простое число?
3. Разложите полином $f(x)$ на простые множители над полем $\mathbb{F}_p$ при $p < 10$.

Используя формулы элементарной алгебры, вычислим два корня уравнения $f(x) = 0$:

$$\alpha = 1 + \sqrt{-1}, \beta = 1 - \sqrt{-1}.$$

1. Поскольку $\sqrt{-1}$ не является действительным числом, полином $f(x)$ оказывается неприводимым над полем $\mathbb{R}$ (а значит, и над полями $\mathbb{Z}$ или $\mathbb{Q}$). Однако, так как $\sqrt{-1} = i \in \mathbb{C}$, полином $f(x)$ является приводимым над полем $\mathbb{C}$.
$$f(x) = (x - 1 - i)(x - 1 + i).$$
2. Полином $f(x)$ является приводимым над полем $\mathbb{F}_p$ при любом нечетном простом p тогда и только тогда, когда число $\sqrt{-1}$ принадлежит полю $\mathbb{F}_p$, т.е. число -1 является квадратом по модулю p.

Число x является квадратом по модулю p тогда и только тогда, когда существует число $y(\mathrm{mod}\, p)$, удовлетворяющее условию $y^2 \equiv x(\mathrm{mod}\, p)$. По малой теореме Ферма (теорема 6.10 из раздела 6.4) все числа $x(\mathrm{mod}\, p)$ удовлетворяют условию $x^{p-1} \equiv 1(\mathrm{mod}\, p)$. Для нечетного простого числа p малая теорема Ферма означает, что

$$x^{\frac{p-1}{2}} \equiv \pm 1(\mathrm{mod}\, p) \tag{5.4.5}$$

для всех $0 < x < p$, где число -1 означает число $p-1$. Если число x является квадратом по модулю p, то условие (5.4.5) принимает следующий вид.

$$x^{\frac{p-1}{2}} \equiv \left(y^2\right)^{\frac{p-1}{2}} \equiv y^{p-1} \equiv 1(\mathrm{mod}\, p).$$

Таким образом, условие (5.4.5) является критерием проверки, является ли число x является квадратом по нечетному простому модулю p: число x — квадрат (соответственно не квадрат) по модулю p, если $x^{\frac{p-1}{2}} \equiv 1 \bmod p$ (соответственно $x^{\frac{p-1}{2}} \equiv -1 \bmod p$).

Итак, для любого нечетного простого числа p полином $f(x)$ является приводимым над полем $\mathbb{F}_p$ тогда и только тогда, когда $(-1)^{\frac{p-1}{2}} \equiv 1 \bmod p$, и неприводимым, когда $(-1)^{\frac{p-1}{2}} \equiv -1 \bmod p$. Иначе говоря, полином $f(x)$ является приводимым (или неприводимым) над полем $\mathbb{F}_p$, если $p \equiv 1(\mathrm{mod}\, 4)$ (или $p \equiv 3(\mathrm{mod}\, 4)$).

3. При $p = 2$ полином $f(x) = x^2 - 2x + 2 = x^2 - 0x + 0 = x^2$ является приводимым над полиномом $\mathbb{F}_2$.

Единственное нечетное простое число, которое меньше 10 и сравнимо с 1 по модулю 4, равно 5. Поскольку $-1 \equiv 4 \equiv 2^2(\mathrm{mod}\, 5)$, т.е. $\sqrt{-1} \equiv 2(\mathrm{mod}\, 5)$, полином $f(x)$ можно разложить на множители над полем $\mathbb{F}_5$.

$$f(x) = (x - 1 - \sqrt{-1})(x - 1 + \sqrt{-1}) = (x - 1 - 2)(x - 1 + 2) = (x + 2)(x + 1).$$

Второй квадратный корень числа -1 в поле $\mathbb{F}_5$ равен 3. Читатели могут самостоятельно убедиться, что при этом получается точно такое же разложение полинома $f(x)$ над полем $\mathbb{F}_5$. □

5.4.2.2 Построение поля с помощью неприводимых полиномов

Построим конечное поле с помощью неприводимого полинома.

Определение 5.21 (Множество $A[x]$ по модулю полинома). *Пусть A — алгебраическая структура, а полиномы $f, g, q, r \in A[x]$, где $g \neq 0$, удовлетворяют условию (5.4.4). Тогда полином r называется остатком от деления полинома f на полином g. Этот факт записывается как $r \equiv f \bmod g$.*

Остатки от деления всех полиномов из множества $A[x]$ по модулю g называются полиномами из множества $A[x]$ по модулю g. Множество таких полиномов обозначается как $A[x]_g$.

Множество $A[x]_f$ состоит из всех полиномов, степени которых меньше $\deg(f)$.

Теорема 5.5. *Пусть F — поле, а f — ненулевой полином из множества $F[x]$. Тогда множество $F[x]_f$ — кольцо и является полем тогда и только тогда, когда полином f неприводим над полем F.*

Доказательство. Во-первых, множество $F[x]_f$ очевидно является кольцом относительно сложения и умножения по модулю f, определенному соотношениями (5.4.2), (5.4.3) и (5.4.4). Нулем и единицей в этом кольце являются нуль и единица поля F.

Во-вторых, пусть $F[x]_f$ — поле. Допустим, что $f = gh$, где g и h — полиномы из множества $F[x]$, не являющиеся константами. Тогда, поскольку $0 < \deg(g) < \deg(f)$ и $0 < \deg(h) < \deg(f)$, полиномы g и h не являются константами в множестве $F[x]_f$, несмотря на то, что элемент f является нулевым полиномом в $F[x]_f$. Это противоречит аксиоме замкнутости для мультипликативной группы $F[x]_f$. Следовательно, множество $F[x]_f$ не может быть полем. Полученное противоречие доказывает, что полином f не может быть приводимым над полем F.

В заключение, пусть полином f является неприводимым над полем F. Поскольку множество $F[x]_f$ — кольцо, достаточно показать, что для любого ненулевого элемента множества $F[x]_f$ в нем существует элемент, обратный относительно умножения. Пусть r — ненулевой полином из $F[x]_f$, причем $\gcd(f, r) = c$. Поскольку $\deg(r) < \deg(f)$, а полином f является неприводимым, полином c должен быть константой. Представим полином r в виде $r = cs$. Тогда $c \in F$, $s \in F[x]_f$ и $\gcd(f, s) = 1$. Как и в целочисленной арифметике, к полиномам можно применить обобщенный алгоритм Евклида и вычислить $s^{-1} (\mathrm{mod}\ f) \in F[x]_f$. Кроме того, поскольку $c \in F$, существует элемент $c^{-1} \in F$. Итак, мы получили элемент $r^{-1} = c^{-1}s^{-1} \in F[x]_f$. □

Неприводимый полином f называется **определяющим полиномом** (definition polynomial) конечного поля $F[x]_f$.

Теорема 5.6. *Пусть F — поле, состоящее из p элементов, а f — неприводимый полином степени n над полем F. Тогда количество элементов поля $F[x]_f$ равно p^n.*

Доказательство. Из определения 5.21 следует, что $F[x]_f$ — множество всех полиномов из множества $F[x]$, степень которых меньше $\deg(f) = n$, а коэффициенты принадлежат полю F, состоящему из p элементов. В поле $F[x]_f$ существует ровно p^n таких элементов. □

Следствие 5.4. *Для каждого простого числа p и каждого положительного числа n существует конечное поле, состоящее из p^n элементов.* □

Как указано в следствии 5.4, если в качестве F выбрать простое поле $\mathbb{F}_p$, структура поля $\mathbb{F}_p[x]_f$ становится очень ясной: оно представляет собой множество всех полиномов, степень которых меньше n, а коэффициенты принадлежат полю $\mathbb{F}_p$. Учитывая изоморфизм, можно даже сказать, что поле $\mathbb{F}_p[x]_f$ — это конечное поле порядка p^n.

Пример 5.17 (Целочисленное представление элемента конечного поля). Полином $f(x) = x^8+x^4+x^3+x+1$ является неприводимым над полем $\mathbb{F}_2$. Множество всех полиномов по модулю $f(x)$ над полем $\mathbb{F}_2$ образует поле, состоящее из 2^8 элементов. Их степени меньше 8. Следовательно, произвольный элемент поля $\mathbb{F}_2[x]_f$ имеет вид

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0,$$

где $b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0 \in \mathbb{F}_2$. Следовательно, любой элемент этого поля можно представить в виде целого числа, состоящего из восьми двоичных цифр $b_7b_6b_5b_4b_3b_2b_1b_0$, т.е. в виде байта. Используя шестнадцатеричную систему счисления, можно представить это число с помощью четырех бит.

$$\text{‘0’} = 0000(=0), \ldots, \text{‘9’} = 1001(=9), \text{‘}A\text{’} = 1010(=10), \ldots, \text{‘}F\text{’} = 1111(=15).$$

Поскольку байт состоит из восьми бит, для шестнадцатеричной кодировки байта можно использовать две буквы, заключенные в одинарные кавычки: ‘XY’, где ‘0’ $\leqslant$ ‘X’ $\leqslant$ ‘F’ и ‘0’ $\leqslant$ ‘Y’ $\leqslant$. Иначе говоря, любой элемент поля $\mathbb{F}_2[x]_f$ можно представить в виде байта, лежащего в интервале [‘00’, ‘FF’].

И наоборот, любой байт из интервала [‘00’, ‘FF’] можно считать элементом поля $\mathbb{F}_2[x]_f$. Например, байт 01010111 (или шестнадцатеричное число ‘57’) соответствует полиному

$$x^6 + x^4 + x^2 + x + 1.$$

□

Следствие 5.4 и пример 5.17 означают, что поле $\mathbb{F}_2[x]_f$ можно считать полем всех неотрицательных целых чисел, двоичная запись которых состоит из менее чем $\deg(f)$ бит. Легко видеть, что это поле содержит $2^{\deg(f)}$ элементов. Следовательно, для любого натурального числа $n > 0$ множество $\{0,1\}^n$ образует поле, состоящее из 2^n элементов. Операции в этом поле являются операциями с полиномами над полем $\mathbb{F}_2$, степень которых меньше n.

Пример 5.18. Пусть f — неприводимый полином степени 8 над полем $\mathbb{F}_2$. В поле восьмибитовых двоичных чисел сложение полиномов сводится к сложению полиномов по модулю 2 (т.е. $1 + 1 = 0$). Например, ‘57’ + ‘83’ = ‘$D4$’.

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2.$$

Итак, сложение в этом поле не зависит от определяющего полинома f. □

Умножение в поле $\mathbb{F}_2[x]_f$ зависит от определяющего полинома f: оно представляет собой умножение двух полиномов по модулю f. Эту операцию можно выполнить, применяя к полиномам обобщенный алгоритм Евклида. Позднее (пример 5.19) мы продемонстрируем другой метод умножения, который использует иное представление элементов поля.

Поле n-битовых двоичных чисел оказалось весьма полезным, поскольку допускает естественную интерпретацию целых чисел. Оно получило широкое распространение в теории кодирования и криптографии. Новый стандарт шифрования — Advanced Encryption Standard (AES) — основан на применении именно восьмибитового двоичного поля. Этот стандарт рассматривается в главе 7.

В заключение отметим, что в теореме 5.6 мы нигде не предполагали, что число p — простое. Фактически в теореме 5.5 поле F сможет быть произвольным, а поле $\mathbb{F}_2[x]_f$ называется **расширенным полем** (extended field), которое получено путем **расширения базового подполя** F (underlying subfield). Поскольку поле F может быть любым, оно само может быть получено в результате расширения другого базового подполя. Во многих приложениях конечных полей необходимо знать взаимосвязи между расширенными и базовыми полями (например, это необходимо при изучении стандарта AES). Кроме того, иное представление элементов конечного поля иногда позволяет упростить вычисления (например, если иначе представить элементы поля в примере 5.18, операцию умножения можно упростить, не прибегая в алгоритму Евклида). В следующем разделе мы подробнее рассмотрим структуру конечных полей.

5.4.3 Конечные поля, построенные с помощью полиномиального базиса

Информация, приведенная в этом разделе, позволит лучше понять некоторые криптосистемы, использующие общее представление конечных полей. Предполагается, что читатель владеет основами линейной алгебры и знаком с понятием векторного пространства. Впрочем, этот раздел можно пропустить без ущерба для дальнейшего изложения.

В разделе 5.4.2 показано, что с учетом изоморфизма поле $\mathbb{F}_p[x]_f$ можно считать конечным полем порядка $p^{\deg(f)}$. Однако часто оказывается неудобным работать с полями по модулю неприводимых полиномов. Завершая изложение алгебраических основ криптографии, построим конечные поля, используя корни неприводимых полиномов над конечным полем F. Такие поля чаще применяются в приложениях.

Пусть F — конечное поле, а n — произвольное целое число. Кроме того, пусть $f(x)$ — неприводимый полином степени n над полем F. Полином $f(x)$ имеет в некотором пространстве ровно n корней, поскольку в этом пространстве его

можно разложить на n линейных полиномов. Пока не будем уточнять, в каком именно пространстве лежат его корни.

Обозначим корни уравнения $f(x) = 0$ через

$$\theta_0, \theta_1, \ldots, \theta_{n-1}. \tag{5.4.6}$$

Поскольку полином $f(x)$ является неприводимым над полем F, эти корни не могут принадлежать полю F.

Теорема 5.7. *Пусть F — конечное поле, а $f \in F[x]$ — неприводимый полином степени n над полем F. Если θ — произвольный корень уравнения $f(x) = 0$, то элементы*

$$1, \theta, \theta^2, \ldots, \theta^{n-1}$$

линейно независимы над полем F, т.е. для $r_i \in F$, где $i = 0, 1, 2, \ldots, n-1$,

$$r_0 + r_1\theta + r_2\theta^2 + \cdots + r_{n-1}\theta^{n-1} = 0 \;\Rightarrow\; r_0 = r_1 = \cdots = r_{n-1} = 0. \tag{5.4.7}$$

Доказательство. Пусть θ — произвольный корень уравнения $f(x) = 0$. Известно, что $\theta \neq 1$, поскольку полином $f(x)$ является неприводимым над полем F, которое содержит единицу. Допустим, что элементы $1, \theta, \theta^2, \ldots, \theta^{n-1}$ не являются линейно независимыми над полем F. Иначе говоря, линейная комбинация (5.4.7) может равняться нулю при некоторых $r_i \in F$, где $i = 0, 1, 2, \ldots, n-1$, которые не обращаются в нуль одновременно. Это значит, что элемент θ является корнем уравнения

$$r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-1} x^{n-1} = 0.$$

Если $r_i \in F (i = 0, 1, 2, \ldots, n-1)$, то по определению 5.21 полином $r(x)$ принадлежит полю $F[x]_f$. Следовательно, условие $r(x) = 0$ равносильно условию $r(x) \equiv 0 (\mathrm{mod}\, f(x))$. Пусть a_n — старший коэффициент полинома $f(x)$. Тогда $a_n \in F, a_n \neq 0$ и $a_n^{-1} f(x) \mid r(x)$. Однако это невозможно, поскольку полином $a_n^{-1} f(x)$ имеет степень n, в то время как степень полинома $r(x)$ меньше n. Следовательно, полином $r(x)$ является нулевым. Это противоречит сделанному предположению о том, что не все $r_i \in F (i = 0, 1, 2, \ldots, n-1)$ равны нулю. □

Определение 5.22 (Полиномиальный базис). *Пусть F — конечное поле, а $f(x)$ — неприводимый полином степени n над полем F. Тогда для любого корня θ уравнения $f(x) = 0$ элементы $1, \theta, \theta^2, \ldots, \theta^{n-1}$ называются полиномиальным базисом конечного векторного пространства над полем F.*

Как известно из курса линейной алгебры, базис, состоящий из n элементов, порождает n-мерное векторное пространство. Для этого используются скалярные

величины, принадлежащие полю F, т.е. векторное пространство, порожденное базисом $1, \theta, \theta^2, \ldots, \theta^{n-1}$, имеет следующую структуру

$$\left\{ \sum_{i=0}^{n-1} r_i \theta^i \,\middle|\, r_0, r_1, \ldots, r_{n-1} \in F \right\}. \tag{5.4.8}$$

Теорема 5.8. *Пусть F — конечное поле, а $f(x)$ — неприводимый полином степени n над полем F. Тогда для любого корня θ уравнения $f(x) = 0$ векторное пространство (5.4.8) является конечным полем и состоит из $(\#F)^n$ элементов.*

Доказательство. Во-первых, покажем, что векторное пространство (5.4.8) является кольцом. Единственный нетривиальный момент в этом доказательстве заключается в проверке аксиомы замкнутости. Для этого отметим, что уравнение

$$f(\theta) = a_n \theta^n + a_{n-1} \theta^{n-1} + \cdots + a_0 = 0, \tag{5.4.9}$$

где $a_n \in F$ и $a_n \neq 0$, можно переписать в следующем виде:

$$\theta^n = a_n^{-1} \left(-a_{n-1} \theta^{n-1} - \cdots - a_0 \right).$$

Таким образом, θ^n является линейной комбинацией элементов базиса $1, \theta, \theta^2, \ldots, \theta^{n-1}$. Умножая θ на выражение (5.4.9), легко показать, что для любого положительного целого числа $m \geqslant n$, элемент θ^m можно представить в виде линейной комбинации элементов того же самого базиса. Следовательно, для любых элементов u, v из пространства (5.4.8) произведение uv, будучи линейной комбинацией базисных элементов $1, \theta, \theta^2, \ldots, \theta^m$ при $m \leqslant 2(n-1)$, должно быть линейной комбинацией базисных элементов $1, \theta, \theta^2, \ldots, \theta^{n-1}$, т.е. принадлежит пространству (5.4.8). Итак, аксиома замкнутости выполняется.

Во-вторых, для того, чтобы показать, что пространство (5.4.8) является полем, необходимо показать, что оно не содержит нулевых делителей. Для этого можно применить условие линейной независимости (5.4.7) и убедиться, что из условия $uv = 0$ следует равенство нулю одного из двух скаляров u и v.

В заключение отметим, что в разложении элементов пространства по базису используются $\#F$ скаляров из поля F, а сам базис состоит из n элементов. Следовательно, пространство (5.4.7) состоит из $(\#F)^n$ элементов. □

Определение 5.23 (Конечное поле $\mathbb{F}_{q^n}$). *Пусть q — количество элементов конечного поля F. Символом $\mathbb{F}_{q^n}$ обозначается конечное поле, построенное по базису, состоящему из n элементов над полем F.*

Теорема 5.9. *Пусть F — конечное поле, состоящее из q элементов, а $\mathbb{F}_{q^n}$ — конечное поле, построенное по базису поля F. Тогда*

1) характеристики полей $\mathbb{F}_{q^n}$ и F совпадают,

2) поле F является подполем поля $\mathbb{F}_{q^n}$,

3) любой элемент $a \in \mathbb{F}_{q^n}$ удовлетворяет условию $a^q = a$ тогда и только тогда, когда $a \in F$.

Доказательство. Пусть $1, \theta, \theta^2, \ldots, \theta^{n-1}$ — базис поля $\mathbb{F}_{q^n}$ над полем F.

1. Обозначим характеристику поля F через $\mathrm{char}(F)$. Тогда, складывая любой элемент поля $\mathbb{F}_{q^n}$ с самим собой $\mathrm{char}(F)$ раз, получаем

$$\sum_{i=0}^{n-1} \mathrm{char}(F) r_i \theta^i = \sum_{i=0}^{n-1} 0\theta^i = 0.$$

   Следовательно, $\mathrm{char}(\mathbb{F}_{q^n}) = \mathrm{char}(F)$.

2. Поскольку базис содержит единицу, то, используя скалярные величины из поля F, любой элемент поля F можно представить в виде линейной комбинации единицы, т.е. в виде линейной комбинации элементов базиса.

3. ($\Leftarrow$). Рассмотрим подполе $F = \{0\} \cup F^*$, где F^* — мультипликативная группа ненулевых элементов. Тогда из условия $a \in F$ следует, что либо $a = 0$, либо $a \in F^*$. Для первого варианта условие $a^q = a$ является тривиальным. Для второго варианта по теореме Лагранжа (следствие 5.2) $\mathrm{ord}(a) \mid \#F^* = q-1$, и, следовательно, $a^{q-1} = 1$. Таким образом, $a^q = a$.

   ($\Rightarrow$). Любой элемент $a \in \mathbb{F}_{q^n}$, удовлетворяющий условию $a^q = a$, должен быть корнем уравнения $x^q - x = 0$. Степень этого полинома равна q, и, следовательно, в поле $\mathbb{F}_{q^n}$ у него существует не больше q корней, включая нуль. Из пункта 2 следует, что поле F является подполем поля $\mathbb{F}_{q^n}$, которое уже содержит корни уравнения $x^q - x = 0$. Ни один другой элемент поля $\mathbb{F}_{q^n}$ не может быть корнем полинома $x^q - x$. □

Создавая поле $\mathbb{F}_{q^n}$ над полем F, состоящим из q элементов, мы не требовали, чтобы число q было простым. Иначе говоря, поле F не обязательно является простым. Следующая теорема раскрывает взаимосвязи между полем F и полем $\mathbb{F}_{q^n}$, построенным над полем F, а также описывает природу числа q.

Теорема 5.10 (Критерий подполя). *Пусть p — простое число. Поле F является подполем поля $\mathbb{F}_{p^n}$ тогда и только тогда, когда поле F состоит из p^m элементов, где m — положительный делитель числа n.*

Доказательство. ($\Rightarrow$) Пусть F — подполе поля $\mathbb{F}_{p^n}$. В этом случае варианты $F = \mathbb{F}_p$ или $F = \mathbb{F}_{p^n}$ являются тривиальными. Предположим, что поле F является собственным подполем поля $\mathbb{F}_{p^n}$, отличным от поля $\mathbb{F}_p$. Тогда по теореме 5.9.1

характеристика поля $\mathbb{F}_{p^n}$ равна p. Следовательно, характеристика поля F также равна p. Итак, поле $\mathbb{F}_p$ является подполем поля F, которое построено по базису поля $\mathbb{F}_p$. Этот базис состоит их m элементов, где $1 \leqslant m \leqslant n$. Требуется лишь показать, что $m \mid n$. Две мультипликативные группы $\mathbb{F}^*_{p^n}$ и F^* состоят из $p^n - 1$ и $p^m - 1$ элементов соответственно. Поскольку группа F^* является подгруппой группы $F^*_{p^n}$, из теоремы Лагранжа (теорема 5.1) следует, что $p^m - 1 \mid p^n - 1$. Это условие выполняется, только если $m \mid n$.
($\Leftarrow$) Пусть m — положительный собственный делитель числа n, а поле F состоит из p^m элементов. Поскольку n/m — положительное целое число, полином степени n/m, неприводимый над полем F, порождает поле, состоящее из $(p^m)^{n/m} = p^n$ элементов. Обозначим это поле F_{p^n}. По теореме 5.9.2 поле F является подполем поля F_{p^n}. □

Пусть $f(x)$ — неприводимый полином произвольной степени n над полем $\mathbb{F}_p$. Из теоремы 5.6 следует, что поле F_{p^n} изоморфно полю $\mathbb{F}_p[x]_f$. Несмотря на то что изоморфные поля не различаются, одно из них иногда оказывается намного удобнее другого. Действительно, ярким подтверждением этого факта является критерий подполя для поля $\mathbb{F}_{p^n}$. Рассмотрим еще один пример.

Пример 5.19 (Поле $\mathbb{F}_{2^8}$).
Разбирая пример 5.18, мы убедились, что поле $\mathbb{F}_2[x]_{x^8+x^4+x^3+x+1}$ представляет собой множество всех полиномов по модулю неприводимого полинома $x^8 + x^4 + + x^3 + x + 1$ над полем $\mathbb{F}_2$ и содержит 2^8 элементов. Теперь нам известно, что множество $\mathbb{F}_{2^8}$ также является полем, состоящим из 2^8 элементов, и может быть представлено в виде векторного пространства

$$\{b_7\theta^7 + b_6\theta^6 + b_5\theta^5 + b_4\theta^4 + b_3\theta^3 + b_2\theta^2 + b_1\theta + b_0\},$$

где θ — корень уравнения $x^8 + x^4 + x^3 + x + 1 = 0$, а скаляры $b_7, b_6, b_5, b_4, b_3, b_2, b_1$ и $b_0 \in \mathbb{F}_2$. Очевидно, эти поля изоморфны. В частности, элемент поля $\mathbb{F}_{2^8}$ можно представить в виде байта.

Как указывалось в примере 5.18, умножение элементов в поле $\mathbb{F}_2[x]_{x^8+x^4+x^3+x+1}$ представляет собой довольно сложную операцию, и для ее выполнения приходится прибегать к делению полиномов с помощью алгоритма Евклида. Умножение элементов в поле $\mathbb{F}^*_{2^8}$, построенном с помощью полиномиального базиса, осуществляется намного проще: путем непосредственного умножения двух элементов и представления результатов в виде линейной комбинации элементов $1, \theta, \ldots, \theta^7$.

Для примера вычислим произведение ‘57’ · ‘83’.

$$(\theta^6 + \theta^4 + \theta^2 + \theta + 1) \cdot (\theta^7 + \theta + 1) = \theta^{13} + \theta^{11} + \theta^9 + \theta^8 + \theta^6 + \theta^5 + \theta^4 + \theta^3 + 1.$$

Поскольку

$$\theta^8 + \theta^4 + \theta^3 + \theta + 1 = 0,$$

получаем следующие линейные комбинации:

$$\begin{aligned}
\theta^8 &= \theta^4 + \theta^3 + \theta + 1,\\
\theta^9 &= \theta^5 + \theta^4 + \theta^2 + \theta,\\
\theta^{11} &= \theta^7 + \theta^6 + \theta^4 + \theta^3,\\
\theta^{13} &= \theta^9 + \theta^8 + \theta^6 + \theta^5.
\end{aligned}$$

Следовательно,

$$\theta^{13} + \theta^{11} + \theta^9 + \theta^8 + \theta^6 + \theta^5 + \theta^4 + \theta^3 + 1 = \theta^7 + \theta^6 + 1.$$

Итак, '57' · '83' = '$C1$'. □

Примечание 5.2. *Мы изучили два метода построения конечных полей: поле по неприводимому полиномиальному модулю (5.4.2) и поле, построенное с помощью полиномиального базиса (5.4.3). В ходе изложения поле, построенное с помощью второго метода, обозначалось как $\mathbb{F}_q$. Однако два изоморфных поля, состоящих из одинакового количества элементов, отождествляются. Это позволяет нам в дальнейшем обозначать через $\mathbb{F}_q$ любое конечное поле, состоящее из q элементов, где q — простая степень.* □

5.4.4 Первообразные корни

В разделе 4.5 указывалось, что при проверке простоты числа n с помощью эффективного детерминированного алгоритма полная факторизация числа $n-1$ порождает "вспомогательную информацию" (т.е. дополнительную информацию для верификации задачи из класса $\mathcal{NP}$). Это утверждение легко доказать, используя факты из теории конечных полей.

Теорема 5.11. *Мультипликативная группа $(\mathbb{F}_{p^n})^*$ поля $\mathbb{F}_{p^n}$ является циклической.*

Доказательство. По теореме 5.9.3 множество корней уравнения $x^{p^n} - 1 = 0$ образует группу $(\mathbb{F}_{p^n})^*$. Однако множество корней этого полинома содержит $p^n - 1$ разных (нетривиальных) корней единицы, распределенных на единичной окружности. Итак, существует $(p^n - 1)$-й корень единицы, порождающий группу $(\mathbb{F}_{p^n})^*$. Следовательно, группа $(\mathbb{F}_{p^n})^*$ является циклической. □

Определение 5.24 (Первообразный корень). *Мультипликативный порождающий элемент группы $(\mathbb{F}_{p^n})^*$ называется первообразным корнем поля $\mathbb{F}_{p^n}$.*

Теорема 5.12. *Пусть n — положительное целое число, а $n-1 = r_1 r_2 \dots r_k$ — полное разложение числа $n-1$ на простые множители (некоторые простые множители могут повторяться). Следовательно число n является простым тогда и только тогда, когда существует положительное целое число $a < n$ такое, что $a^{n-1} \equiv 1 (\mathrm{mod}\, n)$ и $a^{(n-1)/r_i} \not\equiv 1 (\mathrm{mod}\, n)$ при $i = 1, 2, \dots, k$.*

Доказательство. ($\Rightarrow$) Если число n — простое, то по теореме 5.11 группа $(\mathbb{F}_n)^*$ — циклическая и содержит порождающий элемент, который является $(n-1)$-м корнем единицы. Обозначим этот корень буквой a. Число a удовлетворяет условиям теоремы.

($\Leftarrow$) Пусть целое число $a < n$ удовлетворяет условиям теоремы. Тогда числа $a, a^2, \dots, a^{n-1}$ являются решениями уравнения $x^{n-1} - 1 \equiv 0 (\mathrm{mod}\, n)$. Для любого числа $1 \leqslant i < j \leqslant n-1$ должно выполняться условие $a^i \not\equiv a^j (\mathrm{mod}\, n)$. Предположим противоположное: $a^{j-i} \equiv 1 (\mathrm{mod}\, n)$ для некоторых чисел i, j, удовлетворяющих условию $0 < j - i < n - 1$. Тогда по определению 5.9 выполняется условие $\mathrm{ord}(a) | i - j | n - 1$, что противоречит условию теоремы. Как известно, $\langle a \rangle$ — мультипликативная группа, состоящая из $n-1$ элементов (относительно умножения по модулю n). Количество элементов этой группы не превышает $\phi(n)$. Следовательно, $\phi(n) = n - 1$. Таким образом, по определению функции Эйлера число n — простое (определение 5.11). □

В теореме 5.12 предполагается, что существует эффективный алгоритм нахождения первообразного корня по модулю простого числа p, т.е. порождающего элемента группы $\mathbb{Z}_p^*$ (алгоритм 5.1).

Алгоритм 5.1. Случайный первообразный корень по модулю простого числа

ВВОД: p: простое число; $q_1, q_2, \dots, q_k$: все простые множители числа $p-1$.

ВЫВОД: g: случайный первообразный корень по модулю простого числа.

PrimitiveRoot$(p, q_1, q_2, \dots, q_k)$.

1. Извлечь $g \in_U [2, p-1]$.
2. for $(i = 1, i++, k)$ do
   if $(g^{(p-1)/q_i} \equiv 1 (\mathrm{mod}\, p))$ return(PrimitiveRoot$(p, q_1, q_2, \dots, q_k)$).
3. return(g).

По теореме 5.2.4 в группе $\mathbb{Z}_p^*$ существует ровно $\phi(p-1)$ элементов порядка $p-1$. Они являются порождающими элементами группы. Следовательно, количество рекурсивных вызовов алгоритма 5.1 удовлетворяет условию [198]

$$\frac{p-1}{\phi(p-1)} < 6 \log \log p - 1.$$

Поскольку количество простых множителей числа $p-1$ не превосходит $\log p$, временна́я сложность алгоритма имеет порядок $O_B((\log p)^4 \log\log p)$.

5.5 Группы, построенные по точкам на эллиптической кривой

В современной криптографии важную роль играют группы, построенные по точкам **эллиптических кривых** (elliptic curves). Применение этих групп в криптографии с открытым ключом впервые было предложено Миллером (Miller) [203] и Коблицем (Koblitz) [166].

В криптографии рассматриваются эллиптические кривые, определенные на конечных алгебраических структурах, например, на конечных полях. Для простоты изложения будем изучать только простые поля $\mathbb{F}_p$, характеристика которых больше трех. В этом случае эллиптическая кривая представляет собой множество точек $P=(x,y)$ следующего уравнения:

$$E: y^2 = x^3 + ax + b (\mathrm{mod}\, p), \tag{5.5.1}$$

где числа a и b являются константами в поле $\mathbb{F}_p (p > 3)$, удовлетворяющими условию $4a^3+27b^2 \neq 0 (\mathrm{mod}\, p)$[1]. Для того чтобы точки множества E образовывали группу, в него включается точка $\mathcal{O}$. Этот дополнительный элемент называется **бесконечно удаленной точкой** (point at infinity) и представляется в виде

$$\mathcal{O} = (x, \infty).$$

Множество E можно представить так.

$$E = \{P=(x,y) \mid x,y \in \mathbb{F}_p - \text{решения уравнения (5.5.1)}\} \cup \{\mathcal{O}\}. \tag{5.5.2}$$

Это множество точек образует группу с операцией, которую удобно обозначить символом "+". Смысл этой операции будет раскрыт позднее.

Обозначим через $f(x)$ кубический полином, стоящий в правой части уравнения (5.5.1). Если полином $f(x)$ является приводимым над полем $\mathbb{F}_p$, то для того, чтобы элемент $\xi \in \mathbb{F}_p$ был нулем полинома $f(x)$ (т.е. $f(\xi) \equiv 0 (\mathrm{mod}\, p)$), точка $(\xi, 0)$ должна принадлежать множеству E. В свое время мы покажем, что относительно групповой операции "+" эти точки имеют порядок 2. Поскольку $f(x)$ — кубический полином, существует по крайней мере три таких точки (одна или три в зависимости от приводимости полинома $f(x)$ над полем $\mathbb{F}_p$ — см. упражнение 5.13).

[1] Подробное объяснение см. в определении 5.25.

Все другие точки, отличающиеся от $\mathcal{O}$, можно получить из точек $\eta \in \mathbb{F}_p$, таких что числа $f(\eta) \neq 0 (\mathrm{mod}\, p)$ являются квадратичными вычетами в поле $\mathbb{F}_p$ (т.е. квадратами по модулю p (раздел 6.5)). Для каждой такой точки η существуют два разных решения y (каждый квадратичный вычет в поле $\mathbb{F}_p$ имеет два квадратных корня по модулю p — см. следствие 6.2). Поскольку величина $f(\eta)$ является константой, эти два корня равны $\sqrt{f(\eta)}$ и $-\sqrt{f(\eta)}$. Итак, две точки решения, соответствующие элементу η, можно обозначить как $(\eta, \sqrt{f(\eta)})$ и $(\eta, -\sqrt{f(\eta)})$.

Итак, на кривой $E(\mathbb{F}_p)$ лежат точки $\mathcal{O}, (\xi, 0), (\eta, \sqrt{f(\eta)})$ и $(\eta, -\sqrt{f(\eta)})$, где элементы ξ и η из поля $\mathbb{F}_p$ удовлетворяют условиям $f(\xi) \equiv 0 (\mathrm{mod}\, p)$ и $f(\eta)$ — квадратичный вычет в поле $\mathbb{F}_p$.

5.5.1 Групповая операция

Множество E, определенное соотношениями (5.5.2), представляет собой абелеву группу с операцией "+", определенной следующим образом.

Определение 5.25 (Операция в группе, порожденной эллиптической кривой ("метод касательных и хорд")). *Пусть $P, Q \in E$, ℓ — прямая, соединяющая точки P и Q (если $P = Q$, эта прямая является касательной к E) и R — точка пересечения прямой ℓ и кривой E. Пусть ℓ' — прямая, соединяющая точки R и $\mathcal{O}$. Тогда точка P "+" Q является такой точкой, что прямая ℓ' пересекает множество E в точках R, $\mathcal{O}$ и P "+" Q.*

Допустим на время, что пара $(E, \text{"+"})$ образует группу. Следует объяснить, почему необходимо, чтобы коэффициенты кубического полинома (5.5.1) удовлетворяли условию $4a^3 + 27b^2 \not\equiv 0 (\mathrm{mod}\, p)$. Заметим, что величина

$$\Delta = \left(\frac{a}{3}\right)^3 + \left(\frac{b}{2}\right)^2 = \frac{4a^3 + 27b^2}{4 \times 27}$$

является дискриминантом кубического полинома $f(x) = x^3 + ax + b$. Если $\Delta = 0$, то уравнение $f(x) = 0$ имеет по крайней мере двойной нуль X (корень, удовлетворяющий условию $f(X) = 0$). Очевидно, что точка $(X, 0)$ лежит на кривой E. Если $F(x, y) = y^2 - x^3 - ax - b = 0$ эта точка удовлетворяет условию

$$\left.\frac{\partial F}{\partial y} = 2y\right|_{y=0} = \left.\frac{\partial F}{\partial x}\right|_{x=X} = 0.$$

Иначе говоря, точка $(X, 0)$ является сингулярной точкой, в которой касательная прямая не определена. Следовательно, поскольку операция "+" в точке $(X, 0)$ не определена, множество E не может быть группой.

Операция "касательная и хорда" продемонстрирована на рис. 5.1. Верхняя половина рисунка относится к ситуации, когда $\Delta < 0$ (кубический полином,

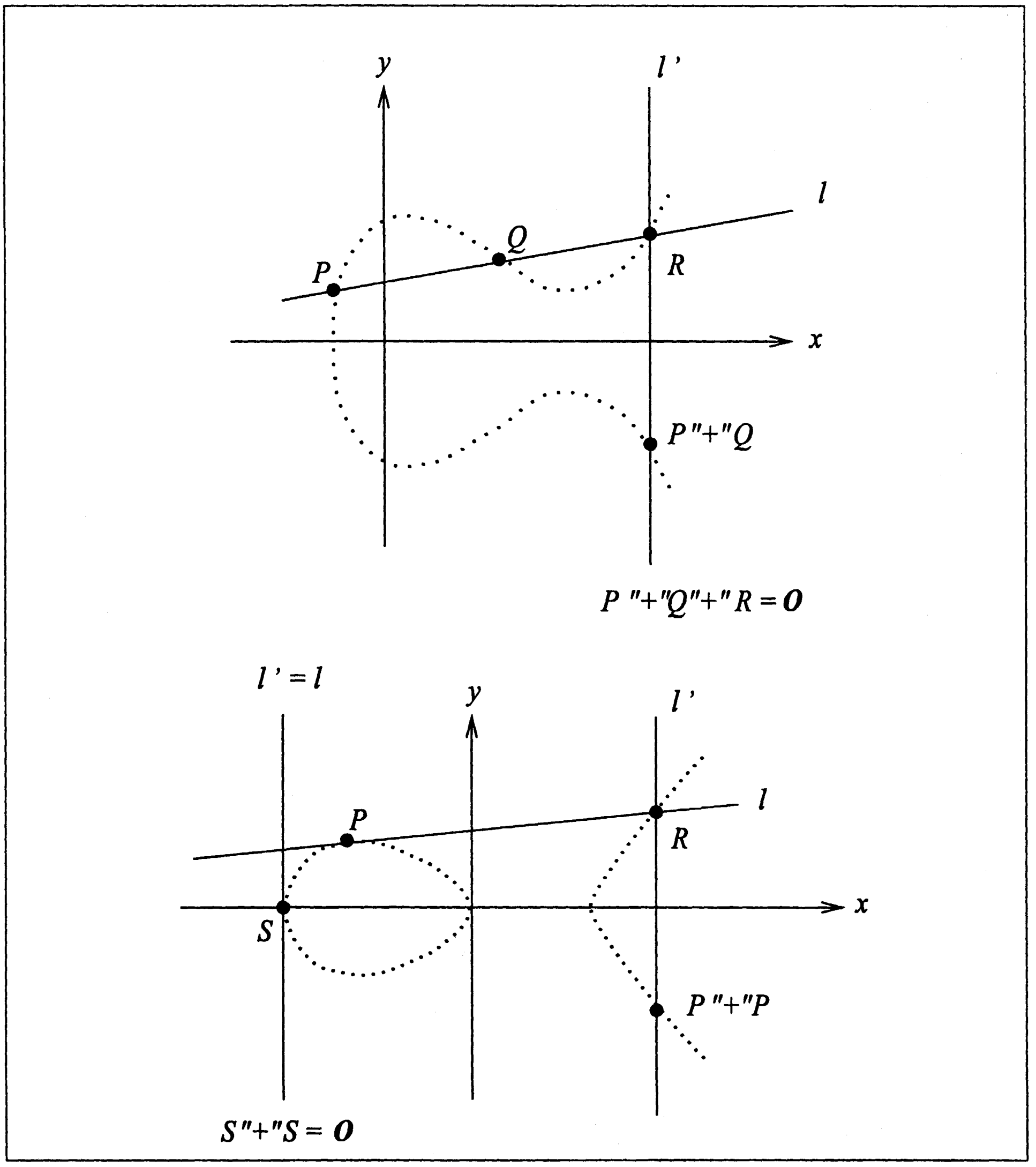


Рис. 5.1. Операция в группе, порожденной эллиптической кривой

имеющий только один действительный корень), а нижняя — когда $\Delta > 0$. Кривые преднамеренно изображены пунктиром, чтобы подчеркнуть, что множество $E(\mathbb{F}_p)$ является дискретным. Элементы этого дискретного множества называются $\mathbb{F}_p$-рациональными точками. Их количество конечно (см. соотношение (5.5.6)).

Покажем, что пара $(E, \text{“+”})$ образует группу относительно операции “касательная и хорда”.

Во-первых, применим к любой точке $P = (X, Y) \in E$ определение 5.25, когда $Q = \mathcal{O}$ (поскольку точка $\mathcal{O}$ включена в множество E). Прямая ℓ, проходящая через точки P и $\mathcal{O}$, имеет вид

$$\ell : x = X.$$

Поскольку из условия $P \in E$ следует, что число $f(X) = X^3 + aX + b (\mathrm{mod}\, p)$ — квадратичный вычет в поле $\mathbb{F}_p$, число $-Y$ является еще одним решением уравнения $y^2 = f(X)$ (т.е. другим квадратным корнем полинома $f(X)$ по модулю p). Иначе говоря, прямая ℓ также проходит через точку $R = (X, -Y) \in E$. Из условия $\ell = \ell'$ следует, что прямая ℓ' проходит через те же три точки, что и прямая ℓ. По определению 5.25 получаем

$$P = P\text{“+”}\mathcal{O}$$

для любой точки $P \in E$. Более того, для всех точек $(x, y) \in E$

$$(x, y)\text{“+”}(x, -y) = \mathcal{O}.$$

Вводя обозначение $(x, -y) = \text{“}-\text{”}(x, y)$, убеждаемся, что точка $\mathcal{O}$ является нулевым элементом относительно операции “+”. Следовательно, в паре $(E, \text{“+”})$ выполняются аксиомы тождества и инверсии.

В частном случае $y_1 = -y_2 = 0$. В этом варианте $P = \text{“}-\text{”}P$ (точка S на нижней кривой рис. 5.1). В этом весьма частном случае $P\text{“+”}P = \mathcal{O}$. Иначе говоря, точка P является элементом порядка 2. Этот вариант уже упоминался ранее: он соответствует решению уравнения $y^2 = f(x) \equiv 0 (\mathrm{mod}\, p)$. Это решение существует, только если полином $f(x)$ имеет нули в поле $\mathbb{F}_p$, т.е. является приводимым над полем $\mathbb{F}_p$.

Рассмотрим теперь общий случай, когда прямая ℓ не является вертикальной. Она задается формулой

$$\ell : y - y_1 = \lambda(x - x_1), \tag{5.5.3}$$

где

$$\lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2}, & \text{если } x_1 \neq x_2, \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{если } x_1 = x_2 \text{ или } y_1 = y_2 \neq 0. \end{cases} \tag{5.5.4}$$

Поскольку прямая ℓ пересекает кривую в точке $R = (x_3, y_3)$, чтобы ее найти, можно применить формулы (5.5.1) и (5.5.3). Координату x точки R можно определить, решив уравнение

$$\ell \cap E : [\lambda(x - x_1) + y_1]^2 - (x^3 + ax + b) = 0.$$

Заметим, что пересечение $\ell \cap E$ является кубическим полиномом, имеющим нули x_1, x_2, x_3. Это позволяет записать его уравнение.

$$\ell \cap E : c(x - x_1)(x - x_2)(x - x_3) = 0,$$

где c — константа. Сравнивая коэффициенты в двух формах записи полинома $\ell \cap E$ (коэффициенты при степенях x^3 и x^2), получаем, что $c = -1$ и

$$x_3 = \lambda^2 - x_1 - x_2.$$

В заключение, из определения 5.25 и того факта, что $P\text{“}+\text{”}Q = \text{“}-\text{”}R$, получаем координаты точки $P\text{“}+\text{”}Q$:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned} \tag{5.5.5}$$

где число λ определяется формулами (5.5.4). Поскольку прямая ℓ проходит через точки P, Q и R, точка R должна лежать на кривой. Следовательно, точка $P\text{“}+\text{”}Q = \text{“}-\text{”}R$ также должна лежать на кривой. Итак, в паре $(E, \text{“}+\text{”})$ выполняется аксиома замыкания.

Аксиому ассоциативности можно проверить, шаг за шагом применяя формулы (5.5.5). Это утомительное упражнение мы предлагаем проделать читателям.

Итак, мы доказали, что пара $(E, \text{“}+\text{”})$ является группой. Более того, она является абелевой группой.

Пример 5.20. Уравнение $E : y^2 = x^3 + 6x + 4$ над полем $\mathbb{F}_7$ определяет группу, порожденную эллиптической кривой, поскольку $4 \times 6^3 + 27 \times 4^2 \equiv 1 \neq 0 (\mathrm{mod}\, 7)$. Группе $E(\mathbb{F}_7)$ принадлежат следующие точки:

$$E(\mathbb{F}_7) = \{\mathcal{O}, (0,2), (0,5), (1,2), (1,5), (3,0), (4,1), (4,6), (6,2), (6,5)\}.$$

Некоторые приложения закона сложения: $(3,0)\text{“}+\text{”}(3,0) = \mathcal{O}$, $(3,0)\text{“}+\text{”}(4,1) = (1,2)$ и $(1,2)\text{“}+\text{”}(1,2) = (0,2)$. Читатели могут сами убедиться, что точка $(1,2)$ является элементом, порождающим группу. Следовательно, группа $E(\mathbb{F}_7)$ является циклической. □

Группы, порожденные эллиптической кривой E, введены в простейшем случае — над простым полем $\mathbb{F}_p$ при $p > 3$. В общем случае группу E можно определить над полем $\mathbb{F}_q$, где q — простая степень. Случаи $p = 2$ и $p = 3$ немного сложнее, но принципы доказательства остаются неизменными. Пытливые читатели могут найти интересующую их информацию в книге [272].

5.5.2 Умножение точек

С этого момента мы больше не будем брать символы $+$ и $-$ в кавычки.

Пусть m — целое число, а $P \in E$. Введем следующее обозначение.

$$[m]P = \begin{cases} \underbrace{P + P + \cdots + P}_{m} & \text{для } m > 0, \\ \mathcal{O}, & \text{для } m = 0, \\ [-m] - P, & \text{для } m < 0. \end{cases}$$

Вычисление величины $[m]P$ (результата m-кратного выполнения групповой операции в любой аддитивной группе) является аналогом возведения в степень в мультипликативной группе. Эта операция описывается следующим алгоритмом.

Алгоритм 5.2. Умножение точек эллиптической кривой

ВВОД: точка $P \in E$; целое число $m > 0$;

ВЫВОД: $[m]P$.

EC_Multiply(P, m)

1. if $m = 0$ return$(\mathcal{O})$.
2. if $m (\mathrm{mod}\, 2) = 0$ return(EC_Multiply$(P + P, m \div 2)$);
   (* $\div$ означает целочисленное деление, т.е. $m \div 2 = [m/2]$ *).
3. return$(P +$ EC_Multiply$(P + P, m \div 2))$.

Например, вычисляя величину EC_Multiply$(P, 14)$, алгоритм 5.2 выполняет четыре рекурсивных вызова.

EC_Multiply$(P, 14) =$

$= \mathrm{EC\_exp}(P + P, 7) =$ (на шаге 2)

$= [2]P + \mathrm{EC\_Multiply}([2]P + [2]P, 3) =$ (на шаге 3)

$= [2]P + [4]P + \mathrm{EC\_Multiply}([4]P + [4]P, 1) =$ (на шаге 3)

$= [2]P + [4]P + [8]P + \mathrm{EC\_Multiply}([8]P + [8]P, 0) =$ (на шаге 3)

$= [2]P + [4]P + [8]P + \mathcal{O}$ (на шаге 1)

Результат равен $[14]P$.

Поскольку $m \approx p$ и вычисления по формулам (5.5.4) и (5.5.5) предусматривают возведение в квадрат величин порядка p, временна́я сложность алгоритма 5.2 имеет порядок $O_B((\log p)^3)$. Следует заметить, что алгоритм 5.2 не использует никаких свойств поля, которому принадлежит точка P. По этой причине его нельзя считать эффективным. Этот алгоритм представляет собой всего лишь краткое описание процесса умножения точки. Более эффективные алгоритмы умножения точки, использующие предварительные вычисления и особые свойства поля, изложены в работе [35].

5.5.3 Дискретное логарифмирование на эллиптической кривой

Операция, обратная к умножению точки, определяет по заданной паре $[P, [m]P]$ целое число m. Она имеет совершенно иную природу. В литературе эта операция называется **дискретным логарифмированием на эллиптической кривой**

(elliptic-curve discrete logarithm problem — ECDLP). Считается, что задача ECDLP является трудноразрешимой (неразрешимой с вычислительной точки зрения), если порядок точки P — большое простое число.

Теорема Хассе (Hasse) утверждает, что

$$\#E(\mathbb{F}_q) = q + 1 - t \quad \text{при} \quad -2\sqrt{q} \leqslant t \leqslant 2\sqrt{q}. \tag{5.5.6}$$

Здесь число t называется "следом Фробениуса" ("trace of Frobenius") числа q. Отсюда следует, что числа $\#E(\mathbb{F}_q)$ и q имеют одинаковый порядок. Для кривой, определенной над полем $\mathbb{F}_q$ (общий случай), легко сконструировать большое простое число p, размер которого немного меньше числа q, такой что группа $E(\mathbb{F}_q)$ содержит подгруппу порядка p. Временна́я сложность наилучшего из всех известных алгоритмов решения задачи ECDLP имеет порядок $O(\sqrt{q})$ (поскольку $p \approx q$). Этот результат достигнут путем применения лобового поиска на основе парадокса дней рождений. Он относится к дискретному логарифмированию в любой абелевой группе, размер которой имеет порядок q. Действительно, λ-метод Полларда (раздел 3.6.1) можно легко модифицировать для решения ECDLP. Следовательно, можно утверждать, что решение задачи ECDLP, порядок сложности которой равен $O(\sqrt{q})$, *вообще не является* решением, поскольку совершенно не учитывает структуру группы, рассматриваемой в задаче.

Существуют арифметические методы дискретного логарифмирования в конечном поле (см. определение 8.2) — методы индексного исчисления. Временна́я сложность таких методов дискретного логарифмирования в конечном поле $\mathbb{F}_q$ имеет субэкспоненциальный порядок sub_exp(q), определенный выражением (8.4.2).

Выражение $O(\sqrt{q})$ экспоненциально зависит от порядка q. При одних и тех же исходных данных выражение $O(\sqrt{q})$ представляет собой функцию, которая возрастает намного быстрее, чем функция sub_exp(q). Это значит, что при решении задачи ECDLP размер исходного поля можно уменьшить, не увеличивая временно́й сложности решения. Как правило, в задаче ECDLP рассматривается число $q \approx 2^{160}$. Это позволяет повысить уровень сложности лобового поиска до 2^{80}. Для того чтобы получить аналогичный порядок сложности при вычислении дискретного логарифма в конечном поле необходимо, чтобы число q в субэкспоненциальном выражении (8.4.2) имело порядок 2^{1000}. Следует, однако, заметить, что прогресс вычислительных технологий вынуждает постоянно повышать порядок числа q. Поскольку асимптотическое поведение величин $\sqrt{q}$ и sub_exp(q) резко отличается друг от друга, в группе точек эллиптической кривой число q может расти намного медленнее, чем в произвольном конечном поле.

Вычислительная неразрешимость задачи ECDLP над относительно малым полем означает, что группы точек эллиптической кривой весьма удобны для реализации более эффективных криптографических систем с открытым ключом. Часто криптография с открытым ключом называется *асимметричной*. Это означает, что

шифрование открытым ключом выполняется легко, а расшифровка без правильного секретного ключа представляет собой трудную задачу. Итак, можно сказать, что криптография с открытым ключом, основанная на группе точек эллиптической кривой, является более асимметричной, чем криптография, основанная на конечных полях.

Однако следует предупредить читателей, что эллиптические кривые имеют слабые места. В этих ситуациях размер поля, равный 2^{160}, слишком мал. Как ни странно, этот факт может иметь положительные последствия (см. главу 13).

5.6 Резюме

Мы рассмотрели абстрактные алгебраические структуры — группу, кольцо и поле, а также конечные варианты арифметических операций. Например, мы показали, что для любого положительного числа n все неотрицательные целые числа, содержащие менее n двоичных цифр, образуют конечное поле, состоящее из 2^n элементов, т.е. структуру, замкнутую относительно сложения и умножения (а значит, и относительно всех других алгебраических операций — возведения в степень, извлечения корня и тому подобного, поскольку все они являются производными от основных операций — сложения и умножения). Алгебраические структуры, обладающие свойством замкнутости в конечных пространствах, являются основными строительными блоками при создании криптографических алгоритмов и протоколов.

Эта глава является не только самодостаточным справочником для большинства читателей, но и содержит большое количество примеров, позволяющих глубже освоить математические основы криптографии. Более подробное изложение тем абстрактной алгебры содержится в работе [177], а эллиптические кривые описаны в книге [272].

Упражнения

5.1. В примере 5.2.5 показано, что группа $\mathrm{Fermat}(n)$ является подгруппой группы $\mathbb{Z}_n^*$. Докажите, что если n — нечетное составное целое число, то $\#\,\mathrm{Fermat}(n) < \#\mathbb{Z}_n^*/2$. Покажите, что это неравенство можно использовать в качестве рабочей основы алгоритма 4.5 для вероятностной проверки простоты числа.

5.2. Докажите, что $\mathrm{DIV3} = \{0\} \cup 3\mathbb{N}$ (множество DIV3 определено в разделе 4.3 (пример 4.1)).

5.3. Выполните следующие задания для группы $\mathbb{Z}_{11}^*$: 1) укажите количество порождающих элементов; 2) перечислите все порождающие элементы; 3) найдите все подгруппы этой группы.

5.4. Пусть n — нечетное составное число, не являющееся степенью простого числа. Существует ли в группе $\mathbb{Z}_n^*$ порождающий элемент?

5.5. Примените "метод шестеренок", описанный в примере 5.10, для того, чтобы доказать, что в группе $\mathbb{Z}_{35}^*$ наибольший порядок равен 12, а порядок любого элемента является делителем числа 12.

5.6. Пусть $n = pq$, где p и q — простые нечетные целые числа, не совпадающие друг с другом. Докажите обобщенный вариант предыдущих утверждений: 1) наибольший порядок элементов в группе $\mathbb{Z}_n^*$ равен $\lambda(n) = \mathrm{lcm}(p-1, q-1)$; 2) порядок любого элемента группы $\mathbb{Z}_n^*$ является делителем числа $\lambda(n)$.

5.7. Почему характеристика конечного кольца или поля должна быть простым числом?

5.8. Выполните обобщенный алгоритм Евклида для деления полиномов, используя процедуру деления столбиком.

5.9. Пусть n — произвольное натуральное число. Постройте конечное поле n-разрядных целых чисел $\{0,1\}^n$. Подсказка: отобразите поле $\mathbb{F}_2[x]_f$ в поле $\{0,1\}^n$, используя отображение, описанное в примере 5.17, где f — полином степени n над полем $\mathbb{F}_2$.

5.10. Сколько изоморфных подполей имеет поле $F_{2^{36}}$? Сколько их у поля $\mathbb{F}_{2^8}$?

5.11. Почему порождающий элемент группы называется первообразным корнем?

5.12. Пусть p — нечетное целое число, причем известно полное разложение числа $p-1$ на простые множители. Постройте эффективный алгоритм проверки простоты числа p, который правильно распознавал бы простые числа с вероятностью 1. (Алгоритм Prime_Test(p) применять нельзя, поскольку у него вероятность правильного ответа не равна единице. Пробное деление также неприемлемо, поскольку оно неэффективно.)

5.13. Докажите, что эллиптическая кривая $E : y^2 = x^3 + ax + b$ над полем $\mathbb{F}_p$, где $p > 3$, не имеет точки порядка 2, если полином $f(x) = x^3 + ax + b$ неприводим над полем $\mathbb{F}_p$. Покажите, что в противном случае кривая E имеет одну или три такие точки.

5.14. Проверьте аксиому ассоциативности для группы $(E, \text{"+"})$, определенной в разделе 5.5.1.

5.15. Докажите, что точка (1, 2) в примере 5.20 является порождающим элементом группы.

Глава 6

Теория чисел

6.1 Введение

Разложение на простые множители и проверка простоты больших целых чисел, извлечение корня, решение систем уравнений в целых числах — вот лишь несколько задач, лежащих в основе современной криптографии. Кроме того, эти задачи представляют большой интерес с точки зрения теории чисел. В главе рассматриваются основные факты и алгоритмы теории чисел, имеющие непосредственное отношение к современной криптографии.

6.1.1 Структурная схема главы

В разделе 6.2 вводятся основные понятия, операции по модулю, а также классы вычетов. В разделе 6.3 описывается функция Эйлера "фи". В разделе 6.4 излагается единообразная точка зрения на теоремы Ферма, Эйлера и Лагранжа. Раздел 6.5 посвящен квадратичным вычетам. В разделе 6.6 рассматриваются алгоритмы извлечения квадратных корней по модулю. Заключительный раздел 6.7 посвящен числам Блюма (Blum).

6.2 Сравнения и классы вычетов

Основы модулярной арифметики изложены в разделе 4.3.2.5. Рассмотрим еще несколько фактов этой теории.

Теорема 6.1. *Для целых чисел $n > 1$ отношение сравнимости по модулю n является рефлексивным, симметричным и транзитивным. Иначе говоря, для любых $a, b, c \in \mathbb{Z}$ выполняются следующие условия.*

1. $a \equiv a (\mathrm{mod}\, n)$.
2. *Если* $a \equiv b (\mathrm{mod}\, n)$, *то* $b \equiv a (\mathrm{mod}\, n)$.
3. *Если* $a \equiv b (\mathrm{mod}\, n)$ *и* $b \equiv c (\mathrm{mod}\, n)$, *то* $a \equiv c (\mathrm{mod}\, n)$. □

Отношение, обладающее тремя указанными свойствами, называется **отношением эквивалентности** (equivalence relation). Как известно, отношение эквивалентности между элементами некоего множества порождает разбиение этого множества на **классы эквивалентности** (equivalence class). Обозначим через "$\equiv_n$" отношение эквивалентности, определяемое сравнением по модулю n. Это отношение определяет во множестве $\mathbb{Z}$ ровно n классов эквивалентности, каждый из которых состоит из целых чисел, сравнимых между собой по модулю n. Обозначим эти n классов следующим образом:

$$\bar{0}, \bar{1}, \ldots, \overline{n-1},$$

где

$$\bar{a} = \{x \in \mathbb{Z} \mid x(\mathrm{mod}\, n) \equiv a\}. \tag{6.2.1}$$

Эти классы называются **классами вычетов по модулю** n (residue class modulo n). Очевидно, что

$$\mathbb{Z}_n = \{\bar{0}, \bar{1}, \ldots, \overline{n-1}\}. \tag{6.2.2}$$

С другой стороны, если рассматривать множество $\mathbb{Z}$ как тривиальное подмножество множества целых чисел $\mathbb{Z}$, то смежный класс $n\mathbb{Z}$ (определение 5.7 в разделе 5.2.1) представляет собой множество всех целых чисел, кратных числу n:

$$n\mathbb{Z} = \{0, \pm n, \pm 2n, \ldots\}. \tag{6.2.3}$$

Рассмотрим фактор-группу (определение 5.8 в разделе 5.2.1) с операцией сложения.

$$\mathbb{Z}/n\mathbb{Z} = \{x + n\mathbb{Z} \mid x \in \mathbb{Z}\}. \tag{6.2.4}$$

Подставляя в выражение (6.2.4) формулу (6.2.3), получаем

$$\begin{aligned}
\mathbb{Z}/n\mathbb{Z} &= \{x + n\mathbb{Z} \mid x \in \mathbb{Z}\} = \\
&= \{0 + \{0, \pm n, \pm 2n, \ldots\}, \\
&\quad 1 + \{0, \pm n, \pm 2n, \ldots\}, \\
&\quad 2 + \{0, \pm n, \pm 2n, \ldots\}, \\
&\quad \ldots \\
&\quad (n-1) + \{0, \pm n, \pm 2n, \ldots\}\} = \\
&= \{\{0, \pm n, \pm 2n, \ldots\}, \\
&\quad \{1, \pm n + 1, \pm 2n + 1, \ldots\}, \\
&\quad \{2, \pm n + 2, \pm 2n + 2, \ldots\}, \\
&\quad \ldots \\
&\quad \{(n-1), \pm n + (n-1), \pm 2n + (n-1), \ldots\}\}.
\end{aligned} \tag{6.2.5}$$

В структуре (6.2.5) существует только n разных элементов. Все другие варианты исключены. Например,

$$n + \{0, \pm n, \pm 2n, \dots\} = \{0, \pm n, \pm 2n, \dots\}$$

и

$$(n+1) + \{0, \pm n, \pm 2n, \dots\} = \{1, \pm n + 1, \pm 2n + 1, \dots\}$$

и т.д. Сравнивая множества (6.2.2) и (6.2.5) и учитывая определение класса $\bar{a}$ в выражении (6.2.1), приходим к выводу, что при $n > 1$

$$\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}.$$

Обозначение $\mathbb{Z}/n\mathbb{Z}$ является стандартным для классов вычетов по модулю n, хотя для удобства в книге используется более краткое обозначение $\mathbb{Z}_n$.

Теорема 6.2. *Для любых чисел $a, b \in \mathbb{Z}$ операции сложения и умножения классов вычетов $\bar{a}$ и $\bar{b}$ определяются следующим образом.*

$$\bar{a} + \bar{b} = \overline{a+b}, \quad \bar{a} \cdot \bar{b} = \overline{ab}.$$

Следовательно, для любого целого числа $n > 1$ отображение $f : \mathbb{Z} \mapsto \mathbb{Z}_n$, определенное отношением сравнимости по модулю n, является гомоморфизмом из поля $\mathbb{Z}$ в фактор-группу $\mathbb{Z}_n$. □

6.2.1 Модулярная арифметика в фактор-группе $\mathbb{Z}_n$

Существование гомоморфизма из поля $\mathbb{Z}$ на фактор-группу $\mathbb{Z}_n$ означает, что арифметические операции в группе $\mathbb{Z}_n$ (по модулю n) обладают свойствами арифметических операций в поле $\mathbb{Z}$.

Теорема 6.3. *Для целых чисел $n > 1$ из того, что $a \equiv b (\mathrm{mod}\, n)$ и $c \equiv d (\mathrm{mod}\, n)$, следует, что $a \pm b \equiv c \pm d (\mathrm{mod}\, n)$ и $ac \equiv bd (\mathrm{mod}\, n)$.*

Несмотря на то что справедливость этой теоремы непосредственно следует из существования гомоморфизма между структурами $\mathbb{Z}$ и $\mathbb{Z}_n$, ее доказательство достойно внимания.

Доказательство. Если $n \mid a - b$ и $n \mid c - d$, то $n \mid (a \pm c) - (b \pm d)$.
Кроме того, $n \mid (a-b)(c-d) = (ac-bd) - b(c-d) - d(a-b)$. Итак, $n \mid (ac-bd)$. □

Свойства арифметических операций в фактор-группе $\mathbb{Z}_n$, описанные в теореме 6.3, называются **свойствами сравнимости** (congruent property). Это значит, что

выполнение одних и тех же вычислений с левой и правой частью равенства сохраняет его справедливость. Однако в теореме 6.3 ничего не говорится о делении. Операция деления в поле $\mathbb{Z}$ обладает следующим свойством сравнимости:

$$\text{если } \forall d \neq 0 : ad = bd, \text{то } a = b. \tag{6.2.6}$$

Операция деления в фактор-группе $\mathbb{Z}_n$ также обладает свойством сравнимости, которое лишь немного отличается от выражения (6.2.6). Прежде чем вывести эту формулу, рассмотрим подробнее свойства деления в поле $\mathbb{Z}$. Представим себе, что поле $\mathbb{Z}$ — это частный случай фактор-группы $\mathbb{Z}_n$, когда $n = \infty$, и что число ∞ делится на любое целое число, причем частное от этого деления снова равно ∞. Следовательно, можно считать, что первое равенство в выражении (6.2.6) выполняется по модулю ∞, а второе — по модулю ∞/d. Поскольку $\infty/d = \infty$, оба равенства в выражении (6.2.6) являются двумя представлениями одной и той же формулы. Это свойство сравнимости для деления в поле $\mathbb{Z}$ наследуется фактор-группой $\mathbb{Z}_n$ и выражается следующей формулой.

Теорема 6.4. *Для целых чисел* $n > 1$ *и* $d \neq 0$ *из тождества* $ad \equiv bd (\mathrm{mod}\, n)$ *следует, что* $a \equiv b \left(\mathrm{mod}\, \frac{n}{\gcd(d,n)}\right)$.

Доказательство. Введем обозначение $k = \gcd(d, n)$. Тогда из $n \mid (ad - bd)$ следует, что $(n/k) \mid (d/k)(a - b)$. Поскольку $\gcd(d/k, n/k) = 1$, из $(n/k) \mid (k/k)(a - b)$ следует, что $(n/k) \mid (a - b)$. □

Итак, арифметические операции в фактор-группе $\mathbb{Z}_n$ полностью сохраняют свойства сравнимости арифметических операций в поле $\mathbb{Z}$.

Следствие 6.1. *Если функция* $f(x)$ *является полиномом над полем* $\mathbb{Z}$ *и* $a \equiv b (\mathrm{mod}\, n)$ *для* $n > 1$, *то* $f(a) \equiv f(b) (\mathrm{mod}\, n)$. □

6.2.2 Решение линейного уравнения в фактор-группе $\mathbb{Z}_n$

В теореме 4.2 (раздел 4.3.2.5) дано определение мультипликативной инверсии по модулю n. Как следует из этой теоремы, для того, чтобы существовало число, обратное к целому числу a по модулю n, т.е. единственное число $x < n$, удовлетворяющее условию $ax \equiv 1 (\mathrm{mod}\, n)$, необходимо и достаточно, чтобы выполнялось условие $\gcd(a, n) = 1$.

Теорема 6.5. *Для того чтобы при* $n > 1$ *уравнение*

$$ax \equiv b (\mathrm{mod}\, n) \tag{6.2.7}$$

имело решение, необходимо и достаточно, чтобы выполнялось условие $\gcd(a, n) \mid b$.

Доказательство. По определению 4.4 (раздел 4.3.2.5) уравнение (6.2.7) можно переписать в виде линейного уравнения

$$ax + kn = b, \tag{6.2.8}$$

где k — некоторое целое число.
($\Rightarrow$) Пусть выполняется условие (6.2.8). Поскольку число $\gcd(a, n)$ является делителем левой части уравнения, оно должно быть также делителем его правой части.
($\Leftarrow$) Используя обобщенный алгоритм Евклида (алгоритм 4.2) для чисел a и n, можно вычислить

$$a\lambda + \mu n = \gcd(a, n).$$

Поскольку число $b/\gcd(a, n)$ является целым, умножая на него обе части уравнения, получаем формулы (6.2.8) или (6.2.7), где $x = \frac{\lambda b}{\gcd(a,n)} (\mathrm{mod}\, n)$ — одно из решений. □

Легко проверить, что при заданном решении x уравнения (6.2.7) уравнение

$$x + \frac{ni}{\gcd(a, n)} (\mathrm{mod}\, n) \quad \text{для } i = 0, 1, 2, \ldots, \gcd(a, n) - 1$$

имеет $\gcd(a, n)$ разных решений, которые меньше числа n. Очевидно, что условие $\gcd(a, n) = 1$ является достаточным для того, чтобы уравнение (6.2.8) имело единственное решение, которое меньше числа n.

Пример 6.1. Уравнение

$$2x \equiv 5 (\mathrm{mod}\, 10)$$

не имеет решения, поскольку $\gcd(2, 10) = 2 \nmid 5$. Фактически левая часть уравнения, $2x$, должна быть четным числом, поскольку правая часть, $10k + 5$, может быть только нечетным числом. Следовательно, пытаясь решить это уравнение, мы ищем число, которое одновременно было бы четным и нечетным, что, разумеется, невозможно.

С другой стороны, уравнение

$$6x \equiv 18 (\mathrm{mod}\, 36)$$

является разрешимым, поскольку $\gcd(6, 36) \mid 18$. Это уравнение имеет шесть решений: 3, 9, 15, 21, 27 и 33. □

Теорема 6.6. *Если для всех чисел* $n > 1$ *выполняется условие* $\gcd(a, n) = 1$, *то* $ai + b \not\equiv aj + b (\mathrm{mod}\, n)$ *для всех чисел* b, i, j, *удовлетворяющих условию* $0 \leqslant i < j < n$.

Доказательство. Допустим противоположное: $ai + b \equiv aj + b (\mathrm{mod}\, n)$. Тогда по теореме 6.4 $i \equiv j (\mathrm{mod}\, n)$, что противоречит условию $0 \leqslant i < j < n$. □

Из этого свойства следует, что для целых чисел a и n, удовлетворяющих условию $\gcd(a, n) = 1$, числа $ai + b (\mathrm{mod}\, n)$, где $i = 0, 1, \ldots, n - 1$, образуют **полную систему вычетов** (complete residue system) по модулю n, т.е. значение выражения $ai + b (\mathrm{mod}\, n)$ пробегает всю фактор-группу $\mathbb{Z}_n$, когда число i принимает значения из этой же фактор-группы.

6.2.3 Китайская теорема об остатках

Итак, нам известно условие разрешимости линейного уравнения (6.2.7). Однако довольно часто возникает необходимость решать систему таких уравнений по разным модулям.

$$\begin{aligned} a_1 x &\equiv b_1 (\mathrm{mod}\, n_1), \\ a_2 x &\equiv b_2 (\mathrm{mod}\, n_2), \\ &\vdots \\ a_r x &\equiv b_r (\mathrm{mod}\, n_r), \end{aligned} \tag{6.2.9}$$

где $a_i, b_i \in \mathbb{Z}, a_i \neq 0$ при $i = 1, 2, \ldots, r$.

Очевидно, система уравнений имеет решение, только если каждое из уравнений разрешимо. Введем обозначение

$$d_i = \gcd(a_i, n_i),$$

где $i = 1, 2, \ldots, r$. Тогда по теореме 6.5 для разрешимости уравнений необходимо, чтобы выполнялось условие $d_i \mid b_i$. В этом случае свойства умножения (теорема 6.3) и деления (теорема 6.4) по модулю n позволяют переписать систему (6.2.9) в эквивалентном, но более простом виде.

$$\begin{aligned} x &\equiv c_1 (\mathrm{mod}\, m_1), \\ x &\equiv c_2 (\mathrm{mod}\, m_2), \\ &\vdots \\ x &\equiv c_r (\mathrm{mod}\, m_r), \end{aligned} \tag{6.2.10}$$

где $i = 1, 2, \ldots, r$,

$$m_i = n_i / d_i$$

и

$$c_i = (b_i / d_i)(a_i / d_i)^{-1} (\mathrm{mod}\, m_i).$$

Заметим, что число $(a_i / d_i)^{-1} (\mathrm{mod}\, m_i)$ существует, поскольку $\gcd(a_i / d_i, m_i) = 1$ (теорема 4.2 из раздела 4.3.2.5).

Из курса линейной алгебры известно, что систему (6.2.10) можно представить в матрично-векторном виде

$$A\vec{X} = \vec{C}, \tag{6.2.11}$$

где

$$A = \begin{pmatrix} \bar{1}_{m_1} & & & & \\ & \bar{1}_{m_2} & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \cdot \\ & & & & \bar{1}_{m_r} \end{pmatrix}, \tag{6.2.12}$$

$$\vec{X} = \begin{pmatrix} x \\ x \\ \cdot \\ \cdot \\ \cdot \\ x \end{pmatrix}, \tag{6.2.13}$$

$$\vec{C} = \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_r \end{pmatrix}. \tag{6.2.14}$$

Заметим, что, поскольку i-е уравнение $(i = 1, 2, \ldots, r)$ в системе (6.2.10) представляет собой сравнение по модулю m_i, диагональные элементы матрицы A обозначают классы вычетов единицы по модулю m_i, т.е.

$$\bar{1}_{m_i} = k_i m_i + 1 \tag{6.2.15}$$

для некоторого целого числа $k_i (i = 1, 2, \ldots, r)$. Остальные элементы матрицы A равны нулю по соответствующему модулю (т.е. нули в i-й строке являются нулями по модулю m_i).

Итак, при заданном r-мерном векторе $\vec{C}$ решение системы уравнений (6.2.10) сводится к идентификации диагональной матрицы A. Иначе говоря, решить систему (6.2.10) значит найти класс вычетов единицы по модулям m_i, где $i = 1, 2, \ldots, r$, как показано в выражении (6.2.15). Из курса линейной алгебры известно, что если такая матрица A существует, то, поскольку все ее диагональные элементы не

равны нулю, ее полный ранг равен r, и, следовательно, система (6.2.11) *имеет единственное решение.*

Если модули в системе (6.2.10) являются попарно взаимно простыми, нетрудно найти систему классов, состоящих из вычетов единицы.

Теорема 6.7 (Китайская теорема об остатках). *Если модули системы линейных уравнений (6.2.10) являются взаимно простыми, т.е.* $\gcd(m_i, m_j) = 1$ *при* $1 \leqslant i < j \leqslant r$, *то существует класс вычетов* $\bar{1}_{m_i}$, *удовлетворяющий условию*

$$\bar{1}_{m_i} \equiv 0 (\mathrm{mod}\, m_j). \tag{6.2.16}$$

Следовательно, существует число $x \in \mathbb{Z}_M$, *являющееся единственным решением системы (6.2.10), где* $M = m_1 m_2 \ldots m_r$.

Доказательство. Сначала докажем существование, а затем — единственность решения.

Существование. Для каждого числа $i = 1, 2, \ldots, r$ выполняется условие $\gcd(m_i, M/m_i) = 1$. По теореме 4.2 (раздел 4.3.2.5) существует число $y_i \in \mathbb{Z}_{m_i}$, удовлетворяющее условию

$$(M/m_i) y_i \equiv 1 (\mathrm{mod}\, m_i). \tag{6.2.17}$$

Более того, поскольку $m_j \mid (M/m_i)$ для всех $j \neq i$, имеем

$$(M/m_i) y_i \equiv 0 (\mathrm{mod}\, m_j). \tag{6.2.18}$$

Итак, число $(M/m_i) y_i$ является искомым. Пусть

$$x \leftarrow \sum_{i=1}^{r} \bar{1}_{m_i} c_i (\mathrm{mod}\, M). \tag{6.2.19}$$

Следовательно, вектор x является решением системы (6.2.10) и образует класс вычетов единицы по модулю M.

Единственность. Рассмотрим такую линейную систему (6.2.11), (6.2.12), (6.2.13) и (6.2.14), в которой все элементы матрицы A и правой части $\vec{C}$ являются целыми числами. Заметим, что в поле $\mathbb{Z}$

$$\det(A) = \bar{1}_{m_1} \bar{1}_{m_2} \ldots \bar{1}_{m_r} \neq 0. \tag{6.2.20}$$

Отсюда следует, что r столбцов матрицы A образуют базис r-мерного векторного пространства $\underbrace{\mathbb{Z} \times \mathbb{Z} \times \ldots \times \mathbb{Z}}_{r}$ (этот базис напоминает так называемый "естественный базис" в линейной алгебре, в котором каждый базисный вектор содержит только одну единицу, а остальные элементы равны нулю). Таким образом, при

любом векторе $\vec{C} \in \underbrace{\mathbb{Z} \times \mathbb{Z} \times \ldots \times \mathbb{Z}}_{r}$ система (6.2.11) имеет единственное решение $\vec{X} \in \underbrace{\mathbb{Z} \times \mathbb{Z} \times \ldots \times \mathbb{Z}}_{r}$. Как показано выше, элементы вектора $\vec{X}$ вычисляются по формуле (6.2.19). □

Доказательство теоремы 6.7 является конструктивным, поскольку мы построили алгоритм, позволяющий находить решение системы (6.2.10).

В алгоритме 6.1 затраты времени происходят только на шаге 2.1, на котором с помощью обобщенного алгоритма Евклида вычисляется мультипликативная инверсия большого целого числа. Если $m_i < M$ при всех $i = 1, 2, \ldots, r$, временна́я сложность алгоритма 6.1 имеет порядок $O_B(r(\log M)^2)$.

Перечислим некоторые выводы из теоремы 6.7.

1. Каждое число $x \in \mathbb{Z}_M$ порождает вектор $\vec{C} \in \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_r}$. Из выражения (6.2.19) следует, что все элементы вектора $\vec{C}$ вычисляются по формуле
$$c_i \leftarrow x (\mathrm{mod}\, m_i) \text{ при } i = 1, 2, \ldots, r.$$

2. В частности, числа 0 и 1 в фактор-группе $\mathbb{Z}_M$ порождают векторы $\vec{0}$ и $\vec{1}$ соответственно из векторного пространства $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_r}$.

3. Пусть числа x и x' порождают векторы $\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_r \end{pmatrix}$ и $\begin{pmatrix} c'_1 \\ c'_2 \\ \vdots \\ c'_r \end{pmatrix}$ соответственно. Тогда число $x \cdot x'$ порождает вектор $\begin{pmatrix} c_1 \cdot c'_1 (\mathrm{mod}\, m_1) \\ c_2 \cdot c'_2 (\mathrm{mod}\, m_2) \\ \vdots \\ c_r \cdot c'_r (\mathrm{mod}\, m_r) \end{pmatrix}$.

Итак, мы доказали следующую теорему.

Теорема 6.8. *Если* $\gcd(m_i, m_j) = 1$ *для* $1 \leqslant i < j \leqslant r$, *то для числа* $M = m_1 m_2 \ldots m_r$ *фактор-группа* $\mathbb{Z}_M$ *является изоморфной векторному пространству* $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_r}$, *причем изоморфизм*

$$f : \mathbb{Z}_M \mapsto \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_r}$$

задается формулой

$$f(x) = (x (\mathrm{mod}\, m_1), x (\mathrm{mod}\, m_2), \ldots, x (\mathrm{mod}\, m_r)).$$ □

Алгоритм 6.1. Китайская теорема об остатках

ВВОД: кортеж целых попарно взаимно простых чисел $(m_1, m_2, \ldots, m_r)$;
кортеж целых чисел $(c_1 \bmod(m_1), c_2 \bmod(m_2), \ldots, c_r \bmod(m_r))$.

ВЫВОД: целое число $x < M = m_1 m_2 \ldots m_r$ — решение системы (6.2.10).

1. $M \leftarrow m_1 m_2 \ldots m_r$;
2. for $(i = 1, i++, r)$ do

   а) $y_i \leftarrow (M/m_i)^{-1} (\bmod m_i)$; (* Обобщенный алгоритм Евклида *)

   б) $\bar{1}_{m_i} \leftarrow y_i M/m_i$;
3. return $\left(\sum_{i=1}^{r} \bar{1}_{m_i} c_i (\bmod M) \right)$.

Теорема 6.8 весьма полезна при изучении криптографических систем и протоколов, в основе которых лежат группы по составным целым модулям. Во многих местах книги мы будем применять изоморфизм между группой $\mathbb{Z}_n^*$ и декартовым произведением групп $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$, где $n = pq$, а p и q — простые числа. Например, мы будем использовать тот факт, что нециклическая группа $\mathbb{Z}_n^*$ порождается двумя элементами циклических групп $\mathbb{Z}_p^*$ и $\mathbb{Z}_q^*$ соответственно.

Рассмотрим одно из приложений китайской теоремы об остатках: упрощение вычислений с помощью изоморфизма.

Пример 6.2. Пока мы не умеем вычислять квадратный корень по целочисленному модулю (этот метод будет рассмотрен в разделе 6.6). Однако иногда квадратные корни в некотором пространстве (например, в поле $\mathbb{Z}$) легко угадать, не применяя операций модулярной арифметики. Применим теорему 6.8 для вычисления одного из квадратных корней числа 29 в группе $\mathbb{Z}_{35}$.

Сведения, которыми мы владеем на данный момент, не позволяют нам утверждать, что число 29 является квадратом в группе $\mathbb{Z}_{35}$, поэтому его квадратный корень нам неизвестен. Однако, применяя теорему 6.8 и отображая число 29 в изоморфное пространство $\mathbb{Z}_5 \times \mathbb{Z}_7$, получаем

$$29(\bmod 5) \mapsto 4, \quad 29(\bmod 7) \mapsto 1,$$

т.е. его образом выступает пара $(4, 1)$. Очевидно, что оба числа 4 и 1 являются квадратами: число 4 — квадрат числа 2, а число 1 — единицы. Благодаря изоморфизму становится очевидным, что один из квадратных корней числа 29 в группе $\mathbb{Z}_{35}$ соответствует паре $(2, 1)$ в пространстве $\mathbb{Z}_5 \times \mathbb{Z}_7$. Применяя китайскую теорему об остатках (алгоритм 6.1), получаем

$$\bar{1}_5 = 21, \quad \bar{1}_7 = 15$$

и

$$\sqrt{29} \equiv 21 \cdot 2 + 15 \cdot 1 \equiv 22 (\mathrm{mod}\, 35).$$

Действительно, $22^2 = 484 \equiv 29 (\mathrm{mod}\, 35)$. □

6.3 Функция Эйлера “фи”

В разделе 5.2.3 определена функция Эйлера “фи” (определение 5.11). Перейдем к изучению ее свойств.

Лемма 6.1. *Пусть $\phi(n)$ — функция Эйлера “фи”. Тогда*

1. $\phi(1) = 1$;
2. *если число p — простое, то $\phi(p) = p - 1$;*
3. *функция Эйлера “фи” является мультипликативной, т.е. если $\gcd(m, n) = 1$, то $\phi(mn) = \phi(m)\phi(n)$;*
4. *если $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ — разложение числа n на простые множители, то*

$$\phi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \ldots \left(1 - \frac{1}{p_k}\right).$$

Доказательство. Утверждения 1 и 2 являются тривиальными и следуют непосредственно из определения 5.11.

Докажем утверждение 3. Поскольку $\phi(1) = 1$, уравнение $\phi(mn) = \phi(m)\phi(n)$ выполняется, когда либо число n, либо число m равно единице. Допустим, что $m > 1$ и $n > 1$, причем $\gcd(n, m) = 1$. Рассмотрим следующий массив.

$$\begin{array}{ccccc} 0 & 1 & 2 & \ldots & m-1 \\ m & m-1 & m+2 & \ldots & m+(m-1) \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ (n-1)m & (n-1)m+1 & (n-1)m+2 & \ldots & (n-1)m+(m-1) \end{array} \tag{6.3.1}$$

С одной стороны, массив (6.3.1) состоит из mn последовательных целых чисел, поэтому все они являются целыми по модулю mn, и, следовательно, среди них есть $\phi(mn)$ чисел, взаимно простых по отношению к числу mn.

С другой стороны, первая строка массива (6.3.1) состоит из чисел, целых по модулю m, а все элементы любого столбца сравнимы по модулю m. Следовательно, в массиве существует $\phi(m)$ столбцов, целиком состоящих из целых чисел, взаимно простых с числом m. Обозначим эти столбцы следующим образом.

$$b, m + b, 2m + b, \ldots, (n-1)m + b$$

Они состоят из n элементов. Поскольку $\gcd(m, n) = 1$, по теореме 6.6 каждый такой столбец образует полную систему вычетов по модулю n. Следовательно, в каждом таком столбце существует $\phi(n)$ элементов, взаимно простых по отношению к числу n. Итак, массив (6.3.1) содержит $\phi(m)\phi(n)$ элементов, взаимно простых по отношению к числу n. Кроме того, заметим, что любое число является взаимно простым по отношению к числам m и n тогда и только тогда, когда оно является взаимно простым по отношению к числу mn.

Объединяя полученные результаты, приходим к выводу, что $\phi(mn) = \phi(m)\phi(n)$.

Докажем утверждение 4. Для любого простого числа p элементы кортежа $1, 2, \ldots, p^e$, не являющиеся взаимно простыми по отношению к числу p^e, являются кратными числу p, т.е. равны $p, 2p, \ldots, p^{e-1}p$. Очевидно, что количество таких чисел равно p^{e-1}. Итак,

$$\phi(p^e) = p^e - p^{e-1} = p^e\left(1 - \frac{1}{p}\right).$$

Эта формула верна для любой простой степени $p^e \mid n$, где $p^{e+1} \nmid n$. Заметим, что две разные простые степени числа n являются взаимно простыми числами. Учитывая утверждение 3, получаем требуемый результат. □

В разделе 4.5 мы рассмотрели задачу SQUARE-FREENESS, в которой требовалось определить, делится ли нечетное составное целое число на квадрат какого-либо простого числа? Для того чтобы доказать, что задача SQUARE-FREENESS принадлежит классу $\mathcal{NP}$, мы применили функцию $\phi(n)$. В соответствии с утверждением 4 леммы 6.1 для любого простого числа $p > 1$ из условия $p^2 \mid n$ следует, что $p \mid \phi(n)$. Вот почему в качестве свидетельства того, что число n не делится на квадрат простого числа, мы использовали условие $\gcd(n, \phi(n)) = 1$. Читатели могут сами рассмотреть вариант, когда $\gcd(n, \phi(n)) > 1$ (особенно интересным является вариант, когда $n = pq$, где $p \mid \phi(q)$, описанный в упражнении 6.5).

Функция Эйлера "фи" имеет одно элегантное свойство.

Теорема 6.9. *Для любого целого числа $n > 0$ выполняется условие* $\sum_{d|n} \phi(d) = n$.

Доказательство. Пусть $S_d = \{x \mid 1 \leqslant x \leqslant n, \gcd(x, n) = d\}$. Очевидно, что при каждом числе $d \mid n$ множество $S = \{1, 2, \ldots, n\}$ разбивается на непересекающиеся подмножества S_d. Следовательно,

$$\bigcup_{d|n} S_d = S.$$

Заметим, что для каждого числа $d \mid n$ выполняется условие $\#S_d = \phi(n/d)$. Следовательно,

$$\sum_{d|n} \phi(n/d) = n.$$

Однако для любого числа $d \mid n$ выполняется условие $(n/d) \mid n$. Следовательно,

$$\sum_{d|n} \phi(n/d) = \sum_{(n/d)|n} \phi(n/d) = \sum_{d|n} \phi(d).$$

□

Пример 6.3. При $n = 12$ условию $d \mid n$ удовлетворяют числа $1, 2, 3, 4, 6$ и 12. Следовательно, $\phi(1) + \phi(2) + \phi(3) + \phi(4) + \phi(6) + \phi(12) = 1 + 1 + 2 + 2 + 2 + + 4 = 12$. □

6.4 Теоремы Ферма, Эйлера и Лагранжа

В главе 4 мы рассмотрели малую теорему Ферма (сравнение (4.4.8)) и уже несколько раз применили ее, не приводя доказательства. Для того чтобы доказать ее, покажем, что она является частным случаем другой знаменитой теоремы теории чисел — теоремы Эйлера.

Теорема 6.10 (Малая теорема Ферма). *Если число p — простое и $p \nmid a$, то* $a^{p-1} \equiv 1 (\mathrm{mod}\, p)$.

Поскольку число p — простое, выполняется условие $\phi(p) = p - 1$. По этой причине малая теорема Ферма является частным случаем следующей теоремы.

Теорема 6.11 (Теорема Эйлера). *Если* $\gcd(a, n) = 1$, *то* $a^{\phi(n)} \equiv 1 (\mathrm{mod}\, n)$.

Доказательство. Из условия $\gcd(a, n) = 1$ следует, что $a (\mathrm{mod}\, n) \in \mathbb{Z}_n^*$. Кроме того, $\#\mathbb{Z}_n^* = \phi(n)$. Учитывая следствие 5.2, получаем, что $\mathrm{ord}_n(a) \mid \#\mathbb{Z}_n^*$. Отсюда следует, что $a^{\phi(n)} \equiv 1 (\mathrm{mod}\, n)$. □

Поскольку следствие 5.2, упомянутое в доказательстве теоремы 6.11, является непосредственным применением теоремы Лагранжа (теорема 5.1), можно утверждать, что малая теорема Ферма и теорема Эйлера являются частными случаями замечательной теоремы Лагранжа.

В главе 4 показана важная роль, которую малая теорема Ферма играет при вероятностной проверке простоты числа, часто применяемой при генерации ключей во многих криптографических системах и протоколах с открытым ключом. В свою очередь, теорема Эйлера нашла применение в криптосистеме RSA, описанной в разделе 8.5.

6.5 Квадратичные вычеты

Квадратичные вычеты играют важную роль в теории чисел. Например, алгоритмы разложения целого числа на множители непременно используют квадратичные вычеты. Кроме того, квадратичные вычеты широко применяются при шифровании и в криптографических протоколах.

Определение 6.1 (Квадратичный вычет). *Пусть n — целое число и $n > 1$. Число a из группы $\mathbb{Z}_n^*$ называется квадратичным вычетом по модулю n (quadratic residue modulo n), если в группе $\mathbb{Z}_n$ существует число x, удовлетворяющее условию $x^2 \equiv a (\mathrm{mod}\, n)$; в противном случае число a называется квадратичным невычетом по модулю n (quadratic non-residue modulo n). Множество квадратичных вычетов по модулю n обозначается как QR_n, а множество квадратичных невычетов по модулю n — как QNR_n.*

Пример 6.4. Вычислим QR_{11} — множество всех квадратичных вычетов по модулю 11. $\mathrm{QR}_{11} = \{1^2, 2^2, 3^2, 4^2, 5^2, 6^2, 7^2, 8^2, 9^2, 10^2\} (\mathrm{mod}\, 11) = \{1, 3, 4, 5, 9\}$. ⊏

В нашем примере мы вычислили множество QR_{11} путем полного перебора элементов группы $\mathbb{Z}_{11}^*$. Читатели могут самостоятельно убедиться в том, что

$$\mathrm{QR}_{11} = \{1^2, 2^2, 3^2, 4^2, 5^2\} (\mathrm{mod}\, 11),$$

т.е. достаточно было возвести в квадрат по модулю 11 лишь половину элементов группы $\mathbb{Z}_{11}^*$. Этот факт обобщается в следующей теореме.

Теорема 6.12. *Пусть p — простое число. Тогда справедливы следующие утверждения.*

1. $\mathrm{QR}_p = \{x^2 (\mathrm{mod}\, p) \mid 0 < x \leqslant (p-1)/2\}$.
2. *Существует ровно $(p-1)/2$ квадратичных вычетов и $(p-1)/2$ квадратичных невычетов по модулю p, т.е. группа $\mathbb{Z}_p^*$ разбивается на два подмножества, QR_p и QNR_p, состоящие из одинакового количества элементов.*

Доказательство. Докажем первое утверждение. Очевидно, что $S = \{x^2 (\mathrm{mod}\, p) \mid 0 < x \leqslant (p-1)/2\} \subseteq \mathrm{QR}_p$. Для того чтобы показать, что $\mathrm{QR}_p = S$, достаточно доказать, что $\mathrm{QR}_p \subseteq S$.

Возьмем произвольное число $a \in \mathrm{QR}_p$. Тогда существует число $x < p$, удовлетворяющее условию $x^2 \equiv a (\mathrm{mod}\, p)$. Если $x \leqslant (p-1)/2$, то $a \in S$. Допустим, что $x > (p-1)/2$. Тогда $y = p - x \leqslant (p-1)/2$ и $y^2 \equiv (p-x)^2 \equiv p^2 - 2px + x^2 \equiv x^2 \equiv \equiv a (\mathrm{mod}\, p)$. Следовательно, $\mathrm{QR}_p \subseteq S$.

Докажем второе утверждение. Для того чтобы показать, что $\#\mathrm{QR}_p = (p-1)/2$, достаточно доказать, что для всех чисел x, удовлетворяющих неравенству $0 < x < y \leqslant (p-1)/2$, выполняется условие $x^2 \not\equiv y^2 (\mathrm{mod}\, p)$. Предположим противное:

$x^2 - y^2 \equiv (x+y)(x-y) \equiv 0 (\mathrm{mod}\, p)$. Следовательно, $p \mid x+y$ или $p \mid x-y$. Поскольку $x + y < p$, возможен лишь последний вариант. Итак, $x = y$, что противоречит условиям теоремы.

Отсюда следует, что $\#\mathrm{QNR}_p = (p-1)/2$, поскольку $\#\mathrm{QNR}_p = \mathbb{Z}_p^* \backslash \mathrm{QR}_p$ и $\#\mathbb{Z}_p^* = p - 1$. □

Следствие 6.2. *Пусть p — простое число. Тогда любое число $a \in \mathrm{QR}_p$ имеет ровно два квадратных корня по модулю p. Обозначим один из них буквой x, тогда второй корень равен $-x(= p - x)$.* □

6.5.1 Задача о квадратичных вычетах

Часто возникает необходимость определить, является ли число n квадратичным вычетом по заданному модулю. Эта проблема называется **задачей о квадратичных вычетах** (quadratic residuosity problem).

Теорема 6.13 (Критерий Эйлера). *Пусть p — простое число. Для любого числа $x \in \mathbb{Z}_p^*$ условие $x \in \mathrm{QR}_p$ выполняется тогда и только тогда, когда*

$$x^{(p-1)/2} \equiv 1 (\mathrm{mod}\, p). \tag{6.5.1}$$

Доказательство. ($\Rightarrow$) Для числа $x \in \mathrm{QR}_p$ существует число $y \in \mathbb{Z}_p^*$, такое что $y^2 \equiv x (\mathrm{mod}\, p)$. Тогда из теоремы Ферма (теорема 6.10) следует, что $x^{(p-1)/2} \equiv y^{p-1} \equiv 1 (\mathrm{mod}\, p)$.

$\Leftarrow$) Пусть $x^{(p-1)/2} \equiv 1 (\mathrm{mod}\, p)$. Тогда число x является корнем полинома $y^{(p-1)/2} - 1 \equiv 0 (\mathrm{mod}\, p)$. Заметим, что группа $\mathbb{Z}_p$ является полем. По теореме 5.9.3 (раздел 5.4.3) каждый элемент этого поля является корнем полинома $y^p - y \equiv 0 (\mathrm{mod}\, p)$. Иначе говоря, каждый ненулевой элемент этого поля (т.е. каждый элемент группы $\mathbb{Z}_p^*$) является корнем полинома

$$y^{p-1} - 1 \equiv (y^{(p-1)/2} - 1)(y^{(p-1)/2} + 1) \equiv 0 (\mathrm{mod}\, p).$$

Все эти корни отличаются друг от друга, поскольку полином, имеющий степень $p-1$, может иметь не более $p-1$ корней. Следовательно, все $(p-1)/2$ корней полинома $y^{(p-1)/2} - 1 \equiv 0 (\mathrm{mod}\, p)$ должны быть разными. В ходе доказательства теоремы 6.12 было показано, что множество QR_p содержит ровно $(p-1)/2$ элементов, каждый из которых удовлетворяет условию $y^{(p-1)/2} - 1 \equiv 0 (\mathrm{mod}\, p)$. Любой другой элемент группы $\mathbb{Z}_p^*$ должен быть корнем уравнения $y^{(p-1)/2} + 1 \equiv 0 (\mathrm{mod}\, p)$. Следовательно, $x \in \mathrm{QR}_p$. □

Доказывая теорему 6.13, мы убедились, что если для $x \in \mathbb{Z}_p$ критерий не выполняется, то

$$x^{(p-1)/2} \equiv -1 (\mathrm{mod}\, p). \tag{6.5.2}$$

Критерий Эйлера позволяет проверить, является ли элемент группы $\mathbb{Z}_p^*$ квадратичным вычетом: если условие (6.5.1) выполняется, то $x \in \mathrm{QR}_p$, в противном случае выполняется условие (6.5.2) и $x \in \mathrm{QNR}_p$.

Пусть n — составное натуральное число, имеющее следующее разложение на простые множители:

$$n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}. \tag{6.5.3}$$

Тогда по теореме 6.8 группа $\mathbb{Z}_n$ изоморфна пространству $\mathbb{Z}_{p_1^{e_1}} \times \mathbb{Z}_{p_2^{e_2}} \times \dots \times \mathbb{Z}_{p_k^{e_k}}$. Поскольку изоморфизм сохраняет свойства арифметических операций, справедлива следующая теорема.

Теорема 6.14. *Пусть n — составное целое число с полным разложением (6.5.3). Следовательно, $x \in \mathrm{QR}_n$ тогда и только тогда, когда $x (\mathrm{mod}\, p_i^{e_i}) \in \mathrm{QR}_{p_i^{e_i}}$ т.е. тогда и только тогда, когда $x (\mathrm{mod}\, p_i) \in \mathrm{QR}_{p_i}$, где $p_i, i = 1, 2, \dots, k$ — простые числа.* □

Следовательно, при известном разложении числа n число $x \in \mathbb{Z}_n^*$ представляет собой квадратичный вычет по модулю n, если число $x (\mathrm{mod}\, p)$ является квадратичным вычетом для каждого простого числа p, удовлетворяющего условию $p \mid n$.

Однако если разложение числа n неизвестно, определить, является ли число квадратичным вычетом по модулю n, довольно трудно.

Определение 6.2 (Задача о квадратичном вычете (задача QRP)).

ВВОД: *n: составное число;*
$x \in \mathbb{Z}_n^*$.

ВЫВОД: *ДА, если $x \in \mathrm{QR}_n$.*

Задача QRP представляет собой хорошо известную трудноразрешимую задачу теории чисел и принадлежит к числу четырех алгоритмических задач, рассмотренных Гауссом в его работе "Disquisitiones Arithmeticae" [119]. Эффективное решение этой задачи зависит от решения других открытых проблем теории чисел. В главе 14 будет описана популярная **криптосистема Голдвассера–Микали** (Goldwasser–Micali). Безопасность этой криптосистемы гарантируется сложностью решения задачи QRP.

Теорема 6.15. *Пусть n — составное число, имеющее $k > 1$ простых множителей. Тогда ровно $\frac{1}{2^k}$-я часть элементов группы $\mathbb{Z}_n^*$ является квадратичным вычетом по модулю n.* □

Итак, для составного числа n эффективный алгоритм для распознавания квадратичного вычета по модулю n образует основу эффективной статистической проверки доли квадратичных вычетов в группе $\mathbb{Z}_n^*$, и, следовательно, теорема 6.15

позволяет создать эффективный алгоритм для ответа на вопрос, имеет ли число n два или три разных простых множителя. Если число n имеет два разных простых множителя, то квадратичными вычетами являются ровно четверть элементов группы $\mathbb{Z}_n^*$, а если три — одна восьмая их количества. Следовательно, существует возможность распознать ансамбли $E_{2-\text{Prime}}$ и $E_{3-\text{Prime}}$ (см. раздел 4.7).

В настоящее время не известно ни одного алгоритма, позволяющего распознать квадратичные вычеты по модулю n при неизвестном разложении числа n за время, полиномиально зависящее от размера числа n.

6.5.2 Символы Лежандра–Якоби

Для распознавания квадратичного вычета с помощью критерия Эйлера (6.5.1) приходится интенсивно выполнять операцию возведения в степень по модулю. Однако эту задачу можно решить намного быстрее. Этот алгоритм использует символы Лежандра и Якоби (Legendre–Jacobi).

Определение 6.3 (Символы Лежандра и Якоби). *Для каждого простого числа p и числа $x \in \mathbb{Z}_p^*$ символом Лежандра числа x по модулю p называется значение, вычисляемое по следующему правилу.*

$$\left(\frac{x}{p}\right) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{если } x \in \text{QR}_p, \\ -1, & \text{если } x \in \text{QNR}_p. \end{cases}$$

Пусть $n = p_1 p_2 \ldots p_k$ — разложение числа n на простые множители (некоторые из этих множителей могут повторяться). Число

$$\left(\frac{x}{n}\right) \stackrel{\text{def}}{=} \left(\frac{x}{p_1}\right)\left(\frac{x}{p_2}\right)\ldots\left(\frac{x}{p_k}\right)$$

называется символом Якоби числа x по модулю n.

В оставшейся части книги обозначение $\left(\frac{a}{b}\right)$ всегда будет относиться к символу Якоби, независимо от того, является ли число b простым.

Если число p — простое, сравнивая формулы (6.5.1) и (6.5.2) с определением 6.3, получаем

$$\left(\frac{x}{p}\right) = x^{(p-1)/2} (\text{mod}\, p). \tag{6.5.4}$$

Теорема 6.16. *Символ Якоби имеет следующие свойства.*

1. $\left(\frac{1}{n}\right) = 1.$
2. $\left(\frac{xy}{n}\right) = \left(\frac{x}{n}\right)\left(\frac{y}{n}\right).$

3. $\left(\frac{x}{mn}\right) = \left(\frac{x}{m}\right)\left(\frac{x}{n}\right)$.
4. *Если* $x \equiv y (\mathrm{mod}\, n)$, *то* $\left(\frac{x}{n}\right) = \left(\frac{y}{n}\right)$.

   Далее числа m *и* n *считаются нечетными числами.*
5. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$.
6. $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$.
7. *Если* $\gcd(m, n) = 1$ *и числа* $m, n > 2$, *то* $\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{(m-1)(n-1)}{4}}$.

Утверждения 6.16.1–6.16.4 непосредственно следуют из определения символа Якоби. Доказательство утверждений 6.16.5–6.16.7 не требует применения особых методов. Однако эти доказательства довольно громоздки и выходят за рамки книги. Любознательные читатели могут найти их в любом учебнике по теории чисел (например, [170, 176]).

Утверждение 6.16.7 известно под названием **закона квадратичной взаимности Гаусса** (Gauss' Law of Quadratic Reciprocity). Благодаря этому закону совсем нетрудно доказать, что вычисление символа Якоби $\left(\frac{x}{n}\right)$, где $\gcd(x, n) = 1$, равносильно вычислению наибольшего общего делителя, и, следовательно, имеет ту же временну́ю сложность.

Примечание 6.1. *При вычислении символа Якоби по теореме 6.16 вычисление правых частей в утверждениях 6.16.1–6.16.7 должно выполняться без возведения в степень. Поскольку* $\mathrm{ord}(-1) = 2$ *(при умножении), требуется лишь проверить равенство степеней. В алгоритме 6.2 эти вычисления сводятся к проверке факта, что эти степени делятся на число 2.* □

Алгоритм 6.2 позволяет вычислить символ Якоби, используя его свойства, перечисленные в теореме 6.16.

В алгоритме 6.2 каждый рекурсивный вызов функции Jacobi, зависящей от двух аргументов, либо делит первый аргумент на два, либо делит второй аргумент на первый по модулю x. Следовательно, алгоритм выполняет не более $\log_2 n$ вызовов, пока первый аргумент не уменьшится до единицы. Строго говоря, поскольку каждая арифметическая операция по модулю имеет сложность $O_B((\log n)^2)$, на выполнение алгоритма 6.2 затрачивается $O_B((\log n)^3)$ единиц времени.

Следует однако заметить, что, упрощая описание алгоритма, мы пожертвовали эффективностью!

При эффективной реализации на выполнение всех операций на шагах 3 и 4 можно затратить $O_B((\log n)^2)$ единиц времени. Эта ситуация совершенно аналогична эффективному вычислению наибольшего общего делителя, основанному

Алгоритм 6.2. Символ Якоби

ВВОД: нечетное целое число $n > 2$, целое число $x \in \mathbb{Z}_n^*$.

ВЫВОД: $\left(\frac{x}{n}\right)$.

Jacobi(x, n)

1. if $(x == 1)$ return (1);
2. if $(2 \mid x)$
   а) if $(2 \mid (n^2 - 1)/8)$ return (Jacobi$(x/2, n)$);
   б) return $(-$ Jacobi$(x/2, n))$;
   (* Далее число x считается нечетным *)
3. if $(2 \mid (x - 1)(n - 1)/4)$ return (Jacobi$(n \bmod x, x)$);
4. return $(-$ Jacobi$(n \bmod x, x))$.

на формуле (4.3.12). Следовательно, для $x \in \mathbb{Z}_n^*$ символ Якоби $\left(\frac{x}{n}\right)$ можно вычислить за $O_B((\log n)^2)$ единиц времени. Эффективная модификация алгоритма 6.2 описана в части I книги [79].

В сравнении со сложностью критерия Эйлера (5.4.5), имеющего временну́ю сложность $O_B((\log p)^3)$, сложность распознавания квадратичного вычета по простому модулю p в $\log p$ раз меньше.

Пример 6.5. Покажем, что $384 \in \mathrm{QNR}_{443}$.

Отслеживая алгоритм 6.2 шаг за шагом, получаем следующее.

$$\begin{aligned}
\text{Jacobi}(384, 443) &= -\,\text{Jacobi}(192, 443) = \\
&= -\,\text{Jacobi}(96, 443) = \\
&= -\,\text{Jacobi}(48, 443) = \\
&= -\,\text{Jacobi}(24, 443) = \\
&= -\,\text{Jacobi}(12, 443) = \\
&= -\,\text{Jacobi}(6, 443) = \\
&= -\,\text{Jacobi}(3, 443) = \\
&= \text{Jacobi}(2, 3) = \\
&= -\,\text{Jacobi}(1, 3) = \\
&= -1.
\end{aligned}$$

Следовательно, $384 \in \mathrm{QNR}_{443}$. □

В заключение необходимо заметить, что для вычисления символа Якоби с помощью алгоритма 6.2 не обязательно знать разложение числа n. Это очень важ-

ное свойство, которое широко применяется в криптографии с открытым ключом, например, в криптосистеме Голдвассера–Микали (14.3.3) и протоколе подбрасывания монеты Блюма (глава 19).

6.6 Квадратные корни по целочисленному модулю

В примере 6.2 мы вычислили квадратный корень по целочисленному модулю. Однако процедуру, которая там описана, нельзя считать алгоритмом, поскольку она использовала изоморфизм между группой и полем, позволивший свести сложную задачу к тривиальной: вычислению квадратных корней единицы и четырех, которые оказались квадратами в поле $\mathbb{Z}$. Эта задача вполне по силам даже школьникам начальных классов. Как правило, такой изоморфизм неизвестен, и существует громадное количество случаев, когда образ числа при изоморфизме не является квадратом в поле $\mathbb{Z}$.

Рассмотрим алгоритмы вычисления квадратных корней по целочисленному модулю. Начнем с простых модулей. По следствию 6.2 два корня квадратичных вычетов являются взаимно дополнительными по простому модулю. Значит, достаточно рассмотреть проблему вычисления одного из квадратных корней квадратичного вычета.

Для большинства нечетных чисел задача очень проста. В частности, к ним относятся простые числа p, такие как $p \equiv 3, 5, 7 (\mathrm{mod}\, 8)$.

6.6.1 Вычисление квадратных корней по простому модулю

Вариант $p \equiv 3, 7 (\mathrm{mod}\, 8)$.

В данном случае число $p+1$ делится на 4. Введем следующее обозначение для числа $a \in \mathrm{QR}_p$.

$$x \stackrel{\mathrm{def}}{=} a^{\frac{p+1}{4}} (\mathrm{mod}\, p).$$

Затем, поскольку $a^{(p-1)/2} \equiv 1 (\mathrm{mod}\, p)$,

$$x^2 \equiv a^{(p+1)/2} \equiv a^{(p-1)/2} a \equiv a (\mathrm{mod}\, p).$$

Итак, действительно, число x является квадратным корнем числа a по модулю p.

Вариант $p \equiv 5 (\mathrm{mod}\, 8)$.

В данном случае число $p+3$ делится на 8. Поскольку число $(p-1)/2$ является четным, число -1 удовлетворяет критерию Эйлера и является квадратичным

вычетом. Введем следующее обозначение для числа $a \in \mathrm{QR}_p$.

$$x \stackrel{\text{def}}{=} a^{\frac{p+3}{8}} (\mathrm{mod}\, p) \tag{6.6.1}$$

Из условия $a^{(p-1)/2} \equiv 1 (\mathrm{mod}\, p)$ следует, что $a^{(p-1)/4} \equiv \pm 1 (\mathrm{mod}\, p)$, поскольку в поле $\mathbb{Z}_p^*$ число 1 имеет два квадратных корня: 1 и -1. Следовательно,

$$x^2 \equiv a^{(p+3)/4} \equiv a^{(p-1)/4} a \equiv \pm a (\mathrm{mod}\, p).$$

Иначе говоря, мы доказали, что число x, определенное по формуле (6.6.1), является квадратным корнем либо числа a, либо числа $-a$. Если число положительное, алгоритм завершен. Если число отрицательное, то

$$-x^2 \equiv \left(\sqrt{-1}x\right)^2 \equiv a (\mathrm{mod}\, p).$$

Следовательно, число

$$x \stackrel{\text{def}}{=} \sqrt{-1} a^{\frac{p+3}{8}} (\mathrm{mod}\, p) \tag{6.6.2}$$

является искомым. Итак, задача сводится к вычислению $\sqrt{-1} (\mathrm{mod}\, p)$. Пусть число b — произвольный квадратичный невычет по модулю p. Тогда по критерию Эйлера

$$\left(b^{(p-1)/4}\right)^2 \equiv b^{(p-1)/2} \equiv -1 (\mathrm{mod}\, p),$$

следовательно, вместо числа $\sqrt{-1}$ можно применять число $b^{(p-1)/4} (\mathrm{mod}\, p)$. Кстати, поскольку

$$p^2 - 1 = (p+1)(p-1) = (8k+6)(8k+4) = 8(4k'+3)(2k''+1),$$

правая часть равенства является результатом умножения нечетного числа на 8. Следовательно, по теореме 6.16.6 выполняется условие $2 \in \mathrm{QNR}_p$. Иначе говоря, если $p \equiv 5 (\mathrm{mod}\, 8)$, вместо числа $\sqrt{-1}$ можно использовать число $2^{(p-1)/4}$. Тогда формула (6.6.2) преобразуется в следующую.

$$2^{(p-1)/4} a^{(p+3)/8} \equiv (4a)^{(p+3)/8} / 2 (\mathrm{mod}\, p). \tag{6.6.3}$$

Используя правую часть формулы (6.6.3), мы можем сэкономить одну операцию возведения в степень по модулю.

Временна́я сложность алгоритма 6.3 имеет порядок $O_B((\log p)^3)$.

Алгоритм 6.3. Квадратный корень по модулю $p \equiv 3, 5, 7 (\mathrm{mod}\, 8)$

ВВОД: простое число p, удовлетворяющее условию $p \equiv 3, 5, 7 (\mathrm{mod}\, 8)$; целое число $a \in \mathrm{QR}_p$.

ВЫВОД: квадратный корень числа a по модулю p.

1. if $(p \equiv 3, 7 (\mathrm{mod}\, 8))$ return $(a^{(p+1)/4} (\mathrm{mod}\, p))$;
   (* Далее считаем, что $p \equiv 5 (\mathrm{mod}\, 8)$. *)
2. if $(a^{(p-1)/4} \equiv 1 (\mathrm{mod}\, p))$ return $(a^{(p+3)/8} (\mathrm{mod}\, p))$;
3. return $((4a)^{(p+3)/8}/2)$.

Вычисление квадратных корней по простому модулю в общем случае

Описываемый ниже метод предложен Шенксом (Shanks) (см. раздел 1.5.1 в книге [179]).

Для произвольного простого числа p справедлива формула

$$p - 1 = 2^e q,$$

где q — нечетное число, причем $e \geqslant 1$. По теореме 5.2 (раздел 5.2.3) циклическая группа $\mathbb{Z}_p^*$ имеет единственную циклическую подгруппу G порядка 2^e. Очевидно, порядки квадратичных вычетов в подгруппе G представляют собой степень числа 2, поскольку эти вычеты являются делителями числа 2^{e-1}. Если $a \in \mathrm{QR}_p$, то, поскольку

$$a^{\frac{p-1}{2}} \equiv (a^q)^{2^{e-1}} \equiv 1 (\mathrm{mod}\, p),$$

число $a^q (\mathrm{mod}\, p)$ принадлежит группе G и, разумеется, является квадратичным вычетом. Следовательно, существует четное число k, удовлетворяющее условию $0 \leqslant k < 2^e$, такое что

$$a^q g^k \equiv 1 (\mathrm{mod}\, p), \tag{6.6.4}$$

где число g является порождающим элементом группы G. Допустим, что нам известны числа g и k. Тогда, полагая

$$x \stackrel{\mathrm{def}}{=} a^{\frac{q+1}{2}} g^{\frac{k}{2}},$$

легко убедиться, что $x^2 \equiv a (\mathrm{mod}\, p)$.

Итак, задача сводится к двум подзадачам: 1) найти порождающий элемент g группы G и 2) найти наименьшее неотрицательное четное число k, удовлетворяющее условию (6.6.4).

Первая подзадача довольно проста. Для любого числа $f \in \mathrm{QNR}_p$, поскольку число q — нечетное, $f^q \in \mathrm{QNR}_p$ и $\mathrm{ord}_p(f^q) = 2^e$, число f^q является порождающим элементом группы G. Найти число f сравнительно легко. Для этого достаточно

извлечь случайный элемент $f \in \mathbb{Z}_p^*$ и проверить условие $\left(\frac{f}{p}\right) = -1$, используя алгоритм 6.2. Поскольку половина элементов поля $\mathbb{Z}_p^*$ является квадратичными невычетами, вероятность найти искомое число равна $1/2$.

Решение второй подзадачи также не вызывает особых затруднений. Число k, удовлетворяющее условию (6.6.4), можно быстро найти, используя тот факт, что порядки неединичных квадратичных вычетов в группе G являются степенями числа 2. Итак, положив

$$b \stackrel{\text{def}}{=} a^q \equiv a^q g^{2^e} (\mathrm{mod}\, p), \tag{6.6.5}$$

приходим к выводу, что $b \in G$. Теперь можно найти наименьшее целое число m, удовлетворяющее условию $0 \leqslant m < e$, такое что

$$b^{2^m} \equiv 1 (\mathrm{mod}\, p), \tag{6.6.6}$$

и модифицировать число b следующим образом:

$$b \leftarrow b g^{2^{e-m}} \equiv a^q g^{2^{e-m}} (\mathrm{mod}\, p). \tag{6.6.7}$$

Заметим, что после модификации (6.6.7) порядок числа b уменьшается по сравнению с определением (6.6.5). При этом число b остается квадратичным вычетом в группе G, а уменьшенный порядок продолжает быть степенью числа 2. Следовательно, скорость убывания порядка является степенью числа 2, и после повторных вычислений по формулам (6.6.6) и (6.6.7) число m в выражении (6.6.6) значительно уменьшается. Если в выражении (6.6.6) $m = 0$, то $b = 1$, и, следовательно, формула (6.6.7) сводится к формуле (6.6.4), а число k можно найти, накапливая степень 2^m на каждом шаге итерации. Количество итераций не превышает числа e.

Итак, теперь можно сформулировать следующий алгоритм.

Поскольку $e < \log_2 p$, временна́я сложность этого алгоритма имеет порядок $O_B((\log p)^4)$.

Примечание 6.2. *Для упрощения изложения мы следовали принципам, предложенным Шенксом, в частности, при поиске четной степени числа k (вторая подзадача). Это не самый эффективный способ решения задачи. Явного определения числа k можно избежать, вычислив число $g^{k/2}$ в качестве побочного результата выполнения шага 3. Такой подход позволяет сэкономить одну операцию возведения в степень по модулю на шаге 4. Оптимизированный вариант алгоритма Шенкса приведен в книге [179] (алгоритм 1.5.1).* □

В заключение отметим, что все три варианта алгоритма 6.3 являются частными случаями алгоритма 6.4.

Алгоритм 6.4. Квадратный корень по простому модулю

ВВОД: простое число p; целое число $a \in \mathrm{QR}_p$.

ВЫВОД: квадратный корень числа a по модулю p.

1. (* Инициализация *)
   Положим $p-1 = 2^e q$, где число q — нечетное; $b \leftarrow a^q (\mathrm{mod}\, p)$; $r \leftarrow e$; $k \leftarrow 0$.
2. (* Подзадача 1, применяется алгоритм 6.2. *)
   Находим $f \in \mathrm{QNR}_p$; $g \leftarrow f^q (\mathrm{mod}\, p)$.
3. (* Подзадача 2, находим четную степень числа k. *)
   while $(b \neq 1)$ do
   а) Найти наименьшее неотрицательное целое число m, удовлетворяющее условию $b^{2^m} \equiv 1 (\mathrm{mod}\, p)$.
   б) $b \leftarrow b g^{2^{r-m}} (\mathrm{mod}\, p)$; $k \leftarrow k + 2^{r-m}$; $r \leftarrow m$.
4. return $(a^{(q+1)/2} g^{k/2} (\mathrm{mod}\, p))$.

6.6.2 Вычисление квадратных корней по составному модулю

Благодаря теореме 6.8 мы знаем, что, если $n = pq$, где p и q — простые числа, группа $\mathbb{Z}_n^*$ изоморфна пространству $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$. Поскольку изоморфизм сохраняет свойства арифметических операций, отношение

$$x^2 \equiv y (\mathrm{mod}\, n)$$

выполняется тогда и только тогда, когда оно справедливо как по модулю p, так и по модулю q. Следовательно, если известно разложение числа n на множители, квадратный корень по модулю n можно вычислить с помощью следующего алгоритма.

Алгоритм 6.5. Квадратный корень по составному модулю

ВВОД: простые числа p, q, удовлетворяющие условию $n = pq$; целое число $y \in \mathrm{QR}_n$.

ВЫВОД: квадратный корень числа y по модулю n.

1. $x_p \leftarrow \sqrt{y (\mathrm{mod}\, p)}$;
   $x_q \leftarrow \sqrt{y (\mathrm{mod}\, q)}$; (* Применяем алгоритмы 6.3 или 6.4. *)
2. return $(\bar{1}_p x_p + \bar{1}_p x_p (\mathrm{mod}\, n))$. (* Применяем алгоритм 6.1. *)

Очевидно, что временна́я сложность алгоритма 6.5 имеет порядок $O_B((\log n)^4)$.

Из следствия 6.2 видно, что число $y(\mathrm{mod}\, p)$ имеет два разных квадратных корня, которые мы обозначим как x_p и $p - x_p$ соответственно. Аналогично обозначим квадратные корни числа $y(\mathrm{mod}\, q)$ как x_q и $q - x_q$ соответственно. Вследствие изоморфизма между группой $\mathbb{Z}_n^*$ и пространством $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ (теорема 6.2) число $y \in \mathrm{QR}_n$ имеет в группе $\mathbb{Z}_n^*$ ровно четыре квадратных корня, которые можно вычислить с помощью алгоритма 6.5.

$$\left.\begin{aligned} x_1 &\equiv \bar{1}_p x_p + \bar{1}_q x_q \\ x_2 &\equiv \bar{1}_p x_p + \bar{1}_q (q - x_q) \\ x_3 &\equiv \bar{1}_p (p - x_p) + \bar{1}_q x_q \\ x_4 &\equiv \bar{1}_p (p - x_p) + \bar{1}_q (q - x_q) \end{aligned}\right\} (\mathrm{mod}\, n) \qquad (6.6.8)$$

Применяя формулу (6.6.8) на шаге 2 алгоритма 6.5, можно вычислить все четыре квадратных корня исходного числа.

В качестве упражнения поставим следующую задачу: сколько разных квадратных корней имеет число $y \in \mathrm{QR}_n$, если $n = pqr$, где p, q и r — разные простые числа?

Итак, если известно разложение числа n на множители, существует эффективный алгоритм, позволяющий вычислить квадратные корни любого заданного числа из множества QR_n. А что делать, если разложение числа n на множители неизвестно? На этот вопрос дает ответ третья часть следующей теоремы.

Теорема 6.17. *Пусть $n = pq$, где числа p и q — разные нечетные простые числа и $y \in \mathrm{QR}_n$. Тогда четыре корня числа y, вычисленные по формуле (6.6.8), обладают следующими свойствами.*

1. *Все они отличаются друг от друга.*
2. $x_1 + x_4 = x_2 + x_3 = n$.
3. $\gcd(x_1 + x_2, n) = \gcd(x_3 + x_4, n) = q, \gcd(x_1 + x_3, n) = \gcd(x_2 + x_4, n) = p$.

Доказательство.

1. Учитывая смысл обозначений $\bar{1}_p$ и $\bar{1}_q$, определенных формулами (6.2.15) и (6.2.16), приходим к выводу, что, например, $x_1(\mathrm{mod}\, q) = x_q$ и $x_2(\mathrm{mod}\, q) = q - x_q$. Напомним, что x_q и $q - x_q$ — два разных квадратных корня числа $y(\mathrm{mod}\, q)$. Из условия $x_1 \not\equiv x_2 (\mathrm{mod}\, q)$ следует, что $x_1 \not\equiv x_2 (\mathrm{mod}\, n)$, т.е. числа x_1 и x_2 отличаются друг от друга. Другие варианты доказываются аналогично.
2. Из формулы (6.6.8) следует, что

$$x_1 + x_4 = x_2 + x_3 = \bar{1}_p p + \bar{1}_q q.$$

Правая часть этого уравнения сравнима с нулем по модулю p и по модулю q. Поскольку эти корни принадлежат группе $\mathbb{Z}_n^*$, выполняется условие $0 < x_1 + x_4 = x_2 + x_3 < 2n$. Очевидно, что число n — единственное число в интервале $(0, 2n)$, сравнимое с нулем по модулю p и по модулю q. Следовательно, $x_1 = n - x_4$ и $x_2 = n - x_3$.

3. Рассмотрим лишь число $x_1 + x_2$. Другие варианты доказываются аналогично. Исследуя формулу (6.6.8), приходим к выводу, что

$$x_1 + x_2 = 2 \cdot \bar{1}_p x_p + \bar{1}_q q.$$

Следовательно, $x_1 + x_2 (\mathrm{mod}\, p) \equiv 2x_p \neq 0$ и $x_1 + x_2 \equiv 0 (\mathrm{mod}\, q)$. Таким образом, число $x_1 + x_2$ не равно нулю и кратно числу q, но не кратно числу p. Отсюда следует, что $\gcd(x_1 + x_2, n) = q$. □

Допустим, что существует алгоритм A, получающий на вход число $y \in \mathrm{QR}_n$ и вычисляющий число x, удовлетворяющее условию $x^2 \equiv y (\mathrm{mod}\, n)$. Следовательно, для вычисления квадратного корня x' числа x^2 можно выполнить алгоритм $A(x^2, n)$. По теореме 6.17.3 вероятность события $0 < \gcd(x + x', n) < n$ равна $1/2$ (вероятностное пространство состоит из четырех квадратных корней числа y). Значит, алгоритм A эффективно выполняет факторизацию числа n.

Объединяя алгоритм 6.5 и теорему 6.17.3, получаем следующее утверждение.

Следствие 6.3. *Пусть $n = pq$, где числа p и q — разные нечетные простые числа. Тогда факторизация числа n эквивалентна вычислению квадратного корня по модулю n.* □

Из теоремы 6.17.2 и того факта, что число n — нечетное, вытекает следующее утверждение.

Следствие 6.4. *Пусть $n = pq$, где числа p и q — разные нечетные простые числа. Тогда для любого числа $y \in \mathrm{QR}_n$ два квадратных корня числа y меньше $n/2$, а другие два корня — больше $n/2$.* □

6.7 Целые числа Блюма

Целые числа Блюма широко применяются в криптографии с открытым ключом.

Определение 6.4 (Целые числа Блюма). *Составное целое число n называется целым числом Блюма, если $n = pq$, где p и q — разные простые числа, удовлетворяющие условию $p \equiv q \equiv 3 (\mathrm{mod}\, 4)$.*

Целые числа Блюма обладают рядом интересных свойств. Некоторые из них оказались весьма полезными для криптографии с открытым ключом и криптографических протоколов.

Теорема 6.18. *Пусть n — целое число Блюма. Тогда выполняются следующие условия.*

1. $\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = -1$, *следовательно,* $\left(\frac{-1}{n}\right) = 1$.
2. *Для* $y \in \mathbb{Z}_n^*$ *если* $\left(\frac{y}{n}\right) = 1$, *то либо* $y \in \mathrm{QR}_n$, *либо* $-y = n - y \in \mathrm{QR}_n$.
3. *Любое число* $y \in \mathrm{QR}_n$ *имеет четыре квадратных корня* $u, -u, v$ *и* $-v$, *удовлетворяющих следующим условиям.* $\left(\frac{u}{p}\right) = 1$, $\left(\frac{u}{q}\right) = 1$, *т.е.* $u \in \mathrm{QR}_n$;
   - *а)* $\left(\frac{-u}{p}\right) = -1$, $\left(\frac{-u}{q}\right) = -1$;
   - *б)* $\left(\frac{v}{p}\right) = -1$, $\left(\frac{v}{q}\right) = 1$;
   - *в)* $\left(\frac{-v}{p}\right) = 1$, $\left(\frac{-v}{q}\right) = -1$.
4. *Функция* $f(x) = x^2 (\mathrm{mod}\, n)$ *является перестановкой над множеством* QR_n.
5. *Для любого числа* $y \in \mathrm{QR}_n$ *существует только один квадратный корень, который меньше* $n/2$, *и символ Якоби которого равен единице.*
6. *Группа* $\mathbb{Z}_n^*$ *разбивается на четыре класса эквивалентности: одну мультипликативную группу,* QR_n, *и три смежных класса* $(-1)\mathrm{QR}_n, \xi\mathrm{QR}_n$ *и* $(-\xi)\mathrm{QR}_n$, *где* ξ — *квадратный корень единицы, символ Якоби которого равен* -1.

Доказательство.

1. Из условия $p \equiv 3 (\mathrm{mod}\, 4)$ следует, что $\frac{p-1}{2} = 2k + 1$. Тогда по критерию Эйлера (6.5.1) получаем

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = (-1)^{2k+1} = -1.$$

   Аналогично $\left(\frac{-1}{q}\right) = -1$.

2. Из условия $\left(\frac{y}{n}\right) = 1$ следует, что $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = 1$ или $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$. В первом случае $y \in \mathrm{QR}_n$ вследствие определения символа Лежандра (определение 6.3) и теоремы 6.14. Во втором случае из утверждения 1 следует, что $\left(\frac{-y}{p}\right) = \left(\frac{-y}{q}\right) = 1$. Следовательно, $-y \in \mathrm{QR}_n$.

3. Прежде всего, из теоремы 6.17.2 следует, что число x действительно имеет четыре квадратных корня. Обозначим их через u, $-u(=n-u)$, v и $-v$. Далее, из условия $u^2 \equiv v^2 (\mathrm{mod}\, n)$ следует, что $(u+v)(u-v) \equiv 0 (\mathrm{mod}\, p)$, т.е. $u \equiv \pm v (\mathrm{mod}\, p)$. Аналогично $u \equiv \pm v (\mathrm{mod}\, q)$. Однако по теореме 6.17.1 $u \not\equiv \pm v (\mathrm{mod}\, n)$, следовательно, возможны только два варианта:

$$u \equiv v (\mathrm{mod}\, p) \text{ и } u \equiv -v (\mathrm{mod}\, q),$$

или

$$u \equiv -v (\mathrm{mod}\, p) \text{ и } u \equiv v (\mathrm{mod}\, q).$$

Отсюда, с учетом утверждения 1, следует, что $\left(\frac{u}{n}\right) = -\left(\frac{v}{n}\right)$.

Итак, если $\left(\frac{u}{n}\right) = 1$, то $\left(\frac{v}{n}\right) = -1$, а если $\left(\frac{u}{n}\right) = -1$, то $\left(\frac{v}{n}\right) = 1$. Не ограничивая общности, можно утверждать, что четыре разных характеризации символа Лежандра 3.1–3.4 следуют из мультипликативного свойства символа Лежандра–Якоби и утверждения 1.

4. Из утверждения 3 следует, что для любого $y \in \mathrm{QR}_n$ существует единственный элемент $x \in \mathrm{QR}_n$, удовлетворяющий условию $f(x) = y$. Значит, $f(x)$ представляет собой взаимно-оуднозначное отображение "на", т.е. перестановку над множеством QR_n.

5. Из утверждения 3 следует, что квадратным корнем, символ Якоби которого равен единице, является число u или $n-u$. Поскольку число n — нечетное, только один из этих корней может быть меньше $n/2$. (Следовательно, только один квадратный корень, символ Якоби которого равен -1, может быть меньше $n/2$, а два остальных — больше $n/2$, причем их символы Якоби имеют противоположный знак.)

6. Легко проверить, что множество QR_n образует группу относительно умножения по модулю n, причем ее единицей является число 1. Из утверждения 3 следует, что четыре разных квадратных корня единицы имеют четыре разных характеризации символа Лежандра 3.1–3.4. Следовательно, четыре множества $\mathrm{QR}_n, (-1)\mathrm{QR}_n, \xi\mathrm{QR}_n$ и $(-\xi)\mathrm{QR}_n$ являются попарно непересекающимися. Эти четыре множества образуют группу $\mathbb{Z}_n^*$, поскольку по теореме 6.15 $\#\mathrm{QR}_n = \frac{\#\mathbb{Z}_n^*}{4}$. □

6.8 Резюме

В главе рассмотрены следующие темы теории чисел.

- Линейная сравнимость.
- Китайская теорема об остатках (с алгоритмом).

- Теоремы Лагранжа, Эйлера и Ферма.
- Квадратичные вычеты и символы Лежандра–Якоби (с алгоритмом).
- Квадратные корни по целочисленному модулю и их связь с разложением числа на множители (с алгоритмом вычисления квадратного корня).
- Целые числа Блюма и их свойства.

Кроме того, изучено несколько важных алгоритмов (применение китайской теоремы об остатках, вычисление символа Якоби, вычисление квадратного корня), изложены их рабочие принципы и приведены оценки их временно́й сложности. Эти алгоритмы имеют не только теоретическое, но и практическое значение: они часто используются в криптографии с открытым ключом и криптографических протоколах.

Факты и алгоритмы, изложенные в этой главе, будут часто упоминаться в остальной части книги.

Упражнения

6.1. Пусть m и n — положительные целые числа, удовлетворяющие условию $m \mid n$. Докажите, что операция "$(\mathrm{mod}\, m)$" разбивает группу $\mathbb{Z}_n$ на n/m классов эквивалентности, состоящих из m элементов каждый.

6.2. При тех же предположениях, что и в предыдущей задаче, докажите, что $\mathbb{Z}_n / m\mathbb{Z}_n = \mathbb{Z}_m$.

6.3. Используя китайскую теорему об остатках (алгоритм 6.1), найдите элемент группы $\mathbb{Z}_{35}$, образом которого при изоморфизме, определенном в теореме 6.8, является пара $(2, 3) \in \mathbb{Z}_5 \times \mathbb{Z}_7$. Докажите, что этот элемент имеет максимальный порядок.

6.4. Применяя метод, описанный в примере 6.2, найдите три остальных квадратных корня числа 29 в группе $\mathbb{Z}_{35}^*$. Вычислите четыре квадратных корня единицы в группе $\mathbb{Z}_{35}^*$.

Подсказка: $29(\mathrm{mod}\, 5) = 4$, причем число 4 имеет квадратные корни 2 и 3 $(= -2(\mathrm{mod}\, 5))$. Кроме того, $29(\mathrm{mod}\, 7) = 1$, причем число 1 имеет квадратные корни 1 и 6 $(= -1(\mathrm{mod}\, 7))$. Четыре квадратных корня числа 29 по модулю 35 изоморфно отображаются в пары $(2, 1), (2, 6), (3, 1)$ и $(3, 6)$ в пространстве $\mathbb{Z}_5 \times \mathbb{Z}_7$.

6.5. Найдите нечетное составное число n, которое не делилось бы на квадрат простого числа, однако удовлетворяло бы условию $\gcd(n, \phi(n)) > 1$.

6.6. Пусть $m \mid n$. Докажите, что для любого числа $x \in \mathbb{Z}_n^*$ выполняется условие $\mathrm{ord}_m(x) \mid \mathrm{ord}_n(x)$.

6.7. Пусть $n = pq$, причем числа p и q — простые. Поскольку $p-1 \mid \phi(n)$, в группе $\mathbb{Z}_n^*$ существуют элементы, порядок которых делится на $p-1$. (Аналогично в группе $\mathbb{Z}_n^*$ существуют элементы, порядок которых делится на $q-1$.) Докажите, что для любого числа $g \in \mathbb{Z}_n^*$ из условий $\mathrm{ord}_n(g) \mid p-1$ и $\mathrm{ord}_n(g) \nmid q-1$ следует, что $\gcd(g-1, n) = q$. (Аналогично для любого элемента $h \in \mathbb{Z}_n^*$ из условий $\mathrm{ord}_n(h) \mid q-1$ и $\mathrm{ord}_n(h) \nmid p-1$ следует, что $\gcd(h-1, n) = p$.)

6.8. Пусть $n = pq$, причем числа p и q — простые. Докажите, что для любого элемента $g \in \mathbb{Z}_n^*$ выполняется условие $g^{p+q} \equiv g^{n+1} (\mathrm{mod}\, n)$. Покажите, что если $|p| \approx |q|$, то верхней оценкой сложности разложения числа n на множители является число $n^{1/4}$.

Подсказка: найдите число $p+q$ из условия $g^{n+1} (\mathrm{mod}\, n)$, используя λ-алгоритм Полларда; затем разложите число n на множители с помощью чисел $p+q$ и pq.

6.9. Пусть p — простое число. Докажите, что порождающий элемент группы $\mathbb{Z}_p^*$ должен быть квадратичным невычетом. Аналогично, допустим, что n — нечетное составное число. Докажите, что элементы группы $\mathbb{Z}_n^*$, имеющие максимальный порядок, должны быть квадратичными невычетами.

6.10. Проверка, является ли число квадратичным невычетом по модулю p, с помощью критерия Эйлера, в $\log p$ раз медленнее, чем проверка с помощью вычисления символа Лежандра. Почему?

6.11. Разложите на множители число 35, используя квадратные корни, вычисленные в упражнении 6.4.

6.12. Покажите, что группа QR_n является подгруппой группы $J_n(1)$, а та, в свою очередь, является подгруппой группы $\mathbb{Z}_n^*$.

6.13. Пусть $n = pq$, где p и q — разные простые числа. При каких условиях $-1 \in \in \mathrm{QR}_n$? При каких условиях $\left(\frac{-1}{n}\right) = -1$?

6.14. Пусть n — целое число Блюма. Найдите функцию, обратную к функции $f(x) = x^2 (\mathrm{mod}\, n)$ над группой QR_n.

Подсказка: примените китайскую теорему об остатках (алгоритм 6.1) к первому случаю алгоритма 6.6.1.

6.15. Пусть $n = pq$ — целое число Блюма, удовлетворяющее условию $\gcd(p-1, q-1) = 2$. Докажите, что группа $J_n(1)$ является циклической.

Подсказка: примените китайскую теорему об остатках и сконструируйте элемент, используя порождающие элементы групп $\mathbb{Z}_p^*$ и $\mathbb{Z}_q^*$. Докажите, что он принадлежит группе $J_n(1)$ и что его порядок равен $\#J_n(1)$.

Часть III

Основные методы криптографии

Эта часть состоит из четырех глав, посвященных основным криптографическим методам обеспечения конфиденциальности и целостности данных. В главе 7 описываются симметричные методы шифрования. В главе 8 излагаются асимметричные методы шифрования. Глава 9 посвящена стойкости основных и популярных асимметричных криптографических функций, применяемых в идеальных условиях (когда данные являются случайными). В заключение, в главе 10 рассматриваются основные методы обеспечения целостности данных.

Основные криптографические алгоритмы и схемы, описанные в этой части, можно считать "учебными", поскольку они изложены во многих учебниках по криптографии. Мы продемонстрируем слабость этих алгоритмов и схем, описав большое количество атак. В некоторых случаях слабые места алгоритмов исправить невозможно. Однако мы не будем останавливаться на уровне "учебных" методов. В последующих главах будут рассмотрены реальные прикладные алгоритмы шифрования и механизмы обеспечения целостности данных. Большинство из этих алгоритмов являются результатом усовершенствования их "учебных" аналогов.

Изучая эту часть, читатели, не собиравшиеся углубляться в методы прикладной криптографии и способы обеспечения их повышенной безопасности, смогут самостоятельно убедиться в слабости "учебной криптографии".

Глава 7

Шифрование — симметричные методы

7.1 Введение

Секретность — основное понятие криптографии. **Шифрование** — практическое средство обеспечения секретности информации. Современные методы шифрования представляют собой математические преобразования (алгоритмы), в которых сообщения рассматриваются как числа или алгебраические элементы в некотором пространстве. Эти алгоритмы отображают область "осмысленных сообщений" в область "бессмысленных сообщений". Сообщения, относящиеся к числу "осмысленных", а также исходные данные алгоритма шифрования называются **открытым текстом** (cleartext), а бессмысленное сообщение, являющееся результатом работы алгоритма шифрования, называется **зашифрованным текстом** (ciphertext). Если пренебречь смыслом сообщения, то входные данные алгоритма шифрования удобно называть **исходным текстом** (plaintext), который не обязан быть осмысленным. Например, исходное сообщение может оказаться как случайным шумом, так и зашифрованным текстом. Мы уже описывали такие протоколы в главе 2. Следовательно, исходный текст и зашифрованный текст образуют взаимосвязанную пару понятий: первое понятие означает входное сообщение, а второе — результат работы алгоритма шифрования.

Для того чтобы иметь возможность восстановить информацию, шифрующие преобразования должны быть обратимыми. Обратное преобразование называется **расшифровкой** (decryption). Весьма удобно параметризовать алгоритмы шифрования и расшифровки с помощью криптографических ключей. Алгоритмы шифрования и расшифровки, а также описание формата сообщений и ключей образуют криптографическую систему, или **криптосистему** (cryptosystem).

Шеннон (Shannon) характеризует желаемое семантическое свойство криптосистемы следующим образом: пространство зашифрованных текстов является пространством всевозможных сообщений, а пространство открытых текстов (но не исходных текстов!) представляет собой разреженную область этого пространства, в которой сообщения обладают чрезвычайно простой статистической структурой,

т.е. являются осмысленными; хороший алгоритм шифрования является преобразованием перемешивания, которое равномерно распределяет осмысленное сообщение, принадлежащее разреженной области, по всему пространству сообщений [264]. Шеннон так описывал это свойство перемешивания:

$$\lim_{n\to\infty} \bigcup_{n} F^n R = \Omega. \tag{7.1.1}$$

Здесь F — отображение (алгоритм шифрования) пространства Ω (пространства сообщений) в самого себя, R — исходная и очень небольшая область (область открытых текстов) пространства Ω. Семантическое определение Шеннона означает, что хороший алгоритм шифрования должен обладать следующим свойством: он может отображать малую исходную область пространства во все пространство.

В настоящее время, особенно после изобретения криптографии с открытым ключом, уже не требуется, чтобы алгоритм шифрования отображал пространство сообщений в самого себя (хотя многие криптосистемы, как с секретным, так и с открытым ключом, обладают этим свойством). Несмотря на это, семантическое требование Шеннона, чтобы алгоритм шифрования перемешивал сообщения, остается весьма актуальным. Современное определение **семантической стойкости** (semantic security) алгоритма шифрования, которое будет дано в разделе 14.3, по существу означает, что распределение зашифрованного текста по пространству сообщений невозможно отличить от равномерного распределения в том же пространстве.

7.1.1 Структурная схема главы

В главе рассматриваются основные понятия, связанные с криптосистемами, несколько известных криптосистем и стандартные операции. Глава начинается с формального определения криптосистемы (раздел 7.2). Затем описываются несколько классических шифров (разделы 7.3, 7.4). В разделе 7.5 продемонстрирована важная роль, которую классические шифры играют при создании современных шифров и криптографических протоколов. После классических шифров в главе описаны два современных блочных шифра: Data Encryption Standard (DES) — в разделе 7.6 и Advanced Encryption Standard (AES) — в разделе 7.7. Кроме того, в этих разделах излагается стратегия их разработки. Раздел 7.7.5 содержит краткое обсуждение положительного влияния стандарта AES на прикладную криптографию. Разделы, посвященные симметричным методам, содержат описание стандартных операций над блочными шифрами, обеспечивающими вероятностное шифрование (раздел 7.8). Введение в современные симметричные методы шифрования завершается постановкой классической задачи об установлении канала для обмена ключами (раздел 7.9).

7.2 Определение

Формальное определение криптосистемы выглядит следующим образом.

Определение 7.1 (Криптографическая система). *Криптографическая система состоит из таких компонентов.*

- *Пространство исходных сообщений $\mathcal{M}$: множество строк над некоторым алфавитом.*
- *Пространство зашифрованных текстов $\mathcal{C}$: множество возможных зашифрованных сообщений.*
- *Пространство ключей шифрования $\mathcal{K}$ — множество возможных ключей шифрования и пространство ключей расшифровки $\mathcal{K}'$ — множество возможных ключей расшифровки.*
- *Эффективный алгоритм генерации ключей $\mathcal{G} : \mathbb{N} \mapsto \mathcal{K} \times \mathcal{K}'$.*
- *Эффективный алгоритм шифрования $\mathcal{E} : \mathcal{M} \times \mathcal{K} \mapsto \mathcal{C}$.*
- *Эффективный алгоритм расшифровки $\mathcal{D} : \mathcal{C} \times \mathcal{K}' \mapsto \mathcal{M}$.*

Для целого числа 1^ℓ результатом алгоритма $\mathcal{G}(1^\ell)$ является пара ключей $(ke, kd) \in \mathcal{K} \times \mathcal{K}'$, имеющих длину ℓ.

Для числа $ke \in \mathcal{K}$ и сообщения $m \in \mathcal{M}$ обозначение

$$c = \mathcal{E}_{ke}(m)$$

относится к преобразованию шифрования и читается как "преобразование c является шифрованием сообщения m ключом ke", а обозначение

$$m = \mathcal{D}_{kd}(c)$$

относится к преобразованию расшифровки и читается как "преобразование m является расшифровкой сообщения c с помощью ключа kd". Необходимо, чтобы для всех сообщений $m \in \mathcal{M}$ и всех ключей $ke \in \mathcal{K}$ существовал ключ $kd \in \mathcal{K}'$, удовлетворяющий условию

$$\mathcal{D}_{kd}(\mathcal{E}_{ke}(m)) = m. \tag{7.2.1}$$

В остальной части книги для описания абстрактной криптосистемы будет использоваться формальное определение, за исключением нескольких мест, где будут применяться общепринятые обозначения. Понятие криптосистемы проиллюстрировано на рис. 7.1.

Определение 7.1 относится к криптосистемам, использующим как секретные (secret), так и открытые (public) ключи (криптосистемы с открытым ключом будут

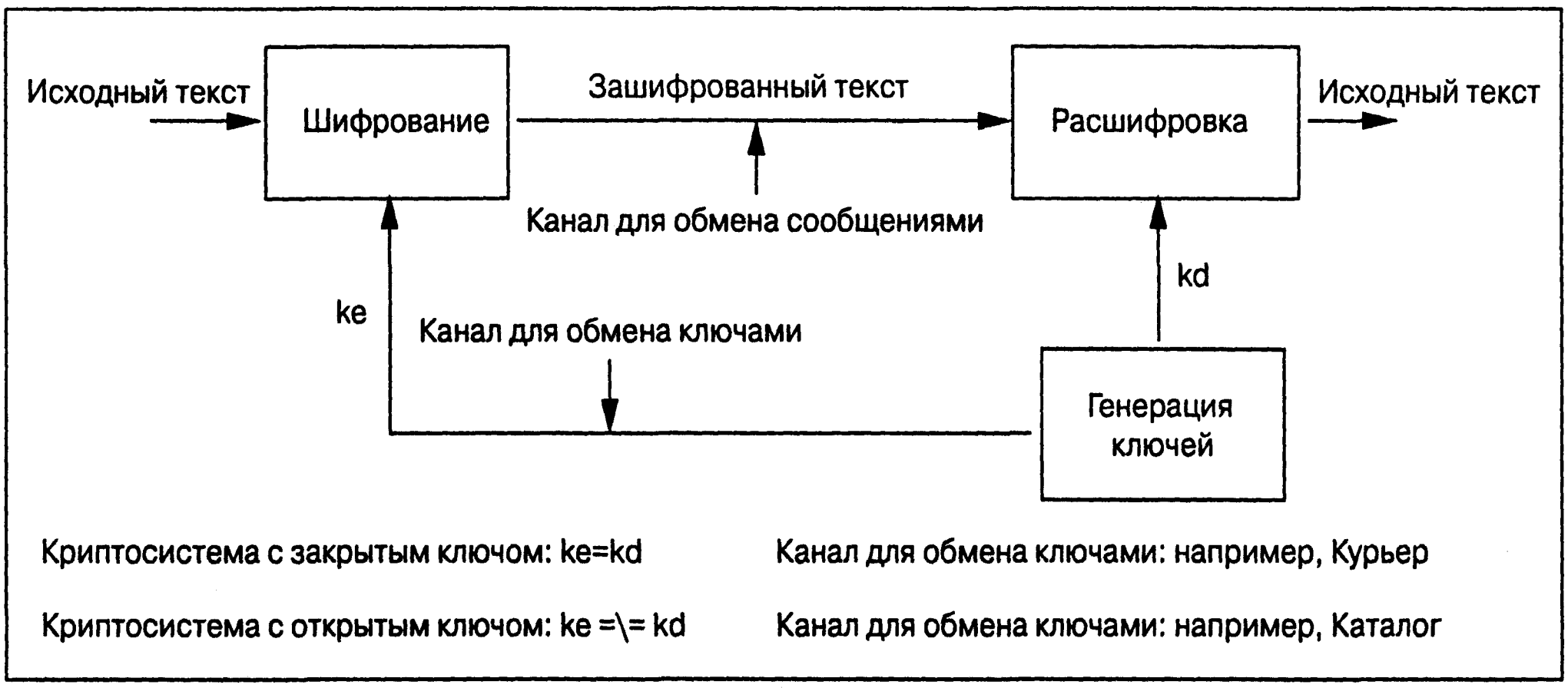


Рис. 7.1. Упрощенная схема криптографической системы

введены в следующей главе). В **криптосистемах с секретным ключом** (secret-key cryptosystems) шифрование и расшифровка осуществляются с помощью одного и того же ключа. Отправитель, шифрующий сообщение, должен передать ключ расшифровки адресату, получающему и расшифровывающему зашифрованное сообщение. Благодаря условию $kd = ke$ криптосистемы с секретным ключом получили другое название: **симметричные криптосистемы** (symmetric cryptosystem). В **криптосистемах с открытым ключом** (public-key cryptosystem) шифрование и расшифровка используют разные ключи: для каждого ключа $ke \in \mathcal{K}$ существует ключ $kd \in \mathcal{K}'$, причем они отличаются друг от друга и взаимосвязаны. Ключ шифрования ke не обязательно держать в секрете, и пользователь, владеющий ключом ke, может расшифровать текст, зашифрованный с помощью этого ключа, используя соответствующий закрытый (private) ключ kd. Благодаря условию $ke \neq kd$ криптосистемы с открытым ключом получили другое название: **асимметричные криптосистемы** (asymmetric cryptosystems).

Требование эффективности, предъявляемое к алгоритмам шифрования, позволяет отнести их к классу полиномиальных алгоритмов. Следовательно, несмотря на то, что абстрактное определение алгоритма $\mathcal{E}$ выглядит как определение детерминированного алгоритма, он может иметь внутренний случайный такт, а значит, зашифрованный текст может быть случайной величиной, зависящей от этого такта. Заметим также, что целочисленный вход алгоритма генерации ключей $\mathcal{G}$ определяет размер результирующих ключей шифрования-расшифровки. Поскольку алгоритм генерации ключей является эффективным, и его временна́я сложность полиномиально зависит от размера исходных данных, входное целое число следует представлять в унарном виде (причины этого требования указаны в разделе 4.4.6.1).

В 1883 году Керхофс (Kerchoffs) написал письмо, регламентирующее разработку криптосистем [198]. Один из пунктов этого письма получил всеобщее признание и стал известен под названием **принципа Керхофса**.

> Знание алгоритма и размера ключа, а также доступ к исходному тексту являются стандартными предположениями современного криптоанализа. Поскольку противник может со временем получить эту информацию, при оценке криптографической стойкости желательно не полагаться на ее секретность.

Объединяя семантическое описание криптосистемы, данное Шенноном, и принцип Керхофса, можно сформулировать критерии качества криптосистемы.

- Алгоритмы $\mathcal{E}$ и $\mathcal{D}$ не содержат секретных компонентов.
- Алгоритм $\mathcal{E}$ равномерно распределяет осмысленные сообщения по всему пространству зашифрованных сообщений, причем это распределение сообщений может обеспечиваться внутренней случайной операцией алгоритма $\mathcal{E}$.
- Если криптографический ключ является правильным, алгоритмы $\mathcal{E}$ и $\mathcal{D}$ *практически* эффективны.
- Если криптографический ключ является неправильным, сложность задачи восстановления исходного текста на основе зашифрованного сообщения зависит исключительно от размера ключа. Размер ключа обычно обозначают буквой s. Таким образом, вычислительная сложность расшифровки оценивается как $p(s)$, где p — некий полином.

Следует заметить, что список желательных свойств криптосистем в данный момент уже устарел. Современные требования будут перечислены при описании конкретных криптосистем.

7.3 Подстановочные шифры

В **подстановочном шифре** (substitution cypher) алгоритм шифрования $\mathcal{E}_k(m)$ представляет собой функцию подстановки, которая заменяет каждое сообщение $m \in \mathcal{M}$ соответствующим текстом $c \in \mathcal{C}$. Функция подстановки зависит от секретного ключа k. Алгоритм дешифровки $\mathcal{D}_k(c)$ представляет собой обратную подстановку. Как правило, подстановку можно задать с помощью отображения $\pi : \mathcal{M} \mapsto \mathcal{C}$, а обратную подстановку — с помощью обратного отображения $\pi^{-1} : \mathcal{C} \mapsto \mathcal{M}$.

7.3.1 Простые подстановочные шифры

Пример 7.1 (Простой подстановочный шифр). Пусть $\mathcal{M} = \mathcal{C} = \mathbb{Z}_{26}$, а буквы алфавита интерпретируются следующим образом: $A = 0, B = 1, \ldots, Z = 25$.

Определим алгоритм шифрования $\mathcal{E}_k(m)$ как следующую перестановку над группой $\mathbb{Z}_{26}$.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 21 & 12 & 25 & 17 & 24 & 23 & 19 & 15 & 22 & 13 & 18 & 3 & 9 \end{pmatrix}$$

$$\begin{pmatrix} 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\ 5 & 10 & 2 & 8 & 16 & 11 & 14 & 7 & 1 & 4 & 20 & 0 & 6 \end{pmatrix}.$$

Тогда соответствующий алгоритм расшифровки $\mathcal{D}_k(c)$ задается следующим образом.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 24 & 21 & 15 & 11 & 22 & 13 & 25 & 20 & 16 & 12 & 14 & 18 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\ 9 & 19 & 7 & 17 & 3 & 10 & 6 & 23 & 0 & 8 & 5 & 4 & 2 \end{pmatrix}.$$

Исходные сообщения

```
proceed meeting as agreed
```

будут преобразованы в следующий зашифрованный текст (пробелы не преобразовываются):

```
cqkzyyr jyyowft vl vtqyyr
```

□

В рассмотренном простом примере пространства сообщений $\mathcal{M}$ и $\mathcal{C}$ совпадали с алфавитом $\mathbb{Z}_{26}$. Иначе говоря, исходное или зашифрованное сообщение представляло собой отдельный символ алфавита. По этой причине строка исходного сообщения `proceedmeetingasagreed` является не одним сообщением, а совокупностью 22 сообщений. Аналогично строка зашифрованного сообщения `cqkzyyrjyyowftvlvtqyyr` также содержит 22 сообщения. Пространство ключей этого шифра имеет размер $26! > 4 \times 10^{26}$. Он намного превышает размер пространства сообщений. Однако на самом деле этот шифр очень слаб: каждый символ в исходном тексте шифруется уникальным символом зашифрованного текста. Эта слабость позволяет расшифровать сообщение с помощью одного из методов **криптоанализа** (cryptoanalysis) — **частотного анализа** (frequency analysis), использующего тот факт, что естественные языки чрезвычайно избыточны (см. раздел 3.8). Стойкость простого подстановочного шифра изучается в разделе 7.5.

В истории известно много примеров простых подстановочных шифров. Простейший и наиболее известный случай — **сдвиговые шифры** (shift cupher). В этих шифрах $\mathcal{K} = \mathcal{M} = \mathcal{C}$. Пусть $N = \#\mathcal{M}$, а отображения шифрования и расшифровки определены следующим образом.

$$\begin{cases} \mathcal{E}_k(m) \leftarrow m + k (\mathrm{mod}\, N), \\ \mathcal{D}_k(c) \leftarrow c - k (\mathrm{mod}\, N), \end{cases} \tag{7.3.1}$$

где $m, c, k \in \mathbb{Z}_N$. Если множество $\mathcal{M}$ — это совокупность прописных букв латинского алфавита (т.е. $\mathcal{M} = \mathbb{Z}_{26}$), сдвиговый шифр называется **шифром Цезаря** (Caesar cipher), поскольку Юлий Цезарь применял его при $k = 3$ (см. раздел 2.2 в книге [93]).

По теореме 6.6 (раздел 6.2.2), если $\gcd(k, N) = 1$, то для каждого $m < N$ число $km (\mathrm{mod}\, N)$ пробегает все пространство сообщений $\mathbb{Z}_N$. Следовательно, при таком числе k и числах m и c, которые меньше N, отображение

$$\begin{cases} \mathcal{E}_k(m) \leftarrow km (\mathrm{mod}\, N), \\ \mathcal{D}_k(c) \leftarrow k^{-1} c (\mathrm{mod}\, N) \end{cases} \tag{7.3.2}$$

является простым подстановочным шифром. Аналогично число

$$k_1 m + k_2 (\mathrm{mod}\, N)$$

также определяет простой подстановочный шифр, известный под названием **аффинного шифра** (affine cipher):

$$\begin{cases} \mathcal{E}_k(m) \leftarrow k_1 m + k_2 (\mathrm{mod}\, N), \\ \mathcal{D}_k(c) \leftarrow k_1^{-1} (c - k_2) (\mathrm{mod}\, N). \end{cases} \tag{7.3.3}$$

Очевидно, что с помощью различных арифметических операций над ключами в пространстве $\mathcal{K}$ и сообщениями в пространстве $\mathcal{M}$ можно создать разнообразные подстановочные шифры. Эти шифры называются **моноалфавитными** (monoalphabetic ciphers): при заданном ключе шифрования каждый элемент в пространстве исходных сообщений заменяется уникальным элементом из пространства зашифрованных сообщений. Следовательно, моноалфавитные шифры чрезвычайно уязвимы для атак на основе частотного анализа.

Однако благодаря своей простоте подобные подстановочные шифры нашли широкое применение в современных алгоритмах шифрования с секретным ключом. Далее будет показано, что простые подстановочные ключи играют основную роль в стандартах DES (раздел 7.6) и AES (раздел 7.7). Общепризнанно, что комбинация простых подстановочных шифров образует алгоритм шифрования, имеющий повышенную стойкость. По этой причине простые шифры широко используются и по сей день. Кроме того, простые подстановочные шифры часто применяются в криптографических протоколах. Демонстрация этого факта содержится в разделе 7.5 и во многих других частях книги.

7.3.2 Полиалфавитные шифры

Подстановочный шифр называется **полиалфавитным** (polyalphabetic cipher), если элемент исходного сообщения в пространстве $\mathcal{P}$ можно заменить несколькими элементами пространства зашифрованных сообщений $\mathcal{C}$.

Наиболее известным из полиалфавитных шифров является **шифр Виженера** (Vigenere cipher). Этот шифр основан для подстановке строк: ключ представляет собой строку, состоящую из большого количества символов. Пусть m — длина ключа. Тогда строка исходного текста разбивается на фрагменты, состоящие из m символов, причем последний фрагмент может содержать меньшее количество символов. Алгоритм шифрования поочередно применяет сдвиговый шифр к строке ключа и каждой строке исходного текста, при необходимости повторяя применение ключа снова. Расшифровка выполняется аналогично.

Пример 7.2 (Шифр Виженера). Пусть ключевой строкой является слово `gold`. Используя правило шифрования $A = 0, B = 1, \ldots, Z = 25$, преобразуем ключевое слово в кортеж $(6, 14, 11, 3)$. Применение шифра Виженера к строке исходного текста

`proceed meeting as agreed`

сводится к выполнению посимвольного сложения по модулю 26.

$$
\begin{array}{cccccccccccc}
15 & 17 & 14 & 2 & 4 & 4 & 3 & 12 & 4 & 4 & 19 \\
6 & 14 & 11 & 3 & 6 & 14 & 11 & 3 & 6 & 14 & 11 \\
\hline
21 & 5 & 25 & 5 & 10 & 18 & 14 & 15 & 10 & 18 & 4
\end{array}
$$

$$
\begin{array}{cccccccccccc}
8 & 13 & 6 & 0 & 18 & 0 & 6 & 17 & 4 & 4 & 3 \\
3 & 6 & 14 & 11 & 3 & 6 & 14 & 11 & 3 & 6 & 14 \\
\hline
11 & 19 & 20 & 11 & 21 & 6 & 20 & 2 & 7 & 10 & 17
\end{array}
$$

Следовательно, зашифрованная строка будет выглядеть так.

`vfzfkso pkseltu lv guchkr` □

Другими примерами известных полиалфавитных шифров является **книжный шифр** (book cipher), называемый также **шифром Биле** (Beale cipher), в котором ключевой строкой является условленная строка книги, и **шифр Хилла** (Hill cipher). Подробное описание этих подстановочных шифров содержится в книгах [93] и [284].

7.3.3 Шифр Вернама и одноразовый блокнот

Шифр Вернама (Vernam cipher) — одна из простейших криптосистем. Предположим, что сообщение и ключ представляют собой строки из n двоичных разрядов:

$$m = b_1 b_2 \ldots b_n \in \{0,1\}^n,$$
$$k = k_1 k_2 \ldots k_n \in_U \{0,1\}^n.$$

(Обратите внимание на символ $\in_U$: число k извлекается из равномерно распределенной генеральной совокупности.) Шифрование выполняется бит за битом, а зашифрованная строка $c = c_1 c_2 \ldots c_n$ образуется путем применения побитовой операции XOR ("исключительное или") к каждому биту сообщения и соответствующему биту ключа:

$$c_i = b_i \oplus k_i,$$

где $1 \leqslant i \leqslant n$, а операция $\oplus$ определена следующим образом:

$\oplus$	0	1
0	0	1
1	1	0

Расшифровка выполняется аналогично шифрованию, поскольку операция $\oplus$ является сложением по модулю 2, а значит, вычитание идентично сложению.

Если $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^*$, шифр Вернама представляет собой частный случай подстановочных шифров. Если ключевая строка используется только один раз, шифр Вернама удовлетворяет двум требованиям повышенной стойкости, предъявляемым к подстановочным шифрам. В разделе 7.5 мы покажем, что секретность, обеспечиваемая однократным применением шифра Вернама, является секретностью в теоретико-информационном смысле, т.е. безусловной. Для того чтобы убедиться в этом, достаточно проверить, что строка зашифрованного текста c не предоставляет перехватчику никакой информации о строке исходного сообщения m, поскольку если ключ k равен $c \oplus m$, то строка c может порождаться любой строкой m (бит за битом).

Одноразовый шифр Вернама называется также **шифром одноразового блокнота** (one-time pad cipher). В принципе, если некий шифр удовлетворяет двум требованиям повышенной стойкости, предъявляемым к подстановочным шифрам (см. раздел 7.5), то любой подстановочный шифр является шифром одноразового блокнота. Однако шифрами одноразового блокнота принято называть только шифры, использующие побитовую операцию XOR.

В сравнении с другими подстановочными шифрами (например, сдвиговым шифром, использующим сложение по модулю 26), побитовую операцию XOR

(т.е. сложение по модулю 2) можно легко реализовать с помощью электронной схемы. По этой причине побитовая операция XOR широко используется при разработке современных алгоритмов шифрования с секретным ключом. Кроме того, она применяется в двух важных современных шифрах: DES (раздел 7.6) и AES (раздел 7.7).

Шифрование с помощью одноразового блокнота широко используется в криптографических протоколах. Один такой протокол мы рассмотрим в разделе 7.5.1.

7.4 Перестановочные шифры

Перестановочный шифр (transposition, or permutation cipher) преобразует сообщение, переставляя его элементы и не изменяя их. Перестановочные шифры в дополнение к подстановочным образуют важную совокупность классических шифров, широко применяемых при создании современных блочных шифров.

Будем считать, что элементы исходного сообщения являются буквами алфавита $\mathbb{Z}_{26}$, b — фиксированное положительное целое число, представляющее собой размер блока сообщения, $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^b$ и $\mathcal{K}$ — множество всевозможных перестановок чисел $(1, 2, \ldots, b)$.

В этом случае перестановка $\pi = (\pi(1), \pi(2), \ldots, \pi(b))$ — ключ, поскольку $\pi \in \mathcal{K}$. Для блока исходного сообщения $(x_1, x_2, \ldots, x_b) \in \mathcal{P}$ алгоритм шифрования с помощью перестановочного шифра выглядит следующим образом.

$$e_\pi(x_1, x_2, \ldots, x_b) = \left(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(b)}\right).$$

Обозначим через π^{-1} отображение, обратное к отображению π, т.е. $\pi^{-1}(\pi(i)) = i$ при $i = 1, 2, \ldots, b$. Тогда соответствующий алгоритм расшифровки с помощью перестановочного шифра принимает следующий вид.

$$d_\pi = (y_1, y_2, \ldots, y_b) = \left(y_{\pi^{-1}(1)}, y_{\pi^{-1}(2)}, \ldots, y_{\pi^{-1}(b)}\right).$$

Если длина сообщения больше, чем размер блока b, сообщение разбивается на несколько блоков, к которым применяется одна и та же процедура.

Поскольку размер блока равен b, существует $b!$ разных ключей. Следовательно, блок исходного текста может быть переставлен в зашифрованном тексте $b!$ способами. Однако, поскольку буквы исходного сообщения не изменяются, перестановочные шифры также чрезвычайно уязвимы для атак на основе частотного анализа.

Пример 7.3 (Перестановочный шифр). Пусть $b = 4$ и

$$\pi = (\pi(1), \pi(2), \pi(3), \pi(4)) = (2, 4, 1, 3).$$

Следовательно, исходное сообщение

proceed meeting as agreed

сначала разбивается на шесть блоков по четыре буквы в каждом:

proc eedm eeti ngas agre ed

Затем каждый блок преобразовывается с помощью перестановочного шифра в следующий текст:

rcpoemedeietgsnagearde

Заметим, что последний блок исходного текста (ed) на самом деле дополняется пробелами ed␣␣, зашифровывается как d␣e␣, а затем пробелы удаляются из блока.

Ключ расшифровки имеет следующий вид.

$$\pi^{-1} = (\pi(1)^{-1}, \pi(2)^{-1}, \pi(3)^{-1}, \pi(4)^{-1}) = (2^{-1}, 4^{-1}, 1^{-1}, 3^{-1}).$$

Тот факт, что в окончательном виде сокращенный зашифрованный блок de содержит только две буквы, означает, что соответствующий блок исходного текста не содержит ни одной буквы на позициях чисел 3^{-1} и 4^{-1}. Следовательно, перед тем, как применять процедуру расшифровки, пробелы следует заново вставить в сокращенный зашифрованный текст в те же самые позиции, чтобы восстановить блок, имеющий вид d␣e␣. □

Заметим, что в ситуациях, когда последний блок исходного сообщения имеет усеченный вид (как в примере 7.3), не следует оставлять в зашифрованном тексте символы-заполнители, такие как ␣, поскольку эти символы-заполнители могут раскрыть информацию об используемом ключе.

7.5 Классические шифры: полезность и стойкость

Прежде всего отметим, что два основных принципа классических шифров — подстановка и перестановка — продолжают лежать в основе современных симметричных алгоритмов шифрования. В разделах 7.6 и 7.7 будет показано, что они используются в двух современных стандартах шифрования: DES и AES.

Рассмотрим шифры, основанные на символьных перестановках. Поскольку пространство исходных сообщений совпадает с алфавитом, каждое сообщение представляет собой символ, а шифрование сводится к последовательной замене символов исходного сообщения символами зашифрованного сообщения в соответствии с секретным ключом. Если для шифрования длинных символьных строк используется фиксированный ключ, то одному и тому же символу исходного сообщения соответствует один и тот же символ зашифрованного сообщения.

Хорошо известно, что буквы естественных языков имеют устойчивую частоту (см. раздел 3.8). Знание распределения частот употребления букв естественного языка образует фундамент криптоанализа — метода, нацеленного на извлечение информации об исходном тексте из зашифрованных сообщений. Это явление было продемонстрировано в примере 7.1. В зашифрованном сообщении, рассмотренном в этом примере, относительно часто встречается буква y. Следовательно, можно предположить, что некая фиксированная буква должна встречаться в соответствующем исходном сообщении с той же самой частотой (на самом деле это — буква e, которая очень часто встречается в англоязычных текстах). Простые подстановочные шифры не в состоянии скрыть информацию о естественном языке, на котором написано исходное сообщение. Подробное описание методов криптоанализа, основанных на исследовании частот употребления букв, можно найти в любом стандартном учебнике по криптографии, например, в книгах [93] и [198].

Полиалфавитные и перестановочные шифры являются более стойкими, чем простые подстановочные шифры. Однако, если ключ — короткий, а сообщение — длинное, методы криптоанализа позволяют взломать такой шифр.

Классические шифры, даже простые подстановочные шифры, могут оказаться *очень стойкими*, если криптографические ключи подчиняются определенным условиям. По этой причине простые подстановочные шифры широко применяются в криптографических системах и протоколах.

7.5.1 Полезность классических шифров

Рассмотрим пример сдвигового шифра (простейший пример подстановочного шифра), который применяется в криптографическом протоколе. Это позволит нам сформулировать два важных условия, обеспечивающих стойкость классических шифров.

Допустим, что над группой $\mathbb{Z}_n$ задана функция $f(x)$, обладающая следующими двумя свойствами.

Однонаправленность. При заданном элементе $x \in \mathbb{Z}_n$ функция $f(x)$ допускает эффективное вычисление (это понятие рассматривается в разделе 4.4.6). В то же время для почти всех элементов $y \in \mathbb{Z}_n$ и для любого эффективного алгоритма A вероятность $P[x \leftarrow A(y) \wedge f(x) = y]$ пренебрежимо мала по сравнению с величиной y (понятие пренебрежимо малой величины описано в разделе 4.6).

Гомоморфизм. Для всех $x_1, x_2 \in \mathbb{Z}_n$ выполняется условие $f(x_1 + x_2) = f(x_1) \cdot f(x_2)$.

Существует много функций, удовлетворяющих такому условию. Используя их, можно построить протокол "доказательства с нулевым разглашением", позволяющий **доказывающей стороне** (например, Алисе) доказать **верификатору** (например, Бобу), что Алиса знает прообраз элемента $f(z)$ (где $z < n$), не сообщая

самого прообраза. Этого можно достичь с помощью простого протокола, использующего сдвиговый шифр.

Протокол 7.1. Протокол доказательства с нулевым разглашением на основе сдвигового шифра

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

1. $f()$: однонаправленная и гомоморфная функция над группой $\mathbb{Z}_n$;
2. $X = f(z)$ для некоторого $z \in \mathbb{Z}_n$.

ВХОДНЫЕ ДАННЫЕ АЛИСЫ:

$z < n$ (* Закрытые входные данные доказывающей стороны *)

ЗАКЛЮЧЕНИЕ БОБА:

Алиса знает элемент $z \in \mathbb{Z}_n$, удовлетворяющий условию $X = f(z)$.

Следующие шаги выполняются m раз.

1. Алиса извлекает число $k \in_U \mathbb{Z}_n$, вычисляет число $\mathsf{Commit} \leftarrow f(k)$ и посылает его Бобу.
2. Боб извлекает число $\mathsf{Challenge} \in_U \{0, 1\}$ и посылает его Алисе.
3. Алиса вычисляет число

$$\mathsf{Response} \leftarrow \begin{cases} k, & \text{если } \mathsf{Challenge} = 0, \\ k + z (\mathrm{mod}\, n), & \text{если } \mathsf{Challenge} = 1 \end{cases}$$

   и посылает его Бобу.

   (* Если $\mathsf{Challenge} = 1$, число Response является результатом шифрования числа z с помощью сдвигового шифра и одноразового ключа k (см. раздел 7.3.1). *)

4. Боб проверяет условие

$$f(\mathsf{Response}) \stackrel{?}{=} \begin{cases} \mathsf{Commit}, & \text{если } \mathsf{Challenge} = 0, \\ \mathsf{Commit} \cdot X, & \text{если } \mathsf{Challenge} = 1. \end{cases}$$

   Если на каком-либо шаге возникает ошибка, Боб отвергает доказательство и прекращает выполнение протокола.

Боб принимает доказательство.

Протокол 7.1 весьма полезен. В приложениях величина $X = f(z)$ может выступать в роли криптографического удостоверения Алисы, позволяющего ее идентифицировать. Это удостоверение может использовать только Алиса, поскольку лишь она знает, как с ним обращаться, учитывая прообраз z. Этот протокол демон-

стрирует, как Алиса должна применять свое удостоверение без предоставления Бобу какой-либо информации о прообразе z.

В главе 18 мы будем интенсивно применять этот протокол и его модификации при описании протокола доказательства с нулевым разглашением.

7.5.2 Стойкость классических шифров

Какой уровень конфиденциальности, позволяющий Алисе скрывать секретную информацию о прообразе z, обеспечивает протокол 7.1? Мы утверждаем, что этот протокол обеспечивает *высший* уровень секретности. Иначе говоря, после запуска протокола Боб *не получает* абсолютно никакой новой информации об элементе $z \in \mathbb{Z}_n$, кроме той, которую он мог получить, анализируя величину $f(z)$ (общие входные данные предоставляют только *априорную* информацию).

Следует заметить, что шифрование с помощью сдвигового шифра

$$\mathsf{Response} = z + k (\mathrm{mod}\, n)$$

образует перестановку над группой $\mathbb{Z}_n$. Если $k \in_U \mathbb{Z}_n = \mathcal{K} = \mathcal{M}$, перестановка переводит это число в элемент $\mathsf{Response} \in_U \mathbb{Z}_n$, поскольку она отображает равномерное распределение в равномерное распределение. Это означает, что заданный зашифрованный текст **Response** мог быть создан с помощью любого ключа из группы $\mathbb{Z}_n$ (вероятностное пространство состоит из пространства ключей и пространства сообщений). Иначе говоря, в сообщении **Response** с одинаковой вероятностью мог быть зашифрован *любой* элемент $x \in \mathbb{Z}_n$. Итак, исходный текст z не зависит от зашифрованного текста Response, т.е. зашифрованный текст не предоставляет никакой информации об исходном тексте.

Если шифр обеспечивает независимость между распределениями исходных и зашифрованных текстов, говорят, что шифр является стойким в **информационно-теоретическом смысле** (information-theoretically secure sense). В отличие от стойкости в смысле вычислительной сложности (complex-theoretic sence), установленной в главе 4, стойкость в теоретико-информационном смысле является *безусловной* и не поддается ни одному из методов криптоанализа. Относительно протокола 7.1 это означает, что запуск протокола не предоставляет Бобу никакой информации о закрытых входных данных Алисы z, кроме того, что этот ввод является секретным.

Понятие теоретико-информационной криптографической безопасности было введено Шенноном [264]. Он сформулировал два условия стойкости классических шифров.

Условия безопасного применения классических шифров

1. $\#\mathcal{K} \geqslant \#\mathcal{M}$.
2. $k \in_U \mathcal{K}$ и используется для шифрования только один раз.

Итак, если строка длины ℓ зашифрована с помощью классического шифра (либо простой подстановки, либо полиалфавитного шифра, либо шифра Вернама), то для надежности шифрования длина ключа должна быть не меньше ℓ, причем ключевую строку следует использовать только один раз. Несмотря на то что эти требования являются не очень практичными (в реальных приложениях шифруются огромные объемы информации), они вполне пригодны для шифрования малых объемов данных, таких как одноразовые случайные числа (nonces) (см. раздел 2.6.4) или сеансовые ключи (см. раздел 2.5). Протокол 7.1 представляет собой именно такой случай.

В остальной части книги будут приведены многочисленные криптографические системы и протоколы, использующие подстановочные шифры, такие как сдвиговые (например, протокол 7.1), шифр Цезаря (7.3.2), аффинные шифры (7.3.3) и подстановочные шифры, а также общий вид перестановок (как в примере 7.1). Многие из этих приложений удовлетворяют двум условиям безопасного применения классических шифров.

7.6 Алгоритм Data Encryption Standard (DES)

Без сомнения, первым и наиболее значительным современным симметричным алгоритмом шифрования является алгоритм Data Encryption Standard (DES) [211]. Этот алгоритм был опубликован Национальным бюро стандартов США в январе 1977 года и был предназначен для шифрования открытых данных (т.е. информации, не имеющей отношения к национальной безопасности). Алгоритм DES получил широкое международное признание и стал интенсивно применяться для безопасного обмена данными в банковских системах. Сначала этот стандарт был принят на пять лет, однако впоследствии его применение было одобрено еще на три пятилетних периода.

7.6.1 Описание алгоритма DES

Алгоритм DES является **блочным шифром** (block cipher), предназначенным для шифрования сообщений, разделенных на блоки фиксированной длины. Каждый блок рассматривается как отдельное сообщение, принадлежащее пространствам $\mathcal{M}$ или $\mathcal{C}$. В алгоритме DES выполняется условие $\mathcal{M} = \mathcal{C} = \{0,1\}^{64}$ и $\mathcal{K} = \{0,1\}^{56}$. В частности, алгоритмы шифрования и расшифровки получают на вход 64-битовый исходный или зашифрованный текст и 56-битовый ключ, а результатом работы этих алгоритмов являются 64-битовые зашифрованные или исходные тексты.

Операции алгоритма DES можно описать с помощью трех шагов.

1. К исходному блоку применяется фиксированная начальная перестановка IP, определенная следующей формулой.

$$(L_0, R_0) \leftarrow \text{IP}(\text{Исходный блок}) \tag{7.6.1}$$

Здесь L_0 и R_0 называются левой и правой половинами блока, каждая из которых представляет собой 32-битовый блок. Заметим, что перестановка IP является фиксированной (т.е. не зависит от исходного блока) и открытой функцией. Следовательно, начальная перестановка не имеет криптографического значения.

2. Выполняются 16 **раундов** (rounds), состоящих из следующих операций.

$$L_i \leftarrow R_{i-1}, \tag{7.6.2}$$

$$R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, k_i), i = 1, 2, \ldots, 16. \tag{7.6.3}$$

Здесь число k_i — ключ раунда (round key), состоящий из 48-битовой подстроки 56-битового исходного ключа, f — функция S-блока (S-box function), представляющая собой подстановочный шифр (см. раздел 7.3). В названии функции f буква S является первой буквой слова *substitution* (подстановка). Операции (7.6.2) и (7.6.3) предназначены для перестановки двух половин блока, т.е. левый блок, поступающий на вход очередного раунда, был правым блоком на выходе предыдущего раунда. Операция перестановки является простым перестановочным шифром (см. раздел 7.4), предназначенным для обеспечения значительной "диффузии сообщения". По существу, эта особенность алгоритма является свойством перемешивания (7.1.1), описанным Шенноном. В дальнейшем будет показано, что этот этап алгоритма DES является комбинацией подстановочного и перестановочного шифра.

3. Результат 16-го раунда (L_{16}, R_{16}) является входными данными для перестановки, обратной к перестановке IP. Результат этого этапа представляет собой результат работы всего алгоритма DES. Заключительный этап алгоритма можно записать в следующем виде.

$$\text{Результирующий блок} \leftarrow \text{IP}^{-1}(R_{16}, L_{16}). \tag{7.6.4}$$

Обратите особое внимание на входные данные обратной перестановки IP^{-1}: перед тем, как поступить на вход обратной перестановки IP^{-1}, две половины блока, являющиеся результатом 16-го раунда, меняются местами еще раз.

Эти три этапа выполняются как при шифровании, так и расшифровке. Единственное отличие заключается в том, что в одном алгоритме используются ключи раундов $k_1, k_2, \ldots, k_{16}$, а в другом — $k_{16}, k_{15}, \ldots, k_1$. Этот способ применения ключей называется "расписанием ключей". Оно обозначается следующим образом.

$$\left(k'_1, k'_2, \ldots, k'_{16}\right) = \left(k_{16}, k_{15}, \ldots, k_1\right). \tag{7.6.5}$$

Пример 7.4. Допустим, что исходное сообщение m преобразовано в зашифрованное сообщение c с помощью ключа шифрования k. Применим алгоритм DES для расшифровки и проверим, получится ли исходное сообщение m из зашифрованного сообщения c при расшифровывании с помощью ключа k.

Алгоритм расшифровки начинает работу с ввода зашифрованного текста c в качестве "исходного блока". По формуле (7.6.1) получаем:

$$\left(L'_0, R'_0\right) \leftarrow \mathrm{IP}(c).$$

Поскольку сообщение c, по существу, является "результирующим блоком", полученным на заключительном этапе алгоритма шифрования, в соответствии с формулой (7.6.4) имеем

$$\left(L'_0, R'_0\right) = (R_{16}, L_{16}). \tag{7.6.6}$$

В первом раунде, применяя операции (7.6.2), (7.6.3) и (7.6.6), получаем

$$\begin{aligned} L'_1 &\leftarrow R'_0 = L_{16}, \\ R'_1 &\leftarrow L'_0 \oplus f\left(R'_0, k'_1\right) = R_{16} \oplus f(L_{16}, k'_1). \end{aligned}$$

По формуле (7.6.2) в правых частях этих двух операций присваивания блок L_{16} должен быть заменен блоком R_{15}, а в соответствии с формулой (7.6.3) блок R_{16} следует заменить блоком $L_{15} \oplus f(R_{15}, k_{16})$. Кроме того, из расписания ключей (7.6.5) следует, что $k'_1 = k_{16}$. Итак, две операции присваивания, описанные выше, принимают следующий вид.

$$\begin{aligned} L'_1 &\leftarrow R_{15}, \\ R'_1 &\leftarrow \left[L_{15} \oplus f\left(R_{15}, k_{16}\right)\right] \oplus f(R_{15}, k_{16}) = L_{15}. \end{aligned}$$

Следовательно, после первого раунда получаем

$$\left(L'_1, R'_1\right) = \left(R_{15}, L_{15}\right).$$

Таким образом, в начале второго раунда две половины блока имеют вид (R_{15}, L_{15}).

Выполняя остальные 15 раундов, приходим к следующим результатам.

$$\left(L'_2, R'_2\right) = \left(R_{14}, L_{14}\right), \ldots, \left(L'_{16}, R'_{16}\right) = \left(R_0, L_0\right).$$

Последние две половины блока (L'_{16}, R'_{16}), полученные в результате 16-го раунда, меняются местами: $(R'_{16}, L'_{16}) = (L_0, R_0)$ и поступают на вход функции IP^{-1}, выполняющей перестановку, обратную по отношению к начальной (учтите, что по формуле (7.6.4) следует выполнить дополнительную перестановку подблоков). В результате начальная перестановка (7.6.1) компенсируется, и мы получаем исходный блок m. □

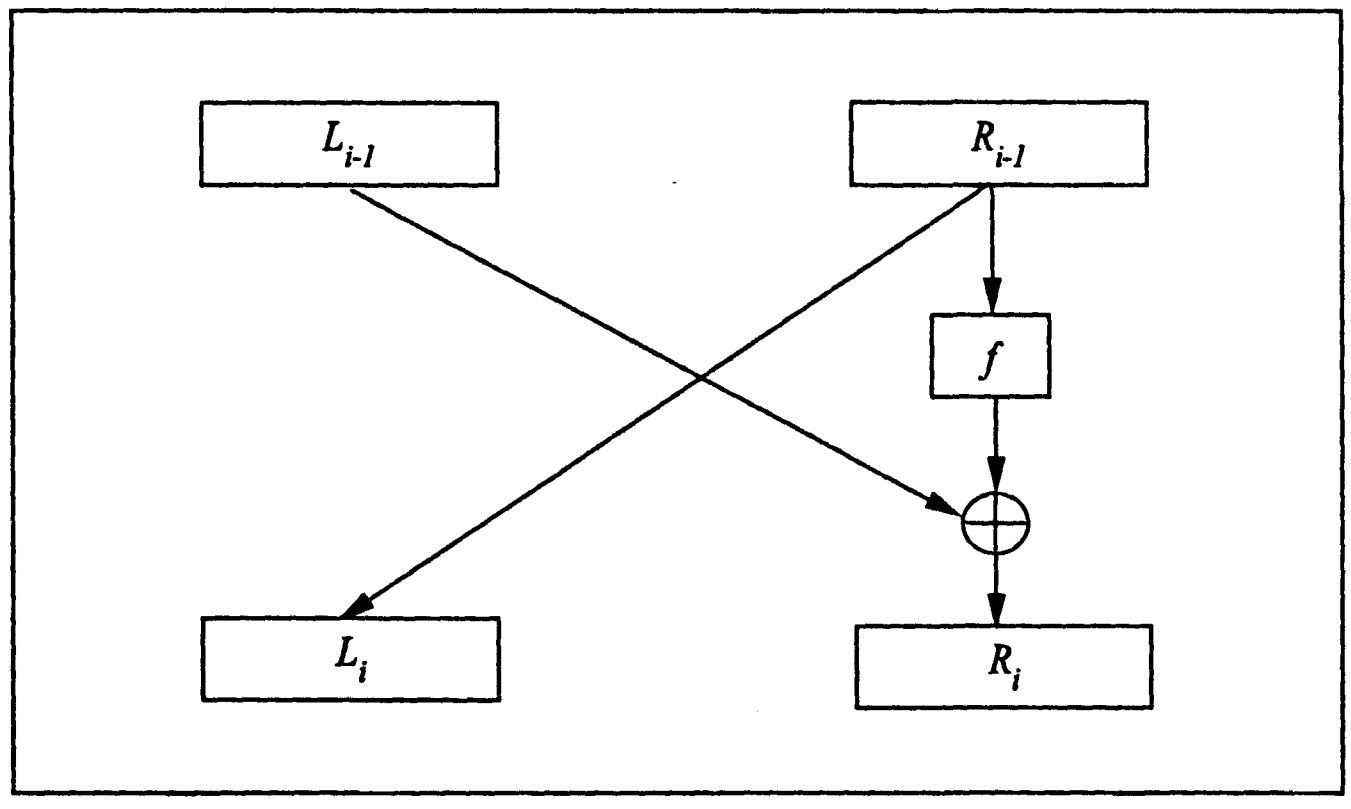


Рис. 7.2. Шифр Файстеля (один раунд)

Мы показали, что алгоритмы шифрования и расшифровки по стандарту DES удовлетворяют условию (7.2.1) при всех $m \in \mathcal{M}$ и $k \in \mathcal{K}$. Очевидно, что работа этих алгоритмов не зависит от внутреннего устройства функции подстановки f и расписания ключей.

Итерации алгоритма DES, использующие операции (7.6.2) и (7.6.3) для перестановки блоков, называются **шифром Файстеля** (Feistel cipher) [107]. На рис. 7.2 показана структура перестановок, выполняемых в течение одного раунда шифра Файстеля. Как указывалось ранее, операция перестановки обеспечивает большую диффузию данных. Шифр Файстеля широко применяется в криптографии с открытым ключом. По существу, структура OAEP (Optimal Asymmetric Encryption Padding) является двухраундовым шифром Фейстеля. Он рассматривается в разделе 15.2.

7.6.2 Основа алгоритма DES — случайное и нелинейное распределение сообщений

Сущность алгоритма DES заключена в функции S-блока f. Именно она обеспечивает случайное и нелинейное распределение исходных сообщений по пространству зашифрованных сообщений.

В i-м раунде функция $f(R_{i-1}, k_i)$ выполняет две следующие вспомогательные операции.

1. Складывает ключ раунда k_i с половиной блока R_{i-1}, применяя побитовую операцию XOR. Это обеспечивает случайность распределения сообщений.
2. Подставляет результат первой операции в фиксированную перестановку, состоящую из восьми “блоков постановки” (S-блоков). Каждый S-блок пред-

ставляет собой нелинейную функцию подстановки. Это гарантирует нелинейность распределения сообщений.

Нелинейность S-блоков очень важна для стойкости алгоритма DES. Заметим, что в общем случае подстановочный шифр (см. пример 7.1 со случайным ключом) является нелинейным, в то время как сдвиговый и аффинный шифры являются линейными. Эта линейность не только резко уменьшает размер пространства ключей, но и делает результирующий зашифрованный текст уязвимым для методов **дифференциального криптоанализа** (DC) [33]. Дифференциальный криптоанализ атакует шифр, используя линейную разность между двумя исходными сообщениями и между двумя зашифрованными сообщениями. Рассмотрим в качестве примера атаку на аффинный шифр (7.3.3). Допустим, что Злоумышленник (атакующий) каким-то образом узнал разность $m - m'$, но не знает ни m, ни m'. Если соответствующие зашифрованные тексты имеют вид $c = k_1 m + k_2 (\mathrm{mod}\, N)$, $c' = k_1 m' + k_2 (\mathrm{mod}\, N)$, Злоумышленник может вычислить величину

$$k_1 = (c - c')/(m - m')(\mathrm{mod}\, N).$$

Зная число k_1, намного легче найти величину k_2: ее можно вычислить, если Злоумышленнику известна хотя бы одна пара "открытый текст – зашифрованный текст". Как было выяснено в 1990 году, дифференциальный криптоанализ оказался очень мощным оружием против многих известных блочных шифров. Однако он не справился с алгоритмом DES. Со своей стороны, разработчики алгоритма DES предвидели возможность атаки с помощью методов дифференциального криптоанализа за 15 лет до этого [81] и защитили его, обеспечив нелинейность S-блоков.

Интересной особенностью алгоритма DES (а фактически — шифра Файстеля) является тот факт, что функция $f(R_{i-1}, k_i)$ не обязана быть обратимой. Как показано в примере 7.4, шифрование и расшифровка могут выполняться с помощью произвольной функции $f(R_{i-1}, k_i)$. Это свойство открывает перспективы для аппаратной реализации алгоритма DES.

Мы пропускаем детали внутреннего устройства S-блоков, расписания ключей и функции, обеспечивающей начальную перестановку. Эти детали выходят за рамки книги. Читатели, интересующиеся этими вопросами, найдут их в книге [93

7.6.3 Стойкость алгоритма DES

Споры о стойкости алгоритма DES начались вскоре после того, как он был принят в качестве стандарта. Подробное описание дискуссий и исторические справки можно найти во многих работах по криптографии [279, 284, 198]. Позднее стало ясно, что все споры сводятся к одному пункту: алгоритм DES имеет относительно короткий ключ. Это свойство считалось наиболее слабым местом алгоритма. Атаки, направленные на это слабое место, основывались на полном перебор

возможных ключей с использованием известной пары "исходное сообщение – зашифрованное сообщение". Такая атака называется **лобовой** (brute-force) или **атакой с помощью полного перебора ключей** (exhaustive key search attack).

Однако лобовую атаку нельзя считать реальной, потому что разработчики предусмотрели эту возможность. К тому же, лобовая атака — теоретически единственное средство взлома алгоритма DES. Таким образом, на уровне вычислительных технологий 1970-х годов алгоритм DES — вполне удачный шифр.

Единственный способ преодолеть ограничение, связанное с малой длиной ключа, — многократно применять алгоритм DES с разными ключами. Одна из таких модификаций получила название "шифрование — расшифровка — шифрование", или **тройной алгоритм DES** (triple DES scheme) [290]. Шифрование по этой схеме выполняется следующим образом:

$$c \leftarrow \mathcal{E}_{k_1}\left(\mathcal{D}_{k_2}\left(\mathcal{E}_{k_1}\left(m\right)\right)\right),$$

а расшифровка —

$$m \leftarrow \mathcal{D}_{k_1}\left(\mathcal{E}_{k_2}\left(\mathcal{D}_{k_1}\left(c\right)\right)\right).$$

Помимо увеличения пространства ключей, при условии, что $k_1 = k_2$, эта схема по простоте сравнима с однократным алгоритмом DES. В тройном алгоритме DES можно использовать три разных ключа, но в этом случае его сложность возрастает.

Слабость алгоритма DES, связанная с небольшой длиной ключа, стала очевидной в 1990-х годах. В 1993 году Винер (Wiener) продемонстрировал, что стоимость создания специализированной машины для поиска ключей алгоритма DES приблизительно равна 1 000 000 долл. При известной паре "исходный текст – зашифрованный текст" эта машина может найти ключ за 3,5 часа [299]. Кроме того, 15 июля 1998 года коалиция компаний Cryptography Research, Advanced Wireless Technology и Electronic Frontier Foundation сообщила об успешной атаке на алгоритм DES. Эти компании создали поисковую машину под названием DES Cracker (известную также под именем Deep Cracker) стоимостью 250 000 долл. и успешно нашли ключ алгоритма DES Challenge за 56 часов [110]. Этот результат продемонстрировал, что с учетом уровня вычислительных технологий конца 1990-х годов 56-битовый ключ слишком короток для обеспечения стойкости шифра с секретным ключом.

7.7 Алгоритм Advanced Encryption Standard (AES)

В январе 1997 года Национальный институт стандартов и технологий США (National Institute of Standards and Technology — NIST) объявил о начале работ над новым алгоритмом шифрования с симметричным блочным ключом, призванным

заменить алгоритм DES в качестве нового стандарта. Новый алгоритм получил название AES (Advanced Encryption Standard). В отличие от засекреченного процесса создания алгоритма DES, 12 сентября 1997 года было сделано открытое заявление о создании алгоритма AES. В заявлении указывалось, что алгоритм AES следует считать незасекреченным, открытым алгоритмом шифрования с симметричным ключом. Он использует (как минимум) блоки, длина которых равна 128 бит, а размеры ключей равны 128, 192 и 256 бит. Стойкость алгоритма AES сравнима со стойкостью тройного алгоритма DES, однако его эффективность намного выше. Кроме того, алгоритм AES предназначался для свободного распространения по всему миру.

20 августа 1998 года институт NIST объявил о создании группы, состоящей из пятнадцати алгоритмов, претендующих на роль нового стандарта AES. Эти алгоритмы были представлены членами криптографического сообщества со всего мира. Институт призвал рецензентов присылать открытые комментарии к этим алгоритмам (период первоначальных комментариев рассматривался как Первый раунд). Этот раунд был завершен 15 апреля 1999 года. На основе проведенного анализа и полученных комментариев институт NIST отобрал пять алгоритмов. Финалистами этого состязания стали алгоритмы MARS [62], RC6 [247], Rijndael [86], Serpent [15] и Twofish [255]. В ходе второго раунда эти алгоритмы подверглись углубленному исследованию. В частности, к ним применялись методы криптоанализа, изучались вопросы интеллектуальной собственности, перспективы внедрения и отзывы специалистов. После завершения второго раунда 15 мая 2000 года институт NIST изучил всю доступную информацию, касающуюся пяти алгоритмов, претендующих на роль нового стандарта. 2 октября 2000 года институт NIST объявил, что победителем конкурса стал алгоритм Rijndael.

Этот алгоритм был разработан двумя бельгийскими криптографами: Даменом (Daemen) и Рийменом (Rijmen).

7.7.1 Обзор алгоритма Rijndael

Алгоритм Rijndael является блочным шифром с переменным размером блоков и переменной длиной ключа. Размеры ключа и блоков могут независимо друг от друга принимать значения 128, 192 и 256 бит. Для простоты мы рассмотрим лишь минимальный вариант, в котором размеры ключа и блока равны 128 бит. При этом основные принципы алгоритма Rijndael будут описаны полностью.

Итак, 128-битовый блок сообщения (исходного или зашифрованного) разбивается на сегменты по 16 байт (один байт состоит из восьми бит, т.е. $128 = 16 \times 8$).

$$\text{InputBlock} = m_0, m_1, \ldots, m_{15}.$$

Аналогичная операция выполняется над блоком ключа.

$$\text{InputKey} = k_0, k_1, \ldots, k_{15}.$$

Для внутреннего представления данных используется матрица 4×4.

$$\text{InputBlock} = \begin{pmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{pmatrix},$$

$$\text{InputKey} = \begin{pmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{pmatrix}.$$

Как алгоритм DES (и большинство современных симметричных блочных шифров), алгоритм Rijndael состоит из большого количества повторяющихся преобразований — раундов. В минимальном варианте, когда размеры блока и ключа равны 128 бит, количество раундов равно 10. Для более крупных сообщений и ключей количество раундов может возрастать, как показано в работе [219].

Преобразование, выполняемое внутри раунда, обозначается как

$$\text{Round}(State, RoundKey).$$

Здесь переменная *State* представляет собой матрицу, содержащую сообщение раунда, и считается как входом, так и результатом раунда, а переменная *RoundKey* представляет собой матрицу, содержащую ключ раунда, и создается на основе входного ключа с помощью расписания ключей. Преобразования внутри раунда должно изменять элементы матрицы *State* (т.е. изменять состояние). При шифровании (соответственно при расшифровке) матрица *State*, поступающая на вход первого раунда, совпадает с матрицей *InputBlock* и содержит исходный текст (соответственно зашифрованный текст). В свою очередь, матрица *State*, возникающая в результате заключительного раунда, содержит зашифрованный текст (соответственно исходный текст).

Преобразования, выполняемые в ходе промежуточных раундов, разбиваются на четыре операции, которые являются внутренними функциями (они будут описаны позднее).

```
Round(State, RoundKey){
  SubBytes(State),
  ShiftRows(State),
  MixColumns(State),
  AddRoundKey(State, RoundKey);
  }
```

Последний раунд обозначается как

$$\mathsf{FinalRound}(State, RoundKey).$$

Он немного отличается от остальных и совпадает с раундом $\mathsf{Round}(State, RoundKey)$ с удаленной матрицей MixColumns. Этот раунд напоминает заключительный раунд алгоритма DES, в котором выполнялась дополнительная перестановка половин результирующего блока.

Для обеспечения расшифровки преобразования раунда являются обратимыми. Обратные преобразования раунда обозначаются как

$$\mathsf{Round}^{-1}(State, RoundKey)$$

и

$$\mathsf{FinalRound}^{-1}(State, RoundKey).$$

Далее будет показано, что все четыре внутренние функции являются обратимыми.

7.7.2 Внутренние функции алгоритма Rijndael

Опишем четыре внутренние функции алгоритма Rijndael. Рассмотрим лишь функции, предназначенные для шифрования. Поскольку каждая из них является обратимой, расшифровка в алгоритме Rijndael просто сводится к применению обратных функций в обратном порядке.

Внутренние функции шифра Rijndael определены на конечном поле. Это поле состоит из всех полиномов по модулю неприводимого полинома

$$f(x) = x^8 + x^4 + x^3 + x + 1$$

над полем $\mathbb{F}_2$. Иначе говоря, шифр использует поле $\mathbb{F}_{2[x]x^8+x^4+x^3+x+1}$. Любой элемент этого поля является полиномом над полем $\mathbb{F}_2$, степень которого меньше восьми, а операции выполняются по модулю $f(x)$. Назовем это поле "полем Rijndael". Благодаря изоморфизму это поле можно переобозначить как $\mathbb{F}_{2^8}$. Оно состоит из $2^8 = 256$ элементов.

На самом деле мы уже изучали поле Rijndael в главе 5 (примеры 5.17, 5.18 и 5.19), где были продемонстрированы следующие операции.

- Отображение между байтом и элементом поля (пример 5.17).
- Сложение двух элементов поля (пример 5.18).
- Умножение двух элементов поля (пример 5.19).

Эти сведения помогут нам изучить внутренние функции шифра Rijndael.

Во-первых, как уже указывалось, блок сообщения (состояние) и блок ключа в шифре Rijndael разбиты на байты. Учитывая взаимно-однозначное отображение, рассмотренное в примере 5.17, эти байты можно рассматривать как элементы поля и обрабатывать внутренними функциями шифра Rijndael.

7.7.2.1 Внутренняя функция **SubBytes**$(State)$

Эта функция выполняет нелинейную подстановку каждого байта (т.е. числа x) в переменной *State*. Любой ненулевой байт $x \in (\mathbb{F}_{2^8})^*$ заменяется с помощью следующего преобразования:

$$y = Ax^{-1} + b, \tag{7.7.1}$$

где

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Если число x является нулевым байтом, результатом преобразования SubBytes является число $y = b$.

Следует отметить, что нелинейность преобразования (7.7.1) обеспечивается исключительно нелинейностью инверсии x^{-1}. Если бы это преобразование применялось непосредственно к элементу x, аффинное уравнение (7.7.1) было бы *абсолютно линейным*!

Поскольку матрица A, имеющая размер 8×8, является обратимой (т.е. ее строки линейно независимы в поле $\mathbb{F}_{2^8}$), преобразование (7.7.1) также обратимо. Следовательно, функция SubBytes(*State*) имеет обратную.

7.7.2.2 Внутренняя функция **ShiftRows**$(State)$

Эта функция применяется к каждой строке матрицы *State*. Если размер блока равен 128 бит, она осуществляет следующее преобразование.

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \to \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}. \tag{7.7.2}$$

Эта операция представляет собой перестановочный шифр (см. раздел 7.4). Она переставляет элементы матрицы, не изменяя их значений: для элементов i-й строки $(i = 0, 1, 2, 3)$ перестановка сводится к циклическому сдвигу вправо на $4 - i$ позиций.

Поскольку перестановочный шифр переставляет только элементы строк, это преобразование является обратимым.

7.7.2.3 Внутренняя функция MixColumns(*State*)

Эта функция применяется к каждому столбцу матрицы *State*. В частности, для четырех столбцов матрицы, стоящей в правой части формулы (7.7.2), функция MixColumns(*State*) выполняет четыре итерации. Рассмотрим эти итерации применительно к одному столбцу. Ее результатом также является столбец.

Во-первых, пусть вектор

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

является столбцом матрицы, стоящей в правой части формулы (7.7.2). Для простоты изложения не будем указывать номер столбца.

Этот столбец можно интерпретировать как полином третьей степени:

$$s(x) = s_3x^3 + s_2x^2 + s_1x + s_0.$$

Заметим, что, поскольку коэффициенты $s(x)$ — это байты, т.е. элементы поля $\mathbb{F}_{2^8}$, функция $s(x)$ представляет собой полином *над* полем $\mathbb{F}_{2^8}$, т.е. не является элементом поля Rijndael.

Операция над столбцом $s(x)$ сводится к умножению этого полинома на фиксированный полином третьей степени $c(x)$ по модулю $x^4 + 1$:

$$c(x) \cdot s(x) (\mathrm{mod}\, x^4 + 1), \tag{7.7.3}$$

где полином $c(x)$ имеет вид

$$c(x) = c_3x^3 + c_2x^2 + c_1x + c_0 = \text{‘03’}x^3 + \text{‘01’}x^2 + \text{‘01’}x + \text{‘02’}.$$

Коэффициенты полинома $c(x)$ являются элементами поля $\mathbb{F}_{2^8}$ (представленными в шестнадцатеричном виде).

Следует отметить, что операция умножения (7.7.3) *не является* операцией в поле Rijndael, поскольку полиномы $c(x)$ и $s(x)$ не принадлежат полю Rijndael. Кроме того, так как полином $x^4 + 1$ представляет собой приводимый полином над полем $\mathbb{F}_2(x^4 + 1 = (x + 1)^4)$, умножение в формуле (7.7.3) не является операцией ни в одном поле (теорема 5.5 из раздела 5.4.2.2). Единственная причина, по которой это умножение должно выполняться по модулю полинома четвертой степени, заключается в том, что ее результатом должен быть полином третьей степени, т.е. для того, чтобы преобразование переводило столбец (полином третьей степени) в такой же столбец (полином третьей степени). Это преобразование можно считать полиалфавитой подстановкой (умножением) с известным ключом.

Читатели могут применить деление столбиков, описанное в примере 5.15, и самостоятельно убедиться, что в поле $\mathbb{F}_2$ выполняется следующее уравнение (учитывая, что вычитание в этом поле идентично сложению):

$$x^i (\mathrm{mod}\, x^4 + 1) = x^{i(\mathrm{mod}\, 4)}.$$

Следовательно, в произведении (7.7.3) коэффициенты при степенях x^i (при $i = 0, 1, 2, 3$) представляют собой суммы произведений $c_j s_k$ по индексам, удовлетворяющим условию $j+k = i(\mathrm{mod}\, 4)$, где $j, k = 0, 1, 2, 3$. Например, коэффициент при степени x^2 в этом произведении имеет следующий вид:

$$c_2 s_0 + c_1 s_1 + c_0 s_2 + c_3 s_3.$$

Операции умножения и сложения в этом выражении считаются операциями в поле $\mathbb{F}_{2^8}$. По этой причине легко проверить, что умножение полиномов в выражении (7.7.3) можно выполнить следующим образом.

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} \text{‘02’} & \text{‘03’} & \text{‘01’} & \text{‘01’} \\ \text{‘01’} & \text{‘02’} & \text{‘03’} & \text{‘01’} \\ \text{‘01’} & \text{‘01’} & \text{‘02’} & \text{‘03’} \\ \text{‘03’} & \text{‘01’} & \text{‘01’} & \text{‘02’} \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}. \quad (7.7.4)$$

Заметим далее, что, поскольку полином $c(x)$ является взаимно простым с полиномом x^4+1 над полем $\mathbb{F}_2$, в поле $\mathbb{F}_2[x]$, существует инверсия $c(x)^{-1}(\mathrm{mod}\, x^4+1)$. Иначе говоря, матрица и преобразование (7.7.4) являются обратимыми.

7.7.2.4 Внутренняя функция **AddRoundKey**$(State, RoundKey)$

Эта функция просто складывает (байт за байтом и бит за битом) элементы переменной *RoundKey* с элементами переменной *State*. Операция сложения трактуется как операция сложения в поле $\mathbb{F}_2$ (т.е., как побитовая операция XOR) и является обратимой, причем обратной к ней также является операция сложения.

Биты переменной *RoundKey* считаются упорядоченными по расписанию, т.е. биты ключей из разных раундов не совпадают друг с другом. Они вычисляются с помощью фиксированного (открытого) расписания ключей. Подробное устройство расписания ключей изложено в работе [219].

Итак, мы завершили описание внутренних функций шифра Rijndael и операции шифрования.

7.7.2.5 Операция расшифровки

Поскольку все четыре внутренние функции являются обратимыми, расшифровка сводится к выполнению шифрования в обратном порядке.

$\mathsf{AddRoundKey}(State, RoundKey)^{-1}$;
$\mathsf{MixColumns}(State)^{-1}$;
$\mathsf{ShiftRows}(State)^{-1}$;
$\mathsf{SubBytes}(State)^{-1}$.

Следует заметить, что в отличие от шифра Файстеля, в котором шифрование и расшифровка используют одну и ту же электронную схему или программу, шифр Rijndael должен применять разные электронные схемы и программы для шифрования и расшифровки соответственно.

7.7.3 Роль внутренних функций алгоритма Rijndael

В заключение опишем роль внутренних функций алгоритма Rijndael.

- Функция SubBytes предназначена для реализации нелинейного подстановочного шифра. Как указывалось в разделе 7.6.2, нелинейность — важное свойство блочного шифра, защищающее его от криптоанализа.
- Функции ShiftRows и MixColumns предназначены для смешивания байтов, размещенных в разных местах блока исходного сообщения. Как правило, распределения исходных текстов в пространстве сообщений имеют низкую энтропию благодаря высокой избыточности натуральных языков и экономических данных (иначе говоря, обычные исходные тексты образуют малое подпространство во всем пространстве сообщений). Смесь байтов, стоящих в разных позициях блока сообщения, расширяет распределение сообщений. По существу, это обеспечивает свойство перемешивания, сформулированное Шенноном (см. раздел 7.1.1).
- Функция AddRoundKey обеспечивает необходимую секретную случайность распределения сообщений.

Эти функции применяются многократно (как минимум 10 раз, если размер ключа и данных равен 128 бит).

7.7.4 Быстродействующая и стойкая реализация

Как мы убедились, внутренние функции алгоритма Rijndael очень просты и действуют в очень простых алгебраических пространствах. В результате реализации этих внутренних функций могут быть чрезвычайно эффективными. Из описания внутренних функций алгоритма Rijndael следует, что только функции SubBytes и MixColumns выполняют нетривиальные алгебраические операции, а значит, необходимо обеспечить их быструю и эффективную реализацию.

Во-первых, величину x^{-1} в функции SubBytes можно эффективно вычислить с помощью табличного поиска: можно однажды построить маленькую таблицу, состоящую из $2^8 = 256$ пар байтов, а затем постоянно ее использовать. (Например, такую таблицу можно “зашить” в электронную схему или реализовать в виде

программы). В этой таблице нулевая пара соответствует нулевому байту, а остальные 255 пар соответствуют 255 парам (x, x^{-1}), в которых обращение выполняется в поле $\mathbb{F}_{2^8}$. Метод табличного поиска не только эффективен, но и предотвращает **атаку на основе временно́го анализа** (timing analysis attack), которая использует разность между временем выполнения операций над разными данными и позволяет распознать, к какому биту применялась операция — 0 или 1 (см. раздел 12.5.4).

Поскольку матрица A и вектор b в выражении (7.7.1) являются постоянными, метод табличного поиска можно распространить на все преобразование (7.7.1), включив в таблицу все пары элементов $x, y \in \mathbb{F}_{2^8}$, причем пара $(0, b)$ представляет собой частный случай пары (x, y).

Очевидно, что обратное преобразование должно использовать обратную таблицу. Следовательно, функцию SubBytes можно реализовать с помощью двух маленьких таблиц, размер каждой из которых равен 256 байт.

Далее, умножение элементов поля $\mathbb{F}_{2^8}$ в функции MixColumn, т.е. коэффициентов полиномов $c(x)$ и $s(x)$, а точнее — элементов фиксированной матрицы и столбца в выражении (7.7.4) также можно реализовать с помощью метода табличного поиска: $z = x \cdot y$ (умножение в поле), где $x \in \{\text{‘01’}, \text{‘02’}, \text{‘03’}\}$ и $y \in \mathbb{F}_{2^8}$. Кроме того, байт ‘01’ представляет собой единицу относительно операции умножения в поле $\mathbb{F}_{2^8}$, т.е. $\text{‘01’} \cdot y = y$. Таким образом, для реализации (аппаратной или программной) необходима таблица с $2 \times 256 = 512$ ячейками. Это довольно маленькая таблица. Описанная выше реализация является не только эффективной, но и снижает риск атаки на основе временно́го анализа.

Линейные алгебраические операции в выражении (7.7.4) и их инверсии также можно эффективно реализовать в виде аппаратного обеспечения. Читатели могут найти подробное изложение этой темы в книге [87].

7.7.5 Благотворное влияние алгоритма AES на прикладную криптографию

Изобретение алгоритма AES привело к позитивным изменениям в состоянии прикладной криптографии.

Во-первых, многократное шифрование, например, с помощью тройного алгоритма DES, с появлением алгоритма AES стало излишним, а переменный размер ключа и блоков сообщения, равный 128, 192 и 256 бит, предоставляет широкий выбор решений, обеспечивающих стойкость в разнообразных приложениях. Поскольку в многократном шифровании используется большое количество ключей, появление AES облегчило управление криптографическими ключами и упростило процесс разработки протоколов и систем защиты данных.

Во-вторых, широкое применение алгоритма AES привело к изобретению новых хэш-функций, имеющих сравнимую степень стойкости. В некотором смысле алгоритмы блочного шифрования напоминают хэш-функции (см. раздел 10.3.1).

Стало повседневной практикой применять блочные алгоритмы шифрования в качестве однонаправленных функций хэширования. В качестве примера можно назвать протокол регистрации пользователей в операционной системе UNIX[1] [206]. В разделе 11.5.1 описывается типичное применение функции алгоритма DES для реализации управления паролями в системе UNIX. Другой пример использования блочных алгоритмов шифрования для реализации однонаправленных функций хэширования, снабженных ключами, описан в разделе 10.3.3. На практике хэш-функции часто применяются в качестве датчиков псевдослучайных чисел для генерации ключей блочных алгоритмов шифрования. Для генерации ключей и блоков данных в алгоритме AES нужны хэш-функции сравнимых размеров. Однако из-за возможности атаки на основе метода квадратного корня (square-root attack), или атаки на основе парадокса "дней рождения" (разделы 3.6 и 10.3.1) размер хэш-функции должен вдвое превышать размер ключа и блока данных. Следовательно, для размеров, равных 128, 192 и 256 бит, хэш-функция должна генерировать данные, имеющие размеры 256, 384 и 512 бит. В настоящее время организация ISO/IES разрабатывает стандарты хэш-функций SHA-256, SHA-384 и SHA-512 [151].

В заключение отметим, что, как и алгоритм DES, алгоритм AES вызывает пристальное внимание со стороны криптоаналитиков, изучающих блочные шифры, что, несомненно, приведет в дальнейшем к появлению новых открытий в этой области.

7.8 Режимы шифрования

В ходе шифрования и расшифровки с помощью блочных шифров сообщения интерпретируются как блоки данных. Как правило, длина строки превосходит размер блока, поэтому сообщения разбиваются на ряд последовательных блоков, которые обрабатываются поочередно.

Для алгоритмов блочного шифрования разработаны разные режимы. Эти режимы (за исключением тривиальных) обеспечивают требуемые свойства блочных шифрованных текстов, например, случайность, возможность выравнивания исходных сообщений до произвольного размера (так чтобы длина зашифрованного текста не была связана с длиной исходного текста), контроль за распространением ошибок, генерацию бегущего ключа для потокового шифра и т.п.

Однако не следует считать, что применение этих режимов шифрования может превратить "учебный" криптографический алгоритм в практичный прикладной метод. В этом мы еще не раз убедимся (в частности, в разделе 7.8.2.1, где рассматривается активная атака на некоторые протоколы, широко используемые в реальных приложениях).

---

[1] UNIX — торговая марка компании Bell Laboratories.

Существуют пять общепринятых видов режимов: **электронная кодовая книга** (electronic codebook — ECB), **сцепление блоков зашифрованного текста** (cipher block chaining — CBC), **обратная связь по выходу** (output feedback — OFB), **обратная связь по зашифрованному тексту** (cipher feedback — CFB) и **счетчик** (counter — CTR). Дальнейшее описание следует новейшим рекомендациям института NIST [218]. В нем используются следующие обозначения.

- $\mathcal{E}()$: алгоритм шифрования с помощью соответствующего блочного шифра;
- $\mathcal{D}()$: алгоритм расшифровки с помощью соответствующего блочного шифра;
- n: бинарный размер блока сообщения, зашифрованного с помощью соответствующего блочного шифра (во всех рассматриваемых нами блочных шифрах пространства исходных и зашифрованных текстов совпадают, и поэтому число n представляет собой размер входных данных и результатов работы алгоритма блочного шифрования);
- $P_1, P_2, \ldots, P_m$ — m последовательных сегментов исходного сообщения, подвергающегося обработке;
  - если m-й сегмент короче других сегментов, он дополняется до стандартной длины;
  - в некоторых режимах шифрования размер сегмента сообщения равен n (размеру блока), а в других режимах он может не превышать число n;
- $C_1, C_2, \ldots, C_m$: m последовательных сегментов зашифрованного сообщения, являющегося результатом применения определенного режима шифрования;
- $\mathrm{LSB}_u(B)$, $\mathrm{MSB}_v(B)$: младший, u, и старший, v, значащие биты блока B соответственно; например

$$\mathrm{LSB}_2(1010011) = 11, \mathrm{MSB}_5(1010011) = 10100;$$

- $A \parallel B$: конкатенация блоков A и B; например,

$$\mathrm{LSB}_2(1010011) \parallel \mathrm{MSB}_5(1010011) = 11 \parallel 10100 = 1110100.$$

7.8.1 Режим электронной кодовой книги (ECB)

Наиболее простой способ шифрования (или расшифровки) последовательности сегментов сообщения — обрабатывать их по отдельности. В этом случае сегмент сообщения становится блоком. Поскольку этот простой и естественный способ напоминает присваивание кодовых слов, взятых из шифровальной книги, он получил название "электронная кодовая книга (ECB)" (в русскоязычной литературе этот метод называется **режимом простой замены**. — *Прим. ред.*). Режим ECB определяется следующим образом.

Шифрование в режиме ECB: $C_i \leftarrow \mathcal{E}(P_i), i = 1, 2, \ldots, m.$

Расшифровка в режиме ECB: $P_i \leftarrow \mathcal{E}(C_i), i = 1, 2, \ldots, m.$

Режим ECB является детерминированным, т.е. если сегменты $P_1, P_2, \ldots, P_m$ шифруются дважды с помощью одного и того же ключа, то результирующие блоки зашифрованного текста будут одинаковыми. В приложениях данные, как правило, содержат информацию, которую легко угадать. Например, нетрудно определить зарплату сотрудников, поскольку она колеблется в небольшом диапазоне. В таких ситуациях зашифрованный текст, полученный в результате применения детерминированной схемы шифрования, позволяет атакующему угадать исходное сообщение методом последовательного перебора вариантов. Например, если известно, что зашифрованный текст, полученный методом ECB, содержит размеры зарплат, то после небольшого перебора вариантов атакующий может расшифровать эти данные. Как правило, детерминированные шифры применять не рекомендуется, и по этой причине режим ECB не следует использовать в большинстве приложений.

7.8.2 Режим сцепления блоков зашифрованного текста (CBC)

Режим сцепления блоков зашифрованного текста представляет собой общепринятый алгоритм блочного шифрования данных, имеющих общий характер. В режиме CBC результат шифрования представляет собой последовательность n-битовых блоков зашифрованного текста, сцепленных друг с другом. Таким образом, все блоки зашифрованного текста зависят не только от соответствующего блока исходного текста, но и от всех предыдущих блоков. Режим CBC состоит из следующих операций.

Шифрование в режиме CBC:

ВВОД: $IV, P_1, P_2, \ldots, P_m$;
ВЫВОД: $IV, C_1, C_2, \ldots, C_m$;

$$C_0 \leftarrow IV;$$
$$C_i \leftarrow \mathcal{E}(P_i \oplus C_{i-1}), \quad i = 1, 2, \ldots, m.$$

Расшифровка в режиме CBC:

ВВОД: $IV, C_1, C_2, \ldots, C_m$;
ВЫВОД: $IV, P_1, P_2, \ldots, P_m$;

$$C_0 \leftarrow IV;$$
$$P_i \leftarrow \mathcal{D}(C_i) \oplus C_{i-1}, \quad i = 1, 2, \ldots, m.$$

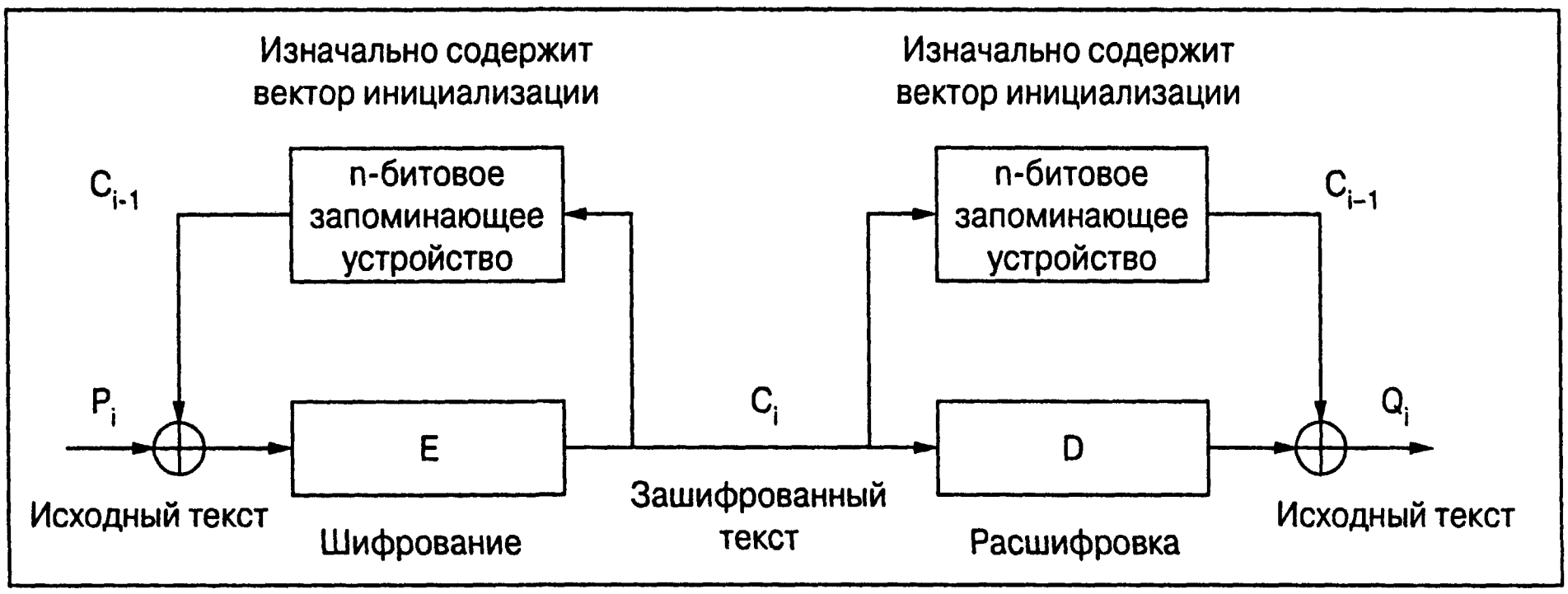


Рис. 7.3. Режим сцепления блоков зашифрованного текста

Для вычисления первого блока C_1 в зашифрованном тексте необходим специальный входной блок C_0, который называется "вектором инициализации" (Initializing Vector — IV). Вектор инициализации (в русскоязычной литературе — **синхропосылка**. — *Прим. ред.*) представляет собой случайный n-битовый блок. В каждом сеансе шифрования должен применяться новый случайный вектор инициализации. Поскольку этот вектор обрабатывается как блок зашифрованного текста, его не обязательно засекречивать, но он должен быть непредсказуемым. В соответствии с процедурой шифрования первый блок зашифрованного текста, C_1, рандомизируется с помощью вектора инициализации. Аналогичным образом все остальные блоки зашифрованного текста поочередно рандомизируются с помощью своего непосредственного предшественника. Следовательно, результатом шифрования в режиме CBC являются рандомизированные блоки зашифрованного текста, который должен содержать в себе вектор инициализации. Таким образом, если исходный текст состоял из m блоков, то результат шифрования в режиме CBC будет содержать $m + 1$ блоков.

Пусть $Q_1, Q_2, \ldots, Q_m$ — блоки, полученные в результате расшифровки блоков зашифрованного текста $C_0, C_1, C_2, \ldots, C_m$. Тогда, поскольку

$$Q_i = \mathcal{D}(C_i) \oplus C_{i-1} = (P_i \oplus C_{i-1}) \oplus C_{i-1} = P_i,$$

алгоритм расшифровки действует правильно. Режим CBC продемонстрирован на рис. 7.3.

7.8.2.1 Распространенное заблуждение

На первый взгляд кажется, что, поскольку в режиме CBC блоки сцеплены друг с другом, он предотвращает несанкционированное изменение данных, например, вставку или удаление (защите **целостности данных** (data integrity) посвящена глава 10). Таким образом, некоторые алгоритмы, использующие режим сцепления

блоков зашифрованного текста, можно считать средствами для защиты целостности данных. Например, режим RC5–CBC–PAD [17] определяет следующую **схему дополнения исходного текста в режиме CBC** (CBC plaintext padding scheme).

1. Строка исходного текста разбивается на последовательность байтов (байт состоит из восьми бит); каждые восемь байтов образуют блок исходного сообщения (таким образом, размер блока равен 64 бит).
2. Последний блок исходного сообщения должен быть дополнен. Его первая часть состоит из a последних байтов исходного сообщения ($0 \leqslant a \leqslant 7$), за которыми следуют $8 - a$ байтов-заполнителей. Каждый байт-заполнитель равен шестнадцатеричному числу $8 - a$. Например, если последний блок состоит из семи байтов исходного сообщения, то к ним присоединяется один байт-заполнитель, равный '01'. Итак, дополненный блок выглядит следующим образом:

   байт сообщения$_1$ || байт сообщения$_2$ || $\cdots$ || байт сообщения$_7$ || '01'.

   Если бы заключительный блок состоял из одного байта исходного сообщения, то дополненный блок принял бы следующий вид.

   байт сообщения || '07' || '07' || '07' || '07' || '07' || '07' || '07'

   Если количество байтов сообщения кратно восьми, то к блокам присоединяется дополнительный блок байтов-заполнителей.

   '08' || '08' || '08' || '08' || '08' || '08' || '08' || '08'

Другие алгоритмы шифрования в режиме CBC используют аналогичные схемы дополнения. Например, в протоколе IPSec (описанном в главе 12) применяются X байтов-заполнителей ($1 \leqslant X \leqslant 255$):

'01' || '02' || $\cdots$ || 'xy'

Здесь '0' $\leqslant$ 'x' $\leqslant$ 'F' и '0' $\leqslant$ 'y' $\leqslant$ 'F', а символ 'xy' является шестнадцатеричным представлением целого числа X. В ходе шифрования распознанные байты-заполнители удаляются из восстановленного исходного сообщения (разумеется, после проверки его целостности).

В двух предварительных сообщениях Международной организации по стандартизации (International Organization for Standards — ISO) [144, 145] были предложены некоторые протоколы аутентификации, предназначенные для защиты целостности данных с помощью режима CBC (принципы, положенные в основу этих протоколов, сформулированы в публикациях [142, 146]).

Однако было бы совершенно неправильно думать, что режим CBC может обеспечить защиту целостности данных *в любом понимании этого термина*.

Воденэ (Vaudene) [294] продемонстрировал атаку на схему дополнения блоков в режиме CBC, которая выявила полное отсутствие защиты. В атаке Воденэ Злоумышленник (атакующий) посылает адресату (хранителю ключа, называемому **оракулом шифрования** (decryption oracle)[2] и предоставляющему **услуги оракула** (oracle service)), два подобранных блока зашифрованного текста

$$r, C_i,$$

где r — случайный блок данных, а $C_i = \mathcal{E}(P \oplus C_{i-1})$ — блок зашифрованного текста, по которому Злоумышленник пытается восстановить исходное сообщение P (например, пароль). Алгоритм шифрования в режиме CBC выполняет операцию

$$P \oplus C_{i-1} \oplus r.$$

Метод проверки целостности данных содержит инструкции, которые должен выполнять оракул шифрования. Анализируя поведение оракула шифрования, Злоумышленник имеет хорошие шансы распознать некоторую информацию в блоке исходного сообщения P. Например, если механизм защиты целостности данных вынуждает оракула шифрования отвечать "Да", если данные дополнены правильно, то наиболее вероятно, что этим дополнением является байт '01'. Вероятность этого события близка к 2^{-8}, поскольку вероятностным пространством является байт, состоящий из восьми бит. Вследствие случайности блока r другие варианты правильного дополнения намного менее вероятны (поскольку в этом случае вероятностное пространство состоит из нескольких байтов), и их можно не рассматривать. Затем Злоумышленник выясняет, что

$$LSB_8(P) = LSB_8(r) \oplus \text{'01'},$$

т.е. Злоумышленник правильно распознает заключительный байт блока P, представляющий собой значительную часть всего блока!

Если в процедуре шифрования выявляется ошибка дополнения (вероятность этого события близка к $1 - 2^{-8}$), оракул может ответить "НЕТ" или не ответить вообще (процедура прекращается так, будто оракул взорвался, поэтому Воденэ назвал такого оракула **заминированным** (bomb oracle)). Однако отсутствие ответа фактически подразумевает отрицательный ответ. Если оракул явно или неявно отвечает "НЕТ", Злоумышленнику не удается распознать последний байт. Однако он может изменить блок r и попытаться снова. Этот способ называется **активной атакой** (active attack). Она, как правило, направлена на адресата, предоставляющего услуги оракула. Формальное определение активной атаки будет дано

[2]Термин "оракул" часто встречается в литературе по криптографии. Как правило, оракулом называют неизвестный алгоритм или метод, которому приписывают способность решить сложную задачу. Услуги оракула означают, что пользователь предоставляет атакующему (как правило, неумышленно) возможность выполнять криптографические операции с помощью ключа, недоступного атакующему.

в разделе 8.6. Кроме того, во многих частях книги будут описаны сценарии атак, в которых важную роль играет принципал, предоставляющий услуги оракула.

Воденэ применил свою атаку для взлома некоторых криптографических протоколов, широко применяемых в реальных приложениях, например, IPSec, SSH и SSL (см. главу 12). В этих протоколах Злоумышленник легко может получить ответ "ДА/НЕТ", даже если они недоступны в явном виде (например, зашифрованы).

В главной разновидности этой атаки оракул шифрования просто возвращает последний байт с вероятностью, близкой к 2^{-8}, если "не взрывается". Тем не менее во многих приложениях существуют способы предотвратить взрыв оракула и получать от него ответы, извлекая из них байты исходного сообщения. Допустим, что, ответив "ДА" на запрос о последнем байте исходного сообщения, оракул остается невредим. Тогда Злоумышленник может заменить блок r блоком r', так что

$$\mathrm{LSB}_8(r') \leftarrow \mathrm{LSB}_8(r) \oplus \text{'01'} \oplus \text{'02'}.$$

Тогда, посылая оракулу пару блоков r', C, Злоумышленник стремится определить все остальные байты исходного сообщения с той же вероятностью, равной 2^{-8}. Если оракул не взрывается, атака может продолжаться до тех пор, пока Злоумышленник не восстановит исходный текст, посылая до $8 \times 2^8 = 2048$ запросов.

В разделе 12.5.4 мы продемонстрируем применение атаки Воденэ для дешифровки электронных сообщений, зашифрованных в режиме CBC с дополнением. В такой атаке роль оракула исполняет почтовый сервер, который никогда не взрывается и, следовательно, позволяет Злоумышленнику извлечь блок исходного сообщения, представляющий собой пароль пользователя. Эта атака использует **вспомогательную информацию** (side channel information), получаемую с помощью **временно́го анализа** (timing analysis). По этой причине она называется **атакой с помощью обходного канала** (side channel attack).

Протокол ISO, использующий режим CBC для защиты целостности данных, также имеет фатально слабое место [184, 185]. Его недостатки будут продемонстрированы в разделе 17.2.1.2 при анализе протокола аутентификации, в котором используется режим CBC. Этот протокол был разработан для защиты целостности данных, однако его слабость заключается именно в том, что он не способен достичь этой цели.

Единственный способ защитить данные — рандомизировать зашифрованный текст. Другие способы защиты целостности данных в текстах, зашифрованных в режиме CBC, будут рассмотрены в главе 10.

7.8.2.2 Предостережение

Кнудсен (Knudsen) выяснил, что режим CBC обладает ограниченными возможностями для обеспечения секретности [165]. Причины этого явления заклю-

чаются в следующем. Если два блока зашифрованных текстов C_i, C'_j совпадают, то в соответствии с алгоритмом шифрования в режиме CBC имеем

$$C_{i-1} \oplus C'_{j-1} = P_i \oplus P'_j.$$

Поскольку исходный текст, как правило, является избыточным, это уравнение позволяет восстановить его по зашифрованному тексту, который доступен взломщику. Для того чтобы предотвратить атаку шифра с помощью указанного уравнения, в каждом сеансе шифрования необходимо использовать случайные векторы инициализации. В этом случае вероятность того, что два зашифрованных текста окажутся одинаковыми, пренебрежимо мала (векторы инициализации образуют очень большое вероятностное пространство).

7.8.3 Режим обратной связи по зашифрованному тексту

Режим обратной связи по зашифрованному тексту (Cipher Feedback Mode — CFB) предполагает возврат последовательных сегментов результирующего зашифрованного текста на вход применяемого алгоритма блочного шифрования. Сообщение (исходное или зашифрованное) имеет размер s, где $0 \leqslant s \leqslant n$. В режиме CFB (в русскоязычной литературе — **гаммирование с обратной связью**. — *Прим. ред.*) вектор инициализации должен быть случайным n-битовым блоком. Поскольку вектор инициализации играет роль зашифрованного текста, его не обязательно держать в секрете. Режим CFB предусматривает следующие операции.

Шифрование в режиме CFB:

ВВОД: $IV, P_1, P_2, \ldots, P_m$;
ВЫВОД: $IV, C_1, C_2, \ldots, C_m$;

$$\begin{aligned}
I_1 &\leftarrow IV; \\
I_i &\leftarrow \mathrm{LSB}_{n-s}(I_{i-1}) \parallel C_{i-1}, && i = 2, \ldots, m; \\
O_i &\leftarrow \mathcal{E}(I_i), && i = 1, 2, \ldots, m; \\
C_i &\leftarrow P_i \oplus \mathrm{MSB}_s(O_i), && i = 1, 2, \ldots, m.
\end{aligned}$$

Расшифровка в режиме CFB:

ВВОД: $IV, C_1, C_2, \ldots, C_m$;
ВЫВОД: $P_1, P_2, \ldots, P_m$;

$$\begin{aligned}
I_1 &\leftarrow IV; \\
I_i &\leftarrow \mathrm{LSB}_{n-s}(I_{i-1}) \parallel C_{i-1}, && i = 2, \ldots, m; \\
O_i &\leftarrow \mathcal{E}(I_i), && i = 1, 2, \ldots, m; \\
P_i &\leftarrow C_i \oplus \mathrm{MSB}_s(O_i), && i = 1, 2, \ldots, m.
\end{aligned}$$

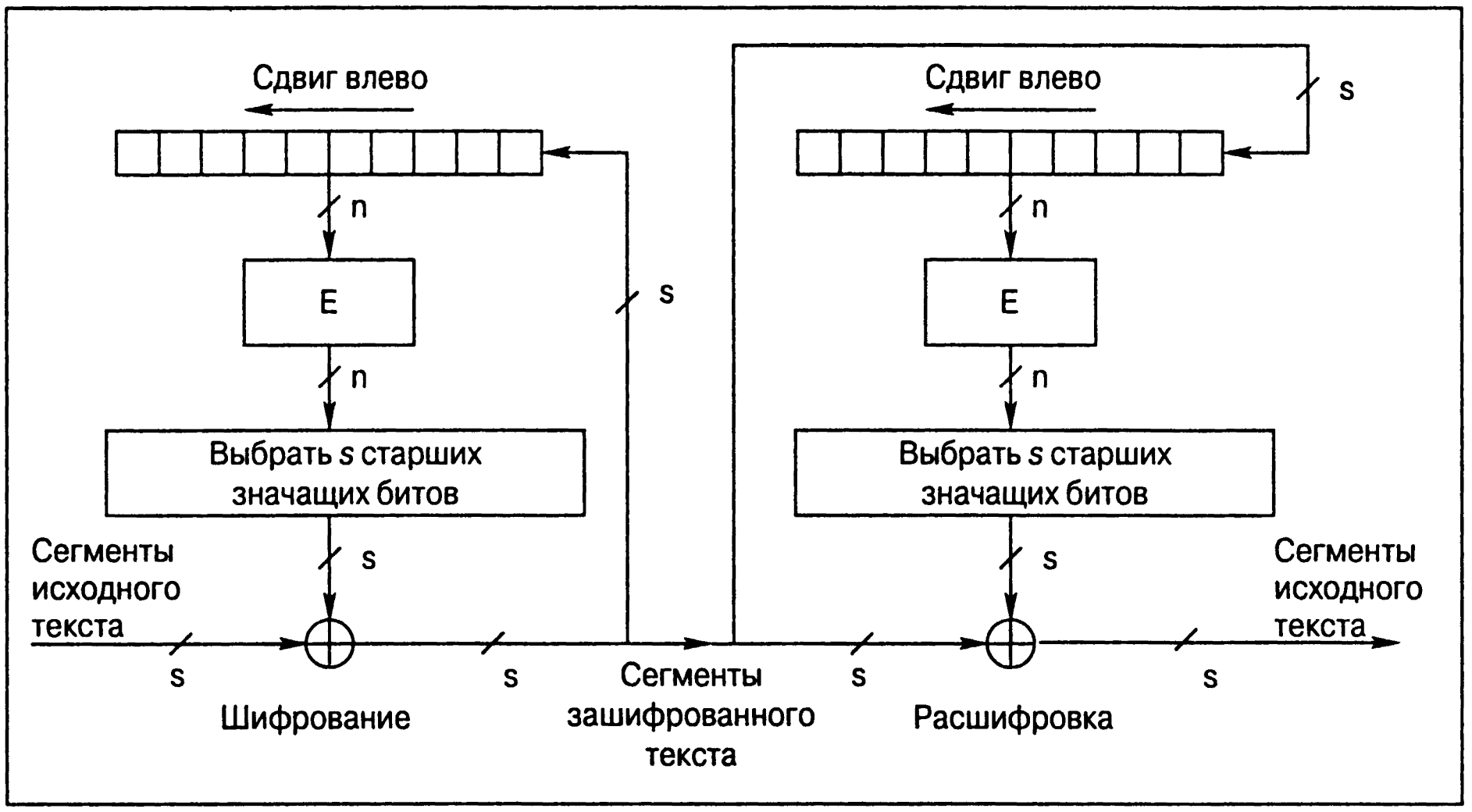


Рис. 7.4. Режим обратной связи по зашифрованному тексту

Заметим, что в режиме CFB при шифровке и расшифровке одного и того же блока используется одна и та же функция шифрования. В результате в качестве функции шифрования E можно использовать любое однонаправленное преобразование, например, однонаправленную хэш-функцию. Режим CFB можно рассматривать в качестве генератора бегущего ключа для потокового шифра, в котором шифрование осуществляется с помощью шифра Вернама. Как и в режиме CBC, сегмент зашифрованного текста зависит от всех предыдущих сегментов исходного текста и вектора инициализации. Иллюстрация режима CFB приведена на рис. 7.4.

7.8.4 Режим обратной связи по выходу

Режим обратной связи по выходу (Output Feedback Mode — OFB) предполагает возврат последовательных результирующих блоков на вход применяемого алгоритма блочного шифрования. Эти блоки образуют строку битов, используемую в качестве бегущего ключа шифра Вернама, т.е. к бегущему ключу и блокам исходного сообщения применяется операция XOR. В режиме OFB вектор инициализации должен быть n-битовым входным блоком. Режим CFB предусматривает следующие операции.

Шифрование в режиме OFB: ВВОД: $IV, P_1, P_2, \ldots, P_m$;
ВЫВОД: $IV, C_1, C_2, \ldots, C_m$;

$$
\begin{aligned}
I_1 &\leftarrow IV; \\
I_i &\leftarrow O_{i-1}, \qquad i = 2, \ldots, m; \\
O_i &\leftarrow \mathcal{E}(I_i), \qquad i = 1, 2, \ldots, m; \\
C_i &\leftarrow P_i \oplus O_i, \quad i = 1, 2, \ldots, m.
\end{aligned}
$$

Расшифровка в режиме OFB:

ВВОД: $IV, C_1, C_2, \ldots, C_m$;
ВЫВОД: $P_1, P_2, \ldots, P_m$;

$$
\begin{aligned}
I_1 &\leftarrow IV; \\
I_i &\leftarrow O_{i-1}, \qquad i = 2, \ldots, m; \\
O_i &\leftarrow \mathcal{E}(I_i), \qquad i = 1, 2, \ldots, m; \\
P_i &\leftarrow C_i \oplus O_i, \quad i = 1, 2, \ldots, m.
\end{aligned}
$$

В режиме OFB шифрование и расшифровка идентичны: к блокам входного сообщения и потоковому ключу, порожденному механизмом обратной связи, применяется операция XOR. Механизм обратной связи представляет собой конечный автомат, состояние которого зависит исключительно от ключа шифрования, соответствующего алгоритму шифрования и вектору инициализации. Таким образом, если в блоке зашифрованного текста возникает ошибка, то искажению подвергнется только соответствующая позиция в исходном тексте. Следовательно, режим OFB можно применять, чтобы зашифровать сообщения, для которых повторная передача невозможна. Как и в режиме CFB, алгоритм блочного шифрования можно заменить подходящей однонаправленной хэш-функцией. Иллюстрация режима OFB приведена на рис. 7.5.

7.8.5 Режим счетчика

Режим счетчика (Counter Mode — CTR) предполагает возврат на вход соответствующего алгоритма блочного шифрования значения счетчика, накопленного с момента старта. Увеличивая значение счетчика, алгоритм блочного шифрования образует строку битов, которая используется в качестве бегущего ключа шифра Вернама, т.е. к бегущему ключу и блокам исходного сообщения применяется операция XOR. Режим CTR предусматривает следующие операции.

Шифрование в режиме CTR:

ВВОД: $Ctr_1, P_1, P_2, \ldots, P_m$;
ВЫВОД: $Ctr_1, C_1, C_2, \ldots, C_m$;

$$C_i \leftarrow P_i \oplus \mathcal{E}(Ctr_i); \quad i = 1, 2, \ldots, m;$$

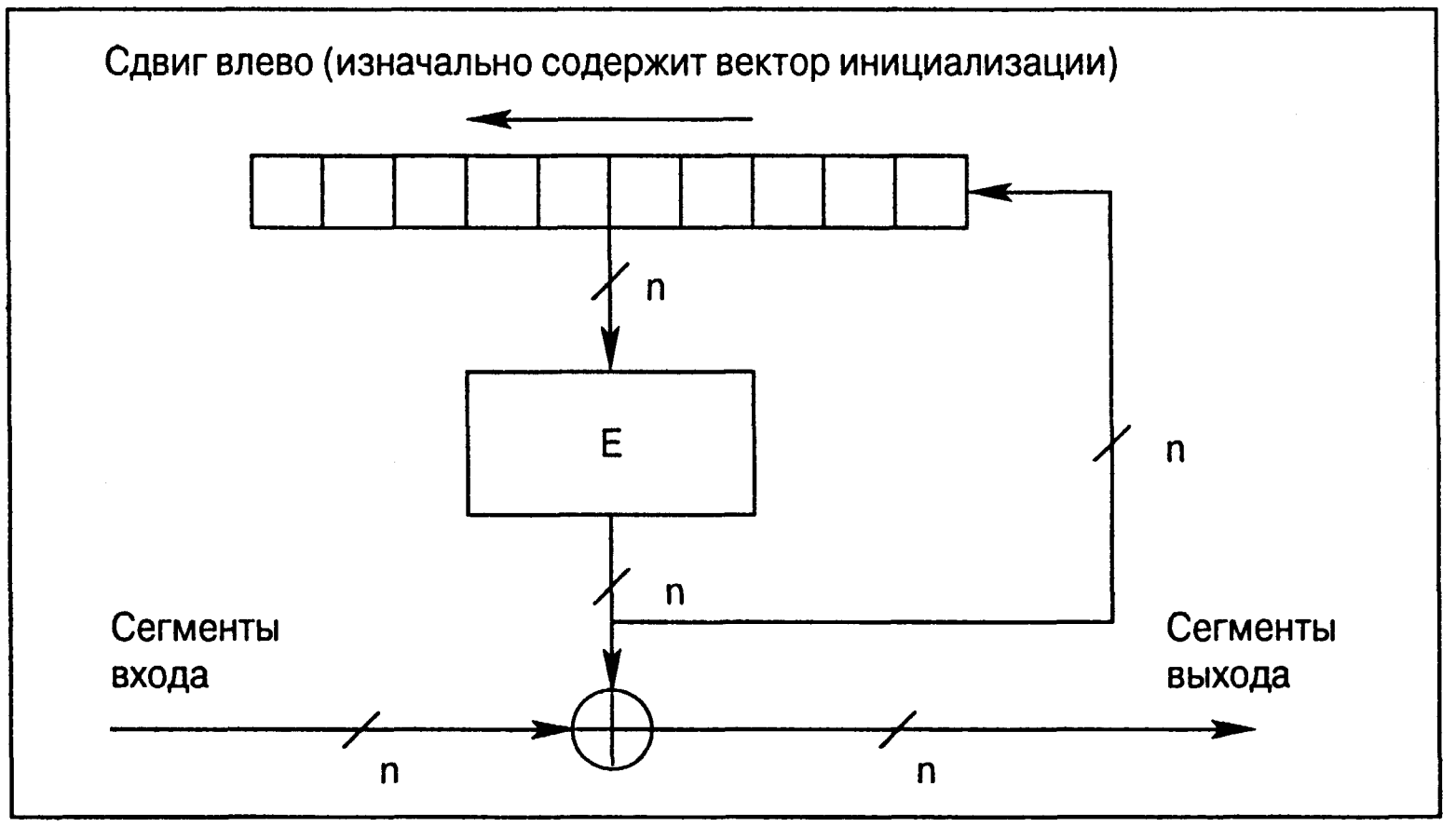


Рис. 7.5. Режим обратной связи по выходу

Расшифровка в режиме CTR:

ВВОД: $Ctr_1, C_1, C_2, \ldots, C_m$;
ВЫВОД: $P_1, P_2, \ldots, P_m$;

$$P_i \leftarrow C_i \oplus \mathcal{E}(Ctr_i); \quad i = 1, 2, \ldots, m;$$

При отсутствии обратной связи алгоритмы шифрования и расшифровки в режиме CTR могут выполняться параллельно. Это обеспечивает режиму CTR преимущество перед режимами CFB и OFB. Поскольку этот режим весьма прост, для его описания не требуются иллюстрации.

7.9 Каналы для обмена ключами в симметричных криптосистемах

Прежде чем приступить к обмену секретной информацией в симметричной криптосистеме, два пользователя должны обменяться правильными криптографическими ключами. Здесь слово "правильный" означает не только то, что ключ содержит корректные биты, т.е. не поврежден, но и то, что данный ключ принадлежит тому, для кого он предназначен.

Канал связи, используемый для установления правильного ключа, называется **каналом для обмена ключами** (key channel) (рис. 7.1). Канал для обмена ключами не совпадает с **каналом для обмена сообщениями** (message channel). Разница между ними заключается в том, что канал для обмена ключами защищен, а канал для обмена сообщениями — нет. Поскольку в симметричных криптосистемах

ключи шифрования и расшифровки совпадают, канал для обмена ключами должен обеспечивать секретность и аутентичность ключа.

Канал для обмена ключами в симметричных криптосистемах можно установить тремя способами: с помощью стандартных методов (conventional techniques). систем с открытым ключом (public-keys techniques) или метода распределения квантовых ключей (Quantum Key Distribution — QKD).

Стандартные методы. В систему установки времени можно внедрить физически безопасный механизм, например, курьерскую службу, которая позволит двум пользователям эксклюзивно использовать общий ключ. Как правило. одним из таких пользователей является доверенный посредник (trusted third party — TTP), обеспечивающий аутентификацию (см. раздел 2.4). После обмена ключами между одним из пользователей и доверенным посредником. использующим долговременный канал, любые два пользователя могут запустить протокол аутентификации для установки защищенного канала, предназначенного для обмена ключами между ними. Привлечение посредника позволяет пользователям избежать обременительной работы: в противном случае пользователям пришлось бы управлять многочисленными ключами. В главе 2 рассмотрено несколько протоколов аутентификации и установки сеансовых ключей, в которых использовался долговременный канал между одним из пользователей и сервером аутентификации. Более подробно эти вопросы будут изучены в главах 11, 12 и 17, посвященных протоколам аутентификации, а также системам и методам формального анализа стойкости.

Серьезным недостатком стандартных методов создания канала для обмена ключами является необходимость применения службы аутентификации. Это ограничивает возможности масштабирования открытых систем, применяющих эти методы. На практике эти методы до сих пор используются в корпоративных сетях (см. раздел 12.4).

Системы с открытым ключом. Криптография с открытым ключом позволяет легко устанавливать канал для обмена ключами между любыми двумя удаленными пользователями без личной встречи или привлечения службы аутентификации. В этом заключается важное преимущество систем с открытым ключом перед стандартными методами. Следовательно, системы с открытым ключом позволяют легко масштабировать приложения. Существует много методов создания канала для обмена ключами в системах с открытым ключом. Криптография с открытым ключом описывается в следующей главе, а методы аутентификации в системах с открытым ключом — в главе 13.

Однако для криптографии с открытым ключом необходим защищенный канал для обмена ключами между пользователем и системой. Термин “защищенный” подразумевает аутентификацию: по открытому ключу можно действительно идентифицировать его владельца. Несмотря на это, установка

канала обмена ключами в системах с открытым ключом не требует манипуляции секретной информацией. Все сводится исключительно к задаче аутентификации. На рис. 7.1 показано, что канал для обмена открытыми ключами можно установить на основе службы каталогов. Некоторые практические методы создания каналов для обмена открытыми ключами и аутентификации рассмотрены в главе 12 (раздел 12.3), а общие методы аутентификации в системах с открытым ключом — в главе 13.

Метод распределения квантовых ключей. В разделе 4.4.5.1 был рассмотрен протокол распределения квантовых ключей (QKD, протокол 4.1). Этот протокол позволяет двум пользователям разделить секретный ключ, не встречаясь лично. Как и в системах с открытым ключом, для этого необходим протокол аутентификации между пользователем и системой. Этот протокол может использовать однонаправленную функцию. Пользователь владеет секретным прообразом однонаправленной функции и может доказать своему партнеру свою личность, не раскрывая ему секрета. Используя протокол аутентификации, участники протокола QKD могут быть уверенными в том, что в обмене информацией участвует партнер, имеющий на это право. Появление коммерческих протоколов QKD ожидалось в 2004 году [268].

Следует подчеркнуть, что протокол QKD является весьма перспективным. Многие системы с открытым ключом, основанные на теории вычислительной сложности (например, системы, использующие трудноразрешимую задачу определения периода периодической функции), с появлением квантовых технологий сойдут с арены. Однако протокол QKD устойчив по отношению к квантовым технологиям (считается, что существуют непериодические однонаправленные функции, прообраз которых трудно восстановить с помощью квантовых компьютеров) и может в дальнейшем применяться для аутентификации. Следовательно, в будущем протокол QKD будет широко использоваться для разделения ключа без личной встречи и без посредников.

В заключение отметим, что системы с открытым ключом и методы распределения квантовых ключей позволяют устанавливать секретный канал связи совершенно открыто [188, 189].

7.10 Резюме

В главе рассмотрены симметричный алгоритм шифрования и описаны некоторые симметричные схемы шифрования.

Рассмотрены классические шифры и показана их относительная стойкость в рамках теории информации, предложенной Шенноном. Изучен основной принцип классических шифров — подстановка, по-прежнему широко применяемая в современных симметричных алгоритмах шифрования.

Изложены два современных блочных алгоритма шифрования: DES и AES. Алгоритм DES представляет интерес с исторической точки зрения. Кроме того, структура шифра Файстеля, положенная в его основу, по-прежнему актуальна. Подробно описан алгоритм AES — современный стандарт шифрования. Рассмотрены методы быстрой и безопасной реализации алгоритма AES, продемонстрировано положительное влияние, которое оказал алгоритм AES на прикладную криптографию.

Далее описаны разнообразные стандартные режимы шифрования на основе блочных шифров. Развеяны заблуждения, связанные с самым распространенным режимом шифрования — режимом CBC. Показано, что режим CBC не может гарантировать целостности данных. Более подробно эта тема рассмотрена в главе 17, посвященной протоколам аутентификации на основе режима шифрования CBC.

В заключение описаны три метода установки защищенного канала для обмена ключами между партнерами, передающими секретную информацию. Показано, что, несмотря на простоту, протокол QKD в будущем получит широкое распространение, благодаря своей устойчивости к квантовым технологиям.

Упражнения

7.1. Почему алгоритм шифрования не должен содержать секретных компонентов?

7.2. Неодинаковые частоты употребления букв в английском языке — яркое доказательство того, что исходные сообщения образуют малую область во всем пространстве сообщений. Приведите еще два примера этого явления.

7.3. Пусть S_P, S_C — источники исходного и зашифрованного текста соответственно. Примените энтропийную формулировку из раздела 3.7 и объясните, почему зашифрованные сообщения, получаемые путем простой подстановки или перестановки, не изменяют распределения исходных сообщений, т.е. зашифрованные сообщения по-прежнему образуют малое подпространство во всем пространстве сообщений.

Подсказка: $H(S_P) = H(S_C)$.

7.4. Является ли шифр Вернама подстановочным? Он моно- или полиалфавитный?

7.5. В чем заключается различие между шифром Вернама и одноразовым блокнотом?

7.6. Почему метод одноразового блокнота абсолютно защищен от взлома?

7.7. Сдвиговый шифр, описанный в протоколе 7.1, представляет собой абсолютно стойкую схему, поскольку он использует одноразовый ключ, причем размер ключа совпадает с размером сообщения. Можно ли считать сдвиговый шифр

абсолютно стойким, если он не использует операции модулярной арифметики?

7.8. Почему простые подстановочные или перестановочные шифры, несмотря на их уязвимость для частотного анализа, по-прежнему широко применяются в современных алгоритмах шифрования и криптографических протоколах?

7.9. Современные шифры часто представляют собой комбинацию нескольких классических шифров. Укажите, в каких местах алгоритмов DES и AES применяются 1) подстановочные шифры, 2) перестановочные шифры и 3) шифр Вернама.

7.10. 1. Почему алгоритм AES считается очень эффективным? 2. Как реализовать умножение в конечном поле $\mathbb{F}_{2^8}$ в алгоритме AES?

7.11. Предположим, что в режиме сцепления блоков pfшифрованного текста (CBC) в результате расшифровки получен “правильно дополненный текст”. Можно ли утверждать, что в переданном исходном тексте сохранена целостность данных?

Глава 8

Шифрование — асимметричные методы

8.1 Введение

Стойкость классических шифров (например, шифра Цезаря) обеспечивалась абсолютной секретностью процесса шифрования. Современные шифры, такие как DES и AES, основаны на принципе Керхофса (см. раздел 7.1): алгоритмическое устройство этих шифров является открытым и доступным для исследования. Тем самым разработчики этих шифров стремились продемонстрировать, что стойкость их криптосистем зависит исключительно от выбора секретных ключей шифрования.

Существуют дополнительные возможности применения принципа Керхофса для уменьшения доли секретных компонентов алгоритма шифрования. Рассмотрим семантическое свойство шифрования, сформулированное Шенноном: преобразование перемешивания, абсолютно равномерно распределяющее осмысленные сообщения, принадлежащие области исходных текстов $\mathcal{M}$, по всему пространству сообщений $\mathcal{C}$ [264]. Как известно, такое распределение можно выполнить, не используя никаких секретов. Этот принцип был впервые реализован в работе Диффи (Diffie) и Хеллмана (Hellmann) в 1975 г. [97] (эта работа была опубликована в 1976 г., однако распространялась как препринт, начиная с декабря 1975 г. [96]). Они назвали свое изобретение **криптографией с открытым ключом** (public-key cryptography). В то время она представляла собой революционное открытие.

В криптосистемах с открытым ключом для шифрования не используется секретный ключ — он необходим только при расшифровке. В работе [97] Диффи и Хеллман кратко описали несколько математических преобразований, названных ими **однонаправленными функциями с секретом** (one-way trapdoor function). Они предназначались для реализации криптографии с открытым ключом и обладали следующими свойствами.

Свойство 8.1 (Однонаправленная функция с секретом). Однонаправленную функцию с секретом $f_t(x) : \mathcal{D} \mapsto \mathcal{R}$ легко вычислить для всех $x \in \mathcal{D}$, но очень трудно инвертировать для почти всех значений из $\mathcal{R}$. Однако, если используется

секретная информация t, то для всех значений $y \in \mathcal{R}$ легко вычислить величину $x \in \mathcal{D}$, удовлетворяющую условию $y = f_t(x)$.

Понятие однонаправленной функции с секретом — основное в криптографии с открытым ключом. Криптосистемы с открытым ключом, использующие однонаправленные функции с секретом, называются **асимметричными** (asymmetric cryptosystems). В первой работе Диффи и Хеллмана, посвященной криптографии с открытым ключом [97], было рассмотрено несколько однонаправленных функций с секретом. Однако эти функции были недостаточно асимметричными, поэтому вскоре Диффи и Хеллман предложили более удачный вариант: возведение в степень по модулю. Эта функция была использована в знаменитом криптографическом протоколе — протоколе обмена ключами Диффи–Хеллмана (Diffie-Hellman key exchange protocol) [98] (см. раздел 8.3). Эта первая удачная реализация криптографического алгоритма с открытым ключом широко используется до сих пор.

В 1974 году Меркл (Merkle) изобрел механизм согласования криптографического ключа путем явных асимметричных вычислений, получивших название *головоломка Меркла* [199]. Асимметричность головоломки Меркла заключается в том, что ее вычислительная сложность для законных участников протокола согласования ключа и для перехватчиков совершенно разная: легальные участники легко выполняют вычисления, а нелегальные — нет. Головоломка Меркла представляет собой первую эффективную реализацию однонаправленной функции с секретом. Несмотря на то что головоломка Меркла не подходит для применения в современных криптографических приложениях (мера ее асимметрии колеблется от n до n^2), ее влияние на криптосистемы с открытым ключом невозможно переоценить.

В настоящее время стало известно, что Кокс (Cocks), британский криптограф, изобрел первую криптосистему с открытым ключом в 1973 году [277]. Алгоритм шифрования Кокса, получивший название *алгоритма с несекретным ключом шифрования*, использует сложность разложения целого числа на простые множители и, по существу, совпадает с криптосистемой RSA (раздел 8.5). К сожалению, алгоритм Кокса был засекречен. В декабре 1997 года Группа по электронной защите средств связи (Communications Services Electronic Security Group — CESG) рассекретила алгоритм Кокса.

Несмотря на то что вначале криптосистемы с открытым ключом были достоянием узкого круга лиц, именно благодаря открытым исследованиям они нашли два важнейших применения: 1) цифровые подписи (раздел 10.4) и 2) обмен секретными ключами через открытые каналы связи (раздел 8.3). Эти два приложения лежат в основе электронной коммерции, осуществляемой через Internet.

8.1.1 Структурная схема главы

В начале главы вводится понятие стойкости "учебной криптографии", которое сопровождается предупреждением о ее практической уязвимости (раздел 8.2). Затем описываются основные примитивы криптографии с открытым ключом: протокол обмена ключами Диффи–Хеллмана (раздел 8.3), учебный вариант алгоритма RSA (раздел 8.5), а также криптосистемы Рабина (раздел 8.10) и Эль-Гамаля (раздел 8.12). Эти алгоритмы и протоколы сопровождаются теоремами об их формальной стойкости при соответствующих предположениях. В частности, формулируется задача Диффи–Хеллмана и задача дискретного логарифмирования (раздел 8.4), задача RSA (раздел 8.7) и задача о разложении целого числа на простые множители (раздел 8.8). Кроме того, в главе вводятся формальные понятия для описания разнообразных атак на криптосистемы с открытым ключом (раздел 8.6). Нестойкость учебных вариантов криптографических алгоритмов демонстрируется на примере алгоритма RSA (раздел 8.9), Рабина (раздел 8.11) и Эль-Гамаля (раздел 8.13). В разделе 8.14 рассматривается необходимость более строгого понятия стойкости шифрования с открытым ключом. В разделе 8.15 описывается комбинация симметричных и асимметричных криптосистем — гибридная криптосистема.

8.2 Нестойкость "учебных" алгоритмов шифрования

Следует отметить, что алгоритмы шифрования, описанные в этой главе, носят учебный характер. Их можно найти в любом учебнике по криптографии. К сожалению, эти алгоритмы непригодны для практического применения. Стойкость "учебного" алгоритма шифрования в криптосистеме с открытым ключом описывается следующим свойством.

Свойство 8.2 (Нестойкость "учебных" алгоритмов шифрования). В рамках этой главы стойкость (секретность) криптосистемы рассматривается с двух точек зрения.

1. **Стойкость по принципу "все или ничего".** Имея текст, зашифрованный определенным алгоритмом, атакующий должен восстановить блок исходного текста, размер которого, как правило, определяется параметром безопасности криптосистемы. Имея исходный и зашифрованный текст, атакующий должен восстановить целый блок секретного ключа. При этом атакующий либо добивается полного успеха, либо не получает ничего. Следует обратить особое внимание на слово "ничего": оно означает, что атакующий не имеет никакой секретной информации ни до, ни после безуспешной атаки.
2. **Пассивная атака.** Атакующий не манипулирует зашифрованным текстом и не модифицирует его, используя имеющиеся данные. Кроме того, он не

обращается к владельцу ключа с просьбой расшифровать или зашифровать сообщение.

Понятие стойкости, сформулированное выше, чрезвычайно расплывчато и непригодно для применения на практике. Точнее было бы назвать его "нестойкостью".

Поясним, почему свойство 8.2.1 на самом деле описывает нестойкость алгоритма. В приложениях исходные данные, как правило, содержат часть несекретной информации, которая может быть известной атакующему. Например, некоторые данные имеют небольшой диапазон изменения: зарплата не может превышать один миллион — числа, весьма небольшого по криптографическим меркам. В качестве второго примера можно указать пароли, состоящие из восьми символов. Довольно часто дополнительная информация позволяет атакующему распознать весь исходный текст.

Теперь покажем, почему свойство 8.2.2 также определяет нестойкость алгоритма. Никогда не следует полагаться на то, что атакующий инертен и пассивен. Как правило, атакующие не стесняются при выборе средств. В частности, они вступают в контакт с атакованным пользователем, посылают ему зашифрованный текст для последующей расшифровки и требуют вернуть исходный текст. Такой способ общения с пользователем (владельцем открытого ключа) предоставляет атакующему **услуги оракула шифрования**. Как будет показано в дальнейшем, избежать этого очень трудно.

Замечательные алгебраические свойства "учебных" криптографических алгоритмов часто позволяют атакующему, прибегающему к услугам оракула, взламывать криптосистему. В главе будет продемонстрировано несколько случаев успешных атак, а в последующих главах показана эффективность таких приемов.

Глава содержит несколько предупреждений об опасности услуг оракула. Следует отметить, что обычные пользователи алгоритмов с открытым ключом часто весьма наивны и предоставляют такие услуги атакующим. Кроме того, избежать этого очень трудно (см. раздел 12.5.4). Правильная стратегия заключается в том, что криптосистема должна оставаться стойкой, даже если с ней работают неопытные пользователи.

Сформулировав свойство 8.2, мы явно указали, что в рамках этой главы не рассматривается строгое понятие стойкости алгоритмов с открытым ключом. Следовательно, "учебные" алгоритмы шифрования не являются стойкими в строгом смысле этого слова. Наоборот, в главе показано, что "учебные" алгоритмы шифрования весьма слабы, поскольку допускают утечку информации. Способы исправления этих недостатков в главе не описываются.

Более строгие определения стойкости, позволяющие противостоять более сильным (т.е. реальным) атакам, будут даны в главе 14. Практические аналоги "учебных" алгоритмов шифрования описываются в главе 15.

8.3 Протокол обмена ключами Диффи–Хеллмана

В симметричных криптосистемах перед началом работы необходимо передать секретный ключ обеим сторонам. До появления криптосистем с открытым ключом распределение секретных ключей между общающимися сторонами всегда представляло собой сложную задачу, поскольку для этого необходим защищенный канал. Как правило, для обмена ключами использовался специальный курьер. Важным преимуществом криптографии с открытым ключом над симметричными криптосистемами является обмен ключами между удаленными пользователями без применения защищенного канала. Первая практичная схема такого обмена была предложена Диффи и Хеллманом и стала называться протоколом обмена экспоненциальными ключами Диффи–Хеллмана [98].

Для начала пользователи, Алиса и Боб, договариваются использовать конечное поле $\mathbb{F}_q$ и элемент $g \in \mathbb{F}_q$, порождающий группу, имеющую большой порядок. Для простоты будем рассматривать поле $\mathbb{F}_p$, в котором число p является большим простым числом. Стороны могут проверить простоту числа p, используя алгоритм 4.5, позволяющий создать число p, для которого известно полное разложение числа $p-1$ на множители. Затем, используя алгоритм 5.1, Алиса и Боб могут найти элемент g, порождающий группу $\mathbb{F}_p^*$. По теореме 5.11 каждое число из интервала $[1, p)$ можно представить в виде $g^x \pmod p$, где x — некоторое число. Теперь числа p и q можно использовать в качестве общих исходных данных в основном варианте протокола обмена ключами Диффи–Хеллмана.

Протокол 8.1. Протокол обмена ключами Диффи–Хеллмана

ОБЩИЕ ИСХОДНЫЕ ДАННЫЕ:

(p, g) : p — большое простое число,
g — порождающий элемент группы $\mathbb{F}_p^*$.

РЕЗУЛЬТАТ:

Элемент группы $\mathbb{F}_p^*$, разделенный между Алисой и Бобом.

1. Алиса генерирует элемент $a \in_U [1, p-1)$, вычисляет число $g_a \leftarrow g^a \pmod p$ и посылает его Бобу.
2. Боб генерирует элемент $b \in_U [1, p-1)$, вычисляет число $g_b \leftarrow g^b \pmod p$ и посылает его Алисе.
3. Алиса вычисляет значение $k \leftarrow g_b^a \pmod p$.
4. Боб вычисляет число $k \leftarrow g_a^b \pmod p$.

Из протокола 8.1 легко видеть, что значение, вычисляемое Алисой, равно

$$k = g^{ba} (\mathrm{mod}\, p),$$

а значение, вычисляемое Бобом, —

$$k = g^{ab} (\mathrm{mod}\, p).$$

Заметим, что, поскольку $ab \equiv ba (\mathrm{mod}\, p-1)$, обе стороны вычисляют одно и то же значение. Это обеспечивает разделение ключа между двумя сторонами.

Таким образом, числа p и g можно разослать всем участникам системы.

Пример 8.1. Допустим, что $p = 43$. Применяя алгоритм 5.1, находим, что число 3 является первообразным корнем по модулю 43. Распределим между Алисой и Бобом открытую пару $(p, g) = (43, 3)$. Для того чтобы обменяться секретными ключами, Алиса генерирует случайную секретную степень, равную 8, и посылает Бобу число $3^8 \equiv 25 (\mathrm{mod}\, 43)$. Боб генерирует секретное число 37 и посылает Алисе число $3^{37} \equiv 20 (\mathrm{mod}\, 43)$. Секретный ключ, согласованный Алисой и Бобом, равен

$$9 \equiv 20^8 \equiv 25^{37} (\mathrm{mod}\, 43).$$

□

Описание протокола Диффи–Хеллмана следует дополнить следующими замечаниями.

- Простое число p следует выбирать так, чтобы число $p-1$ имело достаточно большой простой множитель p'. Слова "достаточно большой" означают, что $p' > 2^{160}$. Необходимость этого условия обсуждается в разделе 8.4.
- Число g не обязано быть порождающим элементом группы $\mathbb{F}_p^*$. Необходимо лишь, чтобы оно было порождающим элементом ее подгруппы, имеющей достаточно большой порядок, например, подгруппы порядка p'. В этом случае Алиса и Боб должны проверить условия $g \neq 1$ и $g^{p'} \equiv 1 (\mathrm{mod}\, p)$. По этой причине число p' должно быть частью общих исходных данных.
- Алиса (соответственно Боб) должна проверить условие $g_b \neq 1$ (соответственно $g_a \neq 1$). Это условие гарантирует, что для выбранной ею степени, принадлежащей интервалу $(1, p')$, разделенный ключ g^{ab} будет элементом подгруппы порядка p' группы $\mathbb{F}_p$, т.е. элементом достаточно большой подгруппы.
- По окончании протокола Алиса (соответственно Боб) должна стереть свою степень a (соответственно степень b). Этим они обеспечивают **заблаговременную секретность** (forward secrecy) ключа g^{ab}. Это свойство обсуждается в разделах 8.15 и 11.6.1.

8.3.1 Атака "человек посередине"

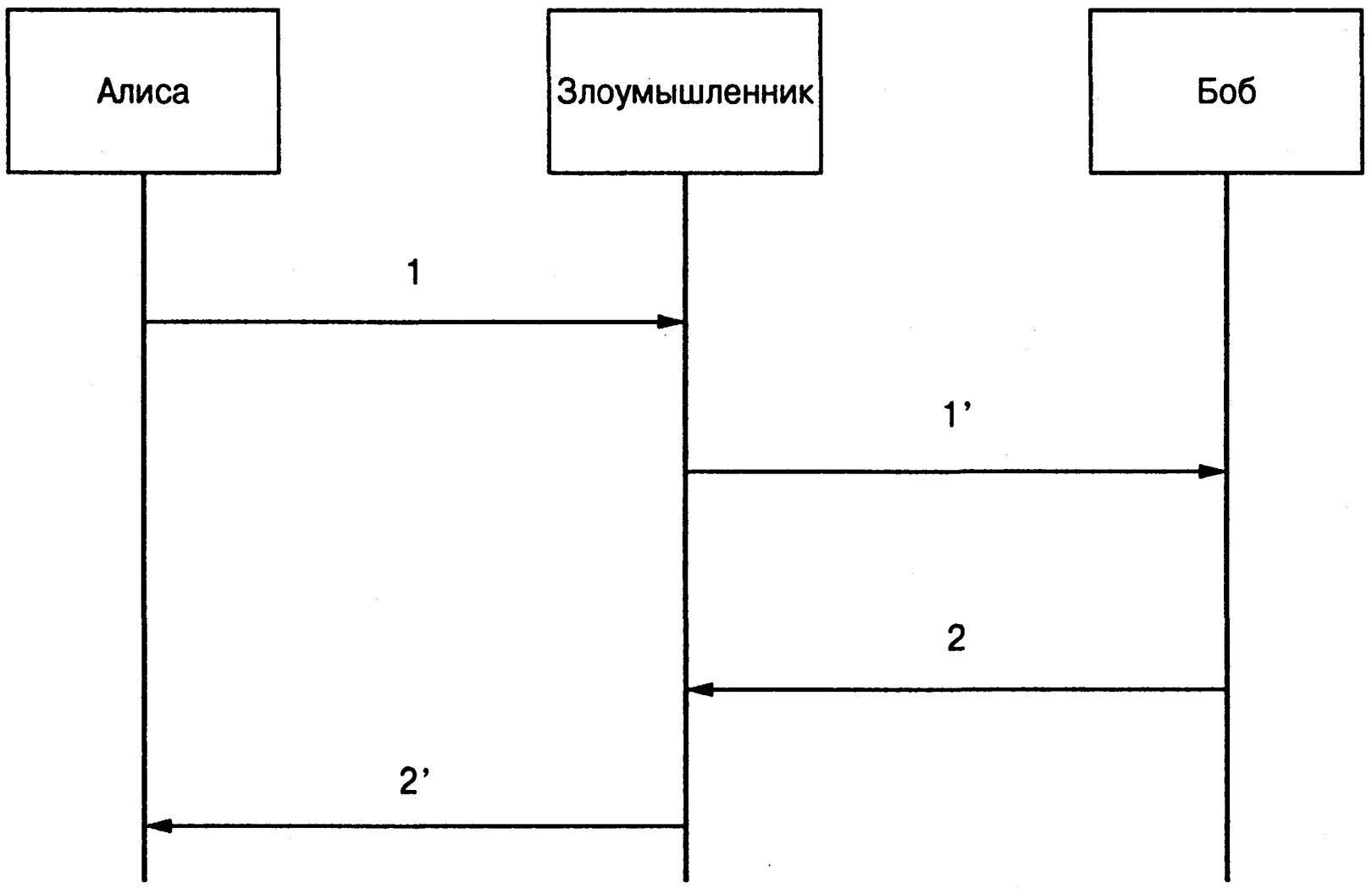
Следует иметь в виду, что протокол обмена ключами Диффи–Хеллмана не обеспечивает аутентичности согласованного ключа. Активный противник, внед-

---

Атака 8.1. Атака "человек посередине" на протокол обмена ключами Диффи–Хеллмана

ОБЩИЕ ИСХОДНЫЕ ДАННЫЕ:

Как в протоколе 8.1.



1. Алиса генерирует элемент $a \in_U [1, p-1)$, вычисляет значение $g_a \leftarrow g^a \pmod p$ и посылает его Злоумышленнику ("Бобу").

1'. Злоумышленник ("Боб") вычисляет значение $g_m \leftarrow g^m \pmod p$, где $m \in \in [1, p-1)$ и посылает его Бобу

2. Боб генерирует элемент $b \in_U [1, p-1)$, вычисляет значение $g_b \leftarrow g^b \pmod p$ и посылает его Злоумышленнику ("Алисе").

2'. Злоумышленник ("Алиса") посылает Алисе число g_m.

3. Алиса вычисляет значение $k_1 \leftarrow g_m^b \pmod p$. (* Этот ключ распределен между Алисой и Злоумышленником, поскольку Злоумышленник может вычислить значение $k_1 \leftarrow g_a^m \pmod p$. *)

4. Боб вычисляет значение $k_2 \leftarrow g_m^b \pmod p$. (* Этот ключ распределен между Бобом и Злоумышленником, поскольку Злоумышленник может вычислить значение $k_2 \leftarrow g_m^b \pmod p$. *)

ренный в канал связи между Алисой и Бобом, может манипулировать сообщениями протокола и начать **атаку "человек посередине"** (man-in-the-middle attack).

В ходе этой атаки Злоумышленник перехватывает и блокирует первое сообщение Алисы Бобу, т.е. число g_a, маскируется под Алису и посылает Бобу следующее сообщение.

Злоумышленник ("Алиса") — Бобу: $g_m \left(\stackrel{\text{def}}{=} g^m\right) (\mathrm{mod}\, p)$.

(Напоминаем соглашения, принятые в разделе 2.6.2.) Боб, следуя инструкциям протокола, возвращает Злоумышленнику ("Алисе") число g_b. Это число снова перехватывается и блокируется Злоумышленником. Теперь Злоумышленник и Боб согласовали между собой ключ $g^{bm} (\mathrm{mod}\, p)$, хотя Боб считает, что он разделил этот ключ с Алисой.

Аналогично Злоумышленник, имитируя Боба, может согласовать другой ключ с Алисой (т.е. число $g^{am} (\mathrm{mod}\, p)$). Впоследствии Злоумышленник может использовать оба ключа для чтения и подмены "секретных" сообщений, которыми обмениваются Алиса и Боб, или поочередно имитировать этих пользователей.

Атака на протокол обмена ключами Диффи–Хеллмана вполне реальна, поскольку этот протокол не предусматривает проверки аутентичности источника сообщений. Для того чтобы ключи были согласованы только между Алисой и Бобом, обе стороны должны быть уверены друг в друге. Способы аутентификации рассматриваются в главе 11. В этой же главе (раздел 11.6) описан метод безопасного применения протокола обмена ключами Диффи–Хеллмана.

8.4 Задача Диффи–Хеллмана и задача дискретного логарифмирования

Стойкость разделенного ключа в протоколе Диффи–Хеллмана обеспечивается вычислением значенияе $g^{ab} (\mathrm{mod}\, p)$ по заданным числам g_a и g_b. Эта задача называется **вычислительной проблемой Диффи–Хеллмана** (CDH problem — Diffie-Hellman problem).

Определение 8.1 (Вычислительная проблема Диффи–Хеллмана) *(в конечном поле).*

ИСХОДНЫЕ ДАННЫЕ:

$desc(\mathbb{F}_q)$: описание конечного поля $\mathbb{F}_q$;
$g \in \mathbb{F}_q^*$ — порождающий элемент группы $\mathbb{F}_q^*$;
$g_a, g_b \in \mathbb{F}_q^*$, где $0 < a, b < q$.

РЕЗУЛЬТАТ:

g^{ab}.

Эта формулировка представляет собой общую постановку задачи в конечном поле $\mathbb{F}_q$. В протоколе обмена ключами Диффи–Хеллмана используется более специальный случай. Общая постановка приводится только для формализации задачи, хотя для большей ясности мы будем изучать лишь частный случай.

Если бы проблему Диффи–Хеллмана было легко решить, то значение $g^{ab} (\mathrm{mod}\, p)$ можно было бы найти, зная числа p, g, g_a и g_b, которые являются частью протокола. Предполагая, что противник обладает возможностями перехвата информации (раздел 2.3), следует предположить, что эти числа не являются для него секретом.

Проблема Диффи–Хеллмана опирается на сложность вычисления **дискретного логарифма** (discrete logarithm problem).

Определение 8.2 (Задача дискретного логарифмирования) *(в конечном поле).*
ИСХОДНЫЕ ДАННЫЕ:

$desc(\mathbb{F}_q)$: описание конечного поля $\mathbb{F}_q$;
$g \in \mathbb{F}_q^*$ — порождающий элемент группы $\mathbb{F}_q^*$;
$h \in_U \mathbb{F}_q^*$.

РЕЗУЛЬТАТ:

Уникальное число $a < q$, удовлетворяющее условию $h = g^a$.

Целое число a обозначается как $\log_g h$.

Дискретное логарифмирование аналогично обычному логарифмированию в поле действительных чисел. Однако в отличие от последней задачи, в которой решение является приближенным, задача о вычислении дискретного логарифма имеет *точное* решение.

Как указывалось в главе 4, в основе современной криптографии лежит теория вычислительной сложности. Это значит, что стойкость криптосистем с открытым ключом является *условной* и зависит от сложности решения некоторых задач. Проблема Диффи–Хеллмана и задача дискретного логарифмирования считаются трудноразрешимыми. Интуитивно ясно, что сложность решения этих задач зависит как от размера поля $\mathbb{F}_q$, так и от выбора параметров (открытого параметра g и секретных чисел a и b). Очевидно, что небольшие варианты этих задач являются разрешимыми. В свое время мы покажем, что существуют другие варианты этих задач, которые относительно легко решить. Таким образом, точное описание понятия сложности решения задачи должно учитывать как размер задачи, так и выбор ее варианта. Сведения из теории сложности, приведенные в главе 4, позволяют сформулировать точные предположения о неразрешимости проблемы Диффи–Хеллмана и задачи дискретного логарифмирования. Рекомендуем читателям вспомнить некоторые понятия и обозначения, используемые в теории вычислительной сложности (в частности, “1^k”, “вероятностный полиномиальный алгоритм” и “пренебрежимо малая величина”).

Предположение 8.1 (Условия неразрешимости проблемы Диффи–Хеллмана). *Алгоритмом решения проблемы Диффи–Хеллмана называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[g^{ab} \leftarrow \mathcal{A}\left(\mathrm{desc}\left(\mathbb{F}_q\right), g, g^a, g^b\right)\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.1.

Пусть $\mathcal{IG}$ — генератор вариантов, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — 1) $\mathrm{desc}(\mathbb{F}_q)$, где $|q| = k$, и 2) порождающий элемент $g \in \mathbb{F}_q^*$.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости проблемы Диффи–Хеллмана, если для вариантов $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по сравнению со всеми достаточно большими числами k.

Предположение 8.2 (Условия неразрешимости задачи дискретного логарифмирования). *Алгоритмом решения задачи дискретного логарифмирования называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[\log_g h \leftarrow \mathcal{A}\left(\mathrm{desc}\left(\mathbb{F}_q\right), g, h\right)\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.2.

Пусть $\mathcal{IG}$ — генератор вариантов, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — 1) $\mathrm{desc}(\mathbb{F}_q)$, где $|q| = k$, 2) порождающий элемент $g \in \mathbb{F}_q^*$ и 3) элемент $h \in \mathbb{F}_q^*$.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости задачи дискретного логарифмирования, если для вариантов $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по отношению ко всем достаточно большим числам k.

Иначе говоря, здесь предполагается, что почти все достаточно большие варианты указанных задач в конечных полях не имеют эффективного алгоритма решения. Доля слабых вариантов этих задач, поддающихся решению, пренебрежимо мала.

И все же эти два предположения требуют уточнения. Для начала сделаем несколько формальных утверждений.

Примечание 8.1.

1. *В предположениях 8.1 и 8.2 соответствующие вероятностные пространства должны учитывать 1) пространство вариантов, из которых извлекаются произвольные конечные поля и элементы (важность этого условия обсуждается в разделе 8.4.1) и 2) пространство случайных операций в эффективном алгоритме. Второе условие необходимо, поскольку в множество*

полиномиальных, или эффективных, алгоритмов входят и рандомизированные алгоритмы (см. определение 4.6 в разделе 4.4.6).

2. *Число k в обоих формулировках называется* ***параметром безопасности*** *(security parameter), а $\mathcal{IG}(1^k)$ — случайный вариант конечного поля и элементов. Изучая вероятностное генерирование простых чисел (раздел 4.4.6.1) и конструкцию поля (раздел 5.4), мы убедились, что вариант $\mathcal{IG}(1^k)$ действительно можно решить за время, полиномиально зависящее от числа k. Считается общепризнанным, что число $k = 1024$ является нижней границей параметра безопасности для задачи дискретного логарифмирования в конечных полях. Эта нижняя граница установлена, исходя из существования субэкспоненциального алгоритма решения этой задачи (индексное исчисление). Определением субэкспоненциальной сложности является выражение (8.4.2). Для числа $|q| = 1024$ это выражение равно 2^{80}. По этой причине в качестве нижней границы используется число $k = 1024$. Таким образом, выражение "для всех достаточно больших чисел k" означает, что числа k больше* 1024.

3. *Предположение о невозможности дискретного логарифмирования означает, что функция*

$$g^x : \mathbb{Z}_q \mapsto \mathbb{F}_q^* \tag{8.4.1}$$

является однонаправленной. Считается, что это условие действительно выполняется (описание гипотезы $\mathcal{P} \neq \mathcal{NP}$ см. в разделе 4.5), т.е. однонаправленная функция существует.

4. *В настоящее время неизвестно, является ли функция (8.4.1) функцией с секретом (см. свойство 8.1 в разделе 8.1). Иначе говоря, никому не удалось внедрить в эту функцию секретную информацию, позволяющую эффективно вычислить обратную функцию (т.е. разработать эффективный метод, позволяющий по числу g^x найти число x, используя секретную информацию). Однако, если эта функция использует составные модули (оставаясь однонаправленной), она становится функцией с секретом, в которой роль секрета играет разложение модуля на простые множители. Технические детали читатели найдут в книгах [224, 228, 229].* □

Сформулированные выше предположения, по сути, означают, что "не существует полиномиальных по параметру k алгоритмов решения этих задач". Однако к этому утверждению следует относиться с большой осторожностью. Если полиномиальный алгоритм $\mathrm{poly}(k)$ существует, его время работы не превышает k^n, где n — некоторое целое число. С другой стороны, известно, что существует субэкспоненциальный алгоритм дискретного логарифмирования, время работы которого определяется выражением

$$\mathrm{sub\_exp}(q) = \exp(c(\log q)^{1/3}(\log\log q)^{2/3}), \tag{8.4.2}$$

где c — небольшая константа (например, $c < 2$). Сравнивая выражения "не существует полиномиального алгоритма дискретного логарифмирования $\mathrm{poly}(k)$" и "существует субэкспоненциальный алгоритм дискретного логарфмирования $\mathrm{sub\_exp}(q)$", приходим к выводу, что число k^n чрезвычайно мало по сравнению с числом $\mathrm{sub\_exp}(k\log 2)$ (если $k = |q| = \log_2 q$, то $q = k\log 2$). Однако выражение "чрезвычайно мало" оказывается истинным, только если число n фиксировано, а число k (будучи функцией числа n) — достаточно велико. Разберемся в этом подробнее.

Допустим, что число k не является достаточно большим. Возьмем натуральный логарифм от выражений $\mathrm{poly}(k)$ и $\mathrm{sub\_exp}(k\log 2)$ и сравним полученные результаты:

$$n(\log k)^{1/3} \text{ и } c'k^{1/3},$$

где $c' = c(\log 2)^{1/3} < c$. Теперь ясно, что, если число n сравнимо с числом $c'k^{1/3}$, субэкспоненциальный алгоритм дискретного логарифмирования работает быстрее, чем гипотетически "несуществующий полиномиальный алгоритм". Выражение "не существует полиномиального алгоритма дискретного логарифмирования $\mathrm{poly}(k)$" приобретает реальный смысл, только если число k не ограничено (и, следовательно, может быть "достаточно большим", если число n фиксировано). В действительности число k не может быть неограниченным. В частности, при $k = 1024$ (общепринятая нижняя граница параметра безопасности) и $c < 2$ существует полиномиальный алгоритм дискретного логарифмирования $\mathrm{poly}(k)$, время работы которого является полиномом 9-й степени от параметра k (см. пример 8.4).

До сих пор мы использовали *асимптотическую* интерпретацию выражения "не существует полиномиального алгоритма дискретного логарифмирования $\mathrm{poly}(k)$": число k считалось неограниченным и достаточно большим. В действительности число k *должно быть ограничено*, а значит, алгоритм $\mathrm{poly}(k)$ *существует*. Несмотря на это, можно установить настолько большую границу для числа k, что полиномиальный алгоритм будет работать немыслимо долго. Считается, что число $k = 1024$ вполне удовлетворяет этому условию.

В остальной части книги будет использоваться именно асимптотическое толкование выражения "не существует полиномиального алгоритма решения $\mathrm{poly}(k)$".

В заключение рассмотрим взаимосвязь между вычислительной проблемой Диффи–Хеллмана и задачей дискретного логарифмирования.

Заметим, что доступность чисел $a = \log_g g_1$ или $b = \log_g g_2$ позволяет найти числа

$$g^{ab} = g_1^b = g_2^b.$$

Иначе говоря, если существует эффективный алгоритм, решающий вычислительную проблему Диффи–Хеллмана, значит, существует эффективный алгоритм дискретного логарифмирования. Следовательно, если не выполняются условия невоз-

можности дискретного логарифмирования, то не выполняются и условия неразрешимости вычислительной проблемы Диффи–Хеллмана. Таким образом, вычислительная проблема Диффи–Хеллмана проще задачи дискретного логарифмирования, т.е. условия неразрешимости вычислительной проблемы Диффи–Хеллмана строже, чем условия неразрешимости задачи дискретного логарифмирования. Обратное утверждение еще не доказано.

Могут ли выполняться условия невозможности дискретного логарифмирования, если условия о неразрешимости вычислительной проблемы Диффи–Хеллмана не выполняются?

Маурер (Maurer) и Вольф (Wolf) привели убедительные эвристические аргументы, обосновывающие взаимозависимость этих задач. Они считают весьма вероятным, что эти две задачи эквивалентны [190].

8.4.1 Произвольность вариантов и условия неразрешимости

Следует подчеркнуть, что в условиях невозможности дискретного логарифмирования варианты должны быть произвольными. Рассмотрим группу $\mathbb{F}_p^*$, где p — простое число, состоящее из k бит, и попробуем найти число a, удовлетворяющее условию $h \equiv g^a (\mathrm{mod}\, p)$.

Известно, что число a является элементом поля $\mathbb{Z}_{p-1}$. Если $p - 1 = q_1 q_2 \ldots q_\ell$, где каждый множитель q_i мал (т.е. q_i не превосходит полином от параметра k при $i = 1, 2, \ldots, \ell$), то задача дискретного логарифмирования сводится к вычислению значений $a_i \equiv a (\mathrm{mod}\, q_i)$ по числам $h^{(p-1)/q_i} (\mathrm{mod}\, p)$, только теперь числа a_i малы и могут быть найдены за полиномиальное время, зависящее от параметра k. После определения чисел $a_1, a_2, \ldots, a_\ell$ число a находится по китайской теореме об остатках (теорема 6.7). Эта идея лежит в основе полиномиального алгоритма Полига (Pohlig) и Хеллмана [231], позволяющего решить задачу дискретного логарифмирования при условии, что число $p - 1$ не имеет большого простого множителя. Очевидно, что если каждый простой множитель числа $p - 1$ ограничен полиномом, зависящим от параметра k, то время выполнения алгоритма Полига–Хеллмана полиномиально зависит от параметра k.

Простое число p, такое что число $p - 1$ не имеет большого простого множителя, называется **гладким** (smooth prime). Правда, иногда в этом же смысле употребляется выражение "число $p - 1$ является гладким". Стандартный способ избежать гладких простых чисел заключается в конструировании такого простого числа p, чтобы $p - 1$ делилось на другое большое простое число p'. По теореме 5.2.2 циклическая группа $\mathbb{F}_p^*$ содержит единственную подгруппу, имеющую порядок p'. Если число p' сделать открытым, пользователи протокола обмена ключами Диффи–Хеллмана могут быть уверенными в том, что этот протокол ра-

ботает в этой большой подгруппе. Все что нужно для этого — найти элемент $g \in \mathbb{F}_p^*$, удовлетворяющий условию

$$g^{(p-1)/p'} \not\equiv 1 (\mathrm{mod}\, p).$$

Элемент g порождает группу, имеющую простой порядок p'. В качестве общих исходных данных протокол обмена ключами Диффи–Хеллмана должен использовать числа (p, p', g), удовлетворяющие указанным выше свойствам. Считается, что размер простого числа p' должен быть не меньше 160 бит, т.е. $p' > 2^{160}$ (см. раздел 10.4.8.1).

Проблема Диффи–Хеллмана и задача дискретного логарифмирования считаются неразрешимыми в конечной абелевой группе большого порядка, например, в подгруппе конечного поля, имеющей большой простой порядок, или в группе точек эллиптической кривой, определенной над конечным полем (конструкция групп рассматривается в разделе 5.5, а задача дискретного логарифмирования на эллиптических кривых — в разделе 5.5.3). Таким образом, протокол обмена ключами Диффи–Хеллмана правильно работает в этих группах.

Существует несколько эффективных субэкспоненциальных алгоритмов дискретного логарифмирования при условии, что вычисляемое значение невелико — например, λ-метод Полларда, описанный в разделе 3.6.1. Вычисление небольших дискретных логарифмов применяется во многих криптографических протоколах.

Проблема Диффи–Хеллмана исследуется весьма активно. Большой список литературы, посвященной этой теме, содержится в обзоре Одлыжко (Odlyzko) [221].

8.5 Криптосистема RSA (учебный вариант)

Наиболее известной криптосистемой с открытым ключом является алгоритм RSA, названный по первым буквам фамилий своих изобретателей — Ривеста (Rivest), Шамира (Samir) и Адлемана (Adleman) [246]. Криптосистема RSA — первая практическая реализация криптографии с открытым ключом на основе понятия однонаправленной функции с секретом, предложенного Диффи и Хеллманом [97, 98].

Криптосистема RSA описывается алгоритмом 8.1. Отметим, что этот алгоритм носит учебный характер.

Покажем, что система, описанная алгоритмом 8.1, действительно является криптографической, т.е. процедура расшифровки, выполненная Алисой, действительно восстанавливает исходный текст, зашифрованный Бобом.

Из определения операций модулярной арифметики (определение 4.4 из раздела 4.3.2.5) следует, что сравнение $ed \equiv 1 (\mathrm{mod}\, \phi(N))$ в алгоритме 8.1 эквивалентно уравнению

$$ed = 1 + k\phi(N),$$

Алгоритм 8.1. Криптосистема RSA

Ключ

Для того чтобы установить ключ, Алисе необходимо выполнить следующие операции.

1. Выбрать два случайных простых числа p и q, удовлетворяющих условию $|p| \approx |q|$.
   (* Для этого можно воспользоваться методом Монте-Карло, описанным алгоритмом 4.7. *)
2. Вычислить $N = pq$.
3. Вычислить $\phi(N) = (p-1)(q-1)$.
4. Выбрать случайное целое число $e < \phi(N)$, удовлетворяющее условию $\gcd(e, \phi(N)) = 1$, и найти целое число d, такое что

$$ed \equiv 1 (\mathrm{mod}\, \phi(N)).$$

   (* Поскольку $\gcd(e, \phi(N)) = 1$, это уравнение имеет решение d, которое можно найти с помощью расширенного алгоритма Евклида (алгоритм 4.2). *)
5. Использовать пару (N, e) в качестве параметров открытого ключа, тщательно уничтожить числа p, q и $\phi(N)$ и запомнить число d в качестве закрытого ключа.

Шифрование

Для того чтобы переслать Алисе секретное сообщение, имеющее длину $m < N$, Боб создает зашифрованный текст c.

$$c \leftarrow m^e (\mathrm{mod}\, N).$$

(* С точки зрения Боба, пространство исходных сообщений представляет собой множество всех положительных чисел, которые меньше числа N, хотя на самом деле этим пространством является группа $\mathbb{Z}_N^*$. *)

Расшифровка

Для того чтобы расшифровать зашифрованный текст c, Алиса вычисляет формулу

$$m \leftarrow c^d (\mathrm{mod}\, N).$$

где k — некоторое целое число. Следовательно, значение, вычисленное Алисой в результате выполнения процедуры расшифровки, определяется по следующей

формуле.

$$c^d \equiv m^{ed} \equiv m^{1+k\phi(N)} \equiv m \cdot m^{k\phi(N)} (\mathrm{mod}\, N). \qquad (8.5.1)$$

Следует отметить, что неравенство $m < N$ практически всегда означает, что $m \in \mathbb{Z}_N^*$ (т.е. почти все числа, которые меньше числа N, принадлежат мультипликативной группе целых чисел, взаимно простых с числом N). Условие $m \in \mathbb{Z}_N^*$ нарушается, если $m = up$ или $m = vq$, где $u < q$ и $v < p$. В этих ситуациях Боб может разложить число N на простые множители, вычислив значение $\gcd(m, N)$. Предполагая, что это является трудноразрешимой задачей (условия, при которых факторизация числа представляет собой трудноразрешимую задачу, будут сформулированы позднее), можно предположить, что любое сообщение $m < N$, созданное Бобом, удовлетворяет условию $m \in \mathbb{Z}_N^*$.

Если $m \in \mathbb{Z}_N^*$, то по теореме Лагранжа (следствие 5.2)

$$\mathrm{ord}_N(m) \mid \#\mathbb{Z}_N^* = \phi(N).$$

Это утверждение справедливо для всех $m \in \mathbb{Z}_N^*$. В соответствии с определением порядка элемента группы (определение 5.9 в разделе 5.2.2) это означает, что для всех $m \in \mathbb{Z}_N^*$ выполняется условие

$$m^{\phi(N)} \equiv 1 (\mathrm{mod}\, N).$$

Отсюда следует, что

$$m^{k\phi(N)} \equiv (m^{\phi(N)})^k \equiv 1 (\mathrm{mod}\, N)$$

для любого целого числа k. Итак, величина в формуле (8.5.1) действительно равна числу m.

Пример 8.2. Допустим, что Алиса выбрала числа $N = 7 \times 13 = 91$ и $e = 5$. Тогда $\phi(N) = 6 \times 12 = 72$. Применяя алгоритм 4.2 к паре $(a, b) = (72, 5)$, Алиса получает следующее число.

$$72 \times (-2) + 5 \times 29 = 1.$$

Иначе говоря, $5 \times 29 \equiv 1 (\mathrm{mod}\, 72)$. Следовательно, секретным показателем степени, используемым Алисой при шифровании, является число 29. Алиса устанавливает пару $(N, e) = (91, 5)$ в качестве параметров открытого ключа криптосистемы RSA.

Допустим, что Боб шифрует исходное сообщение $m = 3$, используя формулу

$$c = 3^5 = 243 \equiv 61 (\mathrm{mod}\, 91).$$

Зашифрованное сообщение представляет собой число 61.

Для того чтобы расшифровать сообщение 61, Алиса вычисляет значение

$$61^{29} \equiv 3 (\mathrm{mod}\, 91).$$

□

8.6 Взлом криптосистем с открытым ключом с помощью криптоанализа

Можно сказать, что "криптосистема X является стойкой по отношению к атаке Y, но нестойкой по отношению к атаке Z", т.е. стойкость криптосистемы зависит от разновидности атаки. Активные атаки можно разделить на три вида.

Определение 8.3 (Активные атаки на криптосистемы).

Атака на основе подобранного открытого текста *(chosen-plaintext attack — CPA). Атакующий выбирает исходные сообщения и передает их шифровальщику для получения зашифрованных текстов. Задача атакующего — взломать криптосистему, используя полученные пары открытых и зашифрованных текстов.*

Атака на основе подобранного зашифрованного текста *(chosen-ciphertext attack — CCA). Атакующий выбирает зашифрованные сообщения и передает их на расшифровку для получения исходных сообщений. Цель атакующего — взломать криптосистему, используя полученные пары открытых и зашифрованных текстов. Атакующий достигает успеха, если он раскрывает ключ и способен в дальнейшем извлекать секретную информацию из зашифрованного текста, не прибегая к посторонней помощи.*

Атака на основе адаптивно подобранного зашифрованного текста *(adaptive chosen-ciphertext attack — CCA2). Это — разновидность атаки CCA, в которой услуги расшифровки доступны для всех зашифрованных текстов, за исключением заданного.*

Эти атаки можно описать следующими сценариями.

- В атаке на основе подобранного открытого текста атакующий владеет блоком шифрования.
- В атаке на основе подобранного зашифрованного текста атакующий имеет ограниченный доступ к блоку расшифровки: после нескольких попыток доступ к блоку закрывается, и расшифровку требуемого текста атакующий должен выполнять без его помощи.
- В атаке на основе адаптивно подобранного зашифрованного текста атакующий владеет блоком расшифровки сколь угодно долго, однако, как и в предыдущем случае, расшифровка требуемого текста должна выполняться без его помощи. Это условие вполне естественно — иначе атакующему незачем взламывать криптосистему.

Все эти атаки основаны на предположении, что атакующему не известен криптографический ключ.

Атаки CPA и CCA изначально предназначались для активного криптоанализа криптосистем с секретным ключом. Целью этого криптоанализа являлся взлом криптосистемы с помощью пар открытых и зашифрованных сообщений, получаемых в ходе атаки [284]. Затем они были адаптированы для криптоанализа криптосистем с открытым ключом. Следует отметить три особенности криптосистем с открытым ключом.

- Услуги шифрования в криптосистеме с открытым ключом доступны любому желающему, поскольку владение *открытым ключом* обеспечивает полный контроль над алгоритмом шифрования. Иначе говоря, против криптосистем с открытым ключом всегда можно организовать атаку на основе подобранного открытого текста. Итак, любую атаку на криптосистему с открытым ключом, в которой не используется блок расшифровки, можно назвать атакой CPA. Очевидно, что любая криптосистема с открытым ключом должна быть устойчивой к атакам на основе подобранного открытого текста, иначе от нее мало пользы.
- Как правило, криптосистемы с открытым ключом имеют стройную алгебраическую структуру, обладающую свойствами замкнутости, ассоциативности, гомоморфизма и т.п. (см. главу 5). Атакующий может использовать эти свойства и взломать зашифрованный текст с помощью хитроумных вычислений. Если атакующий имеет доступ к блоку расшифровки, его вычисления могут дать представление об исходном тексте и даже взломать всю криптосистему. Следовательно, криптосистемы с открытым ключом особенно уязвимы для атак CCA и CCA2. В главе показано, что все учебные криптосистемы с открытым ключом уязвимы для этих атак. Исходя из этого можно сформулировать следующее правило, основанное на свойстве 8.2.2: владелец открытого ключа никому не должен предоставлять услуги расшифровки. Это условие должно выполняться во всех криптосистемах, описанных в главе. В главе 14 будут рассмотрены более строгие криптосистемы с открытым ключом, не требующие от пользователей особой осторожности.
- На первый взгляд, атака CCA слишком ограничивает возможности атакующего. В приложениях пользователь, подвергающийся атаке (т.е. пользователь, к которому обратились за расшифровкой сообщения), на самом деле не знает об этом. Следовательно, пользователю не известно, когда он должен прекратить расшифровку. Как правило, предполагается, что пользователь слишком наивен и не предполагает о существовании атакующего, а значит, должен предоставлять услуги по расшифровке сообщений *постоянно*. С другой стороны, любая криптосистема с открытым ключом должна быть устойчивой к атакам CPA, поскольку атакующий может сам зашифровывать подобранные открытые сообщения. По этой причине, в основном, мы будем рассматривать средства защиты от атаки CCA2.

8.7 Задача RSA

Средства защиты криптосистемы RSA от атак CPA основаны на сложности вычисления корня e-й степени шифрованного текста c по составному целочисленному модулю n. Это — так называемая **задача RSA** (RSA problem).

Определение 8.4 (Задача RSA).
ИСХОДНЫЕ ДАННЫЕ:

$N = pq$, где p и q — простые числа;
e: целое число, удовлетворяющее условию $\gcd(e, (p-1)(q-1)) = 1$.
$c \in \mathbb{Z}_N^*$.

РЕЗУЛЬТАТ:

Единственное целое число $m \in \mathbb{Z}_N^*$, удовлетворяющее условию $m^e \equiv c (\mathrm{mod}\, N)$.

Как и во всех других задачах, обеспечивающих стойкость криптосистем с открытым ключом, сложность задачи RSA зависит от сложности поиска правильных параметров.

Предположение 8.3 (Условия неразрешимости задачи RSA). *Алгоритмом решения задачи RSA называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[m \leftarrow \mathcal{A}\left(N, e, m^e (\mathrm{mod}\, N)\right)\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.4.

Пусть $\mathcal{IG}$ — генератор вариантов задачи RSA, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — 1) $2k$-битовый модуль $N = pq$, где p и q — два разных равномерно распределенных случайных простых числа, состоящих из k бит, и 2) $e \in \mathbb{Z}^*_{(p-1)(q-1)}$.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости задачи RSA, если для вариантов $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по отношению ко всем достаточно большим числам k.

Аналогично замечанию 8.1.3 (в разделе 8.4) можно утверждать, что выполнение условий неразрешимости задачи RSA означает существование однонаправленной хэш-функции. Как и в замечании 8.1.4, из этого следует, что она является функцией с секретом: существует эффективная процедура, обратная разложению модуля на простые множители.

Следует отметить, что вероятностное пространство в этом условии включает в себя пространство вариантов, пространство исходных сообщений и пространство случайных операций рандомизированного алгоритма решения задачи RSA.

Кроме того, в условии неразрешимости задачи RSA на вход предполагаемого алгоритма поступает показатель степени e, используемый при шифровании. Это ясно очерчивает цель атаки: решить задачу RSA при заданном показателе степени e. Существует альтернативная постановка задачи RSA, которая называется **сильной задачей RSA** (strong RSA problem) [85]. Ее цель — найти некоторый нечетный показатель степени $e > 1$ и решить задачу RSA для этого показателя. Очевидно, что решить сильную задачу RSA легче, чем обычную задачу RSA, в которой показатель степени e фиксирован. Считается, что сильная задача RSA является трудноразрешимой. **Условие трудной разрешимости сильной задачи RSA** легло в основу некоторых алгоритмов шифрования и криптографических протоколов.

Очевидно, что если для параметров открытого ключа (N, e) выполняется условие $m < N^{1/e}$, то шифрование сообщения $c = m^e (\mathrm{mod}\, N)$ не использует операции приведения по модулю, и, следовательно, значение m можно эффективно вычислить, извлекая корень e-й степени из целых чисел. Это одна из причин, по которой не следует выбирать вариант $e = 3$. В противном случае, если сообщение m зашифровано с помощью трех разных модулей: $c_i = m^3 (\mathrm{mod}\, N_i)$, где $i = 1, 2, 3$, то, поскольку модули являются попарно взаимно простыми числами, можно применить китайскую теорему об остатках (алгоритм 6.1) и вычислить значение $C = m^3 (\mathrm{mod}\, N_1 N_2 N_3)$. Теперь, поскольку $m < (N_1 N_2 N_3)^{1/3}$, возведение в степень выполняется точно так же, как и в пространстве целых чисел. Итак, расшифровка сообщения C сводится к извлечению кубического корня из целых чисел и может быть эффективно реализована (см. подсказку в упражнению 8.8).

Куперсмит (Coopersmith) [82] распространил этот тривиальный случай на более сложный: для $m' = m + t$, где число m известно, а число t — нет, но при этом выполняется неравенство $t < N^{1/e}$, значение t можно эффективно вычислить по заданному числу $c = m'^e (\mathrm{mod}\, N)$. Поскольку в приложениях часто встречаются ситуации, в которых часть исходного текста является известной (см. главу 15), считается, что при шифровании с помощью алгоритма RSA не следует применять очень маленькие показатели степени e. Как правило, в качестве показателя степени при шифровании используется величина $e = 2^{16} + 1 = 65537$, которая является простым числом. Этот показатель гарантирует достаточную эффективность шифрования и предотвращает атаки, использующие малые показатели степени.

Если показатель степени d, использующийся при расшифровке, невелик, алгоритм RSA устойчив к атакам на основе подобранного открытого текста. Винер (Wiener) изобрел метод, основанный на представлении числа e/N в виде непрерывной дроби. Этот алгоритм позволяет найти число d, если $d < N^{1/4}$ [298]. В дальнейшем этот результат был улучшен для чисел $d < N^{0,292}$ [50].

8.8 Разложение целых чисел на простые множители

Разрешимость задачи RSA зависит от сложности **разложения целого числа на простые множители** (integer factorization problem — IF-problem).

Определение 8.5 (Разложение целого числа на простые множители).
ИСХОДНЫЕ ДАННЫЕ:

N: нечетное составное целое число, имеющее по меньшей мере два разных простых множителя.

РЕЗУЛЬТАТ:

Простое число p, удовлетворяющее условию $p \mid N$.

Предположение 8.4 (Условие неразрешимости задачи о разложении целого числа на простые множители). *Алгоритмом решения задачи о разложении целого числа на простые множители называется вероятностный полиномиальный алгоритм A с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \text{Prob}\,[A(N) \text{ является делителем числа } N \text{ и } 1 < A(N) < N]\,,$$

где входные данные алгоритма A указаны в определении 8.5.

Пусть IG — генератор целых чисел, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — $2k$-битовый модуль $N = pq$, где p и q — два равномерно распределенных случайных простых нечетных числа, состоящих из k бит.

Будем говорить, что генератор IG удовлетворяет условиям неразрешимости задачи о разложении целого числа на простые множители, если для варианта $IG(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по отношению ко всем достаточно большим числам k.

Очевидно, что алгоритм решения задачи о разложении целого числа на простые множители одновременно является алгоритмом решения задачи RSA, поскольку Алиса расшифровывает текст, зашифрованный с помощью алгоритма RSA, вычисляя значение $d \equiv e^{-1}(\text{mod}(p-1)(q-1))$, т.е. используя информацию о факторизации числа N. Как и в задачах дискретного логарифмирования и Диффи–Хеллмана, обратное утверждение является недоказанным: выполняются ли условия неразрешимости задачи о разложении целого числа на множители, если условия неразрешимости задачи RSA не выполняются?

Как и в задаче Диффи–Хеллмана, слабым местом задачи о разложении целых чисел на простые множители являются гладкие простые числа N. Один из таких вариантов был продемонстрирован Поллардом, предложившим алгоритм эффективной факторизации под названием **$(p-1)$-алгоритм Полларда** [237]. В его

основе лежат следующие рассуждения. Пусть p — простой множитель числа N, а наибольший простой множитель числа $p-1$ ограничен величиной $B = \mathrm{Poly}(k)$, где $k = |N|$, а $\mathrm{Poly}(k)$ — полином, зависящий от аргумента k. (Число B называется **границей гладкости** числа $p-1$.) Сконструируем число

$$A = \prod_{\text{простые числа } r < B} r^{\lfloor \log N / \log r \rfloor}.$$

Условие $p-1 \mid A$ выполняется автоматически, поэтому из малой теоремы Ферма (теорема 6.10) следует, что $a^A \equiv 1 (\mathrm{mod}\, p)$ для любого целого числа a, удовлетворяющего условию $\gcd(a,p) = 1$. Если существует простой множитель q числа N, не равный множителю p и такой что $a^A \not\equiv 1 (\mathrm{mod}\, q)$, то существует целое число ℓ, не кратное числу q, такое что $a^A - 1 (\mathrm{mod}\, N) = \ell p$. Следовательно, число $\gcd(a^A - 1 (\mathrm{mod}\, N), N)$ должно быть собственным простым множителем числа N. Если $N = pq$, это число должно равняться числу p. Осталось показать, что размер числа A полиномиально зависит от числа k, а значит, число $a^A (\mathrm{mod}\, N)$ можно найти за полиномиальное время.

По теореме о простом числе [170] существует не более чем $B/\log B$ простых чисел, которые меньше, чем число B. Отсюда следует, что

$$A < B^{\lfloor \log N \rfloor \frac{B}{\log B}} < B^{k \frac{B}{\log B}},$$

т.е.

$$|A| < kB \log 2 < k\,\mathrm{Poly}(k).$$

Очевидно, что правая часть этого неравенства представляет собой полином, зависящий от аргумента k. Следовательно, для вычисления значения $a^A (\mathrm{mod}\, N)$ можно выполнить многократное умножение по модулю N (алгоритм 4.3), причем количество операций умножения полиномиально зависит от аргумента k. Отметим, что число A не обязательно конструировать явно. Число $a^A (\mathrm{mod}\, N)$ можно найти путем вычисления величины $a^{r^{\lfloor \log N)/\log r)\rfloor}} (\mathrm{mod}\, N)$ для всех простых чисел $r < B$.

Очень легко построить модуль $N = pq$ так, чтобы оценка гладкости чисел $p-1$ и $q-1$ была небольшой и неполиномиально зависела от $|N|$. В этом случае алгоритм RSA является стойким по отношению к данному методу факторизации. Можно начать с поиска больших простых чисел p' и q', таких что $p = 2p' + 1$ и $q = 2q' + 1$ также являются простыми. Такие простые числа называются **безопасными простыми числами** (safe prime), а модули алгоритма RSA, имеющие два безопасных простых множителя, называются **безопасными простыми модулями алгоритма RSA** (safe-prime RSA modulus). В настоящее время исследователи ведут дебаты о необходимости применения безопасных простых модулей в криптосистемах RSA. Противники их применения (см. работу [273]) утверждают, что модули алгоритма RSA должны быть как можно более случайными, и для

случайно выбранного простого числа p вероятность того, что число $p-1$ имеет большой простой множитель, является весьма большой. Однако корректность функционирования многих криптографических протоколов, основанных на сложности разложения целых чисел на множители, обеспечивается именно выбором безопасных простых модулей алгоритма RSA.

Хорошо известно, что частичная информация о простом множителе N позволяет разработать эффективный алгоритм факторизации числа N. Например, если $N = pq$, где p и q — простые числа, имеющие приблизительно одинаковый размер, знание половины битов числа p позволяет разложить число N за время, полиномиально зависящее от его размера [82].

При отсутствии априорной информации о простых множителях входного составного числа наилучшим алгоритмом факторизации является метод решета числового поля (number field sieve — NFS), временна́я сложность которого определяется по формуле (4.6.1). Таким образом, как и при выборе параметра безопасности задачи дискретного логарифмирования в конечном поле, для обеспечения надежности в качестве нижней границы размера модуля алгоритма RSA широко используется число 1024.

Эффективность метода решета числового поля, реализованного с помощью большого количества параллельных компьютеров, была продемонстрирована в начале 2000 года: сеть, состоящая из 9 000 рабочих станций, разложила на множители модуль RSA, состоящий из 512 бит (RSA-512 Challenge), после четырех месяцев работы [70] параллельного алгоритма.

Исследования в области целочисленной факторизации ведутся очень интенсивно. Обзор методов решения задачи RSA дан в работе Бонэ (Boneh) [48]. Достижения и перспективы этой области исследований читатели могут найти в главе 3 книги [198].

8.9 Уязвимость учебного алгоритма RSA

В разделе 8.1 указано, что рассматриваемый вариант алгоритма является учебным и описан в большинстве учебников по криптографии. Перейдем к изучению свойств, обусловливающих стойкость (или нестойкость) учебного алгоритма RSA.

При случайном ключе и случайном сообщении, упомянутых в определении 8.4 и предположении 8.3, утверждение о существовании эффективной атаки на основе подобранного открытого текста должно быть ложным.

Теорема 8.1. *Криптосистема RSA устойчива к атаке на основе подобранного открытого текста по принципу "все или ничего" тогда и только тогда, когда выполняются условия неразрешимости задачи RSA.* □

Принцип "все или ничего" объясняется при описании свойства 8.2.1. А применение атаки на основе подобранного открытого текста означает, что атакующий остается пассивным, как и предусмотрено свойством 8.2.2.

Однако такая стойкость имеет не очень высокую практическую ценность.

Во-первых, рассмотрим принцип "все или ничего". Слово "все" означает, что при расшифровке восстанавливается весь блок исходного сообщения: размеры сообщения и модуля совпадают. В приложениях это условие выполняется не всегда. В реальных системах исходный текст, как правило, содержит определенную часть открытой информации, известной атакующему. Учебный вариант алгоритма RSA не скрывает эту информацию. Например, если известно, что зашифрованный текст представляет собой число, не превосходящее 1 000 000 (например, секретное предложение цены или размер оклада), то, имея зашифрованное сообщение, атакующий может восстановить исходный текст менее чем за 1 000 000 попыток.

Как правило, исходный текст $m < N$ с ненулевой вероятностью можно восстановить, использовав только $\sqrt{m}$ попыток, если в распоряжении криптоаналитика есть память, имеющая размер $\sqrt{m}$. Этот факт установлен Бонэ, Жё (Joux) и Нгуеном (Nguen) [52]. Они исходили из того, что факторизация небольшого исходного сообщения не является трудноразрешимой задачей, и использовали мультипликативное свойство функции RSA:

$$(m_1 \times m_2)^e \equiv m_1^e \times m_2^e \equiv c_1 \times c_2 (\mathrm{mod}\, N). \tag{8.9.1}$$

Иначе говоря, факторизация исходного сообщения означает факторизацию соответствующего зашифрованного текста. Как правило, текст, зашифрованный с помощью алгоритма RSA, трудно разложить на простые множители из-за перемешивающего свойства функции шифрования, которое практически всегда приводит к тому, что размер зашифрованного текста равен размеру модуля. Однако из мультипликативного свойства следует, что если исходный текст поддается факторизации, то и зашифрованный текст — тоже. Это позволяет организовать атаку "встреча посередине" ("meet-in-the-middle" attack). Рассмотрим следующий пример.

Пример 8.3. Пусть $c = m^e (\mathrm{mod}\, N)$, причем Злоумышленнику известно число $m < 2^\ell$. Существует ненулевая вероятность того, что число m является составным и удовлетворяет условию

$$m = m_1 \cdot m_2, \text{ где } m_1, m_2 < 2^{\frac{\ell}{2}}. \tag{8.9.2}$$

Из мультипликативного свойства функции RSA следует, что

$$c = m_1^e \cdot m_2^e (\mathrm{mod}\, N). \tag{8.9.3}$$

Злоумышленник может создать упорядоченную базу данных

$$\left\{1^e, 2^2, 3^e, \ldots, \left(2^{\frac{\ell}{2}}\right)^e\right\} (\mathrm{mod}\, N)$$

и искать в ней число $c/i^e (\mathrm{mod}\, N)$, где $i = 1, 2, \ldots, 2^{\frac{\ell}{2}}$. Согласно формулам (8.9.2) и (8.9.3) поиск сводится к проверке условия

$$c/i^e \equiv j^e (\mathrm{mod}\, N)$$

и завершается за $2^{\frac{\ell}{2}}$ шагов, при выполнении которых вычисляется величина $i^e (\mathrm{mod}\, N)$. Поскольку Злоумышленнику известны числа i и j, он восстанавливает число $m = i \cdot j$.

Оценим затраты Злоумышленника. Размер базы данных равен $2^{\frac{\ell}{2}} \log N$ бит. Временны́е затраты складываются из следующих компонентов: сложность создания базы данных имеет порядок $O_B(2^{\frac{\ell}{2}} \log^3 N)$, сортировка — $O_B(\frac{\ell}{2} 2^{\frac{\ell}{2}})$, а поиск числа $j^e (\mathrm{mod}\, N)$ в упорядоченной базе — $O_B(2^{\frac{\ell}{2}} (\frac{\ell}{2} + \log^3 N))$. В последней оценке учитывается время, затраченное на возведение в степень по модулю и бинарный поиск (с помощью алгоритма 4.4). Итак, полная оценка временно́й сложности взлома имеет порядок $O_B(2^{\frac{\ell}{2}+1} (\frac{\ell}{2} + \log^3 N))$. Если Злоумышленнику доступна память, имеющая размер $2^{\frac{\ell}{2}} \log N$ бит, то временна́я сложность взлома намного меньше 2^ℓ. Эта атака по временно́й сложности сравнима с атакой по методу квадратного корня. □

Если размер исходного сообщения колеблется от 40 до 64 бит, то вероятность того, что исходный текст будет разложен на два одинаковых целых числа, изменяется от 18 до 50% [52].

Пример 8.4 (Реальный пример атаки 8.3). Представим себе сценарий, в котором ключ алгоритма DES, состоящий из 56 бит, зашифрован с помощью 1024-битового ключа учебного варианта RSA. Если ключ алгоритма DES случаен, его можно восстановить с ненулевой вероятностью, равной вероятности разложить ключ DES на два целых множителя длиной по 28 бит. Для этого понадобится память, имеющая размер 2^{38} бит (т.е. 32 гигабайт) и 2^{29} операций возведения в степень по модулю. Этим условиям вполне соответствует хороший персональный компьютер. В то же время непосредственный перебор ключей алгоритма DES требует выполнения 2^{56} операций возведения в степень по модулю. Это требование уже не под силу даже специализированному устройству. □

Теперь понятно, почему не следует применять учебный вариант алгоритма RSA для шифрования коротких ключей и паролей, длина которых не превышает 2^{64} бит. Что же произойдет, если в конкретном приложении нам понадобится

зашифровать с помощью алгоритма RSA небольшие числа, даже если размер сообщения равен одному биту? В этой ситуации следует применять методы шифрования, описанные в главе 15.

Следующий пример еще раз демонстрирует уязвимость учебного алгоритма RSA для атаки на основе подобранного открытого текста: против активной атаки этот алгоритм еще более беззащитен.

Пример 8.5. Допустим, что Злоумышленник частично контролирует блок шифрования, принадлежащий Алисе. Это условие вполне "разумно": если результат расшифровки зашифрованного текста, посланный Злоумышленником, окажется бессмысленным набором случайных цифр, Алиса должна вернуть Злоумышленнику исходный текст. "Разумность" этого условия объясняется двумя причинами.

1. "Случайный ответ на случайный запрос" — это вполне стандартный режим во многих криптографических протоколах, и, следовательно, пользователь должен выполнить определенную инструкцию "оклик–отзыв". Действительно, довольно *часто* криптографические протоколы допускают частичный контроль блока расшифровки со стороны пользователей. Например, протокол аутентификации с открытым ключом Нидхема–Шредера (протокол 2.5) функционирует именно так: Алиса обязана выполнить расшифровку зашифрованного текста, полученного от Боба.
2. В любом случае хочется надеяться, что расшифрованный текст, имеющий вид случайного набора цифр, не позволит атакующему извлечь какую-либо полезную информацию.

Предположим теперь, что Злоумышленник желает знать исходный текст, лежащий в основе зашифрованного текста $c \equiv m^e (\mathrm{mod}\, N)$, перехваченного им в ходе предыдущего сеанса связи между Алисой и кем-либо еще (но не с ним!). Злоумышленник извлекает случайное число $r \in_U \mathbb{Z}_N^*$, вычисляет зашифрованный текст $c' = r^e c (\mathrm{mod}\, N)$ и посылает его Алисе. Результат расшифровки, выполненной Алисой, равен

$$c'^d \equiv rm (\mathrm{mod}\, N).$$

Это число может показаться Алисе совершенно случайным, поскольку умножение на число r является **перестановкой** над группой $\mathbb{Z}_N^*$. Итак, Алиса возвращает результат расшифровки rm Злоумышленнику. Увы! Злоумышленнику известно число r, и он может вычислить число m с помощью деления по модулю N. □

Примеры 8.3–8.5 демонстрируют, что учебный алгоритм RSA слишком слаб для реальных приложений. Необходимо систематически исправлять его недостатки. Эта работа выполняется за два этапа.

- В главе 14 вводятся понятия повышенной стойкости схем шифрования с открытым ключом, пригодные для реальных приложений.

- В главе 15 изучается реальная версия алгоритма шифрования RSA, который является стандартным. В ней приводится формальное доказательство стойкости алгоритма RSA на основе понятия сильной стойкости.

8.10 Криптосистема Рабина (учебный вариант)

Криптосистема, разработанная Рабином (Rabin), основана на сложности вычисления квадратного корня по модулю составного числа [240]. Работа Рабина носила теоретический характер. В ней впервые приводилось доказательство стойкости криптосистем с открытым ключом: стойкость криптосистемы Рабина эквивалентна неразрешимости задачи о разложении целых чисел на множители. (Напомним, что эквивалентность разрешимости задачи RSA и разрешимости задачи о разложении целых чисел на множители еще не доказана.) Алгоритм шифрования в криптосистеме Рабина чрезвычайно эффективен и пригоден для многих практических приложений, например, для шифрования с помощью портативных устройств.

Криптосистема Рабина описывается алгоритмом 8.2.

Алгоритм 8.2. Криптосистема Рабина

Ключ

Для создания ключа Алиса должна выполнить следующие действия.

1. Выбрать два случайных простых числа p и q, удовлетворяющих условию $|p| \approx |q|$.
   (* Этот этап совпадает с вычислением модулей алгоритма RSA в алгоритме 8.1. *)
2. Найти число $N = pq$.
3. Извлечь случайное целое число $b \in_U \mathbb{Z}_N^*$.
4. Использовать пару (N, b) в качестве параметров открытого ключа и запомнить пару (p, q) в качестве параметров закрытого ключа.

Шифрование

Для того чтобы послать Алисе секретное сообщение $m \in \mathbb{Z}_N^*$, Боб создает зашифрованный текст c:

$$c \leftarrow m(m + b) (\mathrm{mod}\, N).$$

Расшифровка

Для того чтобы расшифровать зашифрованный текст c, Алиса решает квадратное уравнение

$$m^2 + bm - c \equiv 0 (\mathrm{mod}\, N),$$

где $m < N$.

Покажем, что алгоритм 8.2 действительно описывает криптосистему, т.е. процедура расшифровки, выполняемая Алисой, действительно восстанавливает исходный текст, зашифрованный Бобом.

Из элементарной математики известно, что общее решение квадратного уравнения имеет вид:

$$m \equiv \frac{-b + \sqrt{\Delta_c}}{2} \pmod{N}, \tag{8.10.1}$$

где

$$\Delta_c \stackrel{\text{def}}{=} b^2 + 4c \pmod{N}. \tag{8.10.2}$$

Поскольку число c зависит от элемента $m \in \mathbb{Z}_N^*$, квадратное уравнение

$$m^2 + bm - c \equiv 0 \pmod{N}$$

имеет решения в группе $\mathbb{Z}_N^*$. Одним из этих решений является число m, посланное Бобом. Отсюда следует, что число Δ_c должно быть квадратичным вычетом по модулю N, т.е. элементом группы QR_N.

Вычисления, связанные с расшифровкой, содержат операцию извлечения квадратного корня по модулю N. В разделе 6.6.2 показано, что вычислительная сложность этой задачи эквивалентна факторизации числа N (следствие 6.3). Следовательно, Алиса является единственным пользователем, который может вычислить формулу (8.10.1), поскольку только ей известно разложение числа N на простые множители. Алиса может найти число $\sqrt{\Delta_c}$, используя алгоритм 6.5. В разделе 6.6.2 показано, что для каждого зашифрованного текста c, посланного Бобом, существует четыре разных значения $\sqrt{\Delta_c}$, и, следовательно, существует четыре разных результата шифрования. Как правило, реальное исходное сообщение содержит *избыточную информацию* (redundant information), позволяющую Алисе отличать правильные тексты от неправильных. Смысл понятия "избыточная информация" и способ ее внедрения в текст изложены в разделе 10.4.3.

Отметим, что, если число N является **целым числом Блюма** (Blum integer), т.е. $N = pq$, где $p \equiv q \equiv 3 \pmod{4}$, вычисление квадратных корней по модулю N упрощается и сводится к вычислению квадратных корней по модулю p и q с помощью алгоритма 6.3 для $p \equiv 3, 7 \pmod{8}$ и применения китайской теоремы об остатках. Следовательно, на практике открытые модули, применяемые в криптосистеме Рабина, должны быть целыми числами Блюма.

В алгоритме шифрования Рабина используется только одно умножение и одно сложение. Следовательно, этот алгоритм работает быстрее, чем алгоритм шифрования RSA.

Пример 8.6. Допустим, что Алиса выбирает числа $N = 11 \times 19 = 209$ и $b = 183$. Она объявляет пару $(N, b) = (209, 183)$ параметрами открытого ключа криптосистемы Рабина.

Выберем случайное сообщение m, определим число $c = m(m+b) \pmod N$ и вызовем оракула $\mathcal{O}(c, N)$, который возвращает число $m' \equiv \frac{-b+\sqrt{\Delta_c}'}{2} \pmod N$ с преимуществом ε. Здесь $\sqrt{\Delta_c}'$ обозначает один из четырех квадратных корней числа Δ_c. Из теоремы 6.17 (раздел 6.6.2) следует, что с вероятностью $1/2$ выполняется неравенство

$$m' + \frac{b}{2} \equiv \frac{\sqrt{\Delta_c}'}{2} \not\equiv \pm \frac{\sqrt{\Delta_c}}{2} \equiv \pm \left(m + \frac{b}{2}\right) \pmod N .$$

Однако, поскольку

$$\left(m' + \frac{b}{2}\right)^2 \equiv \frac{\Delta_c}{4} \equiv \left[\pm \left(m' + \frac{b}{2}\right)\right]^2 \pmod N ,$$

из теоремы 6.17 следует, что

$$\gcd\left(m' + \frac{b}{2} \pm \left(m + \frac{b}{2}\right), N\right) = p \text{ или } q. \tag{8.11.1}$$

Иначе говоря, число N можно разложить на множители с ненулевым преимуществом $\varepsilon/2$. Это противоречит предположению о невозможности факторизации модулей алгоритма RSA при условиях неразрешимости задачи о разложении целых чисел на простые множители. Итак, утверждение I доказано.

Утверждение II является тривиальным, если атакующий может получить доступ к блоку расшифровки сообщений: в этом случае блок расшифровки играет роль оракула, использованного в доказательстве утверждения I! Поскольку атакующий генерирует зашифрованный текст для его последующей расшифровки оракулом, такая атака является атакой на основе подобранного зашифрованного текста. □

Из теоремы 8.2 следуют два вывода. Во-первых, криптосистема Рабина является доказуемо стойкой в рамках подхода “все или ничего”, описанного свойством 8.2.1, и зависит от сложности разложения целых чисел на простые множители (если текст не содержит никакой *априорной* информации). Это значительный результат, поскольку он относится к стойкости учебной схемы шифрования Рабина. Если задача о разложении целых чисел на простые множители является трудноразрешимой, то предполагаемый оракул $\mathcal{O}$ в доказательстве утверждения I не может существовать. Однако следует уделить особое внимание модификации принципа “все или ничего” в отношении стойкости криптосистемы по отношению к атакам на основе подобранного открытого текста. Здесь слово “все” означает, что *в общем случае* восстанавливается весь блок исходного текста, т.е. размер сообщения совпадает с размером модуля. Очевидно, что, поскольку шифрование

Предположим, что Боб зашифровал исходное сообщение $m = 31$ с помощью алгоритма шифрования Рабина.

$$c = 31 \times (31 + 183) \equiv 155 (\mathrm{mod}\, 209).$$

Результирующий зашифрованный текст равен 155.

Для расшифровки зашифрованного текста, равного 155, Алиса вычисляет величину Δ_c по формуле (8.10.2).

$$\Delta_c = b^2 + 4c = 183^2 + 4 \times 155 \equiv 42 (\mathrm{mod}\, 209).$$

Теперь, применяя алгоритм 6.5, Алиса определяет четыре квадратных корня числа 42 по модулю 209, т.е. числа 135, 173, 36, 74. В заключение она может применить формулу (8.10.1) и получить четыре результата расшифровки: 185, 204, 31 и 50. В реальной криптосистеме Рабина исходный текст должен содержать дополнительную информацию, позволяющую распознать среди полученных ответов искомый результат. □

8.11 Уязвимость учебной криптосистемы Рабина

Существует чрезвычайно эффективная активная атака против криптосистемы Рабина. В ее основе лежит следующая теорема.

Теорема 8.2.

1. *Криптосистема Рабина является доказуемо стойкой к атаке на основе подобранного открытого зашифрованного текста в рамках подхода "все или ничего", если и только если задача о разложении целого числа на простые множители является трудноразрешимой.*
2. *Криптосистема Рабина является абсолютно беззащитной перед атакой на основе подобранного зашифрованного текста.*

Доказательство. I. Поскольку процедура расшифровки в криптосистеме Рабина использует разложение модулей алгоритма RSA на простые множители, стойкость криптосистемы Рабина означает невозможность факторизации этих модулей. Следовательно, достаточно показать обратное утверждение: неразрешимость задачи о разложении целого числа на простые множители означает стойкость криптосистемы Рабина.

Допустим, что существует оракул $\mathcal{O}$, взламывающий криптосистему Рабина с ненулевым преимуществом ε, т.е.

$$\mathrm{Prob}\left[\mathcal{O}(c, N) = \frac{-b + \sqrt{\Delta_c}}{2} (\mathrm{mod}\, N) \mid c \in_U Z_N^*\right] \geqslant \varepsilon.$$

Рабина является детерминированным алгоритмом, в некоторых случаях расшифровка сообщений не обязательно так же сложна, как и разложение целых чисел на простые множители. Мы еще вернемся к этому утверждению, когда будем рассматривать атаку "встреча посередине" на схему Рабина в конце этого раздела.

Во-вторых, очевидно, что в криптосистеме Рабина *никто* и *никогда* не должен играть роль оракула. Атака на основе подобранного зашифрованного текста уничтожает криптосистему Рабина: в результате такой атаки восстанавливается не только *отдельная* информация об исходном тексте (как в случае атаки CCA2 против криптосистемы RSA в примере 8.5), но и раскрывается секретный ключ. Следовательно, атакующий получает возможность читать *все* секретные сообщения, зашифрованные с помощью открытого ключа.

Пример 8.7. В примере 8.6, посвященном криптосистеме Рабина, были рассмотрены параметры открытого ключа $(N, b) = (209, 183)$ и четыре результата расшифровки сообщения 31: числа $185, 204, 31$ и 50.

Если эти числа станут известны постороннему лицу, не владеющему открытым ключом, например, в результате атаки CCA, он может использовать их для факторизации числа 209. Применяя формулу (8.11.1), получаем, что

$$\gcd(204 - 185, 209) = 19$$

или

$$\gcd((31 + 183/2) + (50 + 183/2), 209) = \gcd(264, 209) = 11.$$ □

Хотя мы предупреждали, что владелец открытого ключа в схеме шифрования Рабина не должен предоставлять услуги расшифровки, в реальных приложениях не стоит полагаться на высокую бдительность пользователей. Следовательно, учебный вариант схемы шифрования Рабина не пригоден для практического применения. Реальные варианты криптосистем Рабина и RSA будут описаны в главе 15. Там же приводится формальное доказательство стойкости практических схем шифрования.

Следует отметить, что в криптосистемах Рабина и RSA используются одни и те же модули. Следовательно, на криптосистему Рабина распространяются меры предосторожности, касающиеся выбора модулей в криптосистеме RSA.

В заключение отметим, что атака "встреча посередине" позволяет взломать еще один вариант схемы шифрования Рабина.

Шифрование. $c = m^2 (\mathrm{mod}\, N)$.

Расшифровка. Вычисление квадратного корня по модулю N.

Как и в учебной криптосистеме RSA, легкость факторизации небольших исходных сообщений (см. раздел 8.9) в криптосистеме Рабина обеспечивает успех атаки на основе перехвата, описанной в примере 8.3.

8.12 Криптосистема Эль-Гамаля (учебный вариант)

Эль-Гамаль разработал весьма остроумную криптосистему с открытым ключом [102], использующую однонаправленную функцию Диффи–Хеллмана с секретом. Работа Эль-Гамаля вызвала большой теоретический и практический интерес, который сохраняется до сих пор. В дальнейшей части книги мы рассмотрим два усовершенствованных варианта этой криптосистемы — схему шифрования Эль-Гамаля (глава 13) и криптосистему Эль-Гамаля с доказуемо высокой стойкостью (глава 15).

Одна из причин широкой популярности этой криптосистемы заключается в использовании задачи Диффи–Хеллмана, неразрешимость которой общепризнанна. Считается, что ее сложность эквивалентна сложности задачи о вычислении дискретного логарифма, которая, в свою очередь, является альтернативой задачи о разложении целого числа на простые множители, лежащей в основе криптосистем RSA и Рабина.

Криптосистема Эль-Гамаля описывается алгоритмом 8.3.

Покажем, что система, описанная алгоритмом 8.3, действительно является криптографической, т.е. в результате расшифровки Алиса восстанавливает тот самый исходный текст, который был послан Бобом.

Поскольку

$$c_1^x \equiv \left(g^k\right)^x \equiv (g^x)^k \equiv y^k \equiv \frac{c_2}{m} (\mathrm{mod}\, p),$$

формула (8.12.2) действительно позволяет восстановить исходный текст m.

Для деления, выполняемого в формуле (8.12.2), необходимо применить расширенный алгоритм Евклида (алгоритм 4.2), что в общем случае сопровождается бóльшими затратами, чем умножение. Однако Алиса может избежать деления, использовав следующие вычисления.

$$m \leftarrow c_2 c_1^{-x} (\mathrm{mod}\, p).$$

Легко проверить, что эта схема расшифровки вполне работоспособна, однако следует иметь в виду, что число $-x$ в этой формуле на самом деле означает число $p - 1 - x$.

Пример 8.8. В примере 8.1 показано, что число 3 является первообразным корнем по модулю 43. Допустим, Алиса выбирает в качестве своего закрытого ключа число 7. Затем она вычисляет свой открытый ключ:

$$37 \equiv 3^7 (\mathrm{mod}\, 43).$$

После этого Алиса оглашает параметры своего открытого ключа $(p, g, y) = (43, 3, 37)$.

Алгоритм 8.3. Криптосистема Эль-Гамаля

Ключ

Для создания ключа Алиса должна выполнить следующие действия.

1. Выбрать случайное простое число p.
2. Вычислить случайный мультипликативный порождающий элемент $g \in \mathbb{F}_p^*$.
3. Извлечь случайное целое число $x \in_U \mathbb{Z}_{p-1}$ и считать его своим закрытым ключом.
4. Вычислить открытый ключ

$$y \leftarrow g^x (\mathrm{mod}\, p).$$

5. Использовать тройку (p, g, y) в качестве параметров открытого ключа и запомнить число x в качестве закрытого ключа.

(* Аналогично протоколу обмена ключами Диффи–Хеллмана системные пользователи могут использовать общие открытые параметры (p, g). *)

Шифрование

Для того чтобы послать Алисе секретное сообщение $m < p$, Боб извлекает случайное целое число $k \in_U \mathbb{Z}_{p-1}$ и вычисляет зашифрованный текст (c_1, c_2):

$$\begin{cases} c_1 \leftarrow g^k (\mathrm{mod}\, p), \\ c_2 \leftarrow y^k m (\mathrm{mod}\, p). \end{cases} \tag{8.12.1}$$

Расшифровка

Для того чтобы расшифровать зашифрованный текст (c_1, c_2), Алиса вычисляет формулу

$$m \leftarrow \frac{c_2}{c_1^x} (\mathrm{mod}\, p). \tag{8.12.2}$$

Предположим, что Боб шифрует исходное сообщение $m = 14$. Он извлекает случайный показатель степени, равный 26, и вычисляет следующие значения.

$$c_2 = 15 \equiv 3^{26} (\mathrm{mod}\, 43), c_2 = 31 \equiv 36^{26} \times 14 (\mathrm{mod}\, 3).$$

В результате возникает зашифрованный текст $(15, 31)$.

Для того чтобы расшифровать сообщение $(15, 31)$, Алиса находит следующее число.

$$14 = 31/36 \equiv 31/15^7 (\mathrm{mod}\, 43).$$

Для деления необходимо применить алгоритм 4.2. Однако Алиса не делает этого и вычисляет значение 14 иначе.

$$14 = 31 \times 15^{42-7} \equiv 31 \times 6 (\mathrm{mod}\, 43).$$

□

8.13 Уязвимость учебной криптосистемы Эль-Гамаля

Алгоритм шифрования (8.12.1) является вероятностным: он использует случайное число $k \in_U \mathbb{Z}_{p-1}$. Допустим, что закрытый ключ Алисы x является числом, взаимно простым с числом $p-1$. Тогда по теореме 5.2.3 (раздел 5.2.3) открытый ключ Алисы $y \equiv g^x (\mathrm{mod}\, p)$, как и число g, является порождающим элементом группы $\mathbb{F}_p^*$. Следовательно, число $y^k (\mathrm{mod}\, p)$ пробегает всю группу $\mathbb{F}_p^*$, когда элемент k пробегает группу $\mathbb{Z}_{p-1}$. Поскольку умножение по модулю p является перестановкой над группой $\mathbb{F}_p^*$, для любого исходного сообщения $m \in \mathbb{F}_p^*$ число $c_2 \equiv y^k m (\mathrm{mod}\, p)$ пробегает всю группу $\mathbb{F}_p^*$, когда число k пробегает группу $\mathbb{Z}_{p-1}$ (теорема 6.6 из раздела 6.2.2). Итак, если $k \in_U \mathbb{Z}_{p-1}$, то $c_2 \in_U \mathbb{F}_p^*$. Это означает, что шифрование Эль-Гамаля *равномерно* распределяет исходное сообщение по всему пространству сообщений. Это — идеальное семантическое свойство алгоритма шифрования.

Однако не следует быть слишком большими оптимистами! Зашифрованный текст в схеме Эль-Гамаля — это не отдельный блок c_2, а пара (c_1, c_2), причем числа c_1 и c_2 статистически *связаны* между собой. Следовательно, как и все криптосистемы с открытым ключом, стойкость криптосистемы Эль-Гамаля зависит от условий неразрешимости задачи, лежащей в ее основе. Более того, как будет показано в разделе 8.13.1, для того, чтобы обеспечить идеальное семантическое свойство, исходное сообщение должно принадлежать группе $\langle g \rangle$. К сожалению, в реальных системах это условие, как правило, не выполняется.

Рассмотрим стойкость криптосистемы Эль-Гамаля по принципу "все или ничего".

Теорема 8.3. *Для исходного сообщения, равномерно распределенного по всему пространству исходных сообщений, криптосистема Эль-Гамаля является стойкой к атаке на основе подобранного открытого текста по принципу "все или ничего" тогда и только тогда, когда задача Диффи–Хеллмана является трудноразрешимой.*

Доказательство. ($\Rightarrow$) Необходимо показать, что, если криптосистема Эль-Гамаля является стойкой, выполняются условия неразрешимости задачи Диффи–Хеллмана.

Допустим противное — условия неразрешимости задачи Диффи–Хеллмана не выполняются. Тогда при заданном тексте $(c_1, c_2) \equiv (g^k, y^k m) (\mathrm{mod}\, p)$, зашифрованном с помощью открытого ключа $y \equiv g^x (\mathrm{mod}\, p)$, оракул задачи Диффи–Хеллмана вычислит по четверке (p, g, g^x, g^k) число $g^{xk} \equiv y^k (\mathrm{mod}\, p)$ с ненулевым преимуществом. Затем с тем же преимуществом вычисляется величина $m \leftarrow c_2 / y^k (\mathrm{mod}\, p)$. Это противоречит стойкости криптосистемы Эль-Гамаля.

($\Leftarrow$) Теперь необходимо показать, что при условиях неразрешимости задачи Диффи–Хеллмана не существует ни одного эффективного алгоритма, позволяющего с преимуществом, которое не является пренебрежимо малым, восстановить исходное сообщение, зашифрованное с помощью криптосистемы Эль-Гамаля.

Допустим противное — существует эффективный оракул $\mathcal{O}$, позволяющий взламывать криптосистему Эль-Гамаля, т.е. для любых параметров открытого ключа (p, g, y) и зашифрованного текста (c_1, c_2) оракул $\mathcal{O}$ с преимуществом δ, которое не является пренебрежимо малым, вычисляет значение

$$m \leftarrow \mathcal{O}(p, g, y, c_1, c_2),$$

удовлетворяющее условию

$$\frac{c_2}{m} \equiv g^{(\log_g y \log_g c_1)} (\mathrm{mod}\, p).$$

Затем для произвольного варианта задачи Диффи–Хеллмана (p, g, g_1, g_2) тройка (p, g, g_1) выбирается в качестве параметров открытого ключа, а пара (g_2, c_2) — в качестве зашифрованной пары, где $c_2 \in \mathbb{F}_p^*$. Тогда с преимуществом δ оракул $\mathcal{O}$ вычисляет значение

$$m \leftarrow \mathcal{O}(p, g, g_1, g_2, c_2),$$

которое удовлетворяет условию

$$\frac{c_2}{m} \equiv g^{(\log_g g_1 \log_g g_2)} (\mathrm{mod}\, p).$$

Это противоречит условиям неразрешимости задачи Диффи–Хеллмана. □

Поскольку стойкость криптосистемы Эль-Гамаля к атаке CPA эквивалентна неразрешимости задачи Диффи–Хеллмана, все утверждения, касающиеся параметров открытого ключа (раздел 8.4), относятся и к криптосистеме Эль-Гамаля. Как и в протоколе обмена ключами Диффи–Хеллмана, криптосистема Эль-Гамаля работает в подгруппе группы $\mathbb{F}_q$, порядок которой является большим простым числом, или в большой группе точек эллиптической кривой, определенной над конечным полем.

8.13.1 Атака "встреча посередине" и активная атака на учебную криптосистему Эль-Гамаля

Алгоритм 8.3, описывающий криптосистему Эль-Гамаля, является учебным, поскольку его схема шифрования очень слаба. Попробуем разобраться в причинах ее слабости.

Обычная схема шифрования Эль-Гамаля, применяемая в реальных приложениях, допускает частичную утечку информации даже при пассивных атаках. На практике в криптосистеме Эль-Гамаля для достижения большей эффективности часто используется элемент g, имеющий порядок $r = \mathrm{ord}_p(g) \ll p$. Если в этом случае сообщение m не принадлежит подгруппе $\langle g \rangle$, то атака "встреча посередине" на криптосистему Эль-Гамаля почти не отличается от подобной атаки на криптосистему RSA (см. пример 8.3). По зашифрованному тексту $(c_1, c_2) = (g^k, y^k m) (\mathrm{mod}\, p)$ Злоумышленник может найти число

$$c_2^r \equiv m^r (\mathrm{mod}\, p).$$

Таким образом, Злоумышленник преобразовал "вероятностную" схему шифрования в криптосистеме Эль-Гамаля в *детерминированную*! Более того, криптосистема Эль-Гамаля обладает таким же мультипликативным свойством, что и криптосистема RSA (см. раздел 8.9). Следовательно, при пересылке коротких сообщений, которые легко разложить на простые множители, Злоумышленник может организовать атаку "встреча посередине" точно так же, как это делается в криптосистеме RSA [52].

Итак, если исходное сообщение не принадлежит подгруппе, порожденной элементом g, криптосистема Эль-Гамаля становится детерминированной. Такие криптосистемы допускают частичную утечку информации, поскольку уязвимы для атаки путем перебора коротких исходных сообщений, например, секретных предложений цены или зарплат сотрудников.

Покажем теперь, что криптосистема Эль-Гамаля уязвима для активной атаки.

Пример 8.9. Допустим, что Злоумышленник имеет частичный доступ к блоку расшифровки Алисы в криптосистеме Эль-Гамаля. Как и в примере 8.5, разумно предположить, что в ответ на бессмысленное сообщение, полученное от Злоумышленника, Алиса должна вернуть ему результат расшифровки.

Предположим, что Злоумышленник владеет зашифрованным текстом $(c_1, c_2) \equiv$ $\equiv (g^k, y^k m) (\mathrm{mod}\, p)$, перехваченным им во время предыдущего сеанса связи между Алисой и кем-либо еще (кроме Злоумышленника!). Если Злоумышленник желает восстановить исходный текст, он генерирует случайное число $r \in_U \mathbb{F}_p^*$, вычисляет $c_2' = r c_2 (\mathrm{mod}\, p)$ и посылает подобранный шифрованный текст (c_1, c_2') Алисе. После расшифровки Алиса восстанавливает число

$$rm (\mathrm{mod}\, p),$$

которое, с ее точки зрения, выглядит совершенно случайным, поскольку умножение на число $r < p$ является перестановкой над группой $\mathbb{F}_p^*$. Алиса возвращает Злоумышленнику число rm. Увы! Злоумышленнику известно число r, и он может восстановить число m, выполняя операцию деления по модулю p. □

8.14 Необходимость понятия повышенной стойкости для криптосистем с открытым ключом

Мы рассмотрели несколько учебных криптосистем с открытым ключом. Их можно считать непосредственными приложениями разных однонаправленных функций с секретом. (Понятие однонаправленной функции с секретом введено при описании свойства 8.1.) Настало время разобраться в причинах уязвимости этих криптосистем. Следует обсудить два аспекта уязвимости учебных криптосистем с открытым ключом.

Во-первых, как указано при описании свойства 8.2.1, в рамках этой главы мы рассматриваем очень слабое понятие стойкости: по принципу "все или ничего". В большинстве приложений криптосистем с открытым ключом такая стойкость совершенно неприемлема. Как правило, исходные сообщения содержат *априорную* информацию, известную атакующему. Например, если шифруется результат голосования, то атакующий априори знает, что ответом может быть либо слово "ДА", либо слово "НЕТ", либо фамилия кандидата. Таким образом, независимо от стойкости функции с секретом, атакующему достаточно применить метод перебора и подобрать исходный текст. В других приложениях атакующий получает дополнительное преимущество за счет знания априорной информации (пример такой атаки будет рассмотрен в разделе 14.3.2). Как правило, учебный алгоритм шифрования недостаточно хорошо скрывает подобную информацию. Следовательно, необходимо разработать криптосистемы с открытым ключом, скрывающие любую априорную информацию.

Во-вторых, как указывалось при описании свойства 8.2.2, в рамках этой главы рассматривается очень слабый вариант атак — пассивные атаки. Однако для каждой из рассмотренных учебных криптосистем с открытым ключом были продемонстрированы и активные атаки (примеры 8.5, 8.7 и 8.9). В ходе атаки CCA или CCA2 Злоумышленник может подготовить очень хитроумное сообщение и переслать его владельцу ключа для дальнейшей расшифровки. Как показано выше, учебные криптосистемы с открытым ключом весьма уязвимы для таких атак. Несмотря на меры предосторожности, которые можно предпринять в таких ситуациях, их явно недостаточно. Невозможно держать пользователей в состоянии

постоянной тревоги и запрещать им отвечать на какие бы то ни было запросы, подлежащие расшифровке.

Различными авторами были разработаны криптосистемы с открытым ключом повышенной стойкости. В главе 14 будут введены понятия, связанные с повышенной стойкостью, и показано, как достичь **формально доказуемой стойкости** (formally provable security). В главе 15 мы рассмотрим практичные криптосистемы с открытым ключом, стойкость которых можно доказать даже в рамках очень строгого подхода.

8.15 Комбинация асимметричной и симметричной криптографии

Криптография с открытым ключом весьма элегантно решает задачу распределения ключей. Однако, как правило, криптографические функции с открытым ключом действуют в очень крупных алгебраических структурах, т.е. связаны с выполнением весьма затратных алгебраических операций. С этой точки зрения симметричные криптографические функции гораздо эффективнее. Например, алгоритм AES работает в поле, состоящем из 256 элементов. Его основные операции, такие как умножение и вычисление обратной функции, можно выполнить с помощью очень эффективного табличного поиска (см. раздел 7.7.4). Кроме того, вычисления в криптосистемах с открытым ключом выполняются намного интенсивнее, чем в их симметричных аналогах.

В приложениях, особенно связанных с шифрованием больших объемов данных, стандартным решением стало использование **гибридных схем** (hybrid scheme). В таких схемах криптография с открытым ключом используется для шифрования так называемого **эфемерного ключа** (ephemeral key) для симметричной криптосистемы. Этот ключ распределяется между отправителем и адресатом, а затем с его помощью шифруется большой массив данных. Гибридные системы обладают лучшими свойствами обеих криптосистем: легкостью распределения ключей в криптосистемах с открытым ключом и эффективностью симметричных криптосистем.

Широкое распространение получила комбинация криптосистем с открытым и симметричным ключами — **цифровой конверт** (digital envelope). Эта схема является комбинацией криптосистемы RSA с симметричной криптосистемой, например, с алгоритмом DES, тройным алгоритмом DES или алгоритмом AES. Такая комбинация (RSA+DES или RSA+тройной DES) представляет собой схему **безопасного переходника** (secure sockets layer — SSL) [136]. Протокол SSL описан в главе 12. Он широко применяется в Web-браузерах, таких как Netscape и Internet Explorer, а также в Web-серверах. Инициатор протокола SSL (допустим, Алиса — как правило, пользователь сети Web) загружает параметры открытого ключа дру-

гой стороны (например, Боба — как правило, Web-сервера). Затем Алиса (точнее, ее Web-браузер) генерирует случайный сеансовый ключ, шифрует его ("запечатывает в конверт"), используя открытый ключ Боба, и посылает Бобу. После этого Боб (точнее, его Web-сервер) расшифровывает сообщение ("распечатывает конверт") и восстанавливает сеансовый ключ. Теперь обе стороны могут использовать этот ключ в симметричной схеме шифрования для дальнейшего обмена секретными сообщениями.

С точки зрения организации протокола гибридная схема очень проста. Однако существует два ограничения. Во-первых, эта схема использует сеансовый ключ, созданный одной из сторон (отправителем сообщения, или инициатором протокола), а другая сторона (получатель сообщения, или адресат протокола) должен целиком полагаться на компетентность и честность инициатора протокола. В некоторых ситуациях это нежелательно: например, в протоколе SSL, в рамках которого клиентом является отправитель, а точнее — его программное обеспечение, которое, как известно, представляет собой весьма слабый датчик случайных чисел.

Во-вторых, гибридные схемы шифрования инертны. В таких системах перехватчик, который может силой заставить получателя раскрыть свой закрытый ключ, получает возможность расшифровывать все сообщения. Эта особенность называется недостатком "заблаговременной секретности". Заблаговременная секретность означает, что перехватчик не в состоянии расшифровать исходное сообщение в будущем, используя зашифрованные сообщения, полученные в прошлом, ни с помощью криптоанализа, ни с помощью *принуждения*.

Эти ограничения можно преодолеть, если в качестве криптосистемы с открытым ключом использовать протокол обмена ключами Диффи–Хеллмана.

Рассмотрим сначала, как преодолевается первое ограничение. При запуске протокола обмена ключами Диффи–Хеллмана между Алисой и Бобом распределяемый секрет g^{ab} представляет собой случайное число, сформированное обеими сторонами: вклад Алисы — число a, вклад Боба — число b. Если число g порождает группу, имеющую простой порядок, и сообщения протокола удовлетворяют условиям $g^a \neq 1$ и $g^b \neq 1$ (детали описаны в разделе 8.3), Алиса (соответственно Боб) может быть уверена, что общий сеансовый ключ, вычисленный на основе числа g^{ab}, является случайным, поскольку она использовала случайный показатель степени. Это возможно благодаря тому, что отображения $g^b \mapsto \left(g^b\right)^a$ и $g^b \mapsto \left(g^a\right)^b$ являются перестановками в группе. Следовательно, если случайный показатель степени не превышает порядок группы, то число g^a (соответственно число g^b) отображается в случайный элемент группы g^{ab}.

Теперь покажем, как снять второе ограничение. Заметим, что гибридная схема, использующая протокол обмена ключами Диффи–Хеллмана, обладает свойством заблаговременной секретности, если Алиса и Боб принимают меры предосторожности, приведенные в разделе 8.3. Для этого Алиса и Боб должны обменяться

сеансовым ключом g^{ab}, а затем стереть показатели a и b до окончания протокола. Кроме того, Алиса и Боб должны уничтожить сеансовый ключ после окончания сеанса связи и правильно разместить исходные сообщения, которыми они обменивались. Если они выполняют эти вполне *стандартные* процедуры, то никакими мерами принуждения перехватчик не сможет взломать исходные сообщения, которыми обменивались Алиса и Боб. Криптоаналитики также не справятся с этой задачей, поскольку свойство заблаговременной секретности (благодаря протоколу обмена ключами Диффи–Хеллмана) является следствием неразрешимости вычислительной проблемы Диффи–Хеллмана (см. раздел 8.4).

В заключение отметим, что можно разработать гибридную схему шифрования, обладающую **доказуемой стойкостью** (provable security) в смысле сильной стойкости. Такие схемы рассматриваются в главе 15.

8.16 Организация канала для обмена открытыми ключами

Атака на протокол обмена ключами Диффи–Хеллмана под названием "человек посередине" (man-in-the-middle attack) — обычный способ взлома криптосистем с открытыми ключами (раздел 8.3.1). Как правило, чтобы переслать секретную информацию, зашифрованную с помощью открытого ключа, отправитель сначала должен убедиться, что сообщение будет доставлено по назначению. Аналогично при получении симметричного ключа шифрования, содержащегося в цифровом конверте, получатель сначала должен убедиться, что он отправлен кем положено.

Итак, независимо от популярности криптографических систем с открытым ключом, необходимо организовать секретный канал связи. Однако в криптографии с открытым ключом выполняется условие $k\ell \neq kd$ (см. рис. 7.1). Следовательно, передача ключа шифрования отправителю сообщения не обязательно связана с раскрытием секретной информации. Таким образом, организация секретного канала для обмена ключами сводится исключительно к задаче аутентификации.

Организация каналов для обмена аутентичными открытыми ключами обсуждается в главе 13. В разделе 13.2 описывается метод организации канала для обмена открытыми ключами на основе каталогов, а в разделе 13.3 — методы личностной криптографии с открытым ключом.

8.17 Резюме

В главе рассмотрено несколько хорошо известных схем шифрования с помощью открытого ключа: протокол обмена ключами Диффи–Хеллмана, а также алгоритмы шифрования RSA, Рабина и Эль-Гамаля. Описаны задачи, неразрешимость которых обусловливает стойкость этих криптосистем.

Стойкость, которая рассматривается в этой главе, основана на принципе "все или ничего", а атаки считаются пассивными. По этой причине все алгоритмы шифрования являются учебными и могут применяться в реальных приложениях только при условии, что все передаваемые данные являются абсолютно случайными, а злоумышленники не ведут себя агрессивно (т.е. не организовывают активных атак). Нестойкость учебных криптосистем, описанных в главе, продемонстрирована на примере разнообразных атак.

Выявлена необходимость более строгого понятия стойкости, которое было бы применимо к реальным криптосистемам. Однако эти системы будут описаны в части V. Читатели, не собирающиеся внимательно изучать часть V, должны тщательно разобрать все атаки, описанные в главе, особенно если они планируют использовать учебные криптосистемы.

Упражнения

8.1. Какими двумя замечательными свойствами обладают учебные криптографические алгоритмы?

8.2. Режим сцепления блоков (CBC) блочного шифра, описанный в разделе 7.8.2, имеет случайные входные данные, и, следовательно, любая утечка информации об исходном тексте исключена. Можно ли считать режим CBC учебным криптографическим алгоритмом? Почему?

8.3. Допустим, что Злоумышленник, организовавший атаку "человек посередине" на протокол обмена ключами Диффи–Хеллмана, только передает сообщения, которыми обмениваются Алиса и Боб (т.е. "человек посередине" не изменяет сообщения, а просто выполняет расшифровку и шифрование с помощью ключей, распределенных между Алисой и Бобом). Какой является эта атака: активной или пассивной?

8.4. В качестве общепринятой нижней границы для размера конечного поля F_q используется число $|q| = 1024$, а в суэкспоненциальном выражении $\mathrm{sub\_exp}(q)$ (8.4.2) число c должно быть меньше двух. Докажите, что при этих условиях существует полиномиальный алгоритм дискретного логарифмирования в поле F_q, причем время работы полиномиального алгоритма решения ограничено полиномом девятой степени, зависящим от размера поля q.

8.5. Допустим, что группа $\langle g \rangle$ имеет несекретный порядок $\mathrm{ord}(g)$. Является ли трудноразрешимой следующая задача? По заданному числу g^c найти числа g^a и g^b, такие что $ab \equiv c(\mathrm{mod}\,\mathrm{ord}(g))$, т.е. создать кортеж Диффи–Хеллмана (g, g^a, g^b, g^c) по заданной паре (g, g^c).

8.6. Как связаны между собой задача дискретного логарифмирования и вычислительная проблема Диффи–Хеллмана?

8.7. Почему в наборе параметров открытого ключа (e, N) криптосистемы RSA показатель степени e должен быть взаимно простым с числом $\phi(N)$?

8.8. Разложение нечетного составного целого числа на множители считается трудноразрешимой задачей. Является ли трудноразрешимой задача о факторизации степени простого числа? (Степенью простого числа называется число $N = p^i$, где p — простое число, а i — целое число. Выполните разложение числа N на простые множители.)

Подсказка: сколько индексов i, где $i > 1$, необходимо перебрать для того, чтобы найти корень i-й степени числа N?

8.9. Если число N является степенью простого числа, одним из методов вычисления корня i-й степени числа N является метод бинарного поиска. Разработайте бинарный алгоритм поиска для вычисления корня степени p^i, где показатель i известен. Докажите эффективность этого алгоритма.

Подсказка: рассмотрите алгоритм бинарного поиска простых чисел, состоящих из $\frac{\log_2}{i}$ битов.

8.10. В криптосистеме RSA функция шифрования является перестановкой в мультипликативной группе по модулю RSA. По этой причине функция RSA называется однонаправленной функцией перестановки с секретом. Является ли функция шифрования в криптосистеме Рабина (Эль-Гамаля) однонаправленной функцией перестановки с секретом?

8.11. Допустим, что $N \approx 2^{1024}$, а элементы из группы $\mathbb{Z}_N^*$ извлекаются случайным образом. Какова вероятность того, что выборочное значение будет меньше, чем 2^{64}? Используя этот результат, докажите, что в алгоритмах шифрования RSA, Рабина и Эль-Гамаля 64-битовый случайный пароль не должен считаться случайным исходным текстом.

8.12. При каких условиях функцию шифрования в криптосистеме Эль-Гамаля можно считать детерминированным алгоритмом?

8.13. Опишите атаки CPA, CCA и CCA2.

8.14. При описании стойкости криптосистем RSA и Рабина к атаке CPA использовался принцип “все или ничего” (теоремы 8.1 и 8.2.1 соответственно). Зачем это нужно?

8.15. Почему любой алгоритм шифрования в криптосистемах с открытым ключом (даже учебный) должен быть устойчивым к атаке CPA?

8.16. Назовите основную причину уязвимости учебных криптографических алгоритмов для активных атак.

8.17. Что такое услуги оракула? Необходимы ли атакующему услуги оракула для расшифровки сообщений в криптосистеме с открытым ключом?

8.18. Поскольку все учебные алгоритмы уязвимы для активных атак, необходимо быть бдительным и не предоставлять услуги оракула для расшифровки сообщений. Насколько практична такая стратегия?

8.19. Активная атака, как правило, сопровождается изменением зашифрованного сообщения, передаваемого по сети. Эффективна ли активная атака, если алгоритм шифрования в криптосистеме с открытым ключом содержит механизм, защищающий целостность данных и распознающий несанкционированные изменения зашифрованных текстов?

8.20. Какие преимущества имеют гибридные системы?

Глава 9

Идеальный мир: битовая стойкость основных криптографических функций с открытым ключом

9.1 Введение

Примеры, рассмотренные нами в предыдущей главе, свидетельствуют о том, что основные криптографические функции с открытым ключом, как правило, допускают утечку частичной информации об исходных сообщениях, особенно, если эти сообщения не случайны. Однако эти функции не настолько плохи, если исходные сообщения носят случайный характер. В таких ситуациях описанные ранее криптографические функции оказываются очень стойкими.

В главе рассматривается понятие **битовой стойкости** (bit security) основных криптографических функций с открытым ключом. Мы покажем, что каждая из этих функций обеспечивает защиту битов. Это означает, что если исходное сообщение является случайным, то расшифровать его отдельный бит так же сложно, как и восстановить весь блок, т.е. вычислить обратную функцию.

Свойство битовой стойкости применялось многими исследователями для создания стойких схем шифрования с открытым ключом на основе распространенных криптографических функций. В основе этих работ лежит идея рандомизации исходных текстов перед шифрованием. В части V описывается общая методология доказательства стойкости — **модель случайного оракула** (random oracle model). В рамках этой модели стойкость схем шифрования с открытым ключом (как и схем цифровой подписи), основанных на применении основных криптографических функций, описанных в предыдущей главе, доказывается в строгом смысле. Важным компонентом доказательства повышенной стойкости таких схем является предположение о том, что исходный текст является рандомизированным.

Заметим, что в идеальном мире все исходные сообщения являются случайными. Однако в реальном мире это условие не выполняется. Кроме того, в идеальном

мире злоумышленники никогда не организовывают активных атак. Следовательно, основные криптографические схемы с открытым ключом слишком слабы для реального мира.

Читатели, не интересующиеся реальными криптографическими схемами, могут пропустить эту главу.

9.1.1 Структурная схема главы

В разделе 9.2 рассматривается битовая стойкость алгоритма RSA. Раздел 9.3 посвящен битовой стойкости алгоритма Рабина и применению этой схемы для генерации псевдослучайных чисел. В разделе 9.4 изучается битовая стойкость алгоритма Эль-Гамаля. В разделе 9.5 описывается битовая стойкость дискретного логарифма.

9.2 Битовая стойкость алгоритма RSA

Если алгоритм RSA используется для шифрования сообщения, не содержащего никакой *априори* угадываемой информации (например, если сообщение является равномерно распределенным случайным числом из группы $\mathbb{Z}_N^*$), то сложность восстановления отдельного бита исходного сообщения из зашифрованного текста сравнима с расшифровкой всего блока [75, 76, 128]. Не ограничивая общности, будем считать, что перед криптоаналитиком стоит задача восстановления младшего значащего бита, т.е. бита четности исходного сообщения.

Теорема 9.1. *Пусть N — модуль алгоритма RSA. Сложность решения следующих двух задач одинакова.*

I. Восстановить сообщение по заданному тексту, зашифрованному с помощью алгоритма RSA.

II. Восстановить младший значащий бит сообщения по заданному тексту, зашифрованному с помощью алгоритма RSA.

Очевидно, что из решения первой задачи следует решение второй. Обратное утверждение не столь очевидно. На первый взгляд вычислительная сложность этих двух задач одинакова: первая задача оказывается вычислительной, если исходное сообщение не является равномерно распределенным случайным сообщением, а вторая относится к задачам принятия решений и в половине случаев сводится к простому угадыванию.

Несмотря на это, если взломщику помогает оракул, который может *надежно* решить вторую задачу, то первую задачу можно решить, вызвав оракула $\log_2 N$ раз. Поскольку размеры чисел N и $\log_2 N$ совпадают, этот метод позволяет свести

первую задачу ко второй за полиномиальное время, зависящее от размера входного числа. По этой причине такой метод называется **полиномиальной редукцией** (polynomial time reduction). Следовательно, с помощью оракула задачу I можно решить за полиномиальное время, зависящее от размера исходных данных и *превышающее* время решения задачи II. И все же эти две задачи имеют одинаковую временну́ю сложность, поскольку мы не дифференцируем понятие сложности по степени полинома.

Рассмотрим метод полиномиальной редукции задачи I к задаче II. Назовем оракул, решающий задачу II, "оракулом четности RSA" и обозначим его как PO_N. Тогда процедура сведения задачи I к задаче II описывается следующим образом.

$$m(\mathrm{mod}\,2) \leftarrow PO_N(m^e(\mathrm{mod}\,N)).$$

В доказательстве теоремы 9.1 обозначим через $x \in (a, b)$ целое число x, принадлежащее интервалу (a, b), где числа a и/или b не обязательно целые. Поскольку x — целое число, из условия $x \in (a, b)$ следует, что оно принадлежит отрезку $[\lceil a \rceil, \lfloor b \rfloor]$.

Основной проблемой в доказательстве является метод **бинарного поиска** (binary search). Его применение обеспечивается следующим утверждением.

Лемма 9.1. *Пусть N — нечетное целое число и $x \in (0, N)$. Тогда число $2x(\mathrm{mod}\,N)$ является четным, если и только если $x(\mathrm{mod}\,N) \in (0, N/2)$.*

Доказательство. Для всех чисел $x \in (0, N/2)$ умножение $2x(\mathrm{mod}\,N)$ не требует выполнения операций по модулю, и, следовательно, результатом является число $2x$ — четное число из интервала $(0, N)$. И наоборот, если число $2x(\mathrm{mod}\,N)$ является четным, оно нацело делится на 2, причем для этого не требуется выполнять никаких операций по модулю. Следовательно, $x \in (0, N/2)$. □

Поскольку все целые числа из интервала $(0, N/2)$ образуют ровно половину целых чисел из интервала $(0, N)$, из леммы 9.1 следует, что число $2x(\mathrm{mod}\,N)$ является нечетным тогда и только тогда, когда $x \in (N/2, N)$.

Перейдем к доказательству теоремы 9.1.

Доказательство. Необходимо показать, что II $\Rightarrow$ I. Доказательство этого факта носит конструктивный характер. Построим алгоритм бинарного поиска, использующий надежный оракул PO_N и определяющий число m по тексту $c = m^e(\mathrm{mod}\,N)$, зашифрованному с помощью алгоритма RSA. Поиск осуществляется в интервале (a, b), который называется *текущим* и обозначается как CI (current interval). Первоначальным текущим интервалом является интервал $(a, b) = (0, N)$. В ходе бинарного поиска выполняются следующие инвариантные условия.

- На каждой итерации интервал CI делится пополам.
- Искомый исходный текст по-прежнему принадлежит интервалу CI.

Для простоты выполним только две итерации этого алгоритма.

Итерация Известно, что исходный текст принадлежит интервалу $(a, b) = (0, N)$. Передадим оракулу PO_N число $2^e c (\mathrm{mod}\, N)$. Заметим, что $2^e c \equiv (2m)^e (\mathrm{mod}\, N)$. Тогда из ответа оракула $PO_N(2^e c)$ и леммы 9.1 следует, что либо $m \in (0, N - N/2)$, либо $m \in (N/2, N)$. Таким образом, возникает новый текущий интервал, содержащий исходный текст и имеющий вдвое меньшую длину. Итак, на вход итерации поступил интервал $(a, b) = (0, N)$, а на выходе получается один из двух интервалов: $(a, b) = (0, N/2)$ или $(a, b) = (N/2, N)$.

Итерация Рассмотрим случай $(a, b) = (N/2, N)$. Передадим оракулу число $2^{2e} c \equiv \equiv (2^2 m)^e (\mathrm{mod}\, N)$. Если $PO_N(2^{2e} c) = 0$, то исходный текст $2^2 m \equiv 4m (\mathrm{mod}\, N)$ является четным числом. По лемме 9.1 получаем, что $2m (\mathrm{mod}\, N) \in (0, N/2)$. Вспомним условие $2m < 2N$. Следовательно, неравенство $2m (\mathrm{mod}\, N) < N/2$ возможно, только если $2m < 2N - N/2 = 3N/2$. Итак, $m < 3N/4$. Итак, $m \in (N/2, 3N/4)$. Обновим текущий интервал, выполнив преобразование $(a, b) \leftarrow \left(a, b - \frac{N}{2^2}\right)$. Таким образом, инвариантные условия выполняются.

Читатели могут самостоятельно проверить два основных варианта преобразования текущего интервала.

- Если оракул PO_N возвращает нуль, исходный текст принадлежит левой половине текущего интервала $CI = (a, b)$, а число b следует уменьшить на величину $|CI|/2$.
- В противном случае исходный текст принадлежит правой половине текущего интервала $CI = (a, b)$, и число a необходимо увеличить на значение $|CI|/2$.

Очевидно, что после $i = \lfloor \log_2 N \rfloor + 1$ шагов поиска длина текущего интервала станет меньше единицы: $|CI| = b - a < 1$. В этом случае алгоритм прекращает работу и выдает число $m = b$. Теорема 9.1 доказана. □

Описание бинарного поиска исходного сообщения суммируется алгоритмом 9.1.

Пример 9.1. Допустим, что открытым ключом алгоритма RSA является пара $(N, e) = (15, 3)$, а шифрованный текст — $c = 13$. Зададим оракулу четыре вопроса и попробуем восстановить секретный текст m. Передадим оракулу следующие запросы:

$$(2^3 \times 13, 4^3 \times 13, 8^3 \times 13, 16^3 \times 13) \equiv (14, 7, 11, 13) (\mathrm{mod}\, 15).$$

Алгоритм 9.1. Бинарный поиск исходного текста, зашифрованного алгоритмом RSA, с помощью оракула четности

ИСХОДНЫЕ ДАННЫЕ:

(N, e): параметры открытого ключа алгоритма RSA;
$c = m^e (\mathrm{mod}\, N)$: текст, зашифрованный алгоритмом RSA;
PO_N: оракул четности, получающий текст, зашифрованный алгоритмом RSA, и возвращающий младший значащий бит соответствующего исходного текста.

РЕЗУЛЬТАТ:

m: исходный текст.

1. Выполнить инициализацию $(a, b) \leftarrow (0, N)$.
   (* Пока выполняется условие $m \in (a, b)$, длина текущего интервала $CI = (a, b)$ на каждой итерации делится пополам. *)
2. for $i = 1, 2, \ldots, \lfloor \log_2 N \rfloor + 1$ do
   {
   (* Длина отрезка (a, b) никогда не превышает число $\frac{N}{2^{i-1}}$. *)
   а) If $(PO_N(2^{ie}c) = 0)$ then $b \leftarrow b - \frac{N}{2^i}$.
      (* Число m лежит в левой половине интервала (a, b). *)
   б) Else $a \leftarrow a + \frac{N}{2^i}$;
      (* Число m лежит в правой половине интервала (a, b). *)
   }
3. return $(\lfloor b \rfloor)$

Ответы оракула: 0, 1, 1, 1. На их основе можно сделать следующие выводы.

Первый ответ: $0 \Rightarrow m \in (0, 15 - 15/2) = (0, 15/2)$, т.е. $m \in [1, 7]$.

Второй ответ: $1 \Rightarrow m \in (0 + 15/4, 15/2) = (15/4, 15/2)$, т.е. $m \in [4, 7]$.

Третий ответ: $1 \Rightarrow m \in (15/4 + 15/8, 15/2) = (45/8, 15/2)$, т.е. $m \in [6, 7]$.

Четвертый ответ: $1 \Rightarrow m \in (48/8 + 15/16, 15/2) = (105/16, 15/2)$, т.е. $m \in [7, 7]$.

Итак, $m = 7$. Действительно, исходным сообщением является число 7: $7^3 = 13 (\mathrm{mod}\, 15)$. □

Из теоремы 9.1 следует, что младший значащий бит в исходном тексте, зашифрованном алгоритмом RSA, найти так же сложно, как и целый блок исходного сообщения.

Пример 8.5 демонстрирует опасность, возникающую, когда пользователь, владеющий открытым ключом алгоритма RSA, предоставляет услуги оракула расшифровки и возвращает в ответ на запрос целый блок исходного текста. Стойкость младшего значащего бита в алгоритме RSA означает, что пользователь не должен

играть роль ни оракула четности, ни $N/2$-оракула (как в лемме 9.1), т.е. не отвечать на шифрованные запросы о бите четности исходного текста и не сообщать, превышает сообщение m число $N/2$ или нет.

Следует предупредить, что атакующий может маскировать свои запросы под видом невинного протокола. Рассмотрим следующий пример.

Пример 9.2. Если Алиса и Злоумышленник согласились разделить секретный сеансовый ключ только между собой, то Злоумышленник может сделать следующее вполне разумное предложение.

> "Алиса, что если мы пошлем друг другу 1 000 сообщений, зашифрованных нашим открытыми ключами? Пусть сеансовый ключ будет строкой битов, представляющей собой результат применения операции XOR к битам четности из каждой пары исходных сообщений. Кстати, для того, чтобы убедиться, что сеансовый ключ является случайным, позволь мне послать свои 1 000 блоков первым!"

Алиса не только согласна, но и очень благодарна Злоумышленнику за оказанное ей высокое доверие (сгенерировать случайный сеансовый ключ!) Однако Злоумышленник посылает ей 1 000 зашифрованных сообщений, имеющих вид $(2^i)^e c (\mathrm{mod}\, N)$ $(i = 1, 2, \ldots, 1\,000)$, где c — шифрованный текст, когда-то посланный Алисой и перехваченный Злоумышленником.

Выполнив протокол, Злоумышленник заявляет об ошибке при вычислении сеансового ключа.

> "Алиса, мне очень жаль, но я запутался в своих вычислениях. Не могла бы ты переслать мне сеансовый ключ? Пожалуйста, зашифруй его моим открытым ключом."

Бедная Алиса спешит на помощь. Увы, имея сеансовый ключ, Злоумышленник может восстановить бит четности и применить алгоритм 9.1 для взлома исходного текста, содержащегося в зашифрованном тексте c! □

В данном случае Злоумышленник организовал активную атаку: он модифицирует зашифрованный текст c, умножая его на число $(2^i)^e (\mathrm{mod}\, N)$. Следовательно, несмотря на то, что восстановить младший значащий бит так же сложно, как и весь блок, функция шифрования в алгоритме RSA остается безнадежно уязвимой для активной атаки.

9.3 Битовая стойкость алгоритма Рабина

Если шифрование имеет вид $c = m^2 (\mathrm{mod}\, N)$ (т.е. показателем степени при шифровании является число $e = 2$), алгоритм 9.1 можно легко модифицировать и применить к схеме Рабина.

Однако существуют некоторые сложности. Если число N имеет два разных простых множителя, то по теореме 6.17 (раздел 6.6.2) любое число $c \in \mathrm{QR}_N$ имеет четыре разных квадратных корня по модулю N, т.е. зашифрованному тексту c соответствуют четыре разных исходных текста. Оракул четности, проверяющий четность случайного квадратного корня числа c (т.е. корня, случайно выбранного из четырех), является ненадежным. Несмотря на это, если квадратный корень обладает определенными свойствами, обеспечивающими детерминированность (а значит, и надежность) оракула четности, то к схеме Рабина также можно применять алгоритм бинарного поиска.

Например, оракул будет детерминированным, если он проверяет бит четности меньшего из квадратных корней положительного символа Якоби. По теореме 6.18 (раздел 6.7), если число N является целым числом Блюма, то любой квадратичный вычет c имеет два корня m и $-m$, являющихся корнями положительного символа Якоби. Поскольку число N — нечетное, только один из этих двух корней меньше $N/2$. Его можно назвать "меньшим из корней числа c, являющихся корнями положительного символа Якоби".

Предположим, что целое число Блюма N удовлетворяет условию $\left(\frac{2}{N}\right) = 1$, где $N = pq$, причем $p \equiv q \pmod 8$. Тогда оракул четности, проверяющий бит четности меньшего из квадратных корней положительного символа Якоби, вполне работоспособен. Заметим, что при условии $\left(\frac{2}{N}\right) = 1$ в ответ на i-й запрос надежный оракул четности вернет исходный текст $\frac{m}{2^i} \pmod N$ и определит знак символа Якоби для всех i запросов об исходном тексте. Детали модифицированного алгоритма бинарного поиска в схеме Рабина приведены в работе [128].

9.3.1 Генератор псевдослучайных битов Блюма–Блюма–Шаба

Применение алгоритма бинарного поиска к функции шифрования Рабина было основано на предположении, что младший значащий бит исходного сообщения в схеме Рабина является стойким, если выполняются условия неразрешимости задачи о разложении целого числа на простые множители (предположение 8.4 из раздела 8.8). Стойкость младшего значащего бита в схеме Рабина имеет важное практическое применение: она используется при генерировании **криптографически стойких псевдослучайных битов** (cryptographically strong pseudo-random bits — CSPRB) [42]. Так называемый генератор псевдослучайных чисел Блюма–Блюма–Шаба (Blum–Blum–Shub pseudo-random number generator — BBS) использует в качестве начального значения число $x_0 \in \mathrm{QR}_N$, где N — k-битовое целое число Блюма. Затем псевдослучайные биты, сгенерированные датчиком BBS с помощью начального значения x_0, образуются из младших значащих битов каждого

числа следующей последовательности.

$$x_0, x_1 = x_0^2, \ldots, x_i = x_{i-1}^2, \ldots \pmod N \tag{9.3.1}$$

Как показано в работах [42, 128], не зная начальное значение x_0, предсказать младшие значащие биты в последовательности (9.3.1) так же сложно, как разложить целое число Блюма N на простые множители.

Примечание 9.1. *В работах [13, 295] доказано, что задача одновременного вычисления* $\log_2 \log_2 N$ *младших значащих битов в зашифрованном тексте Рабина эквивалентна решению задачи о разложении числа* N *на простые множители.* □

Блюм и Голдвассер (Goldwasser) применили этот результат и предложили эффективную криптосистему, обладающую **семантической стойкостью** (semantic security). Эта криптосистема изучается в главе 14.

9.4 Битовая стойкость криптосистемы Эль-Гамаля

Поскольку в криптосистеме Эль-Гамаля пространством исходных сообщений является группа $\mathbb{F}_p^*$, где p — большое простое число (следовательно, нечетное), к ней также можно применить алгоритм бинарного поиска. Для того чтобы найти исходное сообщение, зашифрованное в виде пары (c_1, c_2), оракулу четности посылаются запросы, имеющие следующий вид.

$$\left(c_1, 2^i c_2\right) \pmod p, i = 1, 2, \ldots, \lfloor \log_2 p \rfloor + 1.$$

Если оракулом четности является человек (что вполне вероятно, см. пример 9.2), то для того, чтобы снять подозрения, атакующий может замаскировать запросы, представив их следующим образом.

$$\left(g^{r_i} c_1, 2^i y^{r_i} c_2\right) \pmod p, \text{ где } r_i \in_U \mathbb{Z}_{p-1}, i = 1, 2, \ldots, \lfloor \log_2 p \rfloor + 1,$$

где пара (g, y) содержит параметры открытого ключа, принадлежащего оракулу четности. Хотя эти $\lfloor \log_2 p \rfloor + 1$ пар зашифрованных сообщений абсолютно независимы друг от друга, они шифруют зависимые сообщения $2^i m \pmod p$, где $i = 1, 2, \ldots, \lfloor \log_2 p \rfloor + 1$.

$$\left(g^{k+r_i}, y^{k+r_i} 2^i m\right) \pmod p, i = 1, 2, \ldots, \lfloor \log_2 p \rfloor + 1.$$

Отсюда следует, что распознать младший значащий бит в схеме Эль-Гамаля так же сложно, как взломать весь блок сообщения. С другой стороны, владелец открытого ключа должен проявлять осторожность, чтобы не поддаться на уловки, описанные в примере 9.2.

9.5 Битовая стойкость дискретного логарифма

В разделе 8.4 мы выяснили, что проблема вычисления дискретного логарифма в абелевой группе в общем случае весьма сложна: функция g^x считается однонаправленной. Однако до сих пор неизвестно, является ли она функцией с секретом. По этой причине определять число x по степени g^x — довольно странная идея. Однако для того, чтобы продемонстрировать связь между битовой и блочной стойкостью дискретного логарифма, предположим, что существует оракул, который может сообщать частичную информацию о битах числа x, получая пару (g, g^x).

Если элемент g имеет нечетный порядок $\mathrm{ord}(g)$, не являющийся секретом, можно найти число $\frac{1}{2}(\mathrm{mod}\,\mathrm{ord}(g))$. (Эта ситуация встречается довольно часто.) В этом случае дискретное логарифмирование с помощью оракула четности, по существу, сводится к побитовому выполнению операции, обратной к возведению в степень по модулю (алгоритм 4.3). Поскольку алгоритм возведения в степень по модулю называется также методом "возведения в квадрат и умножения" ("squaring-and-multiplying"), обратный алгоритм можно назвать методом "извлечения квадратного корня и деления" ("square-rooting-and-dividing").

Алгоритм 9.2. Дискретное логарифмирование с помощью оракула четности

ИСХОДНЫЕ ДАННЫЕ:

(g, h) : g — элемент группы, имеющий нечетный порядок, $h = g^x$;
$PO_{\mathrm{desc}(g)}$: оракул четности; $PO_{\mathrm{desc}(g)}(g, h) = \log_g h (\mathrm{mod}\, 2)$.

РЕЗУЛЬТАТ:

x: целое число.

1. Выполнить присваивание $x \leftarrow 0, y \leftarrow \frac{1}{2}(\mathrm{mod}\,\mathrm{ord}(g))$.
2. Повторять следующие шаги, пока $h \geqslant 1$.

{

а) If $\left(PO_{\mathrm{desc}(g)}(g, h) == 1\right)$ then $h \leftarrow h/g; x \leftarrow x + 1$.
(* Если $\log_g h$ — нечетное число, выполнить операцию "поделить и прибавить единицу", обратную по отношению к операции "умножить и вычесть единицу" при возведении в степень по модулю. *)

б) $h \leftarrow h^y; x \leftarrow 2x$.
(* Если $\log_g h$ — четное число, выполнить операцию "извлечь квадратный корень и удвоить", обратную по отношению к операции "возвести в квадрат и поделить пополам" при возведении в степень по модулю. *)

}

3. return (x).

Что произойдет, если элемент g имеет четный порядок? Например, если число g — это порождающий элемент группы $\mathbb{F}_p^*$, где p — простое число, то число $\mathrm{ord}_p(g) = p - 1$ является четным и не является взаимно простым с числом 2. Следовательно, числа $\frac{1}{2}(\mathrm{mod}\, p - 1)$ не существует. Таким образом, квадратный корень из числа h на шаге 2.2 извлечь невозможно.

Однако, если число g является порождающим элементом группы $\mathbb{F}_p^*$, квадратный корень из числа h по модулю p можно вычислить с помощью алгоритма 6.4. Для любого квадратичного вычета $h \in \mathrm{QR}_p$ алгоритм извлечения квадратного корня возвращает два квадратных корня числа h, которые обозначаются как $\pm\sqrt{h}$. Поскольку число g является порождающим элементом группы $\mathbb{F}_p^*$, выполняется условие $g \in \mathrm{QNR}_p$, но $h \in \mathrm{QR}_p$. Следовательно, число $\log_g h$ должно быть четным. Итак, не ограничивая общности, можно утверждать, что дискретные логарифмы двух квадратных корней h по основанию g вычисляются по следующим формулам.

$$\log_g \sqrt{h} = \frac{\log_g h}{2}, \log_g \left(-\sqrt{h}\right) = \frac{p-1}{2} + \frac{\log_g h}{2}. \tag{9.5.1}$$

Заметим: из-за того, что сложение в формуле (9.5.1) выполняется по модулю $p - 1$, только одна из этих величин меньше $\frac{p-1}{2}$, а другая должна быть больше или равной этому числу. Очевидно, что квадратный корень, имеющий меньший дискретный логарифм по основанию g, является искомым. Однако проблема заключается в том, что по числам $\sqrt{h}$ и $-\sqrt{h}$ невозможно определить, какой из этих квадратных корней имеет меньший дискретный логарифм по основанию g!

Несмотря на это, если существует другой "битовый оракул", который по тройке $(g, y, -y)$ определяет, какое из чисел y и $-y$ имеет меньший дискретный логарифм по основанию g (подобный оракул называется "половинчатым" ("half-order oracle")), его можно применить для определения искомого квадратного корня. Функционирование такого "половинчатого" оракула описывается алгоритмом 9.3.

Поскольку проверка символа Лежандра и вычисление квадратных корней по модулю простого числа допускают эффективную реализацию, алгоритм 9.3 означает, что проверка неравенства $\log_g h < \frac{\mathrm{ord}(g)}{2}$ для заданной пары (g, h) эквивалентна вычислению величины $\log_g h$ по заданной паре (g, h).

Алгоритм 9.3 носит более общий характер, чем алгоритм 9.2. Он сохраняет работоспособность, даже если число g имеет нечетный порядок. Рассмотрим небольшой числовой пример.

Пример 9.3. Допустим, мы имеем доступ к половинчатому оракулу. Рассмотрим группу $\mathbb{Z}_{23}^*$, порождающий элемент $g = 5$ и элемент $h = 9$. Вычислим величину $x = \log_5 9(\mathrm{mod}\, 22)$, вызывая "половинчатый" оракул.

Алгоритм 9.3, на вход которого поступает тройка $(5, 9, 23)$, работает так, как показано на рисунке. Каждая двойная стрелка означает "извлечение квадратного

Алгоритм 9.3. Дискретное логарифмирование с помощью "половинчатого" оракула

ИСХОДНЫЕ ДАННЫЕ:

(g, h, p) : g — порождающий элемент группы $\mathbb{F}_p^*$,
p — простое число, $h = g^x \pmod p$,
$PO_{(p,g)}$: "половинчатый оракул".

$$PO_{(p,g)}(g, y, -y) = \begin{cases} y, & \text{если } \log_g y < \log_g(-y), \\ -y & \text{в противном случае.} \end{cases}$$

РЕЗУЛЬТАТ:

x: целое число.

1. Выполнить присваивание $x \leftarrow 0$.
2. Повторять следующие шаги, пока $h \geqslant 1$.
   {
   а) If $(h \in \mathrm{QNR}_p)$ then $h \leftarrow h/g; x \leftarrow x + 1$.
   (* Если $h \in \mathrm{QNR}_p$, то $\log_g h$ — нечетное число. В этом можно убедиться, проверив символ Лежандра $\left(\frac{h}{p}\right) = -1$. *)
   б) $h \leftarrow PO_{(p,g)}\left(g, \sqrt{h}, -\sqrt{h}\right); x \leftarrow 2x$.
   (* Извлечение квадратного корня выполняется без труда, но для этого необходим оракул, сообщающий нам, который из корней является искомым, т.е. имеет меньший дискретный логарифм $\frac{\log_g h}{2}$. *)
   }
3. return (x).

корня", причем горизонтальные двойные стрелки ($\Rightarrow$) обозначают выбор "половинчатого" оракула. Каждая одинарная стрелка ($\rightarrow$) означает "деление числа g", где $g = 5$. Все вычисления производятся по модулю 23.

$$\begin{array}{ccccccccccc} 9 & \Rightarrow & 20 & \rightarrow & 4 & \Rightarrow & 2 & \Rightarrow & 5 & \rightarrow & 1 & \Rightarrow & 1 \\ \Downarrow & & & & \Downarrow & & \Downarrow & & & & \Downarrow & & \\ 3 & & & & 21 & & 18 & & & & 22 & & \end{array}$$

В начале алгоритма переменная x инициализируется нулем (шаг 1), для каждой двойной стрелки $\Rightarrow$ выполняется операция $x \leftarrow 2x$ (шаг 2.2), а для каждой одинарной стрелки $\rightarrow$ — операция $x \leftarrow x + 1$ (шаг 2.1). По завершении алгоритма заключительное значение переменной x равно 10. Действительно, $9 = 5^{10} \pmod{23}$. □

Эти результаты показывают, что распознать отдельный бит дискретного логарифма так же сложно, как и расшифровать весь блок. Теперь можно утверждать, что если порождающий элемент группы является квадратичным вычетом, то все биты дискретного логарифма, включая младший значащий бит, являются стойкими. Это обеспечивает семантическую стойкость криптосистемы Эль-Гамаля, которая будет описана в главе 14.

9.6 Резюме

Исследование битовой стойкости основных криптосистем с открытым ключом неизменно приводит к утешительным выводам: каждый отдельный бит исходного текста, скрытый такими функциями, распознать так же сложно, как и весь блок. Если исходное сообщение является случайным, то извлечь информацию об исходном тексте так же трудно, как вычислить обратную функцию шифрования.

Многие исследователи использовали этот факт для создания более стойких криптосистем с открытым ключом без применения основных примитивов. Идея этих криптосистем заключается в рандомизации исходных сообщений. В части V будет описана общая методология под названием **модель случайного оракула** (random oracle model), позволяющая создавать криптосистемы с открытым ключом (а также схемы цифровой подписи), стойкость которых является сильной и доказуемой.

Изучая основные криптографические функции с открытым ключом, мы убедились, что все они уязвимы для активных атак. Общая методология создания алгоритмов шифрования с открытым ключом, рассмотренная в части V, также предусматривает создание механизмов, позволяющих предотвратить активные атаки.

Упражнения

9.1. Решите еще три варианта задачи о двух итерациях, описанной в теореме 9.1.

   а) $(a, b) = (N/2, N), PO_N(2^{2e}c) = 1.$

   б) $(a, b) = (0, N/2), PO_N(2^{2e}c) = 0.$

   в) $(a, b) = (0, N/2), PO_N(2^{2e}c) = 1.$

9.2. При каких условиях алгоритм шифрования RSA обладает свойством битовой стойкости?

   Подсказка: может ли алгоритм шифрования обладать свойством битовой стойкости, если исходный текст содержит верифицируемую частичную информацию?

9.3. Означает ли свойство битовой стойкости, что основные алгоритмы шифрования с открытым ключом являются стойкими?

9.4. Что обеспечивает стойкость алгоритмов, использующих генератор псевдослучайных чисел Блюма–Блюма–Шаба?

9.5. Допустим, что p — простое число, а g — порождающий элемент группы $\mathbb{F}_p^*$. Сложность вычисления символа Лежандра для числа $g^x \pmod p$ эквивалентна сложности вычисления младшего значащего бита числа x. Почему вычисление числа x по числу $g^x \pmod p$ остается трудноразрешимой задачей?

Глава 10

Методы защиты целостности данных

10.1 Введение

В главе 2 сформулировано вполне реалистичное предположение о уязвимости открытых сетей: все сообщения проходят через Злоумышленника, который может их перехватывать, передавать, модифицировать, фальсифицировать или искажать. Если Злоумышленник модифицирует или фальсифицирует сообщения, он стремится уверить получателя в том, что они посланы законным пользователем. Для того чтобы защитить открытые системы связи и сделать их пригодными для электронной коммерции, недостаточно применять криптографические механизмы, предназначенные для сохранения конфиденциальности сообщений (т.е. для противодействия их перехвату). Для этого необходимы средства, позволяющие получателю убедиться, что сообщение послано из законного источника и не было изменено по дороге. Именно для этого предназначены методы защиты **целостности данных** (data integrity), предотвращающие несанкционированную модификацию сообщений.

Целостность данных в современной криптографии тесно связана с классическим понятием теории связи: кодом контроля ошибок (error-detection code), представляющим собой процедуру для обнаружения ошибок, которые могли возникнуть вследствие дефектов связи. Считается, что вред от использования сообщений, содержащих ошибки из-за несовершенства средств связи, сравним с вредом, возникающим при использовании преднамеренно искаженной информации. По этой причине принципы защиты целостности данных и методы распознавания ошибок, по существу, совпадают: отправитель сообщения присоединяет к нему "контрольное значение", а получатель обрабатывает это значение по определенным правилам, согласованным с отправителем [275]. В кодах с контролем ошибок избыточная информация кодируется так, чтобы получатель мог применить детектор, работающий по методу максимального правдоподобия, и распознать возможные изменения. В системах защиты целостности данных избыточная информация кодируется так, чтобы добавленное контрольное значение было как можно более

равномерно распределено по всему пространству сообщений, а вероятность его преднамеренного искажения была минимальной. Применяемое при этом криптографическое преобразование очень похоже на перемешивание при шифровании (раздел 7.1), хотя перемешивание при шифровании никак не связано с добавлением избыточной информации, предназначенной для верификации сообщений.

Подобно алгоритмам шифрования, криптографические преобразования, предназначенные для обеспечения целостности данных, параметризуются ключами. Следовательно, результат верификации целостности данных предоставляет проверяющему информацию об источнике сообщения, т.е. о пользователе, защитившем эти данные. Однако в последнее время появились методы "защиты данных без идентификации источника". Это новое понятие оказалось весьма полезным для защиты криптосистем с открытым ключом от адаптивных атак. В главе описывается пример такой системы. Это позволит в дальнейшем перейти к изучению стойкости криптосистем с открытым ключом против адаптивных атак.

10.1.1 Структурная схема главы

Сначала в главе вводится формальное определение защиты целостности данных (раздел 10.2) и описываются соответствующие криптографические методы. Затем рассматриваются симметричные (раздел 10.3) и асимметричные (раздел 10.4) методы защиты целостности данных, а также вводится понятие о защите целостности данных без идентификации источника (раздел 10.5).

10.2 Определение

Определение 10.1 (Защита целостности данных). *Пусть* Data *— произвольная информация. Обозначим через* Ke *ключ шифрования, а через* Kv *— ключ верификации, соответствующий ключу шифрования. Защита целостности данных* Data *сводится к следующим криптографическим преобразованиям.*

Создание кода для распознавания манипуляций:

$$\mathsf{MDC} \leftarrow f(Ke, \mathsf{Data}).$$

Верификация кода для распознавания манипуляций:

$$g(Kv, \mathsf{Data}, \mathsf{MDC}) = \\ = \begin{cases} \mathsf{True}, \text{ с вероятностью } 1, & \text{если } \mathsf{MDC} = f(Ke, \mathsf{Data}), \\ \mathsf{False}, \text{ с огромной вероятностью}, & \text{если } \mathsf{MDC} \neq f(Ke, \mathsf{Data}). \end{cases}$$

Здесь f *и* g *— эффективные криптографические преобразования. Преобразование* f *параметризовано вспомогательными данными* Ke *(ключом шифрования),*

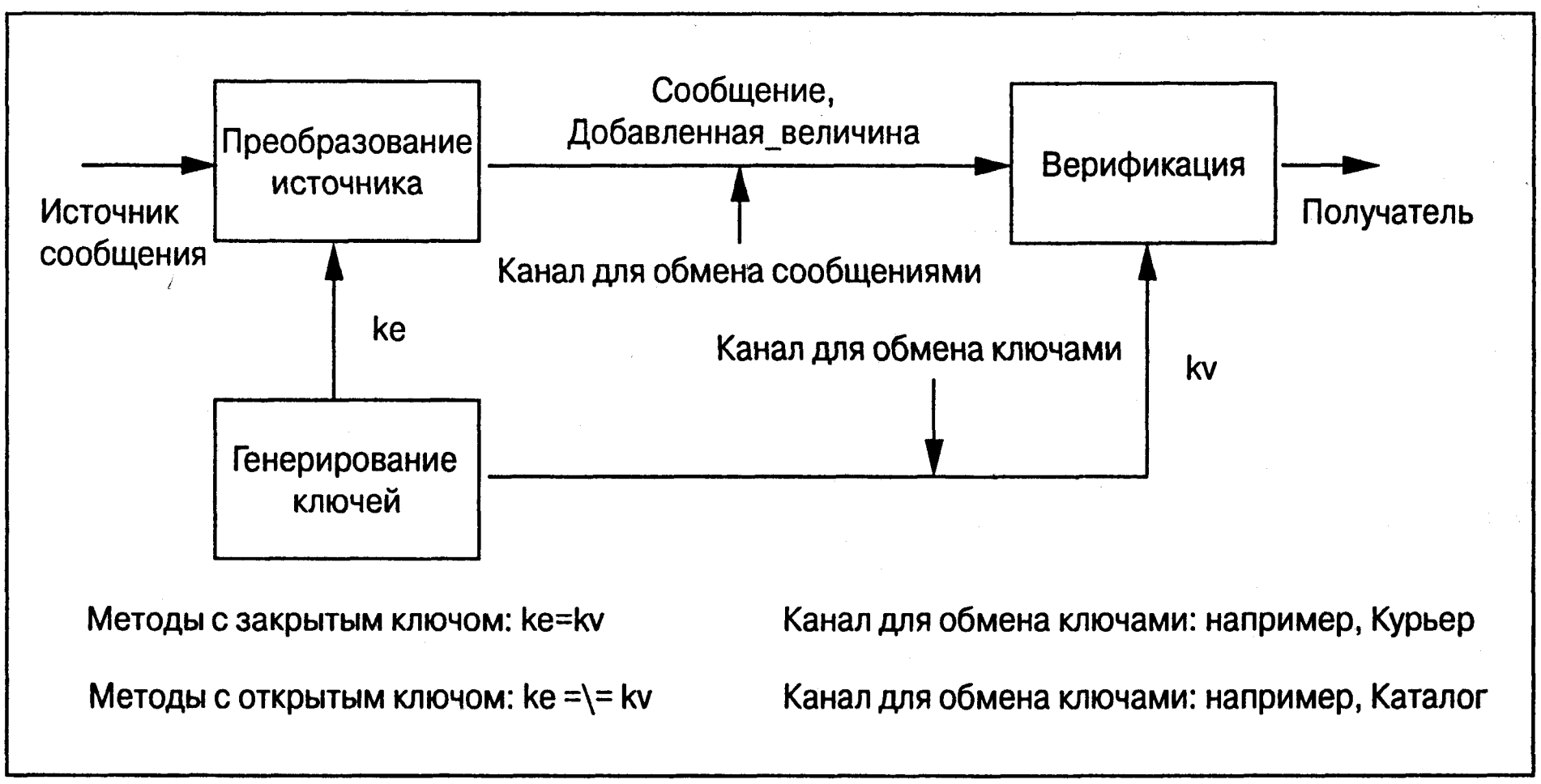


Рис. 10.1. Система защиты целостности данных

а преобразование g — вспомогательными данными Kv (ключом верификации). Аббревиатура MDC *означает код распознавания манипуляций (manipulation detection code). Вероятностное пространство состоит из пространства всевозможных вариантов информации* Data*, кода* MDC *и ключей, а также, возможно, из пространства случайных исходных данных, если алгоритмы подписи/верификации являются вероятностными.*

Система защиты целостности данных приведена на рис. 10.1.

Следует отметить, что в отличие от понятия защиты целостности данных, описанного во введении, определение 10.1 не ограничено системами связи: например, пара (Data, MDC) может храниться или извлекаться из открытого хранилища.

Как и криптосистемы, механизмы защиты целостности данных бывают симметричными и асимметричными. Однако следует подчеркнуть разницу между ними, проявляющуюся при использовании открытого ключа. В криптосистемах, основанных на асимметричных методах, открытый и закрытый ключи имеют фиксированное применение: открытый ключ предназначен для шифрования, а закрытый — для расшифровки. В системах защиты целостности данных, использующих асимметричные методы, открытый и закрытый ключи могут применяться как для шифрования, так и для расшифровки. Эти различия описываются в разделах 10.4 и 10.5.

10.3 Симметричные методы

В симметричных методах защиты целостности данных криптографические преобразования f и g (см. определение 10.1) представляют собой симметричные криптографические алгоритмы, т.е. $f = g$ и $Ke = Kv$. Иначе говоря, создание и верификация соответствия между данными Data и кодом MDC обеспечиваются одними и теми же криптографическими операциями.

Благодаря тесной связи между целостностью данных и аутентификацией сообщений (см. главу 11), код MDC, созданный с помощью симметричного криптографического метода, часто называется **кодом аутентификации сообщений** (message authentication code — MAC). Такой код можно создавать и верифицировать либо с помощью ключевой хэш-функции, либо с помощью алгоритма блочного шифрования.

10.3.1 Криптографические хэш-функции

Общий способ реализации кода MAC заключается в применении так называемой **ключевой хэш-функции** (keyed hash function). Сначала введем понятие криптографической функции хэширования.

Функция хэширования — это детерминированная функция, отображающая строку битов произвольной длины в хэшированное значение, представляющее собой строку битов фиксированной длины. Обозначим через h хэш-функцию, возвращающую строку фиксированной длины $|h|$. Желательно, чтобы функция h обладала следующими свойствами.

Свойство 10.1 (Свойства хэш-функции).

- **Преобразование перемешивания.** При любом аргументе x хэшированное значение $h(x)$ должно быть неотличимым с вычислительной точки зрения от строки битов, равномерно распределенных в интервале $[0, 2^{|h|})$. Понятие неотличимых с вычислительной точки зрения величин введено в определении 4.15 (раздел 4.7). Вследствие предположения 4.2 (раздел 4.7) это свойство вполне разумно.
- **Предотвращение коллизий**. Поиск двух величин x и y, удовлетворяющих условиям $x \neq y$ и $h(x) = h(y)$, должен представлять собой неразрешимую задачу. Для того чтобы это предположение было разумным, необходимо, чтобы область значений функции h была большой. Число $|h|$ не должно быть меньше 128 и, как правило, равно 160.
- **Сложность вычисления прообразов.** Задача вычисления входной строки x по заданному хэшированному значению $h = h(x)$ должна быть неразрешимой вычислительной задачей. В этом случае также необходимо, чтобы область значений функции h была достаточно большой.

- **Практическая эффективность.** Вычисление значения $h(x)$ должно быть ограничено полиномом небольшой степени (в идеальном случае — линейным), зависящим от размера строки x.

Свойства перемешивания и предотвращения коллизий можно обеспечить с помощью тех же функций, которые использовались в алгоритмах блочного шифрования (разделы 7.6, 7.7). Сложность вычисления прообразов достигается за счет сжатия данных, при которой теряется часть исходной информации, и, следовательно, затрудняется вычисление обратной функции.

Мы не будем рассматривать ни одной реальной функции хэширования. Пытливые читатели могут найти их описание в литературе (например, в работе [198]).

10.3.1.1 Применение функций хэширования в криптографии

Хэш-функции широко применяются в криптографии. Перечислим некоторые из их приложений.

- В алгоритмах цифровой подписи хэш-функции обычно используются для создания "конспекта сообщения", или "отпечатка сообщения". Это обеспечивает некоторую избыточность сообщения, подписанного так, что после хэширования оно содержит распознаваемую информацию. Применение хэш-функций для создания цифровых подписей рассматривается в разделе 10.4. В нем показано, что стойкость схемы цифровой подписи к фальсификациям сильно зависит от избыточной информации, содержащейся в подписанном сообщении. Формальное доказательство того, что функции хэширования обеспечивают доказуемую стойкость цифровых подписей, приведено в главе 16.

- В прикладных криптосистемах с открытым ключом хэш-функции широко используются для реализации механизма верификации правильности зашифрованных текстов. Такой механизм необходим для того, чтобы обеспечить доказуемую стойкость к активным атакам. Пример такого применения хэш-функций описан в разделе 10.5. Формальное доказательство того, что хэш-функции обеспечивают доказуемую стойкость криптосистем с открытым ключом, приведено в главе 15.

- В широком круге криптографических приложений функции хэширования используются в качестве псевдослучайных функций. К таким приложениям относятся согласование ключей (например, когда два пользователя используют общее начальное значение в качестве аргумента функции хэширования для получения разделенного значения ключа), протоколы аутентификации (например, когда два участника протокола подтверждают выполнение протокола, обмениваясь хэшированными значениями), протоколы электронной коммерции (например, для обеспечения накопления платежей для биржевой

игры [201, 297]), протоколы доказательства знания (например, для обеспечения автономного режима доказательства (см. раздел 18.3.2.2)). В книге изложено большое количество примеров применения функций хэширования.

10.3.1.2 Случайный оракул

Напомним свойство перемешивания, которое присуще функции хэширования: при любом аргументе хэшированное значение неотличимо с вычислительной точки зрения от строки битов, равномерно распределенных в области значений функции. Если заменить последнее выражение фразой "принадлежит генеральной совокупности равномерно распределенных чисел", мы получим весьма мощную гипотетическую функцию под названием **случайный оракул** (random oracle).

Мы назвали этого оракула весьма *мощным*, поскольку он обладает тремя свойствами: он является *детерминированным*, *эффективным* и обеспечивает *равномерное распределение результирующих значений*. Причина, по которой эта функция считается гипотетической, заключается в том, что все известные вычислительные модели не являются настолько мощными.

С одной стороны, нам известно, как эффективно сгенерировать равномерно распределенные случайные величины, например, подбрасывая монету. Однако этот способ не является детерминированным. С другой стороны, можно рассматривать множество равномерно распределенных независимых величин с детерминированной точки зрения, например, упорядочивая их так, чтобы между любыми двумя числами и расстоянием между ними в упорядоченном списке существовала детерминированная зависимость. Однако эти расстояния невозможно вычислить за полиномиальное время, зависящее от величины случайных чисел (для упорядочения n элементов списка необходимо $n \log n$ операций).

Равномерность и детерминированность величин, вычисляемых случайным оракулом, означает, что энтропия его результатов выше, чем энтропия чисел, поступающих на вход (определение энтропии дано в разделе 3.7). Однако, если теория Шеннона (теорема 3.2 в разделе 3.7) верна, детерминированная функция никогда не сможет увеличить энтропию. Следовательно, случайного оракула в реальном мире не существует.

Поскольку свойство перемешивания, которым обладает хэш-функция, является лишь предположением вычислительного характера (предположение 4.2 в разделе 4.7), реальная хэш-функция должна обеспечивать лишь вычислительную неразличимость в смысле определения 4.15 (раздел 4.7), т.е. ее результаты должны иметь некое распределение вероятностей в области значений, которое невозможно определить за полиномиальное время. Итак, реальные функции хэширования лишь имитируют поведение случайного оракула с высокой точностью.

Несмотря на это, функции хэширования играют важную роль в криптографии с открытым ключом. По существу, хэширование сообщений вносит в них необходимую избыточность, причем эта процедура носит детерминированный характер.

10.3.1.3 Атака на основе парадокса дней рождений

Предположим, что функция хэширования h действительно является случайным оракулом. В атаке по методу квадратного корня (атака на основе парадокса дней рождений, раздел 3.6) предполагается, что для обнаружения коллизии с ненулевой вероятностью достаточно выполнить $2^{\frac{|h|}{2}} = \sqrt{2^{|h|}}$ случайных вычислений значения функции хэширования. Для организации атаки на основе парадокса дней рождений атакующий должен сгенерировать пары "сообщение–хэшированное значение"

$$(m_1, h(m_1)), (m_2, h(m_2)), \ldots,$$

пока не обнаружатся два сообщения m и m', удовлетворяющих условиям

$$m \neq m', h(m) = h(m'). \tag{10.3.1}$$

Такая пара сообщений называется **коллизией** (collision) функции хэширования h. Разумеется, чтобы атака была успешной, сообщения m и m' должны сдержать осмысленную информацию. Например, рассмотрим хэшируемое сообщение, которое должно сопровождаться цифровой подписью (раздел 10.4) и представляет собой санкцию на оплату, имеющую вид

$$M = \text{Цена}, \text{Описание\_товара}, R,$$

где R — случайное число, предназначенное для рандомизации протокольных сообщений (что весьма желательно). Тогда сообщение в атаке на основе парадокса дней рождений выглядит следующим образом.

$$m = \text{Цена\_1}, \text{Описание\_товара}, r$$

и

$$m' = \text{Цена\_2}, \text{Описание\_товара}, r',$$

где Цена_1 $\neq$ Цена_2 и Описание_товара представляют собой фиксированные части сообщений, а коллизия возникает между случайными числами $r \neq r'$. Поиск коллизий между этими сообщениями так же сложен, как и поиск коллизий между случайными сообщениями, удовлетворяющими условиям (10.3.1), поскольку, как легко видеть, функции

$$h'(x) \stackrel{\text{def}}{=} h(\text{Цена\_1}, \text{Описание\_товара}, x)$$

и

$$h''(x) \stackrel{\text{def}}{=} h(\text{Цена\_2}, \text{Описание\_товара}, x)$$

являются случайными.

Очевидно, что, если функции хэширования не являются абсолютно случайными, количество вычислений сократится.

Итак, размер пространства значений криптографической функции хэширования должен быть меньше. В современной криптографии широко используются функции хэширования SHA-1 [217] и RIPEMD-160 [53]. В обеих функциях выполняется условие $|h| = 160$. Их стойкость к атаке на основе парадокса "дней рождения" оценивается величиной 2^{80}. Эта стойкость сравнима со стойкостью алгоритма блочного шифрования с 80-битовым ключом. Более ранняя популярная хэш-функция MD5 [243] использовала длину строки $|h| = 128$, согласованную с алгоритмом DES, в котором ключ состоит из 56 бит и длина блока равна 64 бит.

С появлением алгоритмов AES-128, AES-192 и AES-256 (с ключами, состоящими из 128, 192 и 256 бит соответственно (раздел 7.7)), организации по стандартизации (например, ISO/IES [151]) приступили к стандартизации функций хэширования с подходящей длиной строки $|h| \in \{256, 384, 512\}$.

10.3.2 Коды аутентификации сообщений, использующие функции хэширования с ключом

Криптографические функции хэширования представляют собой естественное средство для защиты целостности данных. В сценарии с распределенными ключами на вход хэш-функции поступает ключ. Остальная часть входной информации состоит из сообщения, подлежащего аутентификации. Таким образом, чтобы аутентифицировать сообщение M, отправитель вычисляет значение

$$\text{MAC} = h(k \parallel M),$$

где k — секретный ключ, распределенный между отправителем и получателем, а символ $\parallel$ обозначает операцию конкатенации битовых строк.

Основываясь на свойствах функции хэширования, перечисленных в разделе 10.3.1, можно предположить, что для создания правильного кода MAC с помощью хэш-функции, получающей в качестве аргументов ключ k и сообщение M, пользователь должен обладать правильным ключом и правильным сообщением. Получатель, владеющий общим с отправителем ключом, должен заново вычислить код MAC на основе полученного сообщения M и проверить, соответствует ли он полученному коду MAC. Если эти коды совпадают, значит, сообщение поступило от законного отправителя.

Поскольку такие коды MAC вычисляются с помощью функции хэширования, они стали называться кодами HMAC (hash message authentication code). Довольно часто коды HMAC вычисляются по формуле

$$\text{HMAC} = h(k \parallel M \parallel k),$$

т.е. ключ присоединяется к сообщению M в качестве префикса и суффикса [288]. Это не позволяет противнику использовать циклическую структуру некоторых функций хэширования. Если сообщение не защищено ключом с обоих концов, такая структура некоторых функций хэширования позволяет противнику, даже не знающему ключа, модифицировать сообщение с помощью произвольного префикса или суффикса.

10.3.3 Коды аутентификации сообщений, использующие алгоритм блочного шифрования

Как правило, функции хэширования с ключом образуются с помощью алгоритмов блочного шифрования в режиме сцепления блоков зашифрованного текста. Функция хэширования с ключом, построенная таким образом, называется **функцией MAC**.

Пусть $\mathcal{E}_k(m)$ — алгоритм блочного шифрования сообщения m с ключом k. Для того чтобы аутентифицировать сообщение M, отправитель сначала разбивает его на блоки:

$$M = m_1 m_2 \ldots m_\ell,$$

где размер каждого блока $m_i (i = 1, 2, \ldots, \ell)$ равен размеру входных данных алгоритма блочного шифрования. Если размер последнего блока m_ℓ не равен размеру полного блока, он дополняется случайной величиной. Пусть $C_0 = IV$ — случайный вектор инициализации. Отправитель применяет алгоритм блочного шифрования в режиме сцепления блоков зашифрованного текста:

$$C_i \leftarrow \mathcal{E}_i(m_i \oplus C_{i-1}), i = 1, 2, \ldots, \ell.$$

Теперь пару (IV, C_i) можно применять в качестве кода аутентификации сообщений, который присоединяется к сообщению M перед отправкой.

Очевидно, что вычисление кода аутентификации сообщений в режиме сцепления блоков зашифрованного текста CBC-MAC предусматривает необратимое сжатие данных (по существу, код CBC-MAC представляет собой "конспект сообщения"). По этой причине преобразование CBC-MAC является однонаправленным. Более того, свойство перемешивания соответствующего алгоритма блочного шифрования придает этому преобразованию характер хэширования (т.е. распределяет код MAC по всему пространству кодов так же равномерно, как исходный алгоритм блочного шифрования распределяет сообщение по всему пространству зашифрованных текстов). Итак, можно предполагать, что для создания корректного кода CBC-MAC пользователь действительно должен владеть ключом k соответствующего алгоритма блочного шифрования. Получатель, разделяющий ключ k с отправителем, должен заново вычислить код MAC на основе полученного сообщения и проверить, совпадает ли он с полученным вариантом. Если совпадает, сообщение было послано законным отправителем.

Код аутентификации сообщений, обеспечивающий целостность данных в сообщении M для пользователей, владеющих ключом k, иногда обозначается как $\mathrm{MAC}(k, M)$. В этом обозначении игнорируются детали реализации, например, однонаправленное преобразование, использованное при вычислении кода.

10.4 Асимметричные методы I: цифровые подписи

В криптографии с открытым ключом пользователь может применять для шифрования сообщения свой закрытый ключ, а для расшифровки — открытый ключ. Очевидно, что зашифрованный текст, созданный таким способом, может использоваться в качестве кода распознавания манипуляций (MDC), сопровождающего зашифрованное сообщение, т.е. обеспечивать защиту целостности данных. Здесь процесс расшифровки с помощью открытого ключа представляет собой верификацию кода MDC.

Более того, верификацию кода может осуществить кто угодно, поскольку открытым ключом владеют все. В противоположность этому считается, что создать код MDC с помощью соответствующего открытого ключа мог только владелец закрытого ключа, использованного для создания этого кода. Такое применение криптографии с открытым ключом — **цифровая подпись** (digital signature) — позволяет установить автора сообщения. Иначе говоря, криптографию с открытым ключом, а точнее, однонаправленную функцию с секретом (см. свойство 8.1 в разделе 8.1) можно применять для реализации цифровой подписи.[1] Первыми цифровую подпись изобрели Диффи и Хеллман [97] (эта работа опубликована в 1976 году, хотя распространялась в качестве препринта с 1975 года [96]).

Возможность создавать цифровые подписи предоставляет криптографии с открытым ключом огромное преимущество над криптографией с секретным ключом (вторым достоинством криптографии с открытым ключом является возможность распределять ключи между удаленными пользователями, как показано в разделе 8.15). Теперь, когда только один пользователь может создать цифровую подпись сообщения, которую может верифицировать кто угодно, легко разрешить вопрос о его авторстве. Это дает возможность предотвратить **отказ от авторства** (non-repudiation), т.е. отрицать свою связь с посланным сообщением. Предотвращение отказа от авторства представляет собой важное требование безопасности в электронной коммерции.

Дадим формальное определение цифровой подписи.

---

[1] Несмотря на то что теоретическую основу цифровых подписей образуют однонаправленные функции [173], на практике для их создания используются однонаправленные функции с секретом.

Определение 10.2 (Схема цифровой подписи). *Схема цифровой подписи состоит из следующих компонентов.*

- *Пространство исходных сообщений $\mathcal{M}$: множество строк над некоторым алфавитом.*
- *Пространство подписей $\mathbb{S}$: пространство возможных подписей.*
- *Пространство ключей подписи $\mathcal{K}$: пространство возможных ключей, применяемых при создании подписи, и пространство ключей верификации $\mathcal{K}'$: множество возможных ключей, применяемых для верификации подписи.*
- *Эффективный алгоритм генерации ключа* $\mathsf{Gen} : \mathbb{N} \mapsto \mathcal{K} \times \mathcal{K}'$, *где $\mathcal{K}$ и $\mathcal{K}'$ — пространства закрытых и открытых ключей соответственно.*
- *Эффективный алгоритм подписи* $\mathsf{Sign} : \mathcal{M} \times \mathcal{K} \mapsto \mathbb{S}$.
- *Эффективный алгоритм верификации* $\mathsf{Verify} : \mathcal{M} \times \mathcal{K} \times \mathcal{K}' \mapsto \{\mathsf{True}, \mathsf{False}\}$.

Для любого ключа подписи $sk \in \mathcal{K}$ и любого сообщения $m \in \mathcal{M}$ операция подписи обозначается как

$$s \leftarrow \mathsf{Sign}_{sk}(m).$$

Это выражение читается следующим образом: "s — подпись сообщения m, созданного с помощью ключа sk".

Для любого секретного ключа подписи $sk \in \mathcal{K}$, соответствующего открытого ключа pk, любого сообщения $m \in \mathcal{M}$ и подписи s должно выполняться условие

$$\mathsf{Verify}_{pk(m,s)} = \begin{cases} \mathsf{True}\ \textit{с вероятностью 1}, & \textit{если } s \leftarrow \mathsf{Sign}_{sk(m)}, \\ \mathsf{False}\ \textit{с огромной вероятностью}, & \textit{если } s \leftarrow \mathsf{Sign}_{sk(m)}, \end{cases}$$

где вероятностное пространство содержит пространства $\mathbb{S}, \mathcal{M}, \mathcal{K}$ и $\mathcal{K}'$ и, возможно, пространство случайных исходных данных, если алгоритмы подписи/верификации являются вероятностными.

Это определение можно считать частным вариантом определения 10.1: пары (Sign, Verify), (sk, pk) и (m, s) соответствуют парам (f, g), (Ke, Kv) и (Data, MDC).

Заметим, что размер ключей подписи/верификации определяется целыми числами, поступающими на вход алгоритма генерации ключей Gen. Поскольку этот алгоритм является эффективным, и время его работы полиномиально зависит от размера целого числа, поступающего на вход, это число следует представлять в унарном виде (причины указаны в определении 4.7 в разделе 4.4.6.1). Это целое число представляет собой параметр безопасности схемы цифровой подписи и определяет размер пространства подписей.

Поскольку размер пространства подписей зависит от параметра безопасности, смысл выражения "огромная вероятность" в формуле "$\mathsf{Verify}_{pk}(m, s) = \mathsf{False}$

с огромной вероятностью, если $s \leftarrow \mathsf{Sign}_{sk}(m)$" полностью соответствует понятию, определенному в разделе 4.6. Однако эта вероятность не должна учитывать вариант, когда подпись допускает легкую фальсификацию, как указано в замечании 10.1. Количественная оценка этой вероятности будет дана в ходе формального доказательства стойкости некоторых реальных схем цифровой подписи в главе 16.

Формально говоря, свойство перемешивания, присущее алгоритмам шифрования (раздел 7.1), также играет важную роль в схемах цифровых подписей. Алгоритм Sign должен равномерно распределять цифровые подписи по всему пространству подписей $\mathcal{S}$. Это предотвращает подделку подписей без помощи соответствующего ключа.

10.4.1 Учебные схемы цифровых подписей

Как и алгоритмы шифрования с открытым ключом, рассмотренные в главе 8, описываемые в этой главе схемы цифровой подписи являются очень нестойкими.

Свойство 10.2 (Учебные схемы цифровой подписи). В рамках этой главы рассматривается ограниченное понятие стойкости схем цифровой подписи. Будем говорить, что цифровая подпись является стойкой, если ее фальсификация требует огромных усилий. Иначе говоря, атакующий по заданному открытому ключу и описанию схемы цифровой подписи не в состоянии создать пару "сообщение-подпись", которая была бы приписана законному владельцу открытого ключа. Атакующий не способен адаптироваться, т.е. он не стремится облегчить процесс фальсификации цифровой подписи, используя доступные ему пары "сообщение-подпись" или взаимодействуя с законным автором сообщения.

Следует заметить, что это понятие стойкости совершенно не пригодно для практического применения, поскольку основано на предположении, что атакующий неестественно слаб и находится в агрессивном окружении. На самом деле атакующий может легко получить в свое распоряжение пары "сообщение-подпись" вместе с открытым ключом и описание цифровой подписи, поскольку это не является секретной информацией. Кроме того, как правило, атакующий имеет возможность обращаться к источнику с просьбой подписать сообщение по своему выбору. Такой атакующий умеет адаптироваться, поскольку он выбирает сообщения по своему усмотрению. В ходе атаки на основе адаптивно подобранного сообщения (adaptive chosen–message attack) атакующий получает **целевое сообщение** (target message), подбирает на его основе подходящие сообщения (возможно, выполняя алгебраические преобразования исходного сообщения) и посылает их автору на подпись. Все это выглядит так, будто автор обучает злоумышленника подделывать свою подпись. Задачей атакующего является подделка подписи в целевом сообщении. Как указано в разделе 8.6 при обсуждении эффективности адаптивных атак на криптосистемы, атака на основе адаптивно подобранного

сообщения не только намного эффективнее обычных атак, но и вполне реальна. Таким образом, ей следует уделить особое внимание.

Напомним, что до сих пор мы рассматривали теоретическую стойкость учебных алгоритмов шифрования с открытым ключом. Мы несколько раз предупреждали, что владелец открытого ключа не должен предоставлять атакующему услуги оракула. Такой уровень бдительности можно обеспечить только в том случае, если владелец ключа достаточно опытен. Следовательно, для предотвращения адаптивных атак не следует полагаться на квалификацию пользователей. При описании схем цифровых подписей мы не будем требовать от пользователя не предоставлять услуги оракула, поскольку этого иногда невозможно избежать: во многих приложениях подпись присланных сообщений считается вполне естественной.

Строгое понятие стойкости схем цифровой подписи, которое можно назвать стойкостью к **атакам на основе адаптивно подобранного сообщения** (adaptive chosen-message attack), будет сформулировано в главе 16. Оно является аналогом понятия практической стойкости криптосистем к атакам CCA2 (определение 8.3 в разделе 8.6).

Отметим простой, но вполне невинный вид подделки подписи.

Примечание 10.1 (Экзистенциальная подделка). *Алгоритмы* $(\mathsf{Sign}_{sk}, \mathsf{Verify}_{pk})$ *образуют пару однонаправленных функций с секретом. Алгоритм* Verify_{pk} *представляет собой однонаправленный компонент пары, а* Sign_{sk} *— функцию с секретом. Как правило, функция* $\mathsf{Verify}_{pk}(s, m)$ *вычисляется в направлении от* s *к* m*. Следовательно, многие схемы цифровой подписи, основанные на однонаправленных функциях с секретом, позволяют эффективно подделывать правильные пары "сообщение-подпись", применяя функцию* Verify_{pk} *в направлении от* s *к* m*. Однако благодаря свойству перемешивания, которым обладает функция* Verify_{pk}*, "сообщение", порожденное на основе "подписи" с помощью функции* Verify_{pk}*, выглядит случайным и бессмысленным. Такой способ подделки цифровых подписей называется* ***экзистенциальным*** *(existential forgery). Схемы цифровой подписи, основанные на применении однонаправленных функций с секретом, как правило, допускают экзистенциальную подделку. Предотвратить такие подделки можно с помощью включения в сообщение избыточной информации, позволяющей верификатору выявить неслучайное распределение сообщений.* □

Рассмотрим несколько схем цифровой подписи.

10.4.2 Цифровая подпись RSA (учебный вариант)

Схема подписи RSA впервые была предложена Диффи и Хеллманом, а затем реализована Ривестом, Шамиром и Адлеманом [246].

Алгоритм 10.1. Схема цифровой подписи RSA

Генерация ключа

Процедура генерации ключа такая же, как и в криптосистеме RSA (алгоритм 8.1). (* Следовательно, параметрами ключа Алисы является пара (N, e), где $N = pq$, p и q — большие простые числа приблизительно одинаковой величины, e — целое число, удовлетворяющее условию $\gcd(e, \phi(N)) = 1$. Алиса также определяет целое число d, удовлетворяющее условию $ed \equiv 1 (\mathrm{mod}\, \phi(N))$. Целое число d является закрытым ключом Алисы. *)

Генерация подписи

Для цифровой подписи сообщения $m \in \mathbb{Z}_N^*$ Алиса генерирует число

$$s = \mathsf{Sign}_d(m) \leftarrow m^d (\mathrm{mod}\, N).$$

Верификация подписи

Допустим, Боб должен удостовериться, что параметры открытого ключа (N, e) действительно принадлежат Алисе. Боб применяет к паре (m, s) следующую процедуру верификации.

$$\mathsf{Verify}_{(N,e)}(m, s) = \mathsf{True}, \text{ если } m \equiv s^e (\mathrm{mod}\, N).$$

(* *Примечание*: сообщение m должно быть распознаваемым (см. раздел 10.4.3). *)

Очевидно, что процедура цифровой подписи RSA совпадает с алгоритмом шифрования и расшифровки RSA (раздел 8.5), за исключением того обстоятельства, что Алиса сначала шифрует сообщение своим закрытым ключом, а Боб (или кто-либо еще) впоследствии расшифровывает его с помощью открытого ключа Алисы. Верификация подписи основана на тех же самых соображениях, что и алгоритмы шифрования и расшифровки RSA в разделе 8.5.

10.4.3 Неформальное обоснование стойкости цифровой подписи RSA

Если бы схема цифровой подписи RSA была настолько простой, как мы ее описали, ее было бы нетрудно подделать. Например, Боб может извлечь случайное число $s \in \mathbb{Z}_N^*$ и вычислить величину

$$m \leftarrow s^e (\mathrm{mod}\, N). \tag{10.4.1}$$

Разумеется, процедура верификации пары (m, s) вернет результат True. Кроме того, благодаря мультипликативному свойству функции RSA (8.9.1) по существующим парам "сообщение-подпись" (m_1, s_1) и (m_2, s_2) нетрудно создать новую, фальсифицированную пару $(m_1 m_2, s_1 s_2)$.

Как сказано в замечании 10.1, описанный выше способ фальсификации подписи называется экзистенциальным. Поскольку сообщение, созданное либо процедурой (10.4.1), либо в результате умножения, выглядит совершенно случайным, экзистенциальную фальсификацию можно предотвратить, включив в сообщение m дополнительную *распознаваемую* информацию так, чтобы сообщение m стало неслучайным или "осмысленным". Простейший способ добавить такую информацию — вставить в сообщение распознаваемую строку, т.е. $m = M \parallel I$, где M — подписанное сообщение, а I — распознаваемая строка, например, имя автора.

Наиболее часто для включения дополнительной информации сообщение хэшируется с помощью криптографической хэш-функции (10.3.1). Пусть h — функция хэширования, отображающая множество $\{0,1\}^*$ в пространство $\mathcal{M}$. Сообщение $m \in \mathcal{M}$ считается распознаваемым или осмысленным, если существует строка $M \in \{0,1\}^*$, удовлетворяющая условию

$$m = h(M).$$

При таких условиях подделка подписи RSA усложняется. Вычисление сообщения m, соответствующего подписи s в формуле (10.4.1), уже не приводит к нужному результату, если атакующий не может придать распознаваемому сообщению осмысленный вид, т.е. не имеет в своем распоряжении прообраз сообщения m до применения к нему функции хэширования. Если предположить, что функция хэширования является случайным оракулом, описанным в разделе 10.3.1.2, то "подделка с нуля" цифровой подписи RSA по заданному сообщению по трудности сравнима с решением задачи RSA, т.е. извлечению корня e-й степени числа N по некоторому модулю (определение 8.4 в разделе 8.7).

Следует однако заметить, что в настоящее время не существует ни одного формального доказательства этого факта. Совершенно очевидно, что учебный вариант цифровой подписи RSA не обладает доказуемой стойкостью. Даже для простейших функций хэширования никому не удалось доказать стойкость цифровой подписи RSA к атакам на основе адаптивно подобранных сообщений. Следовательно, описанную выше схему цифровой подписи RSA следует считать учебной.

Более стойкая схема цифровой подписи RSA, использующая функцию хэширования, описана в главе 16. Этот алгоритм является вероятностным, т.е. генерируемые подписи случайным образом распределяются по всему пространству подписей, причем полученное распределение невозможно отличить от равномерного. Благодаря этому свойству данная схема цифровой подписи вполне пригодна для практического применения. Формальное доказательство строгой стойкости цифровой подписи RSA будет изложено в главе 16.

10.4.4 Цифровая подпись Рабина (учебный вариант)

Схема цифровой подписи Рабина [240] очень напоминает схему цифровой подписи RSA. Разница между ними заключается лишь в процедуре верификации. В цифровой подписи RSA показатель верификации e представляет собой нечетное число, поскольку должно выполняться условие $\gcd(e, \phi(N)) = 1$, где $\phi(N)$ — четное число, а в схеме Рабина $e = 2$.

Схема цифровой подписи Рабина описывается алгоритмом 10.2. Заметим, что она носит исключительно учебный характер.

Алгоритм 10.2. Схема цифровой подписи Рабина

Генерация ключа

Процедура генерации ключа такая же, как и в криптосистеме RSA (алгоритм 8.1). (* Следовательно, выполняется условие $N = pq$, p и q — разные нечетные простые числа приблизительно одинаковой величины. Число N — открытый ключ Алисы, а числа p и q — параметры ее закрытого ключа. *)

Генерация подписи

Для цифровой подписи сообщения $m \in \mathbb{Z}_N^*$ Алиса вычисляет значение

$$s \leftarrow m^{1/2} (\mathrm{mod}\, n).$$

(* Для того чтобы это вычисление было возможным, необходимо, чтобы выполнялось условие $m \in \mathrm{QR}_N$. Из раздела 6.6.2 следует, что если N — модуль алгоритма RSA, то $\#\mathrm{QR}_N = \#\mathbb{Z}_N^*/4$, т.е. четверть элементов группы $\mathbb{Z}_N^*$ принадлежит множеству QR_N. Таким образом, Алиса может использовать подходящий механизм форматирования сообщения так, чтобы выполнялось условие $m \in \mathrm{QR}_N$. Применив к такому сообщению m алгоритм 6.5, Алиса может вычислить его квадратный корень. *)

Верификация подписи

Допустим, Боб должен удостовериться, что открытый ключ N действительно принадлежит Алисе. Боб применяет к паре (m, s) следующую процедуру верификации.

$$\mathsf{Verify}_N(m, s) = \mathsf{True}, \text{ если } m \equiv s^2 (\mathrm{mod}\, B).$$

(* *Примечание*: сообщение m должно быть распознаваемым (см. раздел 10.4.3). *)

Схема цифровой подписи Рабина имеет ряд преимуществ по сравнению со схемой RSA. Во-первых, доказано, что подделка подписи так же сложна, как и разложение целого числа на простые множители (формальное доказательство этого факта будет приведено позднее). Во-вторых, верификация в схеме Рабина выполняется быстрее и вполне пригодна для приложений, в которых верификация

подписей выполняется небольшими вычислительными устройствами, например, портативными.

Обратите внимание на то, что согласно замечанию 10.1, если сообщение m не является распознаваемым, то фальсификация цифровой подписи Рабина становится тривиальной. Эта подделка является экзистенциальной. Для ее предотвращения необходимо хэшировать сообщение, как показано в разделе 10.4.3, так, чтобы сообщение стало распознаваемым.

10.4.5 Парадоксальная стойкость цифровой подписи Рабина

Используя идею, заложенную в теореме 8.2 (раздел 8.11), можно показать, что если существует алгоритм фальсификации цифровой подписи Рабина, то его можно использовать для факторизации составных модулей, использованных в схеме. Это хорошее свойство, поскольку оно позволяет свести задачу о фальсификации цифровой подписи к общеизвестной трудноразрешимой задаче (разложению целого числа на простые множители).

Однако это свойство сильной стойкости также означает, что схема цифровой подписи Рабина является фатально нестойкой к адаптивным атакам, в ходе которых атакующий может обращаться к автору с просьбой подписать специально подобранные сообщения. Например, атакующий может сгенерировать произвольное число $s \in \mathbb{Z}_N^*$ и представить сообщение $m = s^2 (\mathrm{mod}\, N)$ на подпись Алисе. Ответ Алисы, например, сообщение s', представляет собой один из четырех квадратных корней числа m. Если $s' \pm s (\mathrm{mod}\, N)$, то в ходе адаптивной атаки Злоумышленник может разложить ее модуль на множители.

Следовательно, цифровая схема Рабина, описанная алгоритмом 10.2, абсолютно неприменима в реальных приложениях, в которых невозможно избежать адаптивных атак. Схема цифровой подписи Рабина в реальных приложениях не должна позволять атакующему получать два разных квадратных корня одного сообщения.

Более практичной является цифровая схема Рабина с использованием функции хэширования, описанная в главе 16. Этот алгоритм является вероятностным. Он гарантирует, что многочисленные образцы подписи одного и того же сообщения будут рандомизированы, так что атакующий не сможет вычислить два разных квадратных корня одного сообщения. Следовательно, такую схему цифровой подписи можно применять в реальных приложениях. Формальное доказательство стойкости схемы цифровой подписи Рабина в рамках строгого подхода будет приведено в главе 16.

Подведем парадоксальные итоги, касающиеся стойкости схемы цифровой подписи Рабина.

С одной стороны, используя идею теоремы 8.2 (из раздела 8.11), мы показали, что цифровую подпись Рабина в учебном варианте невозможно подделать, так как эта задача по сложности эквивалентна разложению целого числа на простые множители. Данный результат является не только очень сильным (поскольку имеет формальное доказательство), но и крайне желательным, так как сводит фальсификацию цифровой подписи к трудноразрешимой задаче: факторизации целого числа.

С другой стороны, учебный вариант цифровой подписи Рабина безнадежно слаб и абсолютно непригоден для реальных приложений, которые регулярно подвергаются адаптивным атакам. Такие атаки полностью разрушают учебную схему цифровой подписи Рабина. Следовательно, необходимо разработать прикладную схему цифровой подписи Рабина (см. главу 16). К сожалению, как мы убедимся, формальное доказательство стойкости не связано с задачей разложения целого числа на простые множители.

10.4.6 Цифровая подпись Эль-Гамаля

Кроме элегантной криптосистемы с открытым ключом, описанной в разделе 8.12, Эль-Гамаль разработал оригинальную схему цифровой подписи. Как и криптосистема, схема цифровой подписи Эль-Гамаля дала толчок многочисленным исследованиям и породила целое семейство схем цифровой подписи (некоторые из них будут рассмотрены в разделе 10.4.8 и главе 16).

Схема цифровой подписи Эль-Гамаля представлена в алгоритме 10.3.

10.4.7 Неформальное обоснование стойкости схемы цифровой подписи Эль-Гамаля

Оценим стойкость схемы цифровой подписи Эль-Гамаля.

10.4.7.1 Предостережения

Следует сделать несколько предостережений, касающихся схемы цифровой подписи Эль-Гамаля.

Предостережение 1

Во-первых, при верификации цифровой подписи необходимо проверить неравенство $r < p$. Если $r > p$, возможна атака, изобретенная Блайхенбахером (Bleichenbacher) [41]. Пусть (r, s) — подпись сообщения m. Злоумышленник может подделать новую подпись произвольного сообщения m' следующим образом.

1. $u \leftarrow m'm^{-1} (\mathrm{mod}\, p-1)$.
2. $s' \leftarrow su (\mathrm{mod}\, p-1)$.

Алгоритм 10.3. Схема цифровой подписи Эль-Гамаля

Генерация ключа

Процедура генерации ключа такая же, как и в криптосистеме Эль-Гамаля (раздел 8.12).

(* Следовательно, параметрами открытого ключа Алисы является тройка (g, y, p), где p — большое простое число, $g \in \mathbb{F}_p^*$ — случайный мультипликативный образующий элемент, $y_A \equiv y^{x_A} (\mathrm{mod}\, p)$, x_A — секретное целое число, удовлетворяющее условию $x_A < p - 1$, а закрытым ключом является число x_A. *)

Генерация подписи

Для цифровой подписи сообщения $m \in \mathbb{F}_N^*$ Алиса генерирует случайное число $\ell \in_U \mathbb{Z}_{p-1}^*$ (т.е. $\ell < p - 1$ и $\gcd(\ell, p-1) = 1$) и формирует пару (r, s), где

$$\begin{aligned} r &\leftarrow g^{\ell} (\mathrm{mod}\, p), \\ s &\leftarrow \ell^{-1}(m - x_A r) (\mathrm{mod}\, p - 1). \end{aligned} \tag{10.4.2}$$

(* Число ℓ^{-1} можно найти с помощью расширенного алгоритма Евклида (алгоритм 4.2). *)

Верификация подписи

Допустим, Боб должен удостовериться, что параметры открытого ключа (g, y_A, p) действительно принадлежит Алисе. Боб применяет к паре $(m, (r, s))$ следующую процедуру верификации.

$$\mathsf{Verify}_{(g,y_A,p)}(m, (r, s)) = \mathsf{True}, \text{ если } r < p \text{ и } y_A^r r^s \equiv g^m (\mathrm{mod}\, p).$$

(* *Примечание*: сообщение m должно быть распознаваемым (см. раздел 10.4.7.2). *)

3. Вычисляется значение r', удовлетворяющее условию $r' \equiv ru (\mathrm{mod}\, p-1)$ и $r' \equiv r (\mathrm{mod}\, p)$. Это можно сделать с помощью китайской теоремы об остатках (алгоритм 6.1).

Затем можно выполнить следующие рутинные преобразования.

$$y_A^{r'} r'^{s'} \equiv y_A^{ru} r^{su} \equiv (y_A^r r^s)^u \equiv g^{mu} \equiv g^{m'} (\mathrm{mod}\, p).$$

Эта атака невозможна, если $r < p$, поскольку в этом случае значение r', вычисляемое на третьем шаге применения китайской теоремы об остатках, по величине сравнимо с числом $p(p-1)$.

Предостережение 2

Второе предостережение также было сформулировано Блайхенбахером [41], Алиса должна выбрать случайный элемент g из группы $\mathbb{F}_p^*$. Если этот параметр

выбран не Алисой (это возможно, когда системные пользователи обладают одними и теми же открытыми параметрами g и p), то необходимо проверить, насколько случайным является параметр g (например, он может быть результатом применения псевдослучайной функции).

Допустим, что открытые параметры g и p выбраны Злоумышленником. Параметр p можно установить с помощью стандартного способа, описанного в разделе 8.4.1: пусть $p-1=bq$, где q — достаточно большое простое число, но число b может быть гладким (т.е. число b имеет только малые простые множители, и вычисление дискретного логарифма в группе порядка b не представляет труда).

Злоумышленник генерирует параметр g следующим образом

$$g=\beta^t(\bmod p),$$

где $\beta=cq$ и $c<b$.

Как известно, вычисление дискретного логарифма открытого ключа y_A — трудноразрешимая задача. Однако вычисление дискретного логарифма величины y_A^q по основанию g^q не создает никаких сложностей. Дискретный логарифм равен $z\equiv x_A(\bmod b)$, т.е. выполняется следующее сравнение.

$$y_A^q\equiv(g^q)^z\,(\bmod p)\,.$$

Вычислив значение z, Злоумышленник может подделать подпись Алисы с помощью следующих операций.

$$r\leftarrow\beta=cq,$$
$$s\leftarrow t(m-cqz)(\bmod p-1).$$

Затем выполняются рутинные преобразования.

$$y_A^r r^s\equiv y_A^{cq}\left(\beta^t\right)^{(m-cqz)}\equiv g^{cqz}g^{m-cqz}\equiv g^m(\bmod p).$$

Следовательно, пара (r,s) действительно является корректной подписью сообщения m, созданного без помощи закрытого ключа x_A (но с помощью числа $x_A(\bmod b)$).

Заметим, что в этой процедуре фальсификации цифровой подписи число q является делителем числа r. Следовательно, атаку Блайхенбахера можно предотвратить, если во время верификации Боб проверяет условие $q\nmid r$ (предполагается, что в процессе выбора параметра p число q является открытым). В разделе 16.3.2.1 будет показано, что условие $q\nmid r$ является необходимым для доказательства формальной стойкости схемы цифровой подписи Эль-Гамаля к фальсификации.

Предостережение 3

Третье предупреждение касается эфемерного ключа ℓ. Генерация цифровой подписи Эль-Гамаля представляет собой рандомизированный алгоритм, поскольку эфемерный ключ ℓ является случайным.

Алиса никогда не должна применять эфемерный ключ для подписи других сообщений. Если эфемерный ключ ℓ используется повторно для подписи двух сообщений m_1 и m_2, удовлетворяющих условию $m_1 \neq m_2 (\mathrm{mod}\, p-1)$, то из второго уравнения системы (10.4.2) следует, что

$$\ell(s_1 - s_2) \equiv m_1 - m_2 (\mathrm{mod}\, p-1).$$

Поскольку число $\ell^{-1} (\mathrm{mod}\, p-1)$ существует, из неравенства $m_1 \neq m_2 (\mathrm{mod}\, p-1)$ следует, что

$$\ell^{-1} \equiv (s_1 - s_2)/(m_1 - m_2) (\mathrm{mod}\, p-1), \tag{10.4.3}$$

т.е. число ℓ^{-1} раскрыто. В свою очередь, закрытый ключ Алисы x_A можно вычислить из второго уравнения системы (10.4.2) по формуле

$$x_A \equiv (m_1 - \ell s_1)/r (\mathrm{mod}\, p-1). \tag{10.4.4}$$

Заметим также, что эфемерный ключ должен извлекаться из пространства $\mathbb{Z}_{p-1}^*$ с помощью простого случайного выбора. Особую осторожность следует проявлять, когда подпись генерируется небольшими вычислительными устройствами, например, кредитной карточкой с микропроцессором или портативным компьютером: необходимо убедиться, что эти приборы снабжены хорошими датчиками случайных чисел.

Поскольку эфемерный ключ ℓ используется лишь один раз и генерируется с помощью датчика равномерно распределенных случайных чисел, из второго уравнения системы (10.4.2) следует, что для шифрования закрытого ключа x_A, по существу, используется одноразовый шифр. Таким образом, эти два ключа защищают друг друга в теоретико-информационном смысле.

10.4.7.2 Предотвращение экзистенциальной подделки

Экзистенциальную подделку, описанную в замечании 10.1, можно применить и к схеме цифровой подписи Эль-Гамаля, если сообщение не содержит необходимой распознаваемой избыточности. Иначе говоря, нетрудно подделать пару "сообщение-подпись", созданную по схеме Эль-Гамаля, где сообщение не является распознаваемым.

Например, пусть u и v — любые целые числа, которые меньше $p-1$ и удовлетворяют условию $\gcd(v, p-1) = 1$. Выполним следующие операции.

$r \leftarrow g^u y_A^v (\mathrm{mod}\, p)$,
$s \leftarrow -rv^{-1} (\mathrm{mod}\, p-1)$,
$m \leftarrow -ruv^{-1} (\mathrm{mod}\, p-1)$.

Пара $(m,(r,s))$ представляет собой законную подпись, созданную по схеме Эль-Гамаля с помощью открытого ключа Алисы y_A.

$$\begin{aligned} y^r r^s \equiv & \\ & \equiv y_A^r r^{-rv^{-1}} \equiv \\ & \equiv y_A^r \left(g^u y_A^v\right)^{-rv^{-1}} \equiv \\ & \equiv y_A^r \left(g^u\right)^{-rv^{-1}} \left(y_A^v\right)^{-rv^{-1}} \equiv \\ & \equiv y_A^r g^{-ruv^{-1}} y_A^{-r} \equiv \\ & \equiv g^{-ruv^{-1}} \equiv \\ & \equiv g^m (\mathrm{mod}\, p). \end{aligned}$$

Однако, поскольку возведение в степень по модулю обладает свойством перемешивания, подделанное сообщение m не распознается.

Механизм форматирования сообщения предотвращает фальсификацию. В простейшем случае достаточно, чтобы сообщение m одержало распознаваемую часть, т.е. представляло собой конкатенацию $m = M \parallel I$, где M — подписываемое сообщение, а I — распознаваемая строка, например, имя автора.

Наиболее часто для форматирования сообщений применяется хэширование. В качестве примера рассмотрим сообщение

$$m = H(M, r),$$

где H — криптографическая функция хэширования, а M — битовая строка, представляющая сообщение. Теперь подписью является само сообщение M. Этап верификации включает в себя верификацию сообщения $m = H(M, r)$. Поскольку функции хэширования являются однонаправленными, экзистенциальная атака становится невозможной.

Если предположить, что функция хэширования H является случайным оракулом (см. раздел 10.3.1.2), то доказательство стойкости схемы цифровой подписи Эль-Гамаля сводится к доказательству эквивалентности между задачей дискретного логарифмирования (которая, как известно, является трудноразрешимой) и фальсификацией подписи. Формальное доказательство этого факта будет приведено в главе 16. В этой же главе содержатся формальные доказательства других схем цифровой подписи, принадлежащих семейству Эль-Гамаля.

10.4.8 Семейство схем цифровой подписи Эль-Гамаля

После опубликования оригинальной работы Эль-Гамаля появилось несколько вариаций на эту тему. Наиболее значительными из них являются схема цифровой подписи Шнорра (Schnorr) [256, 257] и стандарт цифровой подписи DSS (Digital Signature Standard) [215, 216].

10.4.8.1 Схема цифровой подписи Шнорра

Схема цифровой подписи Шнорра является вариантом схемы цифровой подписи Эль-Гамаля. Однако она обладает свойством, делающим ее весьма ценной для криптографии с открытым ключом: поле простых элементов в этой схеме представлено намного компактнее, чем в схеме Эль-Гамаля, однако это не приводит к уменьшению сложности решения задачи, лежащей в ее основе (например, задачи дискретного логарифмирования). Впоследствии эту идею применили к конечным полям более общего вида, что привело к появлению новой криптосистемы с открытым ключом: XTR [175].

Более компактное представление достигается с помощью конструирования поля $\mathbb{F}_p$, содержащего намного меньшую подгруппу простого порядка q. Заметим, что в настоящее время в криптосистемах семейства Эль-Гамаля выбирается параметр $p \approx 2^{1024}$. Кроме того, с появлением более мощных средств, позволяющих решать задачу дискретного логарифмирования, размер числа p должен увеличиваться. Однако после появления работы Шнорра стало общепринятой практикой устанавливать параметр q равным приблизительно 2^{160}. Вполне возможно, что эта величина не зависит от роста числа p, поскольку информация о подгруппе не играет значительной роли при вычислении дискретного логарифма в поле $\mathbb{F}_p$, даже если известно, что искомый элемент принадлежит данной подгруппе. Величина 2^{160} диктуется исключительно требованием стойкости к атаке по методу квадратного корня (см. раздел 3.6).

Схема цифровой подписи Шнорра описывается алгоритмом 10.4.

Отметим, что при вычислении открытых параметров порождающий элемент g может определяться быстро, поскольку $q \parallel p-1$.

$$\text{Prob}\left[\gcd(\text{ord}(f), q) = 1 \mid f \in_U \mathbb{Z}_p^*\right] \leqslant 1/q,$$

т.е. вероятность найти случайное число f, удовлетворяющее условию $g \leftarrow f^{\frac{p-1}{q}} \equiv \equiv 1 (\text{mod}\, p)$, пренебрежимо мала. Из малой теоремы Ферма (теорема 6.10 из раздела 6.4) следует, что

$$g^q \equiv 1 (\text{mod}\, p).$$

Таким образом, элемент g действительно порождает подгруппу, состоящую из q элементов.

Верификация подписи работает правильно, если пара $(m, (s, e))$ представляет собой правильную пару "сообщение-подпись", созданную Алисой. Тогда

$$r' \equiv g^s y^e \equiv g^{xe+\ell} y^e \equiv y^{-e} g^\ell y^e \equiv g^\ell \equiv r (\text{mod}\, p).$$

Как указывалось ранее, применение подгруппы порядка q группы $\mathbb{F}_p$ позволяет сделать цифровую подпись в схеме Шнорра намного короче, чем подпись в схеме Эль-Гамаля: для передачи подписи Шнорра достаточно $2|q|$ бит, в то

Алгоритм 10.4. Схема цифровой подписи Шнорра

Установка системных параметров

1. Выбрать два простых числа p, удовлетворяющих условию $q \mid p-1$.
   (* Как правило, выбираются числа, имеющие размеры $|p| = 1024$ и $|q| = 160$. *)
2. Выбрать элемент $g \in \mathbb{Z}_p^*$ порядка q.
   (* Для этого достаточно извлечь элемент $f \in_U \mathbb{Z}_p^*$ и выполнить операцию $g \leftarrow f^{(p-1)/q} (\mathrm{mod}\, p)$. Если $g = 1$, повторять операцию, пока $g \neq 1$. *)
3. Выбрать криптографическую функцию хэширования $H : \{0,1\}^* \mapsto \mathbb{Z}_q$.
   (* Например, можно выбрать хэш-функцию SHA-1. *)

Параметры (p, q, g, H) распределяются между системными пользователями.

Генерация открытого/закрытого ключа

Алиса генерирует случайное число $x \in_U \mathbb{Z}_p^*$ и выполняет операцию

$$y \leftarrow g^{-x} (\mathrm{mod}\, p).$$

Параметры открытого ключа Алиса содержатся в кортеже (p, q, g, y, H), а ее закрытым ключом является число x.

Генерация подписи

Для того чтобы подписать сообщение $m \in \{0,1\}^*$, Алиса генерирует случайное число $\ell \in_U \mathbb{Z}_q$ и формирует пару (e, s), где

$$\begin{aligned} r &\leftarrow g^{\ell} (\mathrm{mod}\, p), \\ e &\leftarrow H(m \parallel r), \\ s &\leftarrow \ell + xe (\mathrm{mod}\, q). \end{aligned}$$

Верификация подписи

Допустим, Боб должен удостовериться, что параметры открытого ключа (p, q, g, y, H) действительно принадлежат Алисе. Боб применяет к паре $(m, (e, s))$ следующую процедуру верификации.

$$\begin{aligned} r' &\leftarrow g^s y^e (\mathrm{mod}\, p), \\ e' &\leftarrow H(m \parallel r'), \end{aligned}$$

$$\mathsf{Verify}_{(p,q,g,y,h)}(m, (s, e)) = \mathsf{True}, \text{ если } e' = e.$$

время как для передачи подписи Эль-Гамаля требуется $2|p|$ бит. Более короткая подпись позволяет сократить количество операций, необходимых для генерации

подписи и верификации: в схеме Шнорра — $O_B(\log_2 q \log^2 p)$, а в схеме Эль-Гамаля — $O_B(\log^3 p)$. Заметим также, что в генерации подписи модуль p можно вычислить автономно. Следовательно, для реализации схемы цифровой подписи в режиме реального времени необходимо вычислять только модуль q, а все сложные вычисления можно выполнять в автономном режиме.

Как в схеме Эль-Гамаля, эфемерный ключ ℓ должен применяться только один раз и выбираться с помощью метода простого случайного выбора. В этих условиях эфемерный ключ и закрытый ключ автора защищают друг друга в теоретико-информационном смысле.

10.4.8.2 Стандарт схемы цифровой подписи (DSS)

В августе 1991 г. Национальный институт стандартов и технологий США (National Institute of Standards and Technology — NIST) объявил о новой схеме цифровой подписи под названием стандарт DSS (Digital Signature Standard) [215, 216]. По существу, стандарт DSS представляет собой схему подписи Эль-Гамаля, однако, как и в схеме Шнорра, она работает в намного более компактной подгруппе простого порядка, принадлежащей конечному полю, в котором задача дискретного логарифмирования является трудноразрешимой. Следовательно, стандарт DSS позволяет уменьшить длину цифровой подписи по сравнению со схемой Эль-Гамаля.

Схема DSS описывается алгоритмом 10.5.

Верификация подписи выполняется успешно, если пара $(m, (r, s))$ представляет собой правильную пару "сообщение-подпись", созданную Алисой. Тогда

$$g^{u_1} y^{u_2} \equiv g^{H(m)s^{-1}} y^{rs^{-1}} \equiv g^{(H(m)+xr)s^{-1}} \equiv g^{\ell} (\mathrm{mod}\, p).$$

Если открытые параметры имеют одинаковые размеры, к схеме Шнорра и стандарту DSS предъявляются одинаковые требования, касающиеся полосы пропускания канала связи и вычислительных мощностей.

Стандарт DSS был принят одновременно с хэш-функцией SHA-1 [217]. Применение этой стандартной функции обеспечивает распознавание сообщений и предотвращает экзистенциальную фальсификацию.

В заключение отметим, что как и во всех схемах семейства Эль-Гамаля, эфемерный ключ ℓ должен применяться только один раз и выбираться с помощью метода простого случайного выбора.

10.4.9 Формальное доказательство стойкости схем цифровой подписи

В разделе 8.14 введено понятие повышенной стойкости криптосистем с открытым ключом. Аналогичное понятие можно сформулировать и для схем цифровой подписи.

Алгоритм 10.5. Стандарт DSS

Установка системных параметров

(* Системные параметры идентичны параметрам Шнорра. Параметры (p, q, g, H), как и в алгоритме 10.4, распределяются между системными пользователями. *)

Генерация открытого/закрытого ключа

Алиса генерирует случайное число $x \in_U \mathbb{Z}_q$ в качестве своего закрытого ключа и вычисляет открытый ключ

$$y \leftarrow g^x (\mathrm{mod}\, p).$$

Параметры открытого ключа Алисы содержатся в кортеже (p, q, g, y, H), а ее закрытым ключом является число x.

Генерация подписи

Для того чтобы подписать сообщение $m \in \{0,1\}^*$, Алиса генерирует случайное число $\ell \in_U \mathbb{Z}_q$ и формирует пару (r, s), где

$$r \leftarrow (g^\ell (\mathrm{mod}\, p)) (\mathrm{mod}\, q),$$
$$s \leftarrow \ell^{-1} (H(m) + xr) (\mathrm{mod}\, q).$$

Верификация подписи

Допустим, Боб должен удостовериться, что параметры открытого ключа (p, q, g, y, h) действительно принадлежат Алисе. Боб применяет к паре $(m, (r, s))$ следующую процедуру верификации.

$$w \leftarrow s^{-1} (\mathrm{mod}\, q),$$
$$u_1 \leftarrow H(m) w (\mathrm{mod}\, q),$$
$$u_2 \leftarrow rw (\mathrm{mod}\, q),$$
$$\mathsf{Verify}_{(p,q,g,y,h)}(m, (r, s)) = \mathsf{True}, \text{ если } r = (g^{u_1} y^{u_2} (\mathrm{mod}\, p)) (\mathrm{mod}\, q).$$

Читатели могли заметить, что в главе нет ни одного доказательства стойкости рассматриваемых схем цифровой подписи. Действительно, как сказано в замечании 10.2, формальное доказательство стойкости выходит за рамки этой главы. Для этого существует две причины.

Во-первых, вполне естественно предположить, что подделка подписи "с нуля" должна быть сложнее, чем применение доступных пар "сообщение-подпись", которыми может распоряжаться атакующий. Фальсификацию можно упростить, если атакующий может контактировать с объектом атаки и получать от него специально подготовленные подписанные сообщения. Такой способ фальсифика-

ции подписи называется **атакой на основе адаптивно подобранного сообщения** (adaptive chosen-message attack).

В действительности пары "сообщение-подпись", соответствующие заданному открытому ключу, легкодоступны. Кроме того, адаптивные атаки на цифровые подписи трудно предотвратить: подписание сообщений является неотъемлемой частью многих приложений. Следовательно, для схем цифровой подписи необходимо сформулировать понятие стойкости, приемлемое для практики. Такое понятие приводится в главе 16. Именно по этой причине мы отложили доказательство формальной стойкости цифровых схем подписи на будущее.

Во-вторых, если сообщение не является распознаваемым, то, как правило, пару "сообщение-подпись" легко подделать, даже если они фальсифицируются "с нуля" (см. замечание 10.1 об экзистенциальной подделке и описание конкретных экзистенциальных фальсификаций конкретных схем цифровой подписи). Для того чтобы предотвратить экзистенциальную фальсификацию любая схема цифровой подписи должна предусматривать механизм форматирования сообщений, гарантирующий их распознавание. Обычно в качестве механизма форматирования сообщений применяются криптографические функции хэширования. Следовательно, доказательство формальной стойкости схем цифровой подписи должно сопровождаться формальным описанием криптографических функций хэширования. Без этого доказательство формальной стойкости схем теряет смысл.

Как показано в разделе 10.3.1.2, криптографические хэш-функции имитируют поведение случайных функций. Для криптографических схем, использующих функции хэширования, понятие формальной стойкости связано с **моделью случайного оракула** (random oracle model — ROM). Анализ этого понятия будет проведен в главе 16. Там же будет показано, что с помощью модели случайного оракула формальное доказательство стойкости схемы цифровой подписи (даже к атакам на основе адаптивно подобранных сообщений) сводится к доказательству неразрешимости одной из известных задач.

10.5 Асимметричные методы II: защита целостности данных без идентификации источника

В механизме защиты целостности данных, основанном на схеме цифровой подписи, ключ Ke, как правило, считается закрытым, а ключ Kv — открытым. В таком случае проверка целостности данных сводится к идентификации отправителя сообщения, т.е. владельца ключа Kv.

Однако процедура генерации ключей, будучи неотъемлемым элементом схем цифровой подписи, вовсе необязательна для защиты целостности данных. Факти-

чески определение 10.1 не налагает никаких ограничений на конструкцию ключей, предназначенных для создания и верификации кодов MDC.

Например, можно сгенерировать два ключа, Ke и Kv, используя обратную схему: ключ Ke объявить открытым, а ключ Kv — закрытым. В этом случае любой пользователь может применить открытый ключ Ke для создания целостной пары (Data, MDC) или пары "сообщение-подпись" (m, s), а любой владелец закрытого ключа Kv может верифицировать согласованность пары (Data, MDC) или достоверность пары "сообщение-подпись" (m, s). Разумеется, при таком необычном порядке генерации ключей систему больше нельзя считать схемой цифровой подписи. Однако она полностью соответствует определению 10.1, в котором описывается механизм защиты целостности данных!

Поскольку создать согласованную пару (Data, MDC) с помощью открытого ключа Ke может любой пользователь, такие механизмы называются **системами защиты целостности данных без идентификации источника** (data-integrity without source indetification). Зная особенности поведения противника, можно назвать этот механизм системой "*защиты целостности данных, посланных Злоумышленником*" ("*data integrity from Malice*").

Рассмотрим схему шифрования с открытым ключом, обеспечивающую такую защиту. Эта схема обладает следующим свойством: Злоумышленник может послать Алисе конфиденциальное "стойкое" сообщение (например, через своих союзников). Стойкость сообщения заключается в том, что союзникам Злоумышленника крайне сложно модифицировать текст незаметно для Алисы. Этот алгоритм называется **оптимальным асимметричным шифрованием с дополнением** (optimal asymmetric encryption padding — OAEP). Он был изобретен Белларе (Bellare) и Роджуэем (Rogaway) [24].

Если после отправления шифрованный текст не модифицировался, то алгоритм шифрования гарантирует, что Алиса получит случайное число r. Следовательно,

$$v = s \oplus G(r) = \left(m \parallel 0^{k_1}\right) \oplus G(r) \oplus G(r) = m \parallel 0^{k_1}.$$

Таким образом, Алиса увидит k_1 нулей, присоединенных к восстановленному исходному сообщению.

С другой стороны, любая модификация шифрованного текста приведет к изменению сообщения, запечатанного с помощью функции RSA. Это, в свою очередь, вызовет "неконтролируемое" изменение исходного сообщения, в частности, случайного числа и k_1 нулей, сопровождающих исходное сообщение, которые должны поступить на вход функции OAEP. Интуитивно ясно, что "неконтролируемые" изменения возникают вследствие вмешательства "случайного оракула", которого имитируют две функции хэширования, применяемые в этой схеме (см. раздел 10.3.1.2). Неконтролируемые изменения проявляются в виде повреждения избыточной информации (строки, состоящей из k_1 нулей), присоединяемой к исход-

ному сообщению, с вероятностью не меньше $1 - 2^{-k_1}$. Если число 2^{-k_1} является пренебрежимо малым, то $1 - 2^{-k_1}$ близко к единице. Следовательно, описанная схема обеспечивает защиту целостности данных в зашифрованном сообщении.

Защита целостности данных, обеспечиваемая алгоритмом шифрования RSA-OAEP, выглядит довольно странно: даже увидев строку, состоящую из k_1 нулей, и будучи уверенной, что зашифрованный текст не был модифицирован, Алиса не знает, кто его послал. Вот почему в алгоритме 10.6 мы произвольно назначили отправителем Злоумышленника.

Понятие "целостности данных, посланных Злоумышленником" очень полезно и важно. Оно появилось в результате разработки схем шифрования с открытым ключом, стойких к атаке на основе адаптивного подобранного зашифрованного текста (CCA2). В криптосистемах с открытым ключом, стойких по отношению к атаке CCA2, процедура расшифровки включает в себя этап верификации данных. Такие криптосистемы считаются неуязвимыми даже для следующих экстремальных форм поведения атакующего.

- Атакующий и владелец открытого ключа играют в "вопросы и ответы". Атакующий задает вопросы и может посылать владельцу открытого ключа произвольное количество (разумеется, ограниченное полиномом) сообщений, содержащих "*адаптивно подобранный зашифрованный текст*", предназначенных для шифрования оракулом (см. раздел 8.2).
- Владелец открытого ключа отвечает на вопросы. Если верификация целостности данных завершается успешно, владелец ключа должен просто отослать результаты расшифровки назад, независимо от того, что запрос на расшифровку мог поступить от атакующего, стремящегося либо взломать криптосистему, либо получить исходное сообщение, не предназначенное для посторонних глаз, либо раскрыть секретный ключ владельца).

Если зашифрованный текст содержит корректные защищенные данные, считается, что отправителю известна информация, содержавшаяся в исходном тексте. Это понятие называется **осведомленностью об исходном сообщении** (plaintext awareness). Если атакующему известен зашифрованный исходный текст, то оракул расшифровки не может сообщить ему ничего нового. С другой стороны, если атакующий пытается адаптивно модифицировать зашифрованный текст, то существует огромная вероятность того, что проверка целостности данных завершится неудачей и результатом расшифровки будет пустое сообщение. Таким образом, криптосистемы, предусматривающие защиту целостности данных, устойчивы к активным атакам.

В главе 14 мы рассмотрим формальную модель стойкости криптосистем к атакам на основе адаптивно подобранного зашифрованного текста (CCA2). Там же будут описаны некоторые популярные криптосистемы, формально стойкие по отношению к атакам, описанным в главе 15. К числу таких систем относится и схема

Алгоритм 10.6. Оптимальное асимметричное шифрование с дополнением (RSA-OAEP) [24]

Параметры ключей

Пусть кортеж $(N, e, d, G, H, n, k_0, k_1) \leftarrow_U \mathsf{Gen}(1^k)$ удовлетворяет следующим условиям: тройка (N, e, d) содержит параметры ключа алгоритма RSA, где $d = e^{-1} (\mathrm{mod}\, \phi(N))$, $|N| = k = n + k_0 + k_1$, числа 2^{-k_0} и 2^{-k_1} являются пренебрежимо малыми, функции хэширования G и H удовлетворяют условиям

$$G : \{0,1\}^{k_0} \mapsto \{0,1\}^{k-k_0}, \quad H : \{0,1\}^{k-k_0} \mapsto \{0,1\}^{k_0},$$

n — длина исходного сообщения.

Допустим, что открытым ключом Алисы является пара (N, e), а закрытым ключом — число d.

Шифрование

Для того чтобы послать Алисе сообщение $m \in \{0,1\}^n$, Злоумышленник выполняет следующие операции.

1. $r \leftarrow_U \{0,1\}^{k_0}; s \leftarrow (m \parallel 0^{k_1}) \oplus G(r); t \leftarrow r \oplus H(s);$
2. if [2] $(s \parallel t \geqslant N)$ go to 1;
3. $c \leftarrow (s \parallel t)^e (\mathrm{mod}\, N)$.

Зашифрованным текстом является число c.

(* Здесь символ $\parallel$ обозначает операцию конкатенации, $\oplus$ — побитовую операцию исключающего ИЛИ (XOR), а 0^{k_1} — строку, состоящую из k_1 нулей, обеспечивающих избыточность при проверке целостности данных во время расшифровки. *)

Расшифровка

Получив зашифрованный текст c, Алиса выполняет следующие операции.

1. $s \parallel t \leftarrow c^d (\mathrm{mod}\, N)$, где $|s| = n + k_1 = k - k_0, |t| = k_0$;
2. $u \leftarrow t \oplus H(s); v \leftarrow s \oplus G(u);$
3. Результат $= \begin{cases} m, & \text{если } v = m \parallel 0^{k_1}, \\ \text{ОТКАЗ} & \text{в противном случае.} \end{cases}$

(* В случае отказа зашифрованный текст объявляется недостоверным. *)

шифрования RSA-OEAP. Анализ показывает, что схема шифрования RSA-OAEP устойчива даже к очень сильной атаке на основе адаптивно подобранного зашиф-

[2] Для того чтобы дополненное сообщение никогда не превышало числа N, используется метод проб и ошибок. Вероятность i-кратного повторения теста равна 2^{-i}. В качестве альтернативы для того, чтобы сделать числа r, H и, следовательно, t на один бит короче числа N, можно применить алгоритм "дополнения PSS" (раздел 16.4.2).

рованного текста. Благодаря этому свойству схему RSA-OAEP следует считать не учебной, а прикладной криптосистемой с открытым ключом.

Как показано при описании алгоритма RSA-OAEP, для предотвращения атаки CCA2 криптосистема должна предусматривать механизм проверки целостности данных и не обязана идентифицировать источник.

Механизм идентификации источника является частью более общего механизма аутентификации источника (data-origin authentication), рассмотренного в следующей главе.

10.6 Резюме

В главе рассмотрены основные криптографические методы защиты целостности данных. К ним относятся 1) симметричные методы, основанные на кодах аутентификации сообщения (MAC) с помощью функции хэширования или алгоритмов блочного шифрования, и 2) асимметричные методы, основанные на применении схем цифровой подписи. Эти методы обеспечивают не только защиту целостности данных, но и идентификацию источника сообщения.

Понятие стойкости схем цифровой подписи, рассмотренное в этой главе, является нестрогим. Некоторые схемы цифровой подписи, представленные здесь, сопровождаются предупреждениями об их потенциальной слабости. Более строгое понятие стойкости, касающееся схем цифровой подписи, будет сформулировано в главе 16.

В главе также рассмотрены особые механизмы защиты целостности данных, не предусматривающие идентификацию источника. Приводится пример криптосистемы с открытым ключом, обладающей свойством повышенной стойкости (без доказательства). В главе 15 будет показано, насколько важны такие криптосистемы для формализации общей методологии построения прикладных криптосистем.

Упражнения

10.1. Что такое код распознавания манипуляций (MDC)? Как он генерируется и применяется? Является ли код аутентификации сообщений (MAC) кодом распознавания манипуляций (MDC)?

10.2. Что такое случайный оракул? Существует ли случайный оракул? Как он имитируется в реальных приложениях?

10.3. Допустим, что пространство значений функции хэширования имеет размер 2^{160}. Какое время потребуется для того, чтобы обнаружить коллизию?

10.4. Почему практически невозможно инвертировать функцию хэширования?

10.5. В чем заключается основное различие между симметричными и асимметричными методами защиты целостности данных?

10.6. Что представляет собой экзистенциальная подделка цифровой подписи? Как ее предотвратить?

10.7. Почему нестрогая стойкость учебных схем цифровых подписей непригодна для практического применения?

Подсказка: проанализируйте фатальную уязвимость схемы Рабина для активных атак.

10.8. Сформулируйте понятие "целостности данных, посланных злоумышленником".

10.9. Является ли зашифрованный текст, созданный алгоритмом RSA-OAEP (алгоритм 10.6), корректным кодом MDC?

Часть IV

Аутентификация

В настоящее время многие виды коммерческой деятельности, финансовые операции и обмен документами осуществляется с помощью открытых и уязвимых средств связи, например, через Internet. Чрезвычайно важно, чтобы партнеры доверяли друг другу, а сообщения не фальсифицировались. Для этого необходимо обеспечить аутентификацию с помощью прикладных криптографических средств. Эта часть состоит из трех глав, посвященных различным протоколам аутентификации. В главе 11 рассматриваются основные принципы аутентификации, описываются типичные ошибки и новые достижения. Глава 12 содержит описание наиболее важных методов аутентификации, применяемых в реальных системах. В главе 13 изложены основы аутентификации в инфраструктурах с открытым ключом.

Глава 11

Протоколы аутентификации — принципы

11.1 Введение

Многочисленные примеры протоколов аутентификации рассмотрены в главе 2. Большинство из этих протоколов были фиктивными (кроме двух): мы специально сделали их уязвимыми, чтобы продемонстрировать необходимость повышенной бдительности и описать основные криптографические приемы.

В этой главе мы переходим к более глубокому изучению аутентификации. Тематически главу можно поделить на две части.

Введение в методы аутентификации

В этой части мы изучим различные методы аутентификации, в частности, основные механизмы и конструкции протоколов для аутентификации сообщений и пользователей, методы аутентификации, основанные на использовании паролей, и некоторые важные способы генерации аутентифицированных ключей. Для этого мы выбрали основные механизмы аутентификации и конструкции протоколов, использованные в международных стандартах и описанные в литературе. Таким образом, при изучении аутентификации основное внимание уделяется стандартизованным приемам. Кроме них, в книге рассматриваются другие общепринятые методы аутентификации и протоколы генерации аутентифицированных ключей. Механизмы и протоколы, описанные в этой части книги, можно использовать в качестве строительных блоков для создания новых протоколов. Они будут весьма полезны разработчикам протоколов аутентификации.

Демонстрация многочисленных слабостей протоколов аутентификации

Эта часть чрезвычайно важна. В ней перечислены многие виды атак на протоколы аутентификации. Каждая из этих атак анализируется и обсуждается на примерах уязвимых протоколов. Будет показано, что слабые места обнаруживаются даже у протоколов аутентификации, разработанных экспертами. Подробный список типичных слабостей и связанных с ними атак являет-

ся весьма полезным подспорьем для разработчиков протоколов и позволяет ответить на вопрос: "Знаете ли вы такой вид атаки?".

В отличие от фиктивных протоколов, рассмотренных в главе 2, слабости протоколов, описанные в этой главе, не являются искусственными. Все они взяты из практики! Эти недостатки были вскрыты после того, как протоколы, разработанные уважаемыми авторами, были опубликованы в изданиях, посвященных криптографии и защите информации. Отсюда следует, что стандартизация, подробно разработанная теория проектирования и большой опыт не гарантируют от ошибок даже экспертов.

Поскольку разработка протоколов аутентификации чрезвычайно подвержена ошибкам, мы не оставим эту тему и в дальнейшем. В настоящее время формальные методы доказательства корректности протоколов аутентификации являются предметом интенсивных исследований. Результаты, достигнутые в этой области, будут изложены в главе 17.

11.1.1 Структурная схема главы

В разделе 11.2 рассматриваются основные понятия, связанные с аутентификацией. В разделе 11.3 устанавливаются правила описания компонентов протоколов аутентификации и поведение их участников, принятое по умолчанию. Следующие три раздела посвящены введению в методы аутентификации. В разделе 11.4 изложены основные принципы и стандартные конструкции протоколов аутентификации, раздел 11.5 посвящен методам аутентификации с помощью паролей, а в разделе 11.6 изучаются важные протоколы, обеспечивающие аутентификацию и обмен ключами с помощью альтернативных методов. Раздел 11.7 относится к той части главы, в которой описываются слабости протоколов аутентификации и основные виды атак на них. В заключительном разделе 11.8 перечислены литературные ссылки, относящиеся к этой области.

11.2 Основные понятия аутентификации

Кратко говоря, аутентификация — это процедура, позволяющая одной сущности проверить объявленные свойства другой. Например, аутентификация позволяет первой сущности проверить, является ли вторая сущность законной, а второй сущности — доказать свою законность. Из этого краткого определения следует, что в процессе аутентификации участвуют две стороны. Процедура взаимодействия между общающимися пользователями называется **протоколом**. Таким образом, процедура аутентификации является протоколом аутентификации (authentication protocol).

Аутентификацию можно разделить на три вида: **аутентификация источника данных** (data-origin authentication), **аутентификация сущности** (entity authen-

tication) и **генерация аутентифицированных ключей** (authenticated key establishment). Первый вид аутентификации означает проверку объявленного свойства сообщения, второй больше внимания уделяет проверке достоверности сведений об отправителе сообщения, а третий предназначен для организации защищенного канала для обмена секретными сообщениями.

11.2.1 Аутентификация источника данных

Аутентификация источника данных (также называемая **аутентификацией сообщения** (message authentication)) тесно связана с защитой целостности данных. В первых книгах по криптографии и защите информации эти два понятия практически не различались [89, 93]. Данная точка зрения основывалась на том, что риск, возникающий в результате применения преднамеренно искаженной информацией, сравним с риском получить информацию из ненадежного источника.

Однако на самом деле аутентификация источника данных и защита целостности данных представляют собой *совершенно* разные понятия. Их можно различить по нескольким аспектам.

Во-первых, аутентификация источника данных обязательно связана с каналами связи. Она представляет собой службу безопасности получателя, предназначенную для верификации источников сообщений. Защита целостности данных не обязательно связана с коммуникациями: она может распространяться на данные, хранящиеся в компьютере.

Во-вторых, аутентификация источника данных обязательно связана с его идентификацией, а защита целостности данных — нет. В разделе 10.5 приведен пример защиты целостности данных, не требующей идентификации источника сообщения. Для описания этой ситуации даже использовался термин "защита целостности данных, посланных Злоумышленником". Заметим, что Злоумышленник — это безликий пользователь, никак не связанный с достоверным источником информации. В главе 15 будет показано, что "защита целостности данных, посланных Злоумышленником" представляет собой общий механизм, гарантирующий доказуемую стойкость криптосистем с открытым ключом.

В-третьих, что самое важное, аутентификация источника сообщения обязательно связана с проверкой его "**свежести**" (freshness), а защита целостности данных — нет: фрагмент устаревшей информации может превосходно соответствовать критериям целостности данных. Для того чтобы аутентифицировать источник данных, получатель сообщения должен удостовериться, что сообщение было послано *недавно* (т.е. промежуток времени между его отправлением и получением достаточно мал). Только в этом случае сообщение может считаться "свежим". "Свежесть" информации свидетельствует о хорошей *согласованности* между общающимися сторонами и снижает вероятность диверсии со стороны пользователей, аппаратуры, системы или самого сообщения. В разделе 2.6.4 бы-

ла описана атака Деннинга–Сакко на протокол аутентификации с симметричным ключом Нидхема–Шредера (атака 2.2). В этой атаке устаревшее сообщение полностью соответствовало критериям защиты целостности данных, но не являлось аутентичным. Нарушение аутентичности такого рода называется *целостностью данных без гарантии существования источника сообщения* ("valid data integrity without liveness of the message source").

Заметим, что "свежесть" сообщения зависит от конкретного приложения. В некоторых приложениях устанавливается довольно короткий интервал времени, в течение которого сообщение считается "свежим" (например, во многих системах реального времени этот интервал может измеряться секундами). С другой стороны, в некоторых приложениях допускаются более долгие промежутки времени. Например, во время Второй мировой войны сообщения, передаваемые по военным линиям связи Германии, шифровались с помощью знаменитых машин Энигма, в которых ежедневно применялся новый ключ [277]. До сих пор это правило управления ключами широко используется во многих системах безопасности, хотя в некоторых случаях ключи меняются не только каждый день, но и каждый час и даже каждую минуту. Некоторые приложения допускают намного более долгие интервалы времени, в течение которых сообщение считается "свежим". Например, когда банки проверяют целостность данных на банковском чеке и идентифицируют его источник, аутентичность чека зависит от времени его подписания. Следовательно, интервал времени, в течение которого чек считается действительным, определяется моментом его подписания и моментом его депонирования. Многие банки считают чеки действительными в течение трех месяцев.

В заключение следует отметить тот факт, что аутентификация источника данных и защита целостности данных в некоторых ситуациях различаются наличием анонимных мандатов (например, невидимой подписи (blind signature)). Пользователь может выдать анонимный мандат, позволяющий его владельцу воспользоваться услугами системы, доступными только для ее членов. Заметим, что целостность данных в таком случае также гарантируется, однако система не может идентифицировать пользователя. Этот криптографический метод будет изучен в следующей главе.

Итак, аутентификация источника данных предусматривает следующие действия.

1. Передача сообщения от отправителя к получателю, проверяющему достоверность сообщения перед его принятием.
2. Идентификация отправителя сообщения.
3. Проверка целостности данных, полученных от отправителя.
4. Проверка реальности отправителя сообщения.

11.2.2 Аутентификация сущности

Аутентификация сущности — это процесс обмена информацией (т.е. протокол), в ходе которого пользователь устанавливает *подлинность* (lively correspondence) другого пользователя. Часто слово "сущность" пропускают, как, например, в следующем предложении: "Важной целью протокола аутентификации является установление подлинности пользователя".

Как правило, в ходе протокола аутентификации выясняется подлинность сообщения. В таких ситуациях, чтобы убедиться в подлинности сообщения и его автора, следует воспользоваться механизмом аутентификации источника данных. Действительно, как будет показано в данной главе, если в протоколе проверяется подлинность сообщения, достоверность его автора лучше всего проверять именно с помощью механизма аутентификации источника данных.

Существует несколько сценариев аутентификации сущности в распределенных системах. Перечислим некоторые из них.

Обмен сообщениями между двумя главными компьютерами (host-host type). Участниками протокола являются компьютеры, называемые узлами или платформами распределенной системы. Работа компьютеров, как правило, должна быть согласованной. Например, если одна из удаленных платформ "перезагружается"[1], она должна идентифицировать достоверный сервер и передать ему необходимую информацию, например, достоверную копию операционной системы, достоверные установки таймера или достоверные установки окружения. Определение достоверности информации обычно выполняется с помощью протокола аутентификации. Как правило, протокол обмена сообщениями между двумя компьютерами представляет собой систему клиент-сервер, в которой один из компьютеров (клиент) обслуживается другим компьютером (сервером).

Обмен сообщениями между пользователем и главным компьютером (user-host type). Пользователь получает доступ к компьютерной системе, регистрируясь в главном компьютере. В простейшем случае пользователь регистрируется в главном компьютере через сетевой теледоступ (telnet) или передает файл в соответствии с протоколом передачи файлов (file transfer protocol — ftp). В обеих ситуациях запускается протокол аутентификации пароля. Во многих серьезных приложениях, в которых скомпрометированный главный компьютер может привести к большим потерям (например, когда пользователь осуществляет оплату с помощью кредитной карточки с микропроцессором), необходима **взаимная аутентификация** (mutual authentication).

---

[1] "Перезагрузка" — технический термин, означающий повторную инициализацию компьютерной системы в соответствии с простыми предварительными инструкциями или информацией, которая "зашита" в систему.

Обмен сообщениями между процессом и главным компьютером (process-host type). В настоящее время развитие методов распределенных вычислений предоставило пользователям широкие функциональные возможности. Главный компьютер может предоставлять внешним процессам широкие права. Например, на удаленный компьютер могут передаваться и запускаться на нем фрагменты "кода для мобильного телефона" или "аплет, написанный на языке Java™"[2]. В секретных системах необходимо предусматривать механизмы аутентификации разработчика, чтобы не допустить выполнение враждебных аплетов и предоставлять права доступа только разрешенным аплетам.

Члены клуба (member-club type). Доказательство членства в клубе представляет собой обобщение способа, основанного на обмене сообщениями между пользователем и главным компьютером. В этом случае клуб интересует только мандат его члена, а не информация о нем. В частности, клуб не интересуется подлинностью личности члена клуба, имеющего мандат. Данный сценарий реализуется в протоколах идентификации с нулевым разглашением (zero-knowledge identification protocol) и схемах неоспоримой цифровой подписи (undeniable signature schemes). Эти методы изучаются в главе 18.

11.2.3 Генерация аутентифицированных ключей

Как правило, стороны, обменивающиеся информацией, запускают протокол аутентификации сущности для того, чтобы в дальнейшем перевести общение на более высокий уровень. В современной криптографии в основе организации защищенных каналов связи лежат криптографические ключи. Следовательно, протоколы аутентификации сущностей для дальнейшего обмена информацией по защищенным каналам в качестве составной части должны содержать механизм **генерации аутентифицированных ключей** (authenticated key establishment), или **обмена ключами** (key exchange), или **согласования ключей** (key agreement).

Напомним, что протоколы аутентификации сущностей могут предусматривать аутентификацию источника данных. Аналогично, в протоколах генерации аутентифицированных ключей протокольные сообщения содержат параметры ключей, источник которых также подлежит аутентификации.

В литературе протоколы аутентификации (сущностей), протоколы генерации аутентифицированных ключей (обмена ключами, согласования ключей), протоколы для защиты данных и даже криптографические протоколы часто называют **протоколами связи** (communication protocols).

---

[2] Аплет, написанный на языке Java™, — это выполняемый код, запускаемый под управлением Web-браузера на удаленном компьютере.

11.2.4 Атака на протоколы аутентификации

Поскольку целью протокола аутентификации (источника данных, сущности, ключа) является проверка объявленного свойства, необходимо применить криптографические методы. Атака на протокол преследует противоположную цель. Атака на протокол аутентификации организуется противником или коалицией противников (под общим названием "Злоумышленник"), преследующих незаконные цели. Цель Злоумышленника может быть серьезной, например, взлом секретного сообщения или ключа, или менее серьезной, например, обман получателя сообщения. Как правило, протокол аутентификации считается некорректным, если пользователь считает, что протокол выполняется правильно и связь установлена с подлинным партнером, в то время как подлинный партнер приходит к противоположному выводу.

Следует подчеркнуть, что атаки на протоколы аутентификации, как правило, не связаны со взломом криптографических алгоритмов. Обычные протоколы аутентификации небезопасны не потому, что в них применяются слабые криптографические алгоритмы, а потому, что имеют недостатки, позволяющие Злоумышленнику пройти аутентификацию вообще без взлома криптографического алгоритма. В главе описано много таких атак. По этой причине при анализе протоколов аутентификации обычно предполагают, что лежащий в его основе криптографический алгоритм является "совершенным", и не рассматривают его потенциальные слабости. Анализ слабых мест криптографических алгоритмов является предметом другой отрасли криптографии.

11.3 Соглашения

В протоколах аутентификации, рассмотренных в остальной части главы, приняты следующие соглашения о смысле протокольных сообщений и их синтаксической структуре.

- *Алиса*, *Боб*, *Трент*, *Злоумышленник*, ... — имена пользователей, встречающиеся в протокольных сообщениях. Иногда вместо них используются аббревиатуры $A, B, T, M, \ldots$
- *Алиса* $\rightarrow$ *Бобу* : M. Алиса посылает Бобу сообщение M. Спецификация протокола представляет собой последовательность таких сообщений.
- $\{M\}_K$ — зашифрованный текст, представляющий собой сообщение M, зашифрованное ключом K.
- $K, K_{AB}, K_{AT}, K_A, \ldots$ — криптографические ключи, где K_{XY} — ключ, разделенный между пользователями X и Y, а K_X — открытый ключ, принадлежащий пользователю X.

- $N, N_A, \ldots$ — одноразовые случайные числа (nonces) [61], которые извлекаются из довольно большого пространства. Число N_X генерируется пользователем X.
- T_X — метка времени, поставленная пользователем X.
- $\mathrm{sig}_A(M)$ — цифровая подпись сообщения M, созданная пользователем A.

Примечание 11.1. *Следует отметить, что семантический смысл протокольных сообщений, связанный с их синтаксической структурой (типом), не обязательно должен быть понятным всем участникам протокола (например, Алисе). Как правило, если спецификация протокола не требует, чтобы Алиса выполняла криптографическую операцию над сообщением или частью сообщения, то Алиса (точнее, ее протокольный компилятор) понимает сообщение только на синтаксическом уровне. При этом Алиса может неверно понимать семантический смысл протокольного сообщения. Образцы неверной интерпретации сообщений приведены в примере 11.1.* □

Пример 11.1. На синтаксическом уровне Алиса может неверно интерпретировать смысл протокольных сообщений. Перечислим несколько примеров.

- Алиса может считать часть сообщения зашифрованным текстом и пытаться расшифровать ее, полагая, что владеет правильным ключом, или переслать Бобу, думая, что эта часть сообщения предназначена ему. Однако на самом деле эта часть сообщения может представлять собой имя пользователя (например, *Алиса* или *Боб*), одноразовое случайное число или метку времени.
- Алиса может расшифровать зашифрованный текст и послать результат, "следуя протокольным инструкциям", хотя на самом деле этот текст может представлять собой ее более раннее сообщение, возникшее, возможно, в другом контексте.
- Алиса может принять параметр ключа за одноразовое случайное число и т.п.

Может показаться, что Алиса очень "глупа" и не понимает протокольных сообщений. На самом деле это не так. Скорее, она слишком наивна и не всегда догадывается о существовании коварного Злоумышленника, который, возможно, уже "перекомпилировал" протокол, перепутав фрагменты сообщений так, чтобы их невозможно было понять. □

Примем следующие соглашения о поведении участников протокола, законных и незаконных.

- Подлинный пользователь, участвующий в протоколе, не понимает семантического смысла ни одного протокольного сообщения, пока протокол не завершится успешно.

- Подлинный пользователь, участвующий в протоколе, не способен ни распознать, ни создать, ни разложить сообщение $\{M\}_K$ на составные части, не имея правильного ключа.
- Подлинный пользователь, участвующий в протоколе, не способен распознавать псевдослучайные числа, порядковые номера и криптографические ключи, если они не сгенерированы самим пользователем в рамках текущего сеанса или не являются результатом выполнения протокола.
- Подлинный пользователь, участвующий в протоколе, не записывает протокольные сообщения, если этого не требует спецификация протокола. Как правило, протокол аутентификации *не имеет истории* (stateless), т.е. пользователь не должен запоминать никакой информации о состоянии, возникшем после успешного выполнения протокола, за исключением информации, являющейся результатом выполнения протокола и предназначенной для самого пользователя.
- Злоумышленник, кроме возможностей, указанных в разделе 2.3, знает о "глупости" (а точнее, наивности) подлинных пользователей (см. пример 11.1) и всегда пытается использовать это обстоятельство.

Протоколы аутентификации предназначены для передачи сообщений по открытым каналам связи, предположительно пребывающим под контролем Злоумышленника, и должны препятствовать атакам противника, несмотря на "хитроумие" Злоумышленника и "глупость" подлинных пользователей.

Рассмотрим способы, которые позволяют достичь этой цели.

11.4 Основные методы аутентификации

Существует много методов реализации протоколов аутентификации (источника данных, сущности) и генерации аутентифицированных ключей. Однако основных протокольных конструкций, особенно удачных, не так много.

Начнем с основных протокольных конструкций, уделяя особое внимание конструкциям, принятым в качестве международных стандартов. Эти конструкции могут и должны использоваться при разработке новых протоколов аутентификации. Попробуем разобраться, почему некоторые из этих конструкций более привлекательны, а другие имеют недостатки.

Перечислим основные методы аутентификации, рассматриваемые в этом разделе.

- Стандартные механизмы определения "свежести" сообщения и существования пользователя (раздел 11.4.1).
- Взаимная аутентификация и односторонняя аутентификация (раздел 11.4.2).
- Аутентификация с привлечением доверенного посредника (раздел 11.4.3).

11.4.1 "Свежесть" сообщения и существование пользователя

Проверка "свежести" сообщения — неотъемлемая часть аутентификации источника данных (различия между *идентификацией* и *аутентификацией* источника сообщения описаны в разделе 11.2.1), а также аутентификации сущностей, в процессе которой пользователь должен оживленно обмениваться информацией с подлинным партнером. Следовательно, механизмы, позволяющие установить "свежесть" сообщения или существование пользователя, относятся к основным компонентам протокола аутентификации.

Опишем стандартные механизмы, позволяющие решить поставленную задачу. В наших описаниях Алиса будет заявлять о некотором свойстве (например, своем существовании или "свежести" сообщения), а Боб будет проверять, соответствует ли это действительности. Будем считать, что Алиса и Боб владеют общим ключом K_{AB}, если механизм аутентификации использует симметричные криптографические методы, или Бобу известен открытый ключ Алисы, распространяемый через систему сертификации открытых ключей[3], если в основе механизма аутентификации лежат методы асимметричной криптографии.

11.4.1.1 Стратегия "оклик-отзыв"

В **стратегии "оклик-отзыв"** (challenge-responce mechanism) Боб (верификатор) получает смесь, состоящую из протокольного сообщения и криптографической операции, выполненной Алисой (претендентом) таким образом, чтобы Боб мог убедиться в ее существовании, проверив "свежесть" полученной информации. Как правило, Боб получает одноразовое случайное число (nonce), заблаговременно сгенерированное Бобом и посланное Алисе. Обозначим это одноразовое число символом N_B. Стратегия проверки "свежести" сообщения выглядит следующим образом.

1. Боб $\rightarrow$ Алисе: N_B.
2. Алиса $\rightarrow$ Бобу: $\mathcal{E}_{K_{AB}}(M, N_B)$.
3. Боб расшифровывает порцию шифра и либо принимает ее, если узнает число N_A, либо отказывается принимать в противном случае. (11.4.1)

Первое сообщение, посланное Бобом Алисе, называется **окликом** (challenge), а второе сообщение, посланное Алисой Бобу, называется **отзывом** (response). Боб является **инициатором** (initiator), а Алиса — **ответчиком** (responder).

Описанная выше стратегия использует метод симметричной криптографии, а именно: симметричное шифрование. Следовательно, получив ответ от Алисы,

[3] Система сертификации открытых ключей (public-key certification framework) описывается в главе 13.

Боб должен расшифровать порцию шифрованного текста, используя совместный ключ K_{AB}. Если расшифровка *правильно* восстанавливает случайное число (термин "*правильно*", как будет показано позднее, на самом деле относится к целостности данных), Боб имеет основания считать, что Алиса действительно зашифровала его *после* получения оклика. Если интервал между окликом и отзывом достаточно мал (как показано в разделе 11.2.1, его величина зависит от приложения), сообщение M считается "свежим". Эта стратегия основана на уверенности пользователя в том, что Алиса выполняет шифрование после получения оклика от Боба. Поскольку случайное число, посланное Бобом, было извлечено из достаточно большого пространства, нет никакой возможности предсказать его заранее.

Рассмотрим термин "правильный", который мы использовали, когда Боб расшифровывал свое случайное число. На первый взгляд может показаться, что применение симметричного шифрования гарантирует секретность сообщения. На самом деле секретность сообщения обеспечивает целостность данных. Возникает вопрос: почему бы не засекретить сообщение M, например, оно могло бы быть криптографическим ключом, позволяющим начать секретную переписку. (В этом случае описанную конструкцию можно использовать для обмена сеансовыми ключами.) Это вполне законное предположение. Можно даже предположить, что Боб зашифровывает и свое первое сообщение. Таким образом, мы не утверждаем, что шифрование выполнять не следует. Мы лишь подчеркиваем, что если алгоритм шифрования не обеспечивает необходимой целостности данных (а это на самом деле так!), то описанная выше стратегия является небезопасной. Сформулируем утверждение, которое будет доказано в разделе 17.2.1.

Примечание 11.2. *Если алгоритм шифрования, используемый в стратегии аутентификации (11.4.1), не гарантирует целостности данных, то Боб не может проверить "свежесть" сообщения M.* □

Действительно правильной и стандартной стратегией, гарантирующей целостность данных при использовании симметричных методов шифрования, является применение кода распознавания манипуляций (MDC, см. определение 10.1 в разделе 10.1). Следовательно, в стратегии (11.4.1) шифрование следует сопровождать кодом MDC, зашифрованным с помощью общего ключа и являющимся частью зашифрованного текста. Это обеспечивает защиту целостности сообщения. Если сообщение M не нуждается в защите секретности, для проверки его "свежести" можно применить следующую стратегию.

(11.4.2)

1. Боб → Алисе: N_B.
2. Алиса → Бобу: $M, \mathtt{MDC}(K_{AB}, M, N_B)$.
3. Боб восстанавливает код $\mathtt{MDC}(K_{AB}, M, N_B)$ и либо принимает его, если два кода MDC оказались идентичными, либо отказывается принимать в противном случае.

Для того чтобы Боб мог восстановить код MDC на этапе 3, сообщение M на этапе 2 должно быть послано в виде открытого текста. Разумеется, сообщение M может быть зашифровано.

В разделе 17.2.1 мы покажем, что с точки зрения аутентификации на основе методов симметричной криптографии стратегия (11.4.2) является правильной, а стратегия (11.4.1) — нет. Кроме того, мы докажем, что при нарушении целостности данных секретность сообщения M в стратегии (11.4.1) нарушается, даже если применяется строгий алгоритм шифрования.

Стратегия "оклик-отзыв" может использовать и методы асимметричного шифрования.

1. Боб → Алисе: N_B.
2. Алиса → Бобу: $\mathrm{sig}_A(M, N_B)$.
3. Боб проверяет цифровую подпись, используя свое одноразовое случайное число, и либо принимает его, если подпись является подлинной, либо отказывается в противном случае.

(11.4.3)

Обратите внимание на то, что в этой стратегии свободный выбор Алисой сообщения M является важным обстоятельством. В этом случае Алиса не может непреднамеренно подписать сообщение, подготовленное Бобом. Например, Боб может приготовить свое "случайное число" следующим образом:

$N_B = h$ (Перевести 1 000 фунтов на счет Боба № 123 со счета Алисы № 456),

где h — функция хэширования.

В некоторых приложениях лицо, подписывающее сообщение и играющее роль Алисы в стратегии (11.4.3), не имеет свободы при выборе сообщения M. В таких ситуациях используются специализированные ключи. Например, открытый ключ, предназначенный для верификации подписи Алисы в стратегии (11.4.3), можно уточнить. Специализация криптографических ключей является предметом изучения теории **управления ключами** (key management).

11.4.1.2 Стандартная стратегия "оклик-отзыв"

Международная организация по стандартизации (International Organization for Standardization — ISO) и Международная электротехническая комиссия (International Electrotechnical Commission — IEC) приняли описанные выше три стратегии в качестве стандартных конструкций стратегии **односторонней аутентификации сущности** (unilateral entity authentication). Стандартный вариант стратегии (11.4.1) называется "Двухпроходным односторонним протоколом аутентификации ISO" ("ISO Two-pass Unilateral Authentication Protocol") [147]. Он выглядит следующим образом.

1. $B \to A : R_B \parallel \text{Text1}$;
2. $A \to B : \text{Token}AB$.
   Здесь $\text{Token}AB = \text{Text3} \parallel \mathcal{E}_{K_{AB}}(R_B \parallel B \parallel \text{Text2})$.
   Получив сообщение $\text{Token}AB$, Боб должен расшифровать его. Если в результате расшифровки восстанавливается правильное значение случайного числа R_B, аутентификация считается успешной, а если нет — безуспешной.

Здесь и далее при описании стандартов ISO/IEC используются обозначения, принятые в спецификациях стандартных протоколов, где Text1, Text2 и т.д. — необязательные поля, символ $\parallel$ означает операцию конкатенации, а R_B — одноразовое случайное число, сгенерированное Бобом.

Напомним, что алгоритм шифрования должен обеспечивать защиту целостности данных. Это является необходимым условием для того, чтобы можно было проверить правильность результатов расшифровки (см. замечание 11.2 в разделе 11.4.1.1).

Заметим также, что стратегия (11.4.1) предназначена для проверки "свежести" сообщений, а ее стандартный аналог — для аутентификации сущностей. Следовательно, включение сообщения B, т.е. имени Боба вместо сообщения M в стратегии (11.4.1) является очень важным моментом: это подчеркивает, что механизм ISO/IEC предназначен для проверки существования Боба и представляет собой протокол аутентификации, в котором Боб является субъектом. Абади (Abadi) и Нидхем создали список принципов, которыми должен руководствоваться разработчик криптографических протоколов [1]. Одним из таких принципов является явное выделение сущности, представляющей собой субъект аутентификации. В разделе 11.7.7 мы убедимся, что нарушение этого принципа создает опасную ситуацию.

Стандартный вариант стратегии (11.4.2) называется "двухпроходным односторонним протоколом аутентификации с использованием криптографической тестовой функции (CCF)" ("ISO Two-pass Unilateral Authentication Protocol Using a Cryptographic Check Function (CCF)") [149]. Он выглядит следующим образом.

1. $B \to A : R_B \parallel \text{Text1}$;
2. $A \to B : \text{Token}AB$.
   Здесь[4] $\text{Token}AB = \text{Text2} \parallel f_{K_{AB}}(R_B \parallel B \parallel \text{Text2})$, где f — криптографическая функция хэширования, имеющая ключ.
   Получив сообщение $\text{Token}AB$, Боб должен реконструировать ключевую функцию CCF, используя общий ключ, свое случайное число, свое имя и поле Text2. Если в результате расшифровки восстановленный блок функ-

[4] В работе [149] поле Text2 в открытом тексте ошибочно обозначено как Text3. Если в открытом тексте нет поля Text2, пользователь B не может верифицировать функцию CCF путем ее реконструкции.

ции CCF совпадает с полученным, аутентификация считается успешной, а если нет — безуспешной.

Стандартный вариант стратегии (11.4.3) называется "двухпроходным односторонним протоколом аутентификации с открытым ключом ISO" ("ISO public-Key Two-pass Unilateral Authentication Protocol") [148]. Он выглядит следующим образом.

1. $B \rightarrow A : R_B \parallel \text{Text1}$;
2. $A \rightarrow B : \text{Cert}A \parallel \text{Token}AB$.
   Здесь $\text{Token}AB = R_A \parallel R_B \parallel B \parallel \text{Text3} \parallel \text{sig}_A(R_A \parallel R_B \parallel B \parallel \text{Text2})$, где Cert_A — сертификат открытого ключа Алисы (это понятие вводится в следующей главе).
   Получив сообщение $\text{Token}AB$, Боб должен верифицировать его цифровую подпись. Если подпись проходит проверку, аутентификация считается успешной, а если нет — безуспешной.

Как указывалось при описании стратегии (11.4.3), в данном протоколе ISO/IEC пользователь A может свободно выбирать случайное число R_A для того, чтобы предотвратить непреднамеренное подписание сообщения, подготовленное пользователем B.

11.4.1.3 Метка времени

Используя **механизм метки времени** (timestamp mechanism), Алиса указывает время создания своего сообщения, используя криптографическую операцию. Следовательно, время создания сообщения становится неотъемлемой частью ее сообщения.

Обозначим через T_A метку времени, поставленную Алисой на свое сообщение. Механизм проверки "свежести" сообщений становится автономным.

1. Алиса $\rightarrow$ Бобу: $\mathcal{E}_{K_{AB}}(M, T_A)$.
2. Боб расшифровывает порцию шифра и либо принимает ее, если распознает корректную метку времени T_A, либо отказывается принимать ее в противном случае. (11.4.4)

Аналогично стратегии (11.4.1) расшифровка, выполняемая Бобом, должна сопровождаться проверкой целостности данных (см. раздел 11.4.1.1 и замечание 11.2). После расшифровки Боб может сравнить извлеченную метку времени T_A со своим собственным временем (предполагается, что участники протокола используют общемировое стандартное время, например, по Гринвичу). Если разница во времени относительно мала, сообщение M считается "свежим".

В разделе 11.4.1.1 указывалось, что нарушение целостности данных может привести к неверному применению всего механизма защиты информации. По этой

причине в механизме метки времени лучше использовать методы симметричного шифрования.

1. Алиса → Бобу: $M, T_A, \mathtt{MDC}(K_{AB}, M, T_A)$.
2. Боб реконструирует код $\mathtt{MDC}(K_{AB}, M, T_A)$ и, если два кода MDC оказываются идентичными, а метка времени T_A — корректной, принимает его, либо отказывается принимать в противном случае. (11.4.5)

В нашей версии Боб выполняет проверку целостности данных, оценивая однонаправленность криптографического преобразования между меткой времени и сообщением. Разумеется, если сообщение M также должно быть секретным, следует применять шифрование. Однако шифрование не отменяет необходимости защиты целостности данных.

Очевидно, что механизм метки времени можно создать с помощью методов асимметричного шифрования.

1. Алиса → Бобу: $\mathrm{sig}_A(M, T_A)$.
2. Боб проверяет цифровую подпись и, если подпись оказывается подлинной, а метка времени T_A — корректной, принимает сообщение, либо отказывается принимать в противном случае. (11.4.6)

Механизм метки времени позволяет избежать взаимодействия пользователей и может применяться в приложениях, не предусматривающих интерактивного режима. Однако механизм метки времени имеет недостаток — он требует надежной синхронизации таймеров. Это может оказаться трудной задачей. Сложности, предостережения и возражения, связанные с применением механизма метки времени, описаны в работах [28, 34, 90, 115].

В основных протоколах, разработанных выше, сообщения содержали специальный компонент — одноразовое случайное число или метку времени, предназначенные для гарантии "свежести" сообщений, в которые они интегрированы. По этой причине эти компоненты называются **идентификаторами "свежести" сообщений** (freshness identifiers).

11.4.1.4 Стандартный механизм меток времени

Международная организация по стандартизации и Международная электротехническая комиссия разработали стандартные варианты протоколов аутентификации с помощью механизма меток времени.

Стандартный вариант механизма (11.4.4) называется "Однопроходным односторонним протоколом аутентификации с симметричным ключом ISO" ("ISO Symmetric Key One-Pass Unilateral Authentication Protocol" [147]. Он выглядит следующим образом.

1. $A \rightarrow B$: TokenAB;
   Здесь TokenAB = Text2 $\|$ $\mathcal{E}_{K_{AB}}\left(\begin{smallmatrix} T_A \\ N_A \end{smallmatrix} \| B \| \text{Text1}\right)$.

Поскольку в этой стратегии используется простой механизм шифрования-расшифровки, необходимо напомнить, что алгоритм шифрования должен обеспечивать защиту целостности данных.

Здесь $\begin{smallmatrix} T_A \\ N_A \end{smallmatrix}$ обозначает выбор между применением метки времени T_A и случайного числа N_A, которое называется **порядковым номером** (sequence number). Если применяется порядковый номер, Алиса и Боб должны синхронизировать его (например, используя общий счетчик) так, чтобы Боб знал о каждом увеличении порядкового номера N_A. После успешного получения и проверки порядкового номера каждый из двух участников протокола должен обновить счетчик.

Механизм, применяющий порядковый номер, имеет два недостатка. Во-первых, каждый из партнеров должен хранить информацию о состоянии счетчика. В открытых сетях, где каждый пользователь взаимодействует со многими партнерами, это может оказаться затруднительным. Следовательно, применение порядкового номера недостаточно хорошо масштабируется. Во-вторых, управление порядковым номером может вызвать сложности при наличии помех на линии связи, как случайных, так и преднамеренных (так называемая **атака на основе отказа в обслуживании** (denial-of-service attack)). Напомним о соглашениях, принятых в разделе 11.3. В них говорилось, что протокол аутентификации не должен иметь истории, поскольку протокол с историей не может правильно функционировать во враждебной среде. Таким образом, механизм, основанный на применении порядкового номера, нельзя рекомендовать для применения, хотя он и стандартизован.

Стандартный вариант механизма (11.4.5) называется "Однопроходным односторонним протоколом аутентификации с использованием криптографической тестовой функции (CCF)" ("ISO One-Pass Unilateral Authentication with Cryptographic Check Function" [149]. Он выглядит следующим образом.

1. $A \rightarrow B$: TokenAB;
   Здесь[5] TokenAB = $\|$ B $\|$ Text1 $\|$ $f_{K_{AB}}\left(\begin{smallmatrix} T_A \\ N_A \end{smallmatrix} \| B \| \text{Text1}\right)$, где f — криптографическая функция хэширования, имеющая ключ.

Читатель уже может угадать название следующего протокола — аналога стратегии шифрования с открытым ключом: "Однопроходный односторонний протокол аутентификации с открытым ключом ISO" ("ISO Public Key One-Pass Unilateral Authentication Protocol") [148].

1. $A \rightarrow B$: Cert$A$ $\|$ TokenAB;
   Здесь TokenAB = $\begin{smallmatrix} T_A \\ N_A \end{smallmatrix}$ $\|$ B $\|$ Text2 $\|$ $\text{sig}_A\left(\begin{smallmatrix} T_A \\ N_A \end{smallmatrix} \| B \| \text{Text1}\right)$.

[5] В работе [149] поле Text2 в открытом тексте ошибочно обозначено как Text1, поэтому пользователь B не может верифицировать функцию CCF путем ее реконструкции.

11.4.1.5 Нестандартные стратегии

В предыдущих разделах описаны основные конструкции, применяемые при создании протоколов аутентификации. Нетрудно представить себе другие многочисленные варианты конструкций, предназначенных для достижения той же цели. Например, вариант стратегии (11.4.1), использующий методы симметричной криптографии, может выглядеть следующим образом.

1. Боб $\rightarrow$ Алисе : $Боб, \mathcal{E}_{K_{AB}}(M, N_B)$.
2. Алиса $\rightarrow$ Бобу : N_B.
3. Боб проверяет случайное число. Если оно оказалось правильным, Боб принимает сообщение, в противном случае — отказывается.

(11.4.7)

· Вариант стратегии (11.4.3), использующий методы асимметричной криптографии, может выглядеть следующим образом.

4. Боб $\rightarrow$ Алисе : $Боб, \mathcal{E}_{K_A}(M, Боб, N_B)$.
5. Алиса $\rightarrow$ Бобу : N_B.
6. Боб проверяет случайное число. Если оно оказалось правильным, Боб принимает сообщение, в противном случае — отказывается.

(11.4.8)

Здесь $\mathcal{E}_{K_{AB}}$ — алгоритм шифрования с открытым ключом Алисы. В этих двух вариантах Боб сообщает Алисе о своем существовании, зашифровывая идентификатор "свежести" и проверяя, может ли она выполнить своевременную расшифровку. Этот механизм называется "шифрованием с последующей расшифровкой" (encryption-then-decryption).

Шифрование идентификатора "свежести" с его последующей расшифровкой позволяет проверить, существует ли партнер, участвующий в обмене информацией. Однако этот механизм непригоден для создания протокола аутентификации, поскольку Алиса в нем может играть роль оракула расшифровки (см. разделы 7.8.2.1 и 8.9) и непреднамеренно раскрыть конфиденциальную информацию. Например, Злоумышленник может записать фрагмент зашифрованного текста, которым обменялись Алиса и Боб, вставить его в протокол, использующий механизм шифрования с последующей расшифровкой, а затем обманным путем заставить Алису раскрыть содержание секретной информации. Напомним, что в соответствии с нашими соглашениями (см. раздел 11.3) Алиса может неверно интерпретировать сообщение как случайное число и вернуть его, руководствуясь протокольными инструкциями.

Нежелательность применения механизма шифрования с последующей расшифровкой подтверждается тем фактом, что Международный институт по стандартизации и Международная электротехническая комиссия даже не стали его рассматривать в качестве кандидата на стандартизацию. Именно по этой причине мы называем стратегии (11.4.7) и (11.4.8) нестандартными.

Несмотря на это, многие протоколы аутентификации используют механизм шифрования с последующей расшифровкой. Мы проанализируем эти протоколы в разделе 17.2. Там же будет показано, что их основной слабостью является как раз механизм шифрования с последующей расшифровкой.

11.4.2 Взаимная аутентификация

До сих пор механизмы проверки "свежести" сообщения и существования пользователя обеспечивали только так называемую "одностороннюю аутентификацию", в которой аутентифицировался только один из двух участников протокола. При **взаимной аутентификации** (mutual authentication) пользователи аутентифицируют друг друга.

Международный институт по стандартизации и Международная электротехническая комиссия стандартизировали большое количество протоколов взаимной аутентификации. Протокол 11.1 описывает стратегию, основанную на применении цифровой подписи: "Трехпроходный протокол взаимной аутентификации с открытым ключом ISO" ("ISO Public Key Three-Pass Mutual Authentication Protocol") [148]. Он хорошо иллюстрирует основные заблуждения, связанные со взаимной аутентификацией.

Протокол 11.1. Трехпроходный протокол взаимной аутентификации с открытым ключом ISO

ИСХОДНЫЕ УСЛОВИЯ:

Пользователь A владеет сертификатом открытого ключа Cert_A.
Пользователь B владеет сертификатом открытого ключа Cert_B.

ЦЕЛЬ: Взаимная аутентификация.

1. $B \rightarrow A : R_B$.
2. $A \rightarrow B : \text{Cert}_A, \text{Token}AB$.
3. $B \rightarrow A : \text{Cert}_B, \text{Token}BA$.

Здесь

$\text{Token}AB = R_A \parallel R_B \parallel B \parallel \text{sig}_A(R_A \parallel R_B \parallel B \parallel)$,
$\text{Token}BA = R_B \parallel R_A \parallel A \parallel \text{sig}_B(R_B \parallel R_A \parallel A \parallel)$.

(* Необязательные поля пропущены. *)

Может показаться, что взаимная аутентификация представляет собой двукратную одностороннюю аутентификацию, т.е. для взаимной аутентификации достаточно дважды выполнить протокол односторонней аутентификации 11.4.1 в противоположных направлениях. Однако это не так!

На первых этапах стандартизации эксперты не различали взаимную и одностороннюю аутентификации. В нескольких предварительных вариантах протокола 11.1 [130, 143] сообщение TokenBA имело немного другой вид.

$$\text{Token}BA = R'_B \parallel R_A \parallel A \parallel \text{sig}_B(R'_B \parallel R_A \parallel A \parallel).$$

В первом варианте протокола пользователю B не разрешалось повторно использовать случайное число R_B, чтобы избежать подписания частично определенной строки, которая заранее известна пользователю A. Сообщение TokenBA представляло собой зеркальное отражение сообщения TokenAB. Протокол ISO/IEC 9798-3 несколько раз подвергался пересмотру, пока Винер (Wiener) из канадского отдела института ISO не изобрел знаменитую "канадскую атаку" [143]. В дополнение к стандартной документации Диффи (Diffie), ван Оршот (van Oorschot) и Винер описали эту атаку в работе [99]. В дальнейшем мы будем называть ее атакой Винера.

11.4.2.1 Атака Винера (канадская атака)

Атака Винера на первый вариант трехпроходного протокола взаимной аутентификации с открытым ключом ISO описана ниже (напомним обозначения, принятые в разделе 2.6.2 при описании замаскированных действий Злоумышленника).

Атака 11.1. Атака Винера на трехпроходный протокол взаимной аутентификации с открытым ключом ISO

ИСХОДНЫЕ УСЛОВИЯ:

Кроме условий протокола 11.1, Злоумышленник обладает сертификатом открытого ключа Cert_M.

1. Злоумышленник("B") $\rightarrow A : R_B$.
2. $A \rightarrow$ Злоумышленнику("B") : $\text{Cert}_A, R_A \parallel R_B \parallel B \parallel \text{sig}_A(R_A \parallel R_B \parallel B)$.

1'. Злоумышленник("B") $\rightarrow B : R_A$.

2'. $B \rightarrow$ Злоумышленнику("A") : $\text{Cert}_B, R'_B \parallel R_A \parallel A \parallel \text{sig}_B(R'_B \parallel R_A \parallel A)$.

3. Злоумышленник("B") $\rightarrow A, \text{Cert}_B, R'_B \parallel R_A \parallel A \parallel \text{sig}_B(R'_B \parallel R_A \parallel A)$.

ПОСЛЕДСТВИЯ:

Пользователь A полагает, что протокол был инициализирован пользователем B, и считает идентификацию успешной. На самом деле пользователь B не инициализировал протокол и ожидает завершения протокола, запущенного Злоумышленником("A").

После изобретения атаки Винера к серии протоколов ISO/IEC 9798 стали относиться с осторожностью. Если сообщение TokenAB возникает в одностороннем протоколе аутентификации, то во всех протоколах взаимной аутентификации, построенных на основе их одностороннего варианта, сообщение TokenAB оказывается связанным с сообщением TokenAB. Эта связь обычно возникает вследствие повторного использования идентификатора "свежести" в текущем сеансе.

В текущей версии трехпроходного протокола взаимной аутентификации с открытым ключом ISO (протокол 11.1, защищенный от атаки Винера) в течение всего сеанса пользователь A должен поддерживать состояние, соответствующее одноразовому случайному числу R_B, сгенерированному пользователем B.

11.4.3 Аутентификация с привлечением доверенного посредника

В основных конструкциях протоколов аутентификации, описанных в главе, предполагалось, что между участниками протокола установлен защищенный канал (созданный с помощью методов симметричной криптографии) или они знают открытый ключ партнера (если используются методы асимметричной криптографии). Итак, можно утверждать, что эти протоколы применяются пользователями, уже знакомыми друг с другом. Зачем же они запускают протокол аутентификации? Возможно несколько ответов. Во-первых, они могут "освежить" защищенный канал связи между ними, убедившись в реальном существовании партнера.

Во-вторых, что более важно, эти конструкции являются строительными конструкциями более общих протоколов аутентификации, функционирующих в открытых системах.

В стандартном режиме функционирования открытых систем пользователи взаимодействуют, а затем забывают друг друга. Открытые системы слишком велики, чтобы пользователь мог хранить информацию о своих контактах с другими пользователями. Если два пользователя, незнакомых друг с другом, захотят установить между собой секретную связь, они могут организовать защищенный канал связи. В современной криптографии такой канал связи защищается криптографическим ключом. Следовательно, два пользователя, желающих установить защищенный канал связи между собой, должны запустить протокол аутентификации, который называется протоколом генерации аутентифицированного ключа. Завершив сеанс секретной связи, они разрушают канал. Термин "разрушение канала" означает, что пользователи забывают ключ, защищавший канал, и никогда больше не используют его. Вот почему канал, используемый для генерации аутентифицированного ключа, часто называется сеансовым каналом, а ключ, защищавший его, — сеансовым ключом.

Как правило, для аутентификации и генерации ключей в открытых системах используется централизованная служба аутентификации, известная под названием

доверенный посредник (trusted third party — TTP). Такая служба может быть интерактивной (online) или автономной (offline). В следующей главе мы рассмотрим средства аутентификации с помощью автономного доверенного посредника.

Если аутентификация осуществляется интерактивным доверенным посредником, считается, что между ним и большим количеством пользователей в системе существуют долговременные отношения. Протоколы аутентификации и генерации аутентифицированных ключей в интерактивных системах с использованием доверенного посредника разрабатываются на основе основных конструкций, описанных в разделах 11.4.1 и 11.4.2, в которых одним из "знакомых" пользователей является сам доверенный посредник. Криптографические операции, выполняемые доверенным посредником, могут зависеть от криптографических операций, выполняемых пользователями. С помощью доверенного посредника можно установить защищенный канал даже между пользователями, не знакомыми друг с другом. Большое количество таких протоколов описано в главе 2, где доверенный посредник назывался Трентом.

Стандартные протоколы аутентификации (серия 9798), одобренные организациями ISO/IEC, содержат две стандартные конструкции, в которых участвует интерактивный доверенный посредник [147]. Одна из таких стратегий называется "Четырехпроходным протоколом аутентификации ISO" ("ISO Four-Pass Authentication Protocol"), а другая — "Пятипроходным протоколом аутентификации ISO" ("ISO Five-Pass Authentication Protocol"). Эти протоколы обеспечивают взаимную аутентификацию пользователей и генерацию аутентичных сеансовых ключей. Однако мы не будем описывать их по следующим причинам.

Во-первых, эти протоколы построены на основе стандартных протокольных конструкций, описанных в разделе 11.4.1 и 11.4.2, и, следовательно, не несут никакой новой, полезной для изучения информации. Кроме того, они содержат много особенностей, связанных со стандартизацией, и их рассмотрение лишь затемнило бы простые идеи, лежащие в основе этих протоколов.

Во-вторых, они уже представляют собой полноценные протоколы аутентификации и не должны использоваться в качестве строительных конструкций для создания протоколов аутентификации более высокого уровня. Более того, они обладают некоторыми нежелательными свойствами, например, участники этих протоколов (и даже доверенные посредники!) используют порядковые номера. Таким образом, эти протоколы ни в коем случае не следует использовать в качестве образца для разработки новых протоколов!

По этой причине основное внимание мы уделим протоколу аутентификации сущности с помощью доверенного посредника. Однако следует иметь в виду, что он уязвим для некоторых атак.

11.4.3.1 Протокол Ву–Лама

Этот протокол разработан Ву (Woo) и Ламом (Lam).

Протокол 11.2. Протокол Ву–Лама

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Трент владеют общим симметричным ключом K_{AT}.
Боб и Трент владеют общим симметричным ключом K_{BT}.

ЦЕЛЬ: Алиса аутентифицирует себя Бобу, даже если Боб с ней не знаком.

1. Алиса → Бобу : *Алиса*.
2. Боб → Алисе : N_B.
3. Алиса → Бобу : $\{N_B\}_{K_{AT}}$.
4. Боб → Тренту : $\left\{\textit{Алиса}, \{N_B\}_{K_{AT}}\right\}_{K_{BT}}$.
5. Трент → Бобу : $\{N_B\}_{K_{BT}}$.
6. Боб расшифровывает зашифрованный текст с помощью ключа K_{BT} и принимает его, если случайное число восстановлено правильно, или не принимает в противном случае.

Выбирая для изучения протокол Ву–Лама, мы вовсе *не рекомендуем* его в качестве модели. Наоборот, этот протокол не только фатально уязвим, хотя у него и существует несколько исправленных вариантов, но и неудачно сконструирован. Таким образом, мы выбрали его в качестве отрицательного примера.

Цель этого протокола — Алиса должна доказать Бобу свою подлинность, даже если они не знакомы друг с другом.

Сначала, поскольку Алиса и Боб не знакомы друг с другом, Алиса может продемонстрировать свои криптографические возможности только Тренту: она шифрует одноразовое случайное число N_B, сгенерированное Бобом, с помощью своего долговременного ключа K_{AT}, разделенного с Трентом (этап 3). Трент, будучи доверенным посредником, честно выполняет протокол и расшифровывает зашифрованный текст, присланный Алисой на этапе 4. В заключение, когда Боб убеждается, что его свежее одноразовое случайное число, присланное Трентом, не изменилось, он приходит к выводу, что Трент мог выполнить криптографическую операцию только после того, как Алиса выполнила свою, причем обе эти операции касались числа N_B. Поэтому он аутентифицирует Алису, признавая ее существование.

С одной стороны, протокол Ву–Лама создан на основе стандартных протокольных конструкций, описанных в разделе 11.4.1.1. Например, этапы 2 и 3 похожи на стратегию (11.4.1). То же самое можно сказать об этапах 3 и 4.

Отложим подробный разбор протокола Ву–Лама до раздела 11.7. Кроме того, этот протокол имеет более крупный дефект, который и является причиной остальных недостатков. Однако для того, чтобы понять его сущность, необходимо овладеть формальными методами доказательства стойкости протоколов. По этой причине анализ дефекта проводится только в разделе 17.2.1.

11.5 Аутентификация с помощью пароля

Аутентификация с помощью пароля (password-based authentication) широко применяется при обмене сообщениями между пользователем и удаленным главным компьютером. В таких системах пользователь и главный компьютер устанавливают пароль, представляющий собой долговременный, но относительно короткий симметричный ключ.

Итак, пользователь U, желающий получить доступ к главному компьютеру H, должен сначала зарегистрироваться и установить пароль. Главный компьютер H хранит пароли всех своих пользователей. Каждая запись в его архиве представляет собой пару $(\mathrm{ID}_U, \mathrm{P}_U)$, где ID_U — имя пользователя U, а P_U — его пароль. Простейший протокол, основанный на применении пароля, позволяющего пользователю U получить доступ к компьютеру H, выглядит следующим образом.

1. $U \rightarrow H : \mathrm{ID}_U$.
2. $H \rightarrow U$: “Пароль”.
3. $U \rightarrow H : P_U$.
4. Компьютер H находит в своем архиве запись (ID_U, P_U).

   Пользователь U получает доступ к компьютеру H, если пароль P_U соответствует записи в архиве.

Следует заметить, что этот протокол не обеспечивает аутентификации сущности, даже односторонней, поскольку нигде в протоколе не используется идентификатор “свежести”, позволяющий подтвердить существование пользователя U. Несмотря на это, начиная с первой половины 1970-х годов термин “аутентификация с помощью пароля” используется для описания ситуации, когда пользователь пытается получить доступ к главному компьютеру с удаленного терминала по выделенной линии, не подвергающейся атаке. В таких системах описанный выше протокол обеспечивает одностороннюю аутентификацию пользователя U компьютером H.

Однако в открытых сетях этот протокол порождает серьезные проблемы, поскольку в нем не используются криптографические операции.

Первая проблема возникает из-за уязвимости файлов, записанных на главном компьютере H. Пароли, хранящиеся в архиве, могут стать известными Злоумышленнику (в данном случае Злоумышленником является инсайдер, которым

может быть даже системный администратор). Зная пароли, Злоумышленник обладает всеми правами, принадлежащими каждому из пользователей. Он может обращаться к главному компьютеру, имитируя любого пользователя, и наносить вред как отдельным пользователям, так и всей системе в целом. Очевидно, что, представляясь чужим именем, Злоумышленник не рискует быть узнанным.

Вторая проблема, связанная с протоколом получения удаленного доступа на основе паролей, заключается в том, что пароль перемещается от пользователя U к компьютеру H в открытом виде, и, следовательно, Злоумышленник может его перехватить. Такая атака называется **перехватом оперативного пароля** (online password eavesdropping).

11.5.1 Протокол Нидхема и его реализация в операционной системе UNIX

Нидхем разработал чрезвычайно простой и эффективный метод, обеспечивающий безопасность хранения паролей на главном компьютере [105, 132]. Главный компьютер H должен зашифровать пароли с помощью однонаправленной функции f и заменить записи (ID_U, P_U) зашифрованными текстами $(\mathrm{ID}_U, f(P_U))$.

Протокол 11.3. Протокол Нидхема

ИСХОДНЫЕ УСЛОВИЯ:

Пользователь U и главный компьютер H согласовывают запись $(\mathrm{ID}_U, f(P_U))$,
где f — однонаправленная функция.
Пользователь U запоминает пароль P_U.

ЦЕЛЬ: Получить доступ к компьютеру H.

1. $U \rightarrow H : \mathrm{ID}_U$.
2. $H \rightarrow U$: “Введите пароль”.
3. $U \rightarrow H : P_U$.
4. Главный компьютер H применяет к паролю P_U функцию f и находит запись $(\mathrm{ID}_U, f(P_U))$ в своем архиве.
   Если значение $f(P_U)$ совпадает со значением, записанным в архиве, пользователь U получает доступ к компьютеру H.

Протокол 11.3 реализован в операционной системе UNIX[6]. В протоколе аутентификации паролей системы UNIX однонаправленная функция f реализована с помощью алгоритма шифрования DES (раздел 7.6). Система, установленная

[6]UNIX — торговая марка компании Bell Laboratories.

на компьютере H, хранит в файле паролей универсальный идентификатор пользователя (universal identifier — UID) и зашифрованный текст, полученный в результате применения алгоритма DES к строке, содержащей 64 нуля. В качестве ключа шифрования используется пароль P_U. Для того чтобы предотвратить раскрытие паролей с помощью специальной аппаратуры, предназначенной для взлома шифра DES, однонаправленная функция f на самом деле немного отличается от этого алгоритма. Она повторяет 25 раундов шифрования по алгоритму DES в сочетании с побитовыми перестановками (bit-swapping permutation) в блоках зашифрованного текста. В конце каждого раунда некоторые биты полученного блока зашифрованного текста меняются местами в зависимости от случайного 12-битового числа, которое называется **"солью"** (salt) и хранится в файле паролей. Результирующий блок зашифрованного текста затем используется в качестве исходной информации для каждого следующего раунда алгоритма шифрования DES. Детали этой схемы описаны в работе [206].

Таким образом, преобразование $f(P_U)$, использующее алгоритм DES, можно считать ключевой параметризованной функцией хэширования постоянной строки 0^{64}, где ключом является пароль P_U, а параметром — "соль". Учитывая, что "соль" хранится в файле паролей, регистрационную запись в компьютере H можно представить в виде $(\mathrm{ID}_U, \textit{"соль"}, f(P_U, \textit{"соль"}))$, хотя для простоты записи мы будем по-прежнему использовать обозначение $f(P_U)$, а не $f(P_U, \textit{"соль"})$.

Итак, применение протокола Нидхема в операционной системе UNIX предотвращает атаку Злоумышленника. Во-первых, даже если Злоумышленник узнает число $f(P_U)$, он не сможет применить его в протоколе 11.3, поскольку в этом случае компьютер H вычислит значение $f(f(P_U))$ и не откроет доступ. Во-вторых, однонаправленную функцию f невозможно инвертировать, особенно если учесть, что в конце каждого раунда шифрования выполняется случайная перестановка битов. Следовательно, если пользователи правильно выбрали пароль, который нелегко угадать, Злоумышленнику очень трудно вычислить пароль P_U по значению $f(P_U)$. (Угадывание паролей обсуждается в разделе 11.5.3.)

Несмотря на то что задача защиты паролей решена, остается проблема, связанная с защитой целостности данных. Протокол по-прежнему уязвим для атаки на основе перехвата пароля в интерактивном режиме. Для блокирования такой атаки применяются одноразовые пароли.

11.5.2 Схема с одноразовыми паролями (и ее неудачная модификация)

Лампорт (Lamport) выдвинул простую идею, позволяющую предотвратить перехват паролей [174]. Ее можно рассматривать как схему с одноразовыми паролями. В данном случае термин "одноразовые" означает, что пароли, пересылаемые от пользователя U в компьютер H, никогда не повторяются, но связаны друг

с другом. Теперь пароль, перехваченный Злоумышленником в ходе протокола, бесполезен.

В момент инициализации протокола запись пароля пользователя U имеет вид $(\mathrm{ID}_U, f^n(P_U))$, где

$$f^n(P_U) \stackrel{\text{def}}{=} \underbrace{f(\ldots(f}_{n}(P_U))\ldots),$$

а n — достаточно большое число. Пользователь U, как и в случае применения протокола аутентификации паролей, по-прежнему должен помнить пароль P_U.

Когда пользователь U и компьютер H запускают протокол аутентификации в первый раз, при запросе пароля (этап 2 в протоколе аутентификации паролей), вычислительное устройство пользователя U, представляющее собой операционную систему или микропроцессор, предложит ему ввести пароль P_U, а затем вычислит значение $f^{n-1}(P_U)$. Эта задача эффективно решается даже для больших чисел n (например, для $n = 1\,000$). На третьем этапе протокола аутентификации паролей результат этих вычислений пересылается компьютеру H.

Получив величину $f^{n-1}(P_U)$, компьютер H один раз применит функцию f к полученному значению, получит число $f^n(P_U)$ и сравнит его с записью. Если проверка пройдена успешно, значит, полученное значение действительно было равно $f^{n-1}(P_U)$ и было вычислено по паролю P_U, установленному пользователем. Следовательно, на контакт вышел действительно пользователь U, который имеет доступ к системе. Кроме того, компьютер обновит запись пароля пользователя U, заменив значение $f^n(P_U)$ величиной $f^{n-1}(P_U)$.

При следующем запуске протокола вместо значения $f^{n-1}(P_U)$ пользователь U и компьютер H должны использовать число $f^{n-2}(P_U)$. Таким образом, этот протокол имеет историю и зависит от счетчика, значения которого изменяются от n до 1. Когда счетчик станет равным 1, пользователь U и компьютер H должны установить новый пароль.

В этом методе требуется, чтобы пользователь U и компьютер H синхронизировали пароль: если компьютер H применяет число $f^i(P_U)$, пользователь U должен переслать значение $f^{i-1}(P_U)$. Эта синхронизация может нарушиться, если связь ненадежна или система дала сбой. Учтите, что и помехи на линии связи, и зависание системы может быть результатом происков Злоумышленника!

Лампорт рассмотрел простой метод восстановления потерянной синхронизации [174]. По существу, он предложил "перевести стрелки вперед": синхронизация считается нарушенной, если компьютер H должен вычислить значение $f^j(P_U)$, а пользователь — число $f^k(P_U)$, где $j \neq k+1$. В этом случае система должна "перевести стрелки вперед": компьютер H должен вычислить значение $f^i(P_U)$, а пользователь — число $f^{i-1}(P_U)$, где $i \leqslant \min(j,k)$. Очевидно, что этот способ синхронизации требует взаимной аутентификации компьютера H и пользователя U, однако в короткой заметке Лампорта это требование указано не было.

Схема Лампорта, основанная на использовании пароля для удаленного доступа, была модифицирована и реализована в виде системы "одноразовых паролей" под названием **S/KEY**[7] [134]. Система S/KEY предназначена для решения проблем, связанных с ненадежной связью между компьютером H и пользователем U. Когда пользователь запускает протокол, в компьютере H хранится запись $(\mathrm{ID}_U, f^c(P_U), c)$, где c — счетчик, инициализированный значением n. Схема S/KEY описывается протоколом 11.4.

Протокол 11.4. Протокол S/KEY

ИСХОДНЫЕ УСЛОВИЯ:

Пользователь U и главный компьютер H согласовывают первоначальную запись $(\mathrm{ID}_U, f^n(P_U), n)$, где f — криптографическая хэш-функция.

Пользователь U запоминает пароль P_U.

Текущая запись пользователя U в компьютере H имеет вид $(\mathrm{ID}_U, f^c(P_U), c)$, где $1 \leqslant c \leqslant n$.

ЦЕЛЬ: Получить доступ к компьютеру H, не пересылая пароль P_U открытым текстом.

1. $U \rightarrow H : \mathrm{ID}_U$.
2. $H \rightarrow U : c$, "Введите пароль:".
3. $U \rightarrow H : Q = f^{c-1}(P_U)$.
4. Главный компьютер H находит запись $(\mathrm{ID}_U, f^c(P_U), c)$ в своем архиве.

   Если значение $f(Q) = f^c(P_U)$, пользователь U получает доступ к компьютеру H, а текущая запись заменяется тройкой $(\mathrm{ID}_U, Q, c-1)$.

Очевидно, что в протоколе 11.4 синхронизация не нужна, а значит, ненадежная связь между компьютером H и пользователем U больше не является проблемой.

К сожалению, схема S/KEY является небезопасной. Протоколы удаленного доступа на основе пароля в лучшем случае обеспечивают идентификацию пользователя U компьютером H. Следовательно, значение счетчика, якобы посланное компьютером H, может быть на самом деле послано или модифицировано Злоумышленником. Читатели могут сами попытаться описать действия Злоумышленника и сконструировать его атаку, прежде чем приступать к изучению раздела 11.7.2.

На эти замечания можно было бы возразить: "протокол S/KEY не может быть опасным больше, чем протокол аутентификации паролей Нидхема (протокол 11.3), в котором пароль передается открытым текстом!". Следует, однако, заметить, что

[7] S/KEY — торговая марка компании Bellcore.

протокол Нидхема никогда и не предназначался для противодействия перехвату, а протокол S/KEY создавался именно для этого.

11.5.3 Обмен зашифрованными ключами (EKE)

Большинство систем, использующих пароли, советуют пользователям выбирать пароли, состоящие из восьми символов ASCII. Такие пароли легко запомнить, не записывая. Поскольку каждый ASCII-символ представляет собой байт (8 бит), пароль, состоящий из восьми символов, можно записать в виде строки из 64 бит. Пространство таких строк содержит 2^{64} элементов и, следовательно, довольно велико. Итак, на первый взгляд, пароль, состоящий из восьми символов, нелегко угадать или определить методом последовательного перебора.

Однако это не совсем так. Несмотря на то что энтропия полного набора ASCII-символов не намного меньше 8 бит/символ (см. раздел 3.8), пользователи, как правило, не формируют свои пароли из случайных символов. Наоборот, они стремятся выбрать легко запоминаемые слова. Как правило, плохой пароль — это слово из словаря или имя человека, набранное в нижнем регистре, которое, возможно, сопровождается одной или двумя цифрами. По оценкам Шеннона, энтропия английского языка изменяется от 1,0 до 1,5 бит/символ [265]. При этом он оценивал только слова, не содержащие прописных букв. Итак, на самом деле размер пространства паролей, состоящих из восьми символов, намного меньше 2^{64}. Выбор плохих паролей (символы алфавита, набранные в нижнем регистре, имена людей и т.п.) еще больше уменьшают размер этого пространства и открывают возможность для **автономной атаки по словарю** (offline dictionary attack). В ходе такой атаки Злоумышленник использует значение $f(P_U)$ для поиска слов, совпадающих с паролем P_U. Поскольку атака является автономной, ее можно автоматизировать. Следует отметить, что схема Лампорта не предусматривает защиты против автономной атаки по словарю: Злоумышленник может перехватывать текущие значения i и $f^i(\mathrm{P}_U)$, а затем организовывать поиск.

Белловин (Bellovin) и Мерритт (Merritt) предложили привлекательный протокол, обеспечивающий безопасность паролей на основе аутентификации. Этот протокол называется **обменом зашифрованными ключами** (Encrypted Key Exchange — EKE) [29]. Он защищает пароли не только от перехвата, но и от автономной атаки по словарю. В схеме EKE использовано **вероятностное шифрование** (probabilistic encryption). В полном объеме вероятностное шифрование рассматривается в главе 14, а пока читатели могут считать, что оно сводится к добавлению в пароль своего собственного случайного числа ("соли").

В отличие от протоколов 11.3 и 11.4, в которых компьютер H владеет только образом пароля P_U, полученным в результате применения однонаправленной функции, в рамках протокола EKE пользователь U и компьютер H владеют общим паролем P_U. Этот пароль может использоваться в качестве симметричного

криптографического ключа, несмотря на то, что он принадлежит относительно небольшому пространству.

Оригинальность протокола EKE проявляется на первых двух этапах. На первом этапе зашифрованный текст $P_U(\mathcal{E}_U)$ является результатом шифрования одноразовой и случайной информации $\mathcal{E}_U$ с помощью общего пароля P_U. На втором этапе выполняется двойное шифрование другого одноразового и случайного числа $P_U(\mathcal{E}_U(K))$ — сеансового ключа K. Поскольку пароль P_U выбран человеком и поэтому является относительно небольшим, случайные строки $\mathcal{E}_U$ и K должны быть длиннее. Это позволяет скрыть пароль P_U на первых двух этапах и сделать его статистически независимым от зашифрованных текстов $\mathcal{E}_U$ и K.

Следует подчеркнуть, что роль "соли" играет одноразовое случайное число $\mathcal{E}_U$. Если бы "открытый" ключ не был одноразовым, уникальные функциональные возможности протокола EKE были бы полностью уничтожены: Злоумышленник мог бы определить пароль P_U, используя слабость "учебного" алгоритма шифрования с открытым ключом (например, с помощью атаки "встреча посередине", описанной в разделе 8.9).

Если одноразовые случайные числа N_U и N_H, зашифрованные на этапах 3, 4 и 5, имеют достаточно большой размер (т.е. больше, чем сеансовый ключ K), они скрывают сеансовый ключ точно так же, как пароль P_U на предыдущих этапах. Итак, пароль P_U остается статистически независимым от любых сообщений, передаваемых в рамках протокола EKE. Это означает, что пароль скрыт от Злоумышленника в смысле теоретико-информационной безопасности (раздел 7.5). Следовательно, пассивный перехватчик больше не в состоянии организовать автономную атаку на пароль P_U по словарю, используя протокольные сообщения. Остается либо угадывать пароль P_U непосредственно, либо организовывать активную атаку, модифицируя протокольные сообщения. Атака на основе угадывания неинтересна. Ее невозможно предотвратить, но, к счастью, она не эффективна. С другой стороны, активная атака с высокой вероятностью распознается всеми подлинными пользователями и немедленно пресекается.

Добавление "соли" в пароль P_U осуществляется пользователем U на первом этапе, когда он шифрует случайный открытый ключ, и компьютером H, когда он шифрует случайный сеансовый ключ на втором этапе. Любое изменение этих чисел ставит атакующего в тупик. Следовательно, именно эти два этапа определяют техническую новизну протокола EKE. Этапы 3, 4 и 5 представляют собой обычные протоколы взаимной аутентификации на основе стратегии "оклик-отзыв".

Протокол EKE удобно сочетать с механизмом обмена ключами Диффи–Хеллмана. Пусть число α порождает группу, порядок которой больше, чем $2^{64} > 2^{|P_U|}$. Тогда на первом этапе вычислительное устройство пользователя U должно сгенерировать случайное число $x \in (0, 2^{64})$ и найти число $\mathcal{E}_U = \alpha^x$, а на втором этапе *компьютер H должен сгенерировать случайное число $y \in (0, 2^{64})$ и вычислить значение* $\mathcal{E}_U(K) = \alpha^y$. Согласованным сеансовым ключом между пользователем U

Протокол 11.5. Обмен зашифрованными ключами (EKE)

ИСХОДНЫЕ УСЛОВИЯ:

Пользователь U и главный компьютер H устанавливают пароль P_U.
Система согласована с алгоритмом симметричного шифрования.
Символ $K()$ обозначает симметричное шифрование с помощью ключа K.
Пользователь U и компьютер H применяют схему асимметричного шифрования.
Символ $\mathcal{E}_U$ обозначает асимметричное шифрование с помощью ключа, принадлежащего пользователю U.

ЦЕЛЬ: Осуществить взаимную аутентификацию и согласовать общий закрытый ключ.

1. Пользователь U генерирует случайный "открытый" ключ $\mathcal{E}_U$ и посылает его компьютеру H.

$$U, P_U(\mathcal{E}_U).$$

(* Ключ $\mathcal{E}_U$ на самом деле не является открытым. Он представляет собой ключ шифрования в алгоритме асимметричного шифрования. *)

2. Компьютер H расшифровывает порцию зашифрованного текста, используя ключ P_U, и извлекает из него ключ $\mathcal{E}_U$. Затем он генерирует случайный симметричный ключ K и посылает его пользователю H.

$$P_U(\mathcal{E}_U(K)).$$

3. Пользователь U расшифровывает дважды зашифрованный текст и получает ключ K. Затем он генерирует случайное число N_U и посылает его компьютеру H.

$$K(N_U).$$

4. Компьютер H расшифровывает текст, используя ключ K, генерирует случайное число N_H и посылает его пользователю U.

$$K(N_U, N_H).$$

5. Пользователь U расшифровывает текст с помощью ключа K и возвращает его компьютеру H.

$$K(N_H).$$

6. Если процедура "оклик-отзыв" на этапах 3, 4 и 5 завершилась успешно, осуществляется регистрация, и стороны приступают к дальнейшему секретному обмену информацией, используя общий ключ K.

и компьютером H является число $K = \alpha^{xy}$. Таким образом, каждая из сторон вносит свой вклад в формирование общего сеансового ключа. В рамках описываемой реализации порождающий элемент группы α может согласовываться между пользователем U и компьютером H в виде открытого текста: пользователь U может послать компьютеру H описание группы (в которое входит порождающий элемент α) еще до начала переговоров.

Обратите внимание на то, что к данной реализации выдвигается только одно требование: чтобы порядок группы превышал 2^{64}. Это очень маленькая оценка снизу, позволяющая применять группу в асимметричной криптографической системе. Итак, протокол может оказаться весьма эффективным. Маленький порядок группы позволяет легко вычислить дискретный логарифм и, следовательно, решить вычислительную проблему Диффи–Хеллмана. Однако, не зная чисел α^x, α^y и α^{xy}, несмотря на легкость решения задачи Диффи–Хеллмана, определить пароль невóзможно: он остается статистически независимым элементом пространства, состоящего из 2^{64} чисел. Аналогично при достаточно больших случайных числах, зашифрованных на этапах 3, 4 и 5, сеансовый ключ K остается независимым элементом группы, порядок которой превышает 2^{64}. Следовательно, автономная атака по словарю или простое угадывание сеансового ключа остаются трудной задачей.

По существу, случайное число, добавляемое в пароль, увеличивает размер пространства паролей: из множества слов, перечисленных в словаре, оно превращается в пространство случайных асимметричных ключей. Именно в этом заключается изюминка протокола EKE.

11.6 Обмен аутентичными ключами с помощью асимметричной криптографии

Говорят, что протокол устанавливает распределенный сеансовый ключ с помощью механизма **передачи ключа** (key transport), если результатом протокола является распределенный ключ, поступающий от одного из участников протокола. Говорят, что протокол устанавливает распределенный сеансовый ключ с помощью механизма **обмена ключами** (key exchange), или **согласования ключей** (key agreement), если результатом протокола является распределенный ключ, зависящий от случайных чисел, поступающих от всех участников протокола. Преимущество обмена ключами перед передачей ключа заключается в том, что каждая из сторон контролирует процесс генерации ключа, что гарантирует его высокое качество.

Не считая реализации протокола EKE на основе механизма обмена ключами Диффи–Хеллмана, основные конструкции аутентификации, рассмотренные до сих

пор, являлись механизмами передачи ключа. Рассмотрим теперь механизм обмена ключами.

Обмен ключами достигается путем генерации ключей, которые являются результатами применения псевдослучайной функции или однонаправленной функции хэширования, причем каждая из сторон вносит в его формирование свой вклад. Наиболее часто для этого используется великое открытие Диффи и Хеллмана: протокол обмена ключами Диффи–Хеллмана, который представляет собой однонаправленную функцию (см. замечание 8.1.3 в разделе 8.4), — протокол 8.1. Этот механизм обеспечивает согласование ключей между двумя удаленными пользователями без шифрования.

Протокол 8.1 представляет собой основную версию протокола обмена ключами Диффи–Хеллмана и обеспечивает неаутентифицированное согласование ключей. В главе 8 описана **атака "человек посередине"** (атака 8.1), в ходе которой Злоумышленник делит один ключ между собой и Алисой, а другой — между собой и Бобом. Это позволяет ему передавать "секретные" сообщения от Алисы к Бобу и обратно. Простейшей вариацией протокола обмена ключами Диффи–Хеллмана является протокол с двумя участниками, в рамках которого Алиса точно знает, что число g^b является открытым ключом Боба.

$$1. \quad \text{Алиса} \rightarrow \text{Бобу} : \textit{Алиса}, g^a, \tag{11.6.1}$$

где число a случайным образом извлекается Алисой из достаточно большого целочисленного интервала.

Переслав сообщение, Алиса знает, что число g^{ab} — это ключ, который делят только она и Боб, поскольку любой посторонний пользователь для того, чтобы найти число g^{ab}, должен решить вычислительную проблему Диффи–Хеллмана (см. определение 8.1 в разделе 8.4), которая считается неразрешимой. Поскольку Алиса только что извлекла свой показатель степени случайным образом, согласованный ключ является новым, и Алиса считает его аутентичным. Однако, получив число g^a, Боб не знает, с кем он делит ключ g^{ab} и является ли этот ключ "свежим". Следовательно, описанная выше версия протокола обеспечивает лишь одностороннее согласование аутентичных ключей.

Применяя основные конструкции, описанные ранее, нетрудно усилить механизм (11.6.1) и обеспечить взаимную аутентификацию. Например, Алиса может сопроводить число g^a своим именем или меткой времени.

Рассмотрим широко известный протокол обмена аутентичными ключами, представляющий собой вариант протокола обмена ключами Диффи–Хеллмана.

11.6.1 Протокол "станция-станция"

Протокол **"станция-станция"** (Station-to-Station — STS) был предложен Диффи и его соавторами [99].

В рамках этого протокола Алиса и Боб используют большую конечную абелеву группу, порожденную общим элементом α, который могут использовать все системные пользователи. Ограничения, налагаемые на протокол STS, перечислены в разделе 8.4.1.

Алиса и Боб владеют следующими сертификатами открытых ключей.

$$\text{Cert}_A = \text{sig}_{CA}(\textit{Алиса}, P_A, \text{desc}\langle\alpha\rangle),$$
$$\text{Cert}_B = \text{sig}_{CA}(\textit{Боб}, P_B, \text{desc}\langle\alpha\rangle),$$

где CA — обозначение службы сертификации (certification authority) (см. главу 13), P_A и P_B — открытые ключи Алисы и Боба соответственно, а $\text{desc}\langle\alpha\rangle$ — описание общей группы, порожденной элементом α. Кроме того, обе стороны договариваются использовать алгоритм шифрования симметричным ключом (определение 7.1 из раздела 7.2). Алгоритм шифрования также согласовывается между всеми системными пользователями.

Протокол 11.6. Протокол "станция-станция" (STS)

ИСХОДНЫЕ УСЛОВИЯ:

Алиса владеет сертификатом открытого ключа Cert_A.
Боб владеет сертификатом открытого ключа Cert_B.
Все пользователи системы применяют общую большую конечную абелеву группу $\text{desc}\langle\alpha\rangle$ и симметричный алгоритм шифрования $\mathcal{E}$.

ЦЕЛЬ: Осуществить взаимную аутентификацию и согласовать взаимно аутентифицированный ключ.

1. Алиса генерирует большое случайное число x и посылает его Бобу.

$$\alpha^x.$$

2. Боб генерирует большое случайное число y и посылает его Алисе.

$$\alpha^y, \text{Cert}_B, \mathcal{E}_K(\text{sig}_B(\alpha^y, \alpha^x)).$$

3. Алиса посылает сообщение Бобу.

$$\text{Cert}_A, \mathcal{E}_K(\text{sig}_B(\alpha^x, \alpha^y)).$$

Здесь $K = \alpha^{xy} = \alpha^{yx}$.

ПРЕДУПРЕЖДЕНИЕ:

Этот протокол имеет недостатки, которые рассматриваются в разделе 11.6.3.

Протокол STS должен иметь следующие свойства (некоторые из них достигаются только после исправления недостатков протокола).

Взаимная аутентификация сущностей (mutual entity authentication). На самом деле, если аутентификация понимается в строгом смысле, сформулированном авторами протокола, это свойство не выполняется. В работе [99] Диффи и соавторы сделали две ошибки. Мы обсудим их в разделах 11.6.2 и 11.6.3 соответственно.

Взаимное согласование аутентичного ключа (mutually authenticated key agreement). Согласование ключа очевидным образом достигается с помощью протокола обмена ключами Диффи–Хеллмана. "Свежесть" согласованного ключа обеспечивается, если каждая из сторон правильно генерирует случайные числа. Эксклюзивное распределение ключа гарантируется цифровыми подписями сторон на собственных параметрах ключа. Однако все эти свойства, вместе взятые, не обеспечивают взаимного согласования аутентичного ключа. Оно достигается только после исправления недостатков протокола.

Взаимное утверждение ключа (mutual key confirmation). После завершения протокола каждая из сторон уверена, что партнер использует для шифрования параметров согласованный ключ. Это свойство также зависит от правильной аутентификации, которая достигается только после исправления недостатков протокола.

Полная заблаговременная секретность (perfect forward secrecy). Это привлекательное свойство протокола генерации ключа означает, что компрометация долговременного закрытого ключа не влияет на секретность предыдущих сеансов [99, 133]. Протокол обмена ключами Диффи–Хеллмана гарантирует полную заблаговременную секретность. В протоколе STS долговременными ключами являются закрытые ключи Алисы и Бобы. Поскольку каждый сеансовый ключ, согласованный в ходе протокола, представляет собой однонаправленную функцию, зависящую от двух эфемерных секретов, полностью уничтожаемых после завершения протокола, компрометация любого из подписанных долговременных ключей не влияет на секретность ранее согласованных сеансовых ключей.

Анонимность (anonymity). Если зашифрованный текст содержит зашифрованный сертификат открытого ключа, сообщения, передаваемые в рамках протокола, скрыты от постороннего пользователя, вовлеченного в обмен информацией. Однако необходимо отметить, что информация, пересылаемая в рамках низкоуровневого протокола связи, может раскрыть имена участников протокола. Следовательно, строго говоря, анонимность в данном случае означает возможность отрицать авторство сообщения (deniability). Это означает, что монитор сети не может доказать, что данный протокол выполняется

двумя конкретными пользователями. Поскольку протокол STS лежит в основе обмена ключами через Internet (Internet Key Exchange — IKE), он может использоваться для обеспечения безопасности в Internet [135, 158, 225], гарантируя анонимность пользователей. Свойства протокола IKE рассматриваются в следующей главе.

Хотя протокол STS не лишен недостатков, он представляет собой важное достижение в области аутентификации и обмена аутентичными ключами. На его основе был создан протокол обмена ключами через Internet ("Internet Key Exchange (IKE) Protocol) [135, 225], ставший промышленным стандартом. Протокол IKE описан в разделе 12.2.

В статье [99] содержались две ошибки: одна серьезная, а другая — нет. Серьезной ошибкой являлся упрощенный вариант протокола STS, предназначенный "только для аутентификации", а менее серьезная ошибка заключалась в самом протоколе STS. Если следовать общепризнанным принципам разработки протоколов (принятым после публикации работы [99]), обе ошибки можно устранить. Рассмотрим их поближе.

11.6.2 Дефект упрощенного протокола STS

Для того чтобы доказать свойство взаимной аутентификации, Диффи и его соавторы упростили протокол STS и назвали упрощенную версию "Протоколом STS, предназначенным только для аутентификации" ("Authentication-only STS Protocol"). Они утверждали, что "упрощенная версия, по существу, совпадает с трехпроходным протоколом аутентификации", предложенным институтом ISO. При этом они ссылались на "Трехпроходный протокол взаимной аутентификации с открытым ключом ISO" ("ISO Public Key Three-Pass Mutual Authentication Protocol"), т.е. на протокол 11.1, устойчивый к атаке Винера (см. раздел 11.4.2).

Протокол 11.7. Протокол STS, предназначенный только для аутентификации

ИСХОДНЫЕ УСЛОВИЯ:

Алиса владеет сертификатом открытого ключа Cert_A.
Боб владеет сертификатом открытого ключа Cert_B.

Алиса Боб

$$\xrightarrow{R_A}$$

$$\xleftarrow{\text{Cert}_A, R_B, \text{sig}_B(R_B, R_A)}$$

$$\xrightarrow{\text{Cert}_A, \text{sig}_A(R_A, R_B)}$$

Однако протокол 11.7 существенно отличается от протокола ISO. В упрощенном протоколе STS подписанные сообщения не содержат имен участников протокола, а в протоколе ISO — содержат. Таким образом, упрощенный протокол STS уязвим для атаки с помощью замены сертификата и подписи (certificate–signature–replacement attack).

Атака 11.2. Атака на протокол STS, предназначенный только для аутентификации

ИСХОДНЫЕ УСЛОВИЯ:

В дополнение к условиям протокола 11.7,
Злоумышленник владеет сертификатом Cert_M.
(* т.е. является обычным пользователем системы *)
(* Злоумышленник представляется Алисе своим подлинными именем, а Бобу представляется как Алиса. *)

Алиса		Злоумышленник → Я “Алиса” →		Боб
	$R_A \longrightarrow$		$R_A \longrightarrow$	
	$\text{Cert}_M, R_B, \text{sig}_M(R_B, R_A) \longleftarrow$		$\text{Cert}_B, R_B, \text{sig}_B(R_B, R_A) \longleftarrow$	
	$\text{Cert}_A, \text{sig}_A(R_A, R_B) \longrightarrow$		$\text{Cert}_A, \text{sig}_A(R_A, R_B) \longrightarrow$	

ПОСЛЕДСТВИЯ:

Боб думает, что разговаривает с Алисой, а она думает, что разговаривает со Злоумышленником.

В ходе этой атаки Злоумышленник, являющийся законным пользователем системы и владеющий сертификатом открытого ключа, просит Алису инициализировать протокол. После этого он вступает в разговор с Бобом, представляясь ему “Алисой” и используя ее одноразовое случайное число. Получив ответ от Боба, Злоумышленник заменяет его сертификат и подпись своими копиями. Это побуждает Алису подписать случайное число, сгенерированное Бобом, что, в свою очередь, позволяет Злоумышленнику обмануть Боба. Эта атака безупречна, поскольку ни Алиса, ни Боб ничего не подозревают.

Следует заметить, что Злоумышленник играет в этой атаке весьма активную роль: он подписывает случайное число, сгенерированное Бобом, и убеждает Алису также подписать его, чтобы полностью ввести Боба в заблуждение. Если бы Злоумышленник был пассивным, т.е. играл роль обычного проводника, то Алиса никогда не подписала бы случайное число Боба и не позволила бы себя обмануть.

Протокол STS неуязвим для атаки с помощью замены сертификата и подписи, поскольку шифрование, использованное в его полной версии, не позволяет Злоумышленнику заменить подпись Боба. Протокол ISO также защищен от нее, поскольку имя Злоумышленника должно появиться в сообщении, подписанном Алисой, а значит, Боба обмануть невозможно.

Интересно, что в своей статье [99] Диффи с соавтором обсуждали подобную атаку на упрощенный протокол STS, из которого было исключено шифрование. Однако они не применили атаку с помощью замены сертификата и подписи против протокола STS, предназначенного только для аутентификации. Возможно, это объяснялось тем, что упрощенный протокол STS очень напоминает исправленную версию протокола ISO. Хотя в статье [99] была проанализирована атака Винера на дефектный вариант протокола ISO, она сильно отличается от атаки с помощью замены сертификата и подписи. (Читатели могут самостоятельно убедиться в том, что атака Винера на дефектный вариант протокола ISO не применима к протоколу STS, предназначенному только для аутентификации.) Вся эта запутанная ситуация лишний раз подтверждает уязвимость протоколов аутентификации.

Включение в цифровую подпись имени верифицируемого пользователя позволяет исправить указанный выше недостаток. Разумеется, это не единственный способ исправить протокол. В некоторых приложениях (например, в протоколе IKE, рассмотренном в разделе 12.2.3) имена участников протокола преднамеренно опускаются, чтобы гарантировать их анонимность (см. раздел 12.2.4). Более оригинальные способы устранения дефектов с сохранением анонимности пользователей основаны на применении современных криптографических средств, которые будут описаны в одной из последующих глав.

11.6.3 Незначительный недостаток протокола STS

Лоу (Lowe) изобрел не очень опасную атаку на протокол STS [179]. Прежде чем описать ее, приведем строгое определение аутентификации, сформулированное авторами протокола STS.

В работе [99] Диффи с соавторами описано безопасное выполнение протокола аутентификации с помощью “соответствия протокольных записей” (“matching records of runs”). Допустим, что каждый участник протокола записывает все сообщения, получаемые в ходе его выполнения. “Соответствие протокольных записей” означает, что записи, созданные одним из участников, должны содержаться среди записей другого участника в том же порядке, в котором они пересылались, и наоборот. Тогда некорректное выполнение протокола аутентификации [99] определяется следующим образом.

> Протокол выполняется некорректно, если одна из сторон, участвующих в протоколе, например, Алиса, выполняет протокол правильно и признает подлинность другого участника, записи которого не соот-

ветствуют записям Алисы ни в рамках полного, ни в рамках неполного протокола.

При таком определении некорректного протокола аутентификации, становится возможной следующая атака на протокол STS, хотя ее последствия очень незначительны.

Атака 11.3. Атака Лоу на протокол STS (незначительный недостаток)

(* Злоумышленник представляется Бобу своим подлинным именем, а Алисе представляется как "Боб". *)

Алиса ← Я "Боб" ← Злоумышленник Боб

α^x α^x

$\alpha^x, \text{Cert}_B, \mathcal{E}_K(\text{sig}_B(\alpha^y, \alpha^x))$ $\alpha^x, \text{Cert}_B, \mathcal{E}_K(\text{sig}_B(\alpha^y, \alpha^x))$

$\text{Cert}_A, E_K(\text{sig}_A(\alpha^x, \alpha^y))$

ПОСЛЕДСТВИЯ:

Алиса полностью обманута и полагает, что разговаривает и делит общий сеансовый ключ с Бобом, а Боб думает, что разговаривает со Злоумышленником в рамках неполного протокола. Алиса никогда не догадается о происшедшем, и все ее дальнейшие попытки организовать секретную связь с Бобом будут пресекаться.

Атака Лоу не приводит к серьезным последствиям по двум причинам.

1. Хотя Злоумышленнику и удается обмануть Алису в ходе выполнения протокола, сеансовый ключ остается недоступным. Следовательно, после прекращения протокола Злоумышленник не сможет больше вводить Алису в заблуждение.
2. Злоумышленник не может завершить протокол обмена информацией между ним и Бобом. Следовательно, эту часть атаки следует признать безуспешной.

Однако атака Лоу является вполне возможной по двум причинам.

I. Алиса признает подлинность Боба, хотя Злоумышленник просто бит за битом копирует все сообщения Боба, отправленные Алисе. Однако Боб видит, что контактирует со Злоумышленником, когда тот подписывает сообщения Алисы, содержащие случайный оклик, поскольку записи Боба не соответствуют записям Алисы. Следовательно, атака соответствует определению "некорректного выполнения" протокола, сформулированному авторами протокола STS. Таким образом, взаимная аутентификация не осуществляется.

Разумеется, поскольку понятием аутентификации сущности довольно трудно овладеть, и в этой области сделано немало ошибок, оценка атаки на основе устаревшего определения Диффи [99] является не слишком убедительной. Может возникнуть вопрос: "А правильно ли это определение вообще?". Однако лучше ограничиться вопросом: "Насколько практичной является атака Лоу в настоящее время?". Ответ дан в пункте II.

II. Злоумышленник успешно обманывает Алису, которая считает, что протокол выполняется правильно. Ее последующие попытки установить секретный контакт с Бобом будут пресекаться без объяснений, поскольку Боб считает, что никогда не вступал в контакт с Алисой. Кроме того, никто не может сообщить Алисе о неправильном выполнении протокола. Последствия атаки Лоу можно сравнить с последствиями намного менее интересной атаки, в ходе которой Злоумышленник является пассивным до тех пор, пока Алиса не отправляет Бобу последнее сообщение. Злоумышленник блокирует это сообщение, и Боб не может его получить. Анализируя свои записи, Боб обнаруживает, что в них не хватает последнего сообщения, и уведомляет об этом Алису. Независимо от того, насколько практична атака Лоу в настоящее время, если в роли Алисы выступает централизованный сервер, который подвергается массированной атаке со стороны команды злоумышленников, отсутствие уведомления от конечных пользователей представляет собой проблему: сервер зарезервирует ресурсы для многих конечных пользователей и его мощность резко упадет. Следует подчеркнуть, что в ходе атаки Лоу злоумышленник и его союзники не используют никаких криптографических мандатов (сертификатов). Следовательно, эта атака требует очень небольших затрат. Этим она отличается от обычной атаки на основе отказа в обслуживании (denial-of-service attack), в ходе которой Злоумышленник и его союзник сообщают Алисе свои подлинные имена (т.е. сертификаты).

По причинам, перечисленным в пункте II, атака Лоу называется **совершенной атакой на основе отказа в обслуживании** (perfect denial of service attack), направленной против Алисы: атакующий достигает успеха, используя чужие криптографические мандаты.

Эту атаку можно предотвратить, если модифицировать протокол в соответствии с принципом, предложенным Абади и Нидхемом [1].

> Если смысл сообщения существенно зависит от имени пользователя, это имя следует передавать открыто.

Действительно, подписанные сообщения в протоколе STS должны содержать имена обоих участников! Таким образом, сообщение, подписанное и отправленное Бобом, будет содержать имя "Злоумышленник", поэтому Злоумышленник не сможет переслать его Алисе от имени Боба. Следовательно, Злоумышленник больше не сможет обманывать Алису. Более того, если в упрощенный протокол STS,

предназначенный только для аутентификации, включить имена пользователей, он станет неуязвимым для атаки с помощью замены сертификата и подписи. Вот теперь упрощенный вариант протокола STS, по существу, совпадает с протоколом ISO (протокол 11.1).

Как указывалось ранее, протокол STS, наряду с другими, лежит в основе протокола обмена ключами через Internet (IKE) [135, 158, 225]. Как будет показано в разделе 12.2, "совершенная атака на основе отказа в обслуживании" может успешно взламывать некоторые разновидности протокола IKE.

В заключение следует повторить то, что было сказано в разделе 11.6.2: добавление в цифровую подпись имени верификатора — не единственный способ предотвратить эту атаку. Например, лучших результатов можно добиться, используя **заранее обусловленную подпись верификатора** (designated verifier signature). Этот способ рассматривается в одной из следующих глав.

11.7 Типичные атаки на протоколы аутентификации

В разделе 2.3 мы предположили, что Злоумышленник (возможно, в сотрудничестве со своими союзниками, распределенными по открытой сети) может подслушивать, перехватывать, изменять и рассылать сообщения по открытой сети, маскируясь под других пользователей. С точки зрения уровня приложения способности Злоумышленника по организации атак выглядят сказочными: неужели он так всемогущ?

Однако с точки зрения сетевого уровня для взлома протокола Злоумышленнику вовсе не обязательно применять очень хитроумные методы. Приемы организации атак на низкоуровневые протоколы связи описаны в разделе 12.2. Пока будем считать, что Злоумышленник действительно всемогущ и может организовывать разнообразные атаки на дефектные протоколы.

Несмотря на то что перечислить все приемы, которые Злоумышленник может применять для организации своих атак, в принципе невозможно (ведь он постоянно изобретает что-то новое!), знание основных способов атаки позволяет разрабатывать более устойчивые протоколы. В данном разделе рассматривается несколько хорошо известных методов атаки. Следует отметить, что Злоумышленник может комбинировать разные приемы, конструируя новые, более хитроумные атаки.

Примечание 11.3. *Успешная атака на протокол аутентификации или протокол генерации аутентифицированного ключа обычно не связана со взломом криптографического алгоритма. Наоборот, как правило, атака направлена на несанкционированное и незаметное овладение криптографическими мандатами или уни-*

чтожение криптографического сервиса без *взлома криптографического алгоритма. Разумеется, атака становится возможной вследствие ошибок, сделанных при разработке протокола, а не криптографического алгоритма.* □

11.7.1 Атака с повторной передачей сообщений

В атаке с повторной передачей сообщения (message replay arrack) Злоумышленник заранее записывает старое сообщение, перехваченное в ходе предыдущего сеанса протокола, и повторяет его в рамках нового сеанса. Поскольку цель протокола аутентификации — установить подлинность партнера с помощью обмена свежими сообщениями, повторение старого сообщения не соответствует намерениям участников.

В разделе 2.6.4.2 рассмотрен пример атаки с повторной передачей сообщения — атака на протокол аутентификации с симметричным ключом Нидхема–Шредера (атака 2.2). Мы отметили только одну опасность, связанную с этой атакой: повторное сообщение шифруется старым сеансовым ключом, который считается уязвимым (Злоумышленник мог раскрыть его значение либо в результате небрежности пользователей, либо по другим причинам, указанным в разделе 2.5).

Однако эта атака порождает более серьезную опасность — отсутствие реальной связи между партнерами. Действительно, для успеха атаки 2.2 Злоумышленник не обязан ждать удобного случая, когда Алиса запустит протокол аутентификации Боба. Он может просто начать атаку, сразу перейдя на этап 3, и повторить записанные сообщения, поскольку ему известен старый сеансовый ключ K'.

3. Злоумышленник(“Алиса”) → Бобу : $\{K', Алиса\}_{K_{BT}}$.
4. Боб → Злоумышленнику(“Алисе”) : $\{Я - Боб! N_B\}'_K$.
5. Злоумышленник(“Алиса”) → Бобу : $\{Я - Алиса! N_B - 1\}'_K$.

Теперь Боб думает, что установил контакт с Алисой, в то время как Алиса вообще не выходила на связь.

Повтор сообщения — классическая атака на протоколы аутентификации пользователей и протоколы генерации аутентифицированных ключей. На первый взгляд, мы уже хорошо знаем ее особенности, поскольку повсеместно применяем в основных конструкциях идентификаторы “свежести” (одноразовые случайные числа, метки времени). Однако знание не защищает протоколы от этой атаки. Одна из тонкостей протоколов аутентификации заключается в том, что разработчики часто делают ошибки, даже если они хорошо известны. Рассмотрим одну из вариаций атаки с повторной передачей сообщения.

В работе [293] Варадхараджан (Varadharajan) и его соавторы описали большое количество “протоколов по доверенности” (proxy protocols), в которых пользователь, которому доверяют остальные, выдавал рекомендацию другому пользователю. В одном из протоколов Боб, клиент, разделяет ключ K_{BT} с Трентом, сервером

аутентификации. Боб генерирует метку времени T_B и запрашивает ключ K_{BS} для связи с другим сервером, S. Затем сервер S создает сообщение $\{T_B + 1\}_{K_{BS}}$ и посылает его Бобу.

5. $S \rightarrow$ Боб : $S, B, \{T_B + 1\}_{K_{BS}}, \{K_{BS}\}_{K_{BT}}$.

Вот как авторы объясняют причину этого действия.

> Получив ключ K_{BS}, Боб может с помощью метки времени T_B убедиться в том, что сервер S создал "свежее" сообщение, а значит, сеансовый ключ является действительно "свежим".

Однако, несмотря на идентификатор "свежести", зашифрованный вместе с ключом K_{BS}, Боб может не получить гарантий, что ключ K_{BS} действительно является "свежим". Он может лишь убедиться, что ключ K_{BS} был использован совсем недавно, но мог уже устареть или даже быть скомпрометированным.

Примечание 11.4. *Иногда совместное шифрование идентификатора "свежести" и сообщения может свидетельствовать только о том, что они были зашифрованы недавно, но не о "свежести" сообщения, содержащегося в зашифрованном тексте.* □

11.7.2 Атака "человек посередине"

Атака "человек посередине" (man-in-the-middle attack) хорошо описывается известной проблемой гроссмейстера[8]. В основном, она предназначена для взлома протоколов, не предусматривающих взаимной аутентификации. В ходе этой атаки Злоумышленник может переадресовывать трудные вопросы, задаваемые одним из участников протокола, другому участнику, получать от него ответ, а затем пересылать спрашивающему, и наоборот.

В разделах 2.6.6.3 и 8.3.1 были рассмотрены два варианта атаки "человек посередине": одна — на протокол аутентификации с открытым ключом Нидхема–Шредера, а другая — на протокол обмена ключами Диффи–Хеллмана.

Атака "человек посередине" на протокол S/KEY (атака 11.4 на протокол 11.4) демонстрирует, как Злоумышленник может получить криптографический мандат, не взламывая криптографический алгоритм, лежащий в основе атакованной схемы.

Криптографическая функция хэширования f, использованная в схеме S/KEY, может оказаться настолько стойкой, что инвертировать ее будет практически невозможно. Кроме того, пользователь U может правильно выбрать пароль P_U

[8] Новичок начинает шахматные партии с двумя гроссмейстерами: одну из партий он играет белыми, а другую — черными. Дождавшись хода противника на первой доске он повторяет его на другой доске, ждет ответного хода и воспроизводит его на первой доске. Таким образом, два гроссмейстера практически играют друг с другом. В результате он либо заключит две ничьи, либо одну партию проиграет, а другую выиграет. В любом случае его шахматный рейтинг вырастет.

и затруднить атаку по словарю. Однако этот протокол катастрофически уязвим для активной атаки.

Атака 11.4. Атака на протокол STS

(* Здесь приняты те же обозначения, что и в протоколе 11.4. *)

U	Злоумышленник	H
	$\mathrm{ID}_U \longrightarrow$	$\mathrm{ID}_U \longrightarrow$
	$\leftarrow c-1$, "Пароль"	$\leftarrow c$, "Пароль"
	$f^{c-2}(P_U) \longrightarrow$	$f^{c-1}(P_U) \longrightarrow$

ПОСЛЕДСТВИЯ:

Злоумышленник овладевает значением $f^{c-2}(P_U)$, которое он может использовать для регистрации под именем пользователя U в ходе следующего сеанса.

Атака 11.4 возможна, поскольку в протоколе S/KEY сообщения, поступающие от компьютера H, не подвергаются аутентификации со стороны пользователя U.

Для того чтобы предотвратить атаку "человек посередине", необходимо предусмотреть аутентификацию источника данных в обоих направлениях.

11.7.3 Атака с помощью параллельного сеанса

В атаке с помощью параллельного сеанса под управлением Злоумышленника выполняется сразу несколько протоколов. Параллельные сеансы позволяют Злоумышленнику использовать для ответа на трудные вопросы, возникающие в ходе одного сеанса, информацию, которую он получает в ходе других сеансов.

Примером атаки с помощью параллельных сеансов является атака на протокол Ву–Лама (протокол 11.2), изобретенная Абади и Нидхемом [1].

Эта атака возможна, если Боб устанавливает контакты с Алисой и Бобом практически одновременно. Тогда Злоумышленник может блокировать сообщения, направленные Алисе. В сообщениях 1 и 1' Боба просят ответить на два запроса: один — от Злоумышленника, а другой — от "Алисы". В сообщениях 2 и 2' Боб отвечает на два случайных оклика, посылая оба ответа Злоумышленнику (один из ответов, число N_B, Злоумышленник просто перехватывает). Злоумышленник игнорирует одноразовое случайное число N'_B, предназначенное для него, а вместо него использует перехваченное число N_B, предназначавшееся Алисе. Итак, в сообщениях 3 и 3' Боб получает зашифрованный текст $\{N_B\}_{K_{MT}}$. Обратите

Атака 11.5. Атака на протокол Ву–Лама с помощью параллельного сеанса

ИСХОДНЫЕ УСЛОВИЯ:

В дополнение к условиям протокола 11.2,
Злоумышленник и Трент владеют общим долговременным ключом K_{MT}.
(* Иначе говоря, Злоумышленник является обычным пользователем системы. *)

1. Злоумышленник(“Алиса”) → Бобу : *Алиса*.

1’. Злоумышленник → Бобу : *Злоумышленник*.

2. Боб → Злоумышленнику(“Алисе”) : N_B.

2’. Боб → Злоумышленнику : N'_B.

3. Злоумышленник(“Алиса”) → Бобу : $\{N_B\}_{K_{MT}}$.

3’. Злоумышленник → Бобу : $\{N_B\}_{K_{MT}}$.

4. Боб → Тренту : $\left\{\textit{Алиса}, \{N_B\}_{K_{MT}}\right\}_{K_{BT}}$.

4’. Боб → Тренту : $\left\{\textit{Злоумышленник}, \{N_B\}_{K_{MT}}\right\}_{K_{BT}}$.

5. Трент → Бобу : $\{\text{“мусор”}\}_{K_{BT}}$.
(* “Мусор” возникает после того, как Трент расшифровывает сообщение $\{N_B\}_{K_{MT}}$ с помощью ключа K_{AT}. *)

5’. Трент → Бобу : $\{N_B\}_{K_{BT}}$.

6. Боб прекращает связь со Злоумышленником
(* поскольку результатом расшифровки является “мусор”, а не случайное число N'_B *).

6’. Боб вступает в контакт с “Алисой”, хотя на самом деле под ее именем скрывается Злоумышленик
(* поскольку случайное число N_B расшифровано правильно *).

ПОСЛЕДСТВИЯ:

Боб думает, что установил контакт с Алисой, хотя Алиса вообще не выходила на связь.

внимание на то, что два зашифрованных текста, содержащихся в сообщениях 3 и 3’, могут и не быть идентичными — это зависит от деталей алгоритма шифрования (см. главы 14 и 15). В любом случае Боб должен просто следовать протокольным инструкциям: он расшифровывает сообщения 4 и 4’ и пересылает их Тренту. Даже если оба зашифрованных текста, полученных Бобом в сообщени-

ях 3 и 3', идентичны (если алгоритм шифрования является детерминированным, что маловероятно в современных системах), Боб не должен этого замечать, поскольку зашифрованный текст им не распознается и не считается сообщением, подлежащим интерпретации. "Отказ от интерпретации" постороннего зашифрованного текста полностью соответствует кодексу поведения законопослушного пользователя (см. раздел 11.3): Боб не подозревает об атаке и не может распознать зашифрованный текст, не предназначенный для расшифровки. Разумеется, такое поведение глупо, но мы приняли соглашение, в соответствии с которым Боб должен быть "наивным". Итак, получив от Трента сообщения 5 и 5', Боб находит в одном из них правильное случайное число N_B и приходит к выводу, что между ним и Алисой установлен контакт, хотя Алиса вообще не выходила на связь. Второе сообщение содержит мусор, поскольку оно является результатом расшифровки сообщения $\{N_B\}_{K_{MT}}$ с помощью ключа K_{AT}, выполненного Трентом. В результате Боб разрывает связь со Злоумышленником, устанавливая контакт с "Алисой".

В атаке с помощью параллельного сеанса последовательность двух сеансов не важна. Например, если Боб получит сообщение 3' до сообщения 3, атака будет развиваться точно так же. Боб может определить, кем было послано то или иное сообщение, по сопутствующей информации на сетевом уровне (см. раздел 12.2).

Абади и Нидхем предложили исправить протокол Ву–Лама. Их предложение будет рассмотрено позднее. Они также сообщили Ву и Ламу о своем изобретении [1]. В ответ Ву и Лам разработали ряд усовершенствований, в которые включили предложение Абади и Нидхема, названное в работе [302] символом Π^3. Наиболее серьезное исправление этого протокола называется Π^f. Оно заключается в добавлении имен обоих субъектов, *Алисы* и *Боба*, во все зашифрованные тексты. Авторы утверждают, что исправленный вариант протокола является безопасным. К сожалению, все эти исправления имеют дефекты (включая исправление Абади и Нидхема). Каждый из этих вариантов уязвим для активной атаки, описанной в следующем разделе.

11.7.4 Атака с помощью отражения сообщений

В атаке с помощью отражения сообщений (reflection attack) Злоумышленник перехватывает сообщение, посланное законопослушным пользователем своему партнеру для дальнейшей криптографической обработки, и посылает его обратно. Отметим, что отраженное сообщение не является "возвращенным сообщением": Злоумышленник изменяет имя и адрес, обработанный низкоуровневым протоколом, поэтому автор сообщения не узнает своего же текста. Технические детали этой атаки описаны в разделе 12.2.

В ходе этой атаки Злоумышленник пытается обмануть автора сообщения и убедить его, что отраженное сообщение пришло от подлинного партнера либо в каче-

стве ответа, либо в качестве запроса. Если обман удается, автор сообщения либо отвечает на свой же вопрос, либо предоставляет Злоумышленнику необходимые услуги оракула.

Изобретя атаку на протокол Ву–Лама с помощью параллельного сеанса (атака 11.5), Абади и Нидхем предложили его исправить [1]: последнее сообщение, посланное Трентом Бобу в протоколе 11.2, должно содержать имя Алисы.

5. Трент → Бобу : $\{Алиса, N_B\}_{K_{BT}}$. (11.7.1)

Это исправление действительно предотвращает атаку с помощью параллельного сеанса, поскольку теперь атака Злоумышленника приведет к следующему результату

5. Трент → Бобу : $\{Злоумышленник, N_B\}_{K_{BT}}$.

Поскольку Боб ожидает сообщения (11.7.1), атака захлебывается.

Однако, несмотря на то, что явное указание имен участников протокола вполне уместно и полностью соответствует принципам разработки протоколов аутентификации, это только одно из исправлений, которое следует сделать. Довольно часто одно исправление предотвращает одну атаку, но не спасает от другой. Вариант протокола Ву–Лама, исправленный Абади и Нидхемом, остается уязвимым для атаки с помощью отражения сообщений (Кларк (Clark) и Джекоб (Jacob) [77]).

Атака 11.6. Атака на исправленный вариант протокола Ву–Лама с помощью отражения сообщений

ИСХОДНЫЕ УСЛОВИЯ:

Те же, что и в протоколе 11.2.

1. Злоумышленник(“Алиса”) → Бобу: *Алиса*.
2. Боб → Злоумышленнику(“Алисе”): N_B.
3. Злоумышленник(“Алиса”) → Бобу: N_B.
4. Боб → Злоумышленник(“Трент”): $\{Алиса, Боб, N_B\}_{K_{BT}}$.
5. Злоумышленник(“Трент”) → Бобу: $\{Алиса, Боб, N_B\}_{K_{BT}}$.
6. Боб аутентифицирует Алису.

ПОСЛЕДСТВИЯ:

Боб думает, что установил контакт с Алисой, хотя Алиса вообще не выходила на связь.

В данном случае Злоумышленник организовывает атаку с помощью отражения сообщений дважды: сообщение 3 является отражением сообщения 2, а сообщение 5 — отражением сообщения 4. Атака является успешной, если в сообщениях 3 и 5 Боб не может обнаружить ничего подозрительного. Это условие хорошо согласуется с кодексом поведения законопослушного пользователя (раздел 11.3).

Во-первых, случайное число, которое Боб получает в сообщении 3, на самом деле было сгенерировано им самим и послано вместе с сообщением 2. Однако Боб не может распознавать никаких посторонних сообщений и должен бездумно следовать протоколу. Аналогично случайное число, которое Боб получает в сообщении 5, также было сгенерировано им самим и послано вместе с сообщением 4. Однако Боб не помнит сообщения 4. Это также полностью соответствует кодексу поведения законопослушного пользователя (раздел 11.3). Итак, Боб не может заподозрить атаку.

Исправления, внесенные Ву и Ламом [302], не спасают положения: все они уязвимы для атаки с помощью отражения сообщений. Наиболее существенное исправление, Π^f, в котором каждый зашифрованный текст должен содержать имена обоих участников протокола, по-прежнему не предотвращает атаку с помощью отражения сообщений, если Боб не реагирует на размер постороннего зашифрованного текста. Это вполне разумное предположение, поскольку по умолчанию Боб считается "наивным".

Более фундаментальная причина уязвимости протокола Ву–Лама и его вариантов исследуется в разделе 17.2.1, посвященном формальному доказательству корректности протоколов аутентификции. Правильный подход к разработке протоколов аутентификации изложен в разделе 17.2. Он позволяет исправить не только протокол Ву–Лама, но и многие другие протоколы. В разделе 17.2.3.2 будет показано, что исправленный протокол Ву–Лама (протокол 17.2) неуязвимым для любых известных атак.

11.7.5 Атака с помощью чередования сообщений

В ходе этой атаки Злоумышленник реализует несколько протоколов, чередуя сообщения. Злоумышленник составляет сообщение и посылает его одному из участников протокола, ожидая ответа. Затем он пересылает полученный ответ другому пользователю в рамках другого протокола, получает его ответ, пересылает очередному пользователю и т.д.

Некоторые авторы [34] считают, что атака с помощью чередования сообщений (interleaving attack) на самом деле является коллективным названием двух предыдущих атак, т.е. атаки с помощью параллельного сеанса и атаки с помощью отражения сообщений. Мы не разделяем эту точку зрения, поскольку атака с помощью чередования сообщений имеет более сложную конструкцию. Для того чтобы организовать успешную атаку с помощью чередования сообщений, Злоумышленник должен использовать последовательно связанные между собой сообщения, пересылаемые в рамках разных протоколов.

Атака Винера (атака 11.1) на первый вариант трехпроходного протокола взаимной аутентификации с открытым ключом, рассмотренный в разделе 11.4.2, представляет собой яркий пример атаки с помощью чередования сообщений. В ходе

атаки Винера Злоумышленник инициирует сеанс связи с пользователем A, маскируясь пользователем B(сообщение 1). Получив ответ от пользователя A (сообщение 2), Злоумышленник инициирует новый сеанс связи с пользователем B, маскируясь пользователем A (сообщение 1'). В ответ пользователь B пересылает Злоумышленнику сообщение, которого ожидает пользователь A. Таким образом, Злоумышленник может переслать пользователю A ожидаемый им ответ и завершить выполнение протокола. В отличие от атаки с помощью параллельного сеанса (например, атаки 11.5), атака с помощью чередования сообщений очень чувствительна к последовательности передаваемых сообщений.

Еще одним примером атаки с помощью чередования сообщений является атака на протокол STS, предназначенный только для аутентификации, осуществляемая с помощью замены сертификата и подписи. Эта атака тесно связана с атакой Винера. Кроме того, атака Лоу на протокол аутентификации с открытым ключом Нидхема–Шредера также относится к семейству атак с помощью чередования сообщений.

Как правило, такие атаки становятся возможными благодаря дефектам, присущим протоколам взаимной аутентификации.

11.7.6 Атака на основе неправильной интерпретации

В атаке на основе неправильной интерпретации (attack due to type flaw) Злоумышленник использует тот факт, что законопослушный пользователь не способен распознавать семантический смысл сообщения или группы сообщений (замечание 11.1 и пример 11.1 в разделе 11.3).

В большинстве случаев неправильная интерпретация проявляется в том, что пользователя путем обмана вынуждают неправильно интерпретировать случайное число, метку времени, имя, зашифрованное в ключе, и т.п. Такая ошибка может возникнуть вследствие неудачной разработки протокола. Продемонстрируем атаку на основе неправильной интерпретации [69, 285] на примере протокола, предложенного Нойманом (Neuman) и Стаблбайном (Stubblebine) [214]. Этот протокол выглядит следующим образом.

1. Алиса $\rightarrow$ Бобу: A, N_A.
2. Боб $\rightarrow$ Тренту: $B, \{A, N_A, T_B\}_{K_{BT}}, N_B$.
3. Трент $\rightarrow$ Алисе: $\{B, N_A, K_{AB}, T_B\}_{K_{AT}}, \{A, K_{AB}, T_B\}_{K_{BT}}, N_B$.
4. Алиса $\rightarrow$ Бобу: $\{A, K_{AB}, T_B\}_{K_{BT}}, \{N_B\}_{K_{AB}}$.

Этот протокол должен обеспечить взаимную аутентификацию Алисы и Боба, а также генерацию аутентифицированного ключа с помощью доверенного посредника, Трента. Если случайное число и ключ имеют одинаковую длину, этот протокол позволяет Злоумышленнику организовать атаку на основе неправильной интерпретации.

1. Злоумышленник ("Алиса") → Бобу: A, N_A.
2. Боб → Злоумышленнику("Тренту"): $B, \{A, N_A, T_B\}_{K_{BT}}, N_B$.
3. Ничего.
4. Злоумышленник ("Алиса") → Бобу: $\{A, N_A, T_B\}_{K_{BT}}, \{N_B\}_{N_A}$.

В ходе этой атаки Злоумышленник вместо сеансового ключа K_{AB} использует одноразовое случайное число N_A. Если Боб не способен различать тип информации, он может ошибочно принять это число в качестве сеансового ключа, и предотвратить этот обман невозможно.

Ошибочная интерпретация информации, как правило, зависит от особенностей реализации протокола. Если спецификация протокола недостаточно ясно описывает тип переменных, используемых в протоколе, ошибка практически неизбежна. Бойд (Boyd) [226] проиллюстрировал эту проблему на примере протокола аутентификации Отвея–Рииса (Otway–Rees) [256] и показал, насколько важно избегать при разработке криптографических протоколов предположений, принятых по умолчанию.

11.7.7 Атака на основе безымянных сообщений

В протоколах аутентификации имя автора сообщения и ключ шифрования, как правило, можно выяснить по контексту. Однако в ситуациях, когда это сделать невозможно, отсутствие имени порождает большие проблемы.

Эксперты в области криптографии, защиты информации и разработки протоколов весьма часто делают эту ошибку. Возможно, это связано с их желанием создать элегантный протокол, не страдающий избыточностью. Яркими примерами атаки на основе безымянных сообщений (attack due to name omission) являются две атаки на варианты протокола STS, рассмотренные в разделах 11.6.2 и 11.6.3 соответственно. Рассмотрим еще один пример.

Деннинг (Denning) и Сакко (Sacco) предложили протокол с открытым ключом, представляющий собой альтернативу протоколу аутентификации с симметричным ключом Нидхема–Шредера [94]. Протокол Деннинга–Сакко выглядит следующим образом.

1. Алиса → Тренту: A, B.
2. Трент → Алисе: Cert_A, Cert_B.
3. Алиса → Бобу: Cert_A, Cert_B, $\{\text{sig}_A(K_{AB}, T_A)\}_{K_B}$.

В этом протоколе третье сообщение шифруется, чтобы обеспечить его секретность и аутентичность. Когда Боб получает сообщение от Алисы, он видит, что сеансовый ключ K_{AB} должен быть разделен только между ним и Алисой, поскольку он видит подпись Алисы и следы применения ее открытого ключа.

К сожалению, протокол не гарантирует, что ключ будет разделен только между Бобом и Алисой. Абади и Нидхем изобрели простую, но впечатляющую атаку [1],

в ходе которой Боб, получив сообщение от Алисы, может заставить другого пользователя поверить, будто имеет место следующее "свойство".

3'. Боб ("Алиса") → Чарли: Cert_A, Cert_C, $\{\text{sig}_A(K_{AB}, T_A)\}_{K_C}$.

Чарли поверит, что сообщение пришло от Алисы, и может послать ей ответ, зашифрованный сеансовым ключом K_{AB}. Увы, Боб сможет его прочитать!

Сообщение 3 имеет следующий смысл: "В момент T_A Алиса говорит, что ключ K_{AB} является правильным ключом для связи между Алисой и Бобом". Очевидно, что правильная спецификация должна выглядеть так.

3. Алиса → Бобу: Cert_A, Cert_C, $\{\text{sig}_A(A, B, K_{AB}, T_A)\}_{K_B}$.

Разрабатывая протоколы, проектировщики должны предусмотреть явное указание имен участников протокола, особенно при выполнении криптографических операций. Однако даже опытные конструкторы часто игнорируют это требование, поэтому Абади и Нидхем внесли это правило в список основных принципов разработки протокола.

> Если имя автора существенно для понимания смысла, его необходимо указывать явно.

Наше навязчивое повторение этого принципа совершенно не излишне: в разделе 12.2 мы продемонстрируем, что протокол IKE [135] также страдает от этого недостатка, хотя его разрабатывали умудренные эксперты в области защиты информации, имеющие многолетний опыт разработки протоколов.

11.7.8 Атака на основе неправильного выполнения криптографических операций

В заключение рассмотрим очень распространенный дефект протоколов — неправильное выполнение криптографических операций, которое заключается в отсутствии необходимой защиты. Этот недостаток проявляется в двух ситуациях.

1. **Атака вследствие отсутствия защиты целостности данных.** Мы продемонстрируем атаку на ослабленный протокол, которая становится возможной из-за отсутствия защиты целостности данных. Более многочисленные примеры атак на криптографические протоколы с открытым ключом содержатся в главе 14, в которой рассматривается понятие стойкости против адаптивных атак. Кроме того, этот дефект протоколов подробно изучается в разделе 17.2, посвященном формальному анализу протоколов аутентификации.
2. **Нарушение секретности вследствие отсутствия "семантической стойкости".** Используя дефект протоколов (и криптосистем), Злоумышленник

может извлекать частичную информацию о секретном сообщении, содержащемся в зашифрованном тексте, и организовывать атаки без взлома криптографического протокола по принципу "все или ничего" (см. свойство 8.2 в разделе 8.2). В главе 14 мы рассмотрим понятие семантической стойкости и продемонстрируем большое количество атак на нее. Криптографические средства, обеспечивающие семантическую стойкость, изучаются в главе 15.

Эти две распространенные ситуации часто упоминаются в литературе, посвященной протоколам аутентификации. Очевидно, разработчики, неправильно выполняющие криптографические операции, не знакомы со слабостями "учебной" криптографии.

Рассмотрим недостаток протоколов, связанный с отсутствием защиты целостности данных. Дефектный протокол, рассмотренный ниже, представляет собой вариант протокола Отвея–Рииса [226] и создан на основе предположений, сформулированных в работе [61].

Протокол 11.8. Ослабленный вариант протокола Отвея–Рииса

ИСХОДНЫЕ УСЛОВИЯ:

Алиса и Трент владеют общим ключом K_{AT}.
Боб и Трент владеют общим ключом K_{BT}.

ЦЕЛЬ: Алиса и Боб выполняют взаимную аутентификацию

1. Алиса → Бобу: M, *Алиса*, *Боб*, $\{N_A, M, Алиса, Боб\}_{K_{AT}}$.
2. Боб → Тренту: M, *Алиса*, *Боб*, $\{N_A, M, Алиса, Боб\}_{K_{AT}}$, $\{N_B\}_{K_{BT}}$, $\{M, Алиса, Боб\}_{K_{BT}}$.
3. Трент → Бобу: M, $\{N_A, K_{AB}\}_{K_{AT}}$, $\{N_B, K_{AB}\}_{K_{BT}}$.
4. Боб → Алисе: M, $\{N_A, K_{AB}\}_{K_{AT}}$.

(* Строка M называется *идентификатором сеанса.* *)

Для взаимной аутентификации и организации аутентифицированного сеанса между двумя пользователями в протоколе 11.8 используются довольно стандартные методы, основанные на применении сервера оперативной аутентификации (Трента). Рассмотрим ход протокола с точки зрения Боба (для Алисы он выглядит точно так же). Анализируя ключ и случайное число, зашифрованные в одном сообщении, Боб может прийти к выводу, что контакт, установленный на этапе 3, является "свежим", а сеансовый ключ принадлежит ему и Алисе. Это достигается путем объединения имен обоих пользователей с идентификатором сеанса M. Это объединение создается самим Бобом и верифицируется Трентом.

Рассмотренный вариант очень мало отличается от исходного протокола Отвея–Рииса: на втором шаге Боб шифрует сообщения (зашифрованные с помощью ключа K_{AT}) в виде двух отдельных порций: в одной из них содержится случайное

число N_A, а во второй — остальные компоненты сообщения. В оригинальном протоколе Отвея–Рииса одноразовое случайное число и остальные компоненты сообщения шифровались в виде одного сообщения: $\{N_B, M, Алиса, Боб\}_{K_{BT}}$.

Интересно, что авторы некоторых реализаций данного протокола вообще не считают этот вариант какой-то модификацией, поскольку при шифровании длинное сообщение всегда разбивается на блоки. Эта точка зрения заслуживает отдельного анализа.

Предложенная модификация протокола не настолько агрессивна, как вариант, описанный в работе [61], в котором предполагалось, что Боб не держит свое случайное число в тайне и может пересылать его открытым текстом. Действительно, послав на втором этапе идентификатор "свежести" в виде открытого текста, Боб может по-прежнему использовать случайное число N_B, полученное на этапе 3, для идентификации "свежести" ключа K_{AB}. Однако мы настаиваем на том, что на этапе 2 должно выполняться шифрование. Причины будут указаны позднее.

Протокол 11.8 фатально уязвим для атаки Бойда (Boyd) и Мао (Mao) [55].

Атака 11.7. Атака на ослабленный протокол Отвея–Рииса

ИСХОДНЫЕ УСЛОВИЯ:

Те же, что и в протоколе 11.8.

Злоумышленник и Трент владеют общим ключом K_{MT}.

(* Итак, Злоумышленник является обычным пользователем системы. *)

1. Злоумышленник ("Алиса") → Бобу: M, *Алиса*, *Боб*, $\{N_M, M, Злоумышленник, Боб\}_{K_{MT}}$.
2. Боб → Злоумышленнику ("Тренту"): M, *Алиса*, *Боб*, $\{N_M, M, Злоумышленник, Боб\}_{K_{MT}}, \{N_B\}_{K_{BT}}, \{M, Алиса, Боб\}_{K_{BT}}$.

2'. Злоумышленник ("Боб") → Тренту: M, *Злоумышленник*, *Боб*, $\{N_M, M, Злоумышленник, Боб\}_{K_{MT}}$, $\{N_B\}_{K_{BT}} \{M, Злоумышленник, Боб\}_{K_{BT}}$.

(* Здесь сообщение $\{M, Злоумышленник, Боб\}_{K_{BT}}$ представляет собой старый зашифрованный текст, который Злоумышленник сохранил в ходе нормального протокола связи между ним и Бобом. *)

3. Трент → Бобу: M, $\{N_M, K_{MB}\}_{K_{MT}}, \{N_B, K_{MB}\}_{K_{BT}}$.
4. Боб → Злоумышленнику ("Трент"): M, $\{N_M, K_{MB}\}_{K_{MT}}$.

ПОСЛЕДСТВИЯ:

Боб думает, что установил контакт с Алисой и разделил с ней сеансовый ключ, хотя на самом деле он общался со Злоумышленником и разделил сеансовый ключ с ним.

В начале этой атаки Злоумышленник маскируется под Алису, чтобы инициировать протокол связи с Бобом. Затем он перехватывает сообщение, посланное Бобом Тренту (этап 2), подменяет имя Алисы своим собственным, не изменяет первую порцию сообщения Боба (Злоумышленник не интересуется зашифрованным случайным числом) и заменяет вторую порцию $\{M, Алиса, Боб\}_{K_{BT}}$ сообщением $\{M, Злоумышленник, Боб\}_{K_{BT}}$, записанным в ходе предыдущего нормального сеанса протокола связи между ним и Бобом. Маскируясь Бобом, Злоумышленник пересылает модифицированное сообщение Тренту (этап 2'). Ни Боб, ни Трент ничего не подозревают: Трент уверен, что аутентификацию запрашивают Злоумышленник и Боб, а Боб думает, что установил связь с Алисой. Увы, Боб разделил сеансовый ключ не с Алисой, а со Злоумышленником, и открыл ему доступ к секретным сообщениям, которые предназначены для Алисы!

Эта атака выявила весьма важный момент: защита идентификатора "свежести" N_B ничего не дает! Намного лучше защитить целостность данных. Для этого необходимо объединить одноразовое случайное число N_B с именами пользователей. Если в протоколе предусмотрена защита целостности данных, число N_B действительно можно пересылать открытым текстом. Без защиты целостности данных шифрование одноразового случайного числа N_B бесполезно!

Выше мы отметили, что некоторые авторы вообще не считают протокол 11.8 модификацией протокола Отвея–Рииса. Это правда, поскольку при шифровании длинное сообщение всегда разбивается на несколько блоков. Если эти блоки не объединяются в одно целое, оба протокола сводятся к одному и тому же коду и не отличаются друг от друга.

В обычных реализациях блочных шифров последовательные блоки зашифрованных текстов криптографически связаны друг с другом, при этом режим CBC (см. раздел 7.8.2) применяется чаще других. Следует заметить, что в этом случае целостность сцепленных блоков зашифрованного текста никак не защищена, хотя многие уверены в обратном. Без защиты целостности данных блок зашифрованного текста можно модифицировать так, что это будет незаметно при расшифровке. Этот недостаток режима CBC рассматривается в разделе 17.2.1.2.

Это еще не все

Можно перечислить еще несколько атак на протоколы аутентификации, например, атаку с помощью обходного канала ("side channel attack"), которая будет рассмотрена в разделе 12.5.4 на примере протокола TLS/SSL (в этом разделе она называется атакой с помощью временно́го анализа ("time analysis attack")), атаку, зависящую от реализации ("implementation dependent attack"), атаку с помощью связывания ("binding attack") и атаку с помощью инкапсуляции ("encapsulation attack") [198]. Многие из этих атак перекрываются с атаками, рассмотренными выше. Кроме того, они не исчерпывают всего списка возможных атак. По этой причине мы не будем их рассматривать.

Уязвимость протоколов аутентификации, даже созданных экспертами, побудила многих исследователей разработать систематический подход к их созданию и анализу. Формальные методы разработки и анализа протоколов аутентификации изучаются в главе 17.

11.8 Краткий обзор литературы

Аутентификация — очень важное свойство криптографических протоколов. Мы рекомендуем следующую литературу для ее изучения.

- Логика аутентификации исследуется в работе Барроуза (Burrows), Абади и Нидхема [61]. Эта книга оказалась весьма полезной и цитируется во многих других публикациях. Она содержит подробное описание ранних протоколов аутентификации и их многочисленных недостатков.
- Обзор разнообразных криптографических протоколов содержится в работах Мура (Moore) [204, 205]. Это — очень важные статьи. В них описаны разнообразные дефекты криптографических протоколов, не связанные со слабостями криптографических алгоритмов. Эти недостатки, скорее, связаны с необоснованными предположениями о применении криптографических операций, которые на самом деле не выполняются.
- Принципы разработки криптографических протоколов рассмотрены Абади и Нидхемом в работе [1]. В этой книге перечислены одиннадцать эвристических принципов, которыми должны руководствоваться разработчики протоколов.
- Обзор литературы, посвященной протоколам аутентификации, сделан Кларком и Джекобом в работе [77]. В ней приведена большая библиотека протоколов аутентификации и протоколов генерации аутентифицированых ключей, а также описаны атаки на них, о которых следует знать всем разработчиками протоколов. В дальнейшем работа [77] легла в основу Web-сайта "Security Protocols Open Repository" (SPORE) `http://www.lsv.ens-cachan.fr/spore/`.
- В скором времени ожидается выход книги Бойда (Boyd) и Матуриа (Mathuria) *Protocols for Key Establishment and Authentication* (Information Security and Cryptography Series, Publisher: Springer, ISBN: 3-540-43107-1). Эта книга представляет собой первое полное описание протоколов генерации аутентифицированных ключей, в частности, основных протоколов, использующих симметричные и асимметричные методы криптографии, групповых протоколов, протоколов генерации ключей для проведения конференции, а также протоколов на основе паролей. Книга представляет собой превосходный справочник. Кроме ясного и подробного анализа протоколов, в книге дано

описание многих атак и перечислены уязвимые места большинства протоколов.

11.9 Резюме

В главе рассмотрен широкий круг тем, связанных с аутентификацией, в частности, основные понятия (целостность данных, сущность, генерация аутентифицированных ключей, односторонность, взаимность, подлинность), конструкции, рекомендованные международными стандартами, стандартные протоколы, некоторые интересные и полезные протоколы (например, однократный пароль, EKE, STS) и атаки на них.

Несмотря на то что протоколы аутентификации вызывают активный академический интерес, они представляют собой еще только развивающуюся область знаний, связанных с криптографическими протоколами. Наше изложение не претендует на полноту, поэтому в конце главы приведен краткий обзор литературы, посвященной этой теме. Кроме того, читатели, интересующиеся протоколами аутентификации, могут найти полезную информацию в главе 17, посвященной формальному доказательству их стойкости.

Протоколы аутентификации — важная часть реальных приложений. В этой главе рассмотрены лишь некоторые аспекты этой темы. Более подробно реальные приложения протоколов аутентификации описываются в следующей главе.

Упражнения

11.1. Укажите разницу между следующими понятиями: целостность данных, аутентификация сообщений, аутентификация сущностей.

11.2. Что такое идентификатор “свежести”?

11.3. Предположим, что криптографическая операция была выполнена недавно. Гарантирует ли это “свежесть” сообщения, посланного пользователем?

11.4. После расшифровки зашифрованного текста (например, созданного с помощью шифра AES-CBC) Алиса обнаружила идентификатор “свежести” (например, посланное ею случайное число). Может ли она утверждать, что полученный зашифрованный текст является “свежим”?

11.5. Почему защита данных в зашифрованном протокольном сообщении так важна для обеспечения секретности?

11.6. В разделе 11.4 описаны основные конструкции протоколов аутентификации. Чем стандартные конструкции отличаются от нестандартных?

11.7. Укажите нестандартную конструкцию в протоколе Ву–Лама (протокол 11.2).

Подсказка: посмотрите, каким образом обеспечивается секретность связи между Бобом и Трентом в сообщениях 4 и 5, и сравните их с конструкцией (11.4.7).

11.8. Что общего у атаки Винера на ослабленный вариант протокола ISO (атака 11.1), атаки с помощью замены сертификата и подписи на протокол STS, предназначенный только для аутентификации (атака 11.2), и атаки Лоу на протокол аутентификации с открытым ключом Нидхема–Шредера (атака 2.3)?

11.9. Каждый ASCII-символ в компьютере представляется с помощью восьми бит. Почему, как правило, в восьми ASCII-символах содержится намного меньше информации, чем 64 бит?

11.10. Что такое "соль" протокола аутентификации с помощью пароля? Какую роль она играет?

11.11. В протоколе аутентификации, применяемом в операционной системе UNIX (см. раздел 11.5.1 и протокол 11.3), криптографическое преобразование $f(P_U)$ генерируется с помощью алгоритма шифрования DES. Можно ли утверждать, что в протоколе применяется функция шифрования DES? В чем заключается разница между этим преобразованием и нестандартным механизмом аутентификации (11.4.7)?

11.12. В протоколе S/KEY (протокол 11.4) используется, по существу, то же самое криптографическое преобразование, что и в протоколе аутентификации операционной системы UNIX (протокол 11.3). Почему второй протокол уязвим, а первый — нет?

11.13. В протоколе EKE (протокол 11.5) используются асимметричные криптографические методы. Можно ли утверждать, что этот протокол является протоколом аутентификации с открытым ключом?

11.14. В главе продемонстрированы недостатки упрощенного протокола STS, предназначенного только для аутентификации (атака 11.2). Исправьте этот протокол.

11.15. В разделе 11.6.3 описано, почему указание адресата устраняет незначительный дефект протокола STS (см. атаку 11.3). Однако при этом исчезает анонимность протокола. Предложите другой способ устранения этого дефекта, сохраняя анонимность пользователей.

Подсказка: пользователи не обязаны объединять общий сеансовый ключ и имя адресата в одно целое. Вот почему мы не считаем, что согласованный сеансовый ключ является взаимно согласованным (см. обсуждение протокола в разделе 11.6).

Глава 12

Протоколы аутентификации — реальный мир

12.1 Введение

Наше обсуждение протоколов аутентификации в предыдущей главе носило академический характер: в ней изучались стандартные конструкции, рассматривались важные примеры и методы, описанные в литературе, и проводился систематический анализ разных "академических" атак. Однако прикладной аспект практически не застрагивался. Несомненно, прикладные протоколы аутентификации должны решать реальные задачи, некоторые из которых весьма сложны.

Эта глава посвящена некоторым проблемам аутентификации, возникающим в реальном мире. В ней описывается большое количество протоколов аутентификации, предложенных для решения прикладных задач. Все эти протоколы фактически являются промышленными стандартами.

В начале главы описывается протокол обмена ключами через Internet (IKE) [138, 158], представляющий собой механизм аутентификации, использованный в протоколе обеспечения безопасности в Internet (Internet Security Protocol — IPSec), принятом группой IETF (Internet Engineering Task Force — проблемная группа проектирования Internet) в качестве стандарта. Эта система состоит из протоколов аутентификации и обмена аутентифицированными ключами, функционирующих на сетевом уровне. Показано, как осуществляется связь, каким образом Злоумышленник может манипулировать адресами и организовывать разнообразные атаки, а также продемонстрирована высокая безопасность сетевого уровня. Мы убедимся, что обеспечить повышенную секретность связи на сетевом уровне, гарантирующую безопасность приложений на более высоких уровнях, чрезвычайно сложно.

Затем описывается протокол SSH (Secure Shell Protocol) [304–308]. Этот протокол аутентификации с открытым ключом предназначен для обеспечения безопасного доступа к ресурсам удаленного компьютера (безопасной удаленной регистрации) с ненадежного компьютера (т.е. с ненадежной клиентской машины — к удаленному серверу). Фактически протокол SSH стал стандартом для безопас-

ной удаленной регистрации в открытых системах и применяется во всемирном масштабе. Протокол SSH является протоколом "клиент-сервер". Как правило, его серверная часть запускается на компьютерах под управлением операционной системы UNIX[1] или ее популярного варианта LINUX, а клиентская — на компьютерах под управлением других операционных систем, например, Windows. Целью протокола SSH является гармоничное обеспечение безопасности: защита должна внедряться так, чтобы не препятствовать работе незащищенных систем (обратная совместимость).

После этого описывается еще один важный и популярный протокол аутентификации — Kerberos [168, 202]. Он обеспечивает сетевую аутентификацию в другой популярной операционной системе — Windows 2000. Эта операционная система широко используется в корпоративных сетях, в которых каждый пользователь имеет право доступа к распределенному корпоративному сервису, не имея возможности хранить многочисленные криптографические мандаты для доступа к разным серверам (пользователь не должен помнить большое количество разных паспортов и хранить большое количество пластиковых карточек с микропроцессором). Архитектура протокола Kerberos, основанная на принципе однократной регистрации ("single–signon"), очень хорошо подходит для обеспечения безопасности в корпоративных сетях.

В заключение рассматривается протокол SSL (Secure Socket Layer) [136], который организация IETF называет протоколом TLS (Transport Layer Security). Во время написания книги этот протокол считался наиболее популярным методом аутентификации с открытым ключом. В настоящее время он интегрирован в каждый Web-браузер и Web-сервер, хотя во многих случаях он используется только для односторонней аутентификации (клиент аутентифицирует сервер). Протокол SSL является типичным протоколом аутентификации в среде "клиент-сервер". Хотя в его основе лежит весьма простая идея (это самый простой среди четырех прикладных протоколов аутентификации, описываемых в данной главе), его реализация в реальных приложениях представляет собой довольно сложную задачу.

12.1.1 Структурная схема главы

Система IPSec и протокол IKE описываются в разделе 12.2. Раздел 12.3 посвящен протоколу SSH. Сценарий однократной регистрации, применяемый в операционной системе Windows, и протокол Kerberos обсуждаются в разделе 12.4. В заключительном разделе 12.5 описывается протокол SSL (TLS).

[1] UNIX – торговая марка компании Bell Laboratories.

12.2 Протоколы аутентификации для обеспечения безопасности в Internet

В книге рассмотрены разнообразные методы защиты сообщений, передаваемых в открытых сетях. Методы, описанные до сих пор, защищали верхний уровень связи — уровень приложений. Защита распространялась только на содержательную часть сообщения, а его адресная часть, представляющая собой низкоуровневую информацию, оставалась незащищенной.

Однако при передаче информации через Internet необходимо обеспечить эффективную защиту как содержательной, так и адресной части сообщений на сетевом уровне. Как мы убедились, изучая раздел 11.7, именно манипуляции с адресной информацией являются основным источником трюков, позволяющих Злоумышленнику организовывать разнообразные атаки.

В данном разделе будет показано, как именно осуществляется обработка сообщения низкоуровневым протоколом связи, и как Злоумышленник может использовать недостатки этого протокола для организации атак. Затем будут описаны стандартные протоколы защиты информации в Internet. Совокупность этих протоколов получила название "Обмен ключами через Internet" (Internet Key Exchange — IKE). Они предназначены для защиты сообщений в низкоуровневом протоколе связи с помощью методов аутентификации, описанных в предыдущей главе. Будут рассмотрены разные режимы обмена ключами через Internet и продемонстрирована их уязвимость. Кроме того, приводятся критические замечания и комментарии, касающиеся протоколов IKE, опубликованные в научных источниках.

12.2.1 Обмен сообщениями на уровне протокола Internet

Internet представляет собой огромную открытую сеть компьютеров, называемых узлами. Каждый узел имеет уникальный сетевой адрес, так что сообщение, посланное на этот узел или отосланное с этого узла, содержит его сетевой адрес. Протокол, обеспечивающий передачу сообщений в сети, называется **протоколом Internet** (Internet Protocol — IP). По этой причине уникальный сетевой адрес называется **IP-адресом узла**. В соответствии с моделью взаимодействия в открытых системах на семи уровнях ("Open Systems Interconnection (ISO-OSI) Sevenlayer Reference Model") [159, 227] протокол IP работает на третьем уровне. Этот уровень называется сетевым или IP-уровнем. Многие протоколы связи, в том числе протоколы аутентификации, работают на седьмом уровне, т.е. уровне приложения. Это еще одна причина, по которой протокол IP называется низкоуровневым протоколом связи, а остальные — высокоуровневыми.

Связь на IP-уровне осуществляется с помощью IP-пакетов. На рис. 12.1 изображен IP-пакет, не имеющий криптографической защиты. Предназначение первых трех полей IP-пакета совершенно очевидно. Четвертое поле, "Поля верхнего

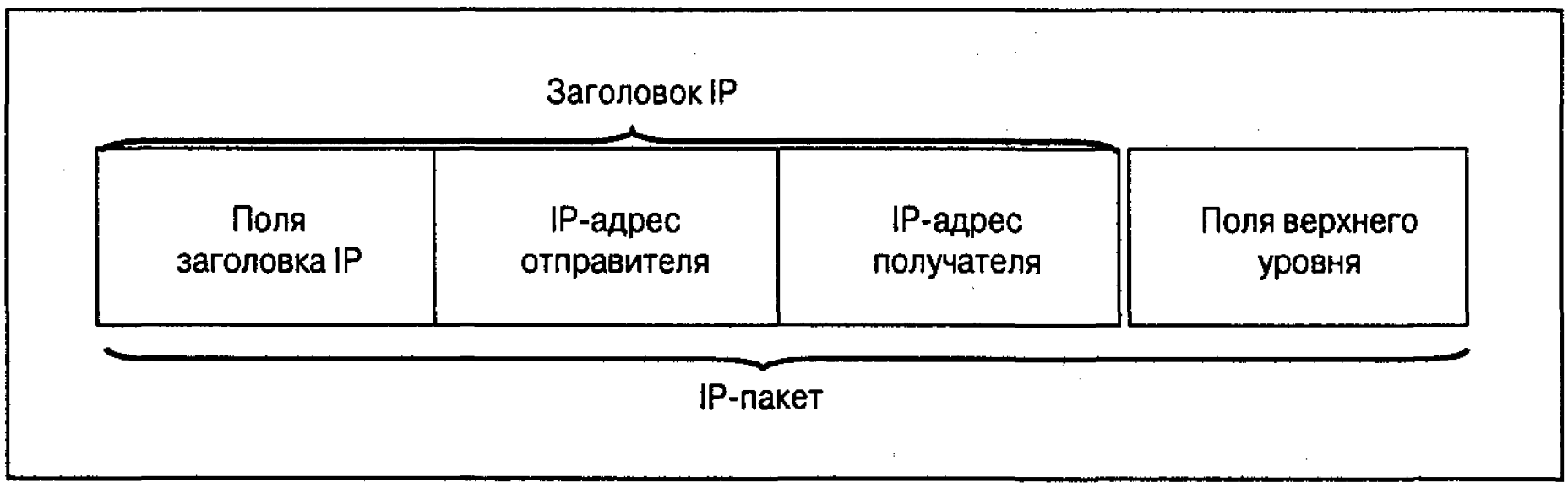


Рис. 12.1. Незащищенный IP-пакет

уровня", содержит, во-первых, спецификацию протокола, запускаемого на бли-жайшем верхнем уровне и обрабатывающего IP-пакет (т.е. "протокола управления передачей"— (Transmission Control Protocol — TCP)) и, во-вторых, данные, содержащиеся в IP-пакете.

В качестве примера обмена сообщениями с помощью IP-пакетов через Internet рассмотрим электронную почту. Начнем с варианта, в котором IP-пакеты не имеют криптографической защиты. Пусть `James_Bond@007.654.321` и `Miss_Moneypenny@123.456.700` — два электронных адреса. Здесь `James_Bond` и `Miss_Moneypenny` — имена конечных пользователей (endpoint identity), а 007.654.321 и 123.456.700 — два IP-адреса [2]. Например, первый из них может представлять собой IP-адрес карманного компьютера, а второй — IP-адрес офисного компьютера. Электронное сообщение, посланное с адреса `Miss_Moneypenny@123.456.700` по адресу `James_Bond@007.654.321`, на IP-уровне выглядит следующим образом.

Поля заголовка IP	123.456.700	007.654.321	...	Дорогой Джеймс, ...

Чтобы упростить изложение, на рисунке показаны только поля верхнего уровня, а спецификация обрабатывающего протокола пропущена (в данном случае пропущена спецификация протокола "SMTP", т.е. простого протокола передачи почтовых отправлений (simple mail transfer protocol) [164]). Отметим, что в полях верхнего уровня указываются имена двух конечных пользователей. Следовательно, когда `James_Bond` получит электронное сообщение, ему станет известно, кто его послал, и он сможет ответить.

Стороны, обменивающиеся сообщениями, могут установить секретную связь, применив оперативное шифрование (end-to-end encryption), используя общий или

[2] Часто IP-адресу ставят в соответствие "имя домена", которое намного легче запомнить. Например, IP-адрес 007.654.321 можно заменить именем домена `spy1.mi.five.gb`.

открытый ключи. Поскольку оперативное шифрование осуществляется на уровне приложений, зашифровывается только четвертое поле IP-пакета. Если протокол Internet, применяемый пользователями, не обеспечивает безопасности, поля данных в IP-заголовке не защищаются. Изменение данных, содержащихся в этих полях, открывает много возможностей для организации разнообразных атак, перечисленных в разделе 11.7.

12.2.2 Протокол обеспечения безопасности в Internet (IPSec)

Организация IETF разработала протокол для обеспечения безопасности в Internet, известный под названием **IPSec** [161, 163]. Протокол IPSec предназначен для криптографической защиты IP-заголовка, состоящего из трех полей IP-пакета (см. рис. 12.1). Этот протокол предусматривает обязательную аутентификацию IP-заголовка и возможную защиту информации об отправителе и адресате, содержащейся в некоторых полях IP-заголовка.

Следует отметить, что без защиты на уровне протокола Internet передача IP-заголовка осуществляется открыто, что позволяет Злоумышленнику организовать разнообразные атаки, такие как получение доступа путем обмана (spoofing[3]), перехват сообщения (sniffing) и захват сеанса (session hijacking). Например, если Злоумышленник перехватил IP-пакет, посланный с адреса `James_Bond@007.654.321`, заменил адрес получателя адресом отправителя и послал его обратно, пакет вернется к отправителю. Если эта модификация осталась незамеченной из-за отсутствия средств защиты, становится возможной атака с помощью отражения сообщений, рассмотренная в разделе 11.7.4. Более того, если Злоумышленник способен подделать адрес отправителя и имя адресата (например, `Miss_Moneypenny`), то отправитель `James_Bond@007.654.321` окажется обманутым и будет думать, что сообщение пришло от фальшивого адресата. Этот вид атаки на уровне приложения обозначается так.

Злоумышленник (“`Miss_Moneypenny`”) → `James_Bond: ...`

Во всех атаках, рассмотренных в разделе 11.7, Злоумышленник должен выполнять некоторые манипуляции с IP-адресами и именами пользователей, хранящимися в IP-заголовке. Следовательно, защита, предусмотренная на уровне протокола Internet, может эффективно предотвращать такие атаки, поскольку манипуляции с IP-заголовком становятся невозможными. Вообще, безопасность на уровне протокола Internet обеспечивает защиту всех приложений на более высоких уровнях.

---

[3] Атака, в ходе которой Злоумышленник пытается соединиться с сервером Internet, прокси-сервером или брандмауэром, используя ложный IP-адрес. — *Прим. ред.*

Более того, поскольку каждый **брандмауэр** (firewall[4]) одновременно защищает несколько узлов, в ходе обмена информацией между двумя брандмауэрами IP-адрес каждого защищенного узла должен шифроваться. Это значит, что любое несанкционированное вмешательство в работу брандмауэра можно предотвратить с помощью криптографических средств, представляющих собой очень сильное средство защиты информации. Без защиты информации на уровне протокола Internet эффективность брандмауэров падает, а вероятность вторжения возрастает.

В настоящее время необходимость защиты информации на уровне протокола Internet стала общепризнанным фактом.

12.2.2.1 Аутентификация в протоколе IPSec

Протокол Internet (IP) эволюционировал от версии 4 (IPv4) к версии 6 (IPv6). В протоколе IPv6 данные хранятся в виде 32-битовых блоков, называемых **дейтаграммами** (datagram). Кроме того, в защищенном протоколе IPv6 IP-пакет содержит дополнительное поле, называемое **заголовком аутентификации** (Authentication Header — AH) (рис. 12.2.2.2, ср. с рис. 12.1). Этот заголовок расположен между IP-заголовком и полями верхнего уровня. Длина заголовка аутентификации может изменяться, однако она всегда является кратной 32 бит, т.е. содержит целое число дейтаграмм, которые образуют несколько подполей, хранящих данные для криптографической защиты IP-пакета.

Аутентификация (фактически защита целостности данных и идентификация их источника) представляет собой неотъемлемую часть протокола IPSec. Эта защита достигается за счет данных, хранящихся в двух подполях заголовка аутентификации. Одно из этих подполей называется **индексом параметра безопасности** ("Security Parameter Index" — SPI). Это поле содержит произвольное 32-битовое значение, однозначно указывающее криптографические алгоритмы, применяемые при аутентификации IP-пакета. Другое поле содержит **данные аутентификации** ("Authentication Data"), созданные отправителем сообщения и предназначенные для проверки целостности данных после получения (поэтому эти данные также называются **контрольными значениями для проверки целостности данных** (Integrity Check Value — ICV)). Получатель IP-пакета восстановит данные аутентификации с помощью секретного ключа и алгоритма, однозначно заданного индексом SPI, и сравнит их с полученными значениями. Применение секретного ключа обсуждается в разделе 12.2.3.

Подполе "порядковый номер" ("Sequence Number") используется для предотвращения повторных пересылок IP-пакетов. Другие подполя в первой дейтаграмме заголовка аутентификации под названиями "следующий заголовок" ("Next

[4]Брандмауэр — это специализированный компьютер, соединяющий кластер защищенных компьютеров и устройств с Internet, так что для доступа к этому кластеру из Internet необходимо знать некоторое имя и IP-адрес.

Рис. 12.2. Структура заголовка аутентификации и его место в IP-пакете

Header"), "величина полезной нагрузки" ("Payload Length") и "резерв для дальнейшего применения" ("Reserved for future use") не используются для обеспечения защиты и потому не представляют интереса.

12.2.2.2 Обеспечение секретности в протоколе IPSec

Обеспечение секретности (шифрование) — необязательная часть протокола IPSec. Для этого в IP-пакет включается целое число 32-битовых дейтаграмм, обеспечивающих "**инкапсуляцию зашифрованных данных**" (Encapsulating Security Payload — ESP) [162]. Эти дейтаграммы изображены на рис. 12.2 рядом с заголовком аутентификации ("Поля верхнего уровня"). Формат дейтаграмм ESP показан на рис. 12.3.

Первое подполе называется "индексом параметра безопасности" (Security Parameters Index — SPI). Оно однозначно определяет алгоритм шифрования. Второе подполе — "порядковый номер" — имеет то же предназначение, что и в заголовке аутентификации (см. раздел 12.2.2.1). Третье подполе, "полезная нагрузка", имеет переменную длину и представляет собой зашифрованный текст, содержащий секретные данные. Поскольку длина пакета в протоколе IPv6 должна быть кратной 32 бит, исходный текст в поле "полезная нагрузка" должен быть дополнен

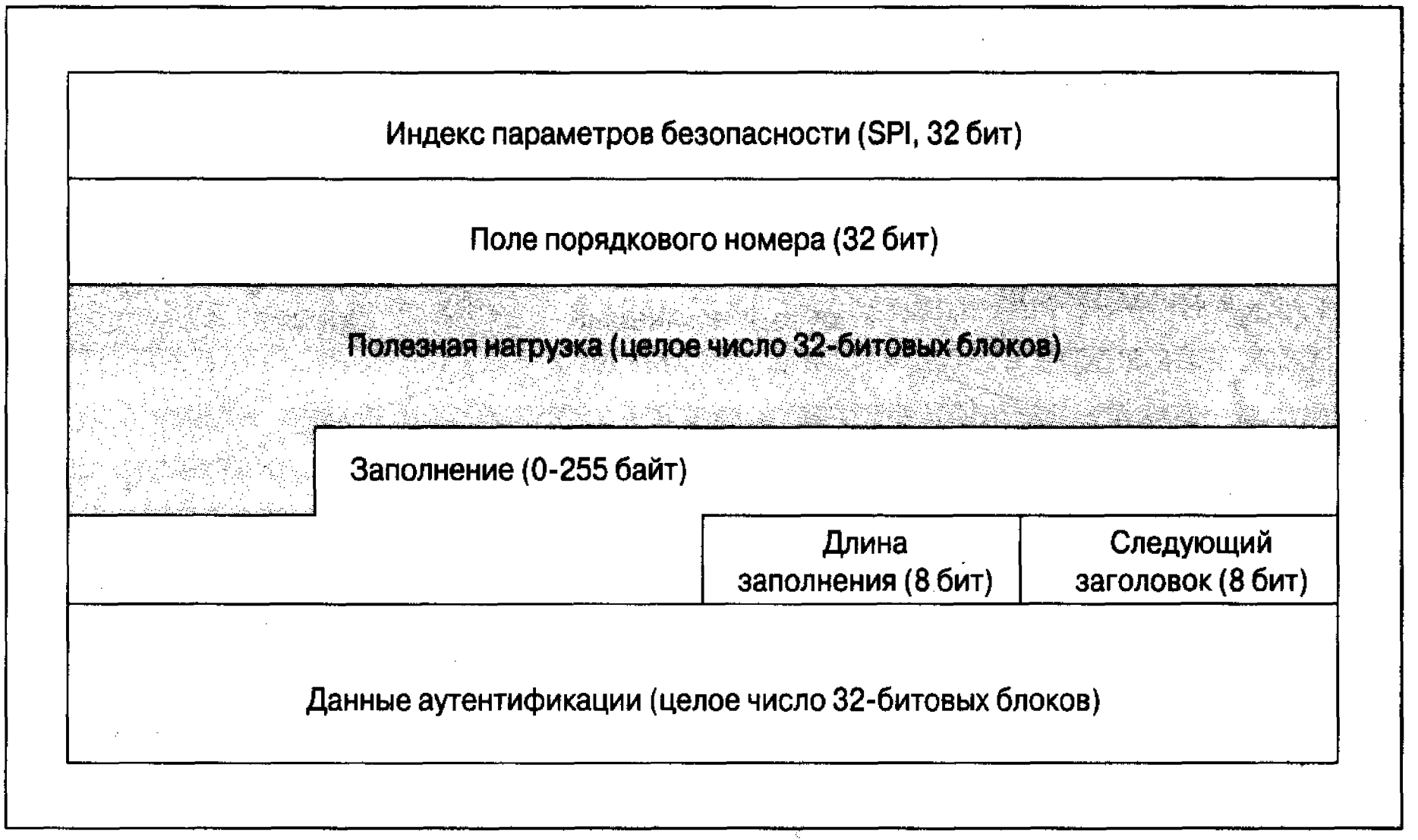


Рис. 12.3. Структура инкапсулированной защиты и ее место в IP-пакете

с помощью заполнителей, хранящихся в поле "заполнение". Байты заполнения инициализируются однобайтовыми целыми числами без знака. Первый байт заполнения, добавленный к исходному тексту, нумеруется числом 1, а последующие байты заполнения образуют монотонно возрастающую последовательность:

$$\text{‘01’} \parallel \text{‘02’} \parallel \cdots \parallel \text{‘xy’},$$

где ‘xy’ — шестнадцатеричное число, удовлетворяющее условию ‘01’ $\leqslant xy \leqslant$ ‘FF’. Следовательно, максимальное количество байтов заполнения равно ‘FF’ $= 255_{(10)}$. Количество байтов заполнения называется "длиной заполнения". Заключительный блок, "данные аутентификации", имеет тот же смысл, что и в заголовке аутентификации.

Отметим разницу между данными аутентификации в пакетах ESP и AH. В первом случае они предназначены для защиты целостности данных, содержащихся в зашифрованном тексте (т.е. полей пакета ESP минус подполе "данные аутентификации"), и являются необязательными [162], а во втором — для защиты целостности данных в IP-пакете и являются необходимыми.

Необязательность данных аутентификации в пакете ESP на самом деле является ошибкой. Она обсуждается в разделе 12.2.5.

12.2.2.3 Параметры защиты

Основным понятием в протоколе IPSec является **защищенное соединение** (Security Association — SA), которое однозначно определяется тройкой

(SPI, IP-адрес получателя, служебный идентификатор),

где служебный идентификатор ("Service Identifier") обозначает либо аутентификацию либо инкапсулированную защиту.

По существу, протокол IPSec представляет собой объединение аутентификации и инкапсулированной защиты. Если два узла устанавливают между собой связь с помощью протокола IPSec, они должны обязательно согласовать либо одно (для аутентификации), либо два (для аутентификации и обеспечения конфиденциальности) защищенных соединения, а также секретные криптографические ключи, разделенные между двумя узлами для обеспечения криптографической защиты. Согласование достигается с помощью протокола обмена ключами через Internet (Internet Key Exchange Protocol — IKE Protocol).

12.2.3 Протокол обмена ключами через Internet (IKE)

Включение в протокол IPSec заголовка аутентификации и инкапсулированной защиты обеспечивает криптографическую защиту IP-пакета. Однако два узла, устанавливающих между собой связь, сначала должны согласовать средства защиты (криптографические ключи, алгоритмы, параметры). Для этого предназначен протокола обмена ключами через Internet (IKE) [135, 158]. В настоящее время он является стандартным протоколом обмена ключами в рамках протокола IPSec, одобренным группой IETF. .

Протокол IKE представляет собой набор протоколов аутентификации и обмена аутентифицированными ключами. Каждый протокол из этого набора представляет собой гибрид, использующий часть протокола Окли (Протокол определения ключа Окли — Oakley Key Determination Protocol [225]), часть механизма SKEME (Механизм для многостороннего обмена секретными ключами через Internet — Versatile Secure Key Exchange Mechanism for Internet [172]) и часть протокола ISAKMP (Протокол установления защищенных соединений и управления ключами — Internet Security Association and Key Management Protocol [187]).

Окли описал несколько процессов обмена ключами, называемых **режимами**, и указал их предназначение (например, полная заблаговременная секретность сеансовых ключей, сокрытие имен конечных пользователей и взаимная аутентификация). Механизм SKEME представляет собой способ обмена аутентифицированными ключами и обеспечивает анонимность соединений между партнерами (благодаря использованию разделенных ключей в рамках протоколов IKE и IKEv2). Протокол ISAKMP обеспечивает аутентификацию и обмен аутентифицированными ключами между двумя общающимися сторонами, позволяя им согласовывать

и утверждать различные защитные средства, криптографические алгоритмы, параметры безопасности, механизмы аутентификации и тому подобные, т.е. все то, что в совокупности называется **защищенными соединениями** (secure associations). Однако в протоколе ISAKMP не определены конкретные способы обмена ключами, что позволяет применять его в сочетании с разными протоколами.

Будучи гибридом, протокол IKE представляет собой набор двусторонних протоколов, предназначенных для обмена аутентифицированными сеансовыми ключами, большинство из которых использует механизм обмена ключами Диффи–Хеллмана и предоставляет пользователям широкие возможности для переговоров в интерактивном режиме.

Протокол IKE состоит из двух фаз — первой и второй.

Первая фаза предполагает, что каждая из двух сторон, участвующих в обмене ключами, знает имя другой стороны. Кроме того, другая сторона может продемонстрировать партнеру определенные криптографические возможности. К их числу относятся владение заранее разделенным секретным ключом в симметричных криптосистемах или закрытым ключом, соответствующим открытому ключу в криптосистемах с открытым ключом. Целью первой фазы является взаимная аутентификация[5], основанная на демонстрации криптографических возможностей, и установка общего сеансового ключа. Этот ключ применяется на протяжении текущего сеанса первой фазы протокола, но его можно применять и на второй фазе, и для защиты связи более высокого уровня.

После обмена ключами в рамках первой фазы стороны могут приступить к многократным обменам ключами во второй фазе. Вторая фаза часто называется "быстрым режимом" ("Quick Mode"). В ее основе лежит общий ключ, согласованный в рамках первой фазы. Многократные обмены ключами во второй фазе протокола позволяют пользователям установить многочисленные контакты с разным уровнем безопасности, например, "только защита целостности данных", "только обеспечение секретности", "шифрование с помощью короткого ключа" или "шифрование с помощью стойкого ключа".

Для того чтобы продемонстрировать протокол IKE, рассмотрим режимы его первой фазы.

12.2.3.1 Первая фаза протокола IKE

Первая фаза протокола IKE имеет восемь вариантов, поскольку существует три разновидности ключей (заранее разделенные симметричные ключи, открытые ключи для шифрования и открытые ключи для верификации цифровой подписи) и две версии взаимозаменяемых протоколов, основанных на применении открытых ключей шифрования и обеспечивающих обратную совместимость. Итак, на

[5]Как мы увидим далее, в некоторых режимах первая фаза протокола IKE не обеспечивает взаимной аутентификации, т.е. пользователь абсолютно уверен в том, что разделяет сеансовый ключ с подлинным партнером, в то время как на самом деле ключ разделен с другой стороной.

самом деле существует четыре типа ключей (заранее разделенные симметричные ключи, устаревшие открытые ключи для шифрования, современные открытые ключи для шифрования и открытые ключи для верификации цифровой подписи). Каждому из этих типов ключей соответствует два вида обменов в первой фазе протоколе: "основной режим" и "агрессивный режим".

Основной режим предусматривает шесть обменов сообщениями: три сообщения передаются инициатором (сокращенно I (Initiator)) ответчику (сокращенно R (Responder)), а остальные три сообщения — передаются ответчиком R инициатору I. Основной режим протокола IKE является обязательным, т.е. два пользователя не могут перейти в агрессивный режим, не выполнив протокол IKE в основном режиме.

В агрессивном режиме предусмотрено только три сообщения: пользователь I инициирует сообщение, пользователь R отвечает, а затем пользователь I передает заключительное сообщение и прекращает выполнение протокола. Агрессивный режим является необязательным и может быть пропущен.

Для первой фазы протокола IKE мы опишем и проанализируем только "режимы, основанные на цифровой подписи". В других режимах для аутентификации, как правило, используется шифрование и расшифровка идентификатора "свежести". В разделе 11.4.1.5 этот механизм назван нестандартным. Его дальнейший анализ содержится в разделе 17.2.

12.2.3.2 Основной режим первой фазы протокола IKE, основанный на цифровой подписи

Основной режим первой фазы протокола IKE, основанный на цифровой подписи, называемый также **аутентификацией с помощью цифровых подписей** ("Autheticated with Signatures" [135]), описан в протоколе 12.1. Этот режим является порождением двух протоколов: протокола STS (протокол 11.6) и протокола SIGMA, предложенного Кравчиком (Krawchyk) [171] (см. раздел 12.2.4).

Первые два сообщения обеспечивают обмен заголовками и параметрами защиты: пользователь I передает пользователю R заголовок HDR_I и набор параметров SA_I и в ответ получает заголовок HDR_R и набор параметров SA_R. Заголовки сообщений HDR_I и HDR_R содержат строки C_I и C_R ("cookies"): строка C_I предназначена для пользователя R и содержит информацию о сеансе связи с пользователем I, а вторая — наоборот. Набор параметров защиты SA_I перечисляет атрибуты системы безопасности, которые хотел бы использовать инициатор протокола I. Пользователь выбирает один из них и возвращает набор параметров SA_R.

Вторая пара сообщений обеспечивает обмен параметрами ключей Диффи–Хеллмана.

Пятое и шестое сообщения предназначены для согласования алгоритма шифрования, цифровой подписи и псевдослучайных функций, применяемых для хэширования подписанных сообщений.

Протокол 12.1. Основной режим первой фазы протокола IKE, основанный на цифровой подписи

1. $I \to R$: HDR_I, SA_I.
2. $R \to I$: HDR_R, SA_R.
3. $I \to R$: HDR_I, g^x, N_I.
4. $R \to I$: HDR_R, g^y, N_R.
5. $I \to R$: HDR_I, $\{ID_I, \mathrm{Cert}_I, \mathrm{Sig}_I\}_{g^{xy}}$.
6. $R \to I$: HDR_R, $\{ID_R, \mathrm{Cert}_R, \mathrm{Sig}_R\}_{g^{xy}}$.

Обозначения. (* Чтобы упростить изложение, мы опускаем некоторые детали. Это не уменьшает функциональных возможностей протокола, в частности, не делает его уязвимым для атаки, описанной в дальнейшем. *)

I, R: инициатор и ответчик соответственно.

HDR_I, HDR_R: заголовки сообщений, посланных инициатором I и ответчиком R соответственно. Эти данные содержат строки C_I и C_R, ("cookies"[6]), представляющие собой информацию о сеансе связи между двумя сторонами.

SA_I, SA_R: защищенные соединения, установленные инициатором I и ответчиком R. Две стороны используют соединения SA_I и SA_R для согласования параметров, используемых в рамках текущего сеанса, в частности, алгоритмов шифрования, алгоритмов цифровой подписи, псевдослучайных функций для хэширования подписываемых сообщений и т.п. Инициатор I может предложить несколько вариантов, но ответчик R должен выбрать только один из них.

g^x, g^y: элементы ключей Диффи–Хеллмана, принадлежащие инициатору I и ответчику R соответственно.

ID_I, ID_R: имена конечных пользователей I и R соответственно.

N_I, N_R: одноразовые случайные числа, сгенерированные пользователями I и R.

Sig_I, Sig_R: подписи, созданные пользователями I и R соответственно. Подписанные сообщения обозначены символами M_I и M_R:

$$M_I = \mathrm{prf}_1(\mathrm{prf}_2(N_I \mid N_R \mid g^{xy}) \mid g^x \mid g^y \mid C_I \mid C_R \mid \mathrm{SA}_I \mid \mathrm{ID}_I),$$
$$M_R = \mathrm{prf}_1(\mathrm{prf}_2(N_I \mid N_R \mid g^{xy}) \mid g^y \mid g^x \mid C_R \mid C_I \mid \mathrm{SA}_R \mid \mathrm{ID}_R),$$

где prf_1 и prf_2 — псевдослучайные функции, используемые в защищенных соединениях.

[6] Cookie — это текстовая строка, записанная в памяти или файле удаленного головного компьютера для сохранения информации о сеансе между клиентом и сервером.

Основной режим первой фазы протокола IKE, основанный на цифровой подписи, похож на протокол STS (протокол 11.6). Однако следует отметить два существенных отличия.

1. Протокол STS оставляет сертификаты за рамками шифрования, в то время как в основном режиме первой фазы протокола IKE сертификаты шифруются, обеспечивая анонимность пользователей (раздел 11.6.1). Это свойство особенно полезно, если пользователи I и R защищены брандмауэрами.
2. Для подписания сообщений в рамках основного режима первой фазы протокола STS не нужен согласованный сеансовый ключ, а в протоколе IKE к подписанному сообщению применяется псевдослучайная функция prf, в которой в качестве начального значения используется согласованный сеансовый ключ g^{xy}. Следовательно, в этом режиме цифровые подписи могут верифицироваться только партнерами, согласовавшими общий сеансовый ключ.

12.2.3.3 Отказ в аутентификации в основном режиме первой фазы протокола IKE, основанного на цифровой подписи

Как и в протоколе STS, сообщение, подписанное в основном режиме первой фазы протокола IKE, содержит информацию только об отправителе, но не о получателе. Это может спровоцировать атаку, похожую на атаку Лоу (атака 11.3). Аналогичный недостаток был обнаружен Мидоузом (Meadows) в работе [195].

Используя обнаруженный недостаток, Злоумышленник может успешно обмануть пользователя R, заставив его поверить в то, что пользователь I инициировал и полностью завершил сеанс связи с ним, хотя на самом деле пользователь I прекратил выполнение протокола досрочно. Обратите внимание на то, что пользователь R оказывается обманутым дважды: во-первых, он вступает в контакт с фальшивым партнером и делит с ним общий ключ, и, во-вторых, он никогда не узнает об этом. Этот недостаток ярко демонстрирует атака 12.2.3.4.

Эта атака называется также “атакой с помощью отказа в обслуживании” (“denial of service attack”). В рамках протокола IKE после успешного обмена сообщениями в первой фазе сервер, играющий роль ответчика R, сохранит текущее состояние сеанса с пользователем I, чтобы применить согласованный сеансовый ключ во второй фазе. Однако после атаки 12.2.3.4 пользователь I никогда не сможет вступить в контакт с сервером R. Следовательно, сервер R может сохранить состояние сеанса, выделить ресурсы для пользователя I и ожидать, когда он вновь приступит к обмену сообщениями. Если Злоумышленник сможет организовать массированную атаку, используя большое количество сообщников одновременно, ресурсы сервера быстро окажутся исчерпанными. Обратите внимание на то, что в этой атаке Злоумышленник не обязан проводить изощренные манипуляции или сложные вычисления. Следовательно, распределенная атака может оказаться весьма эффективной.

Атака 12.1. Отказ в аутентификации в основном режиме первой фазы протокола IKE, основанного на цифровой подписи

(* Злоумышленник представляется инициатору I своим подлинным именем, а ответчику R — именем инициатора I. *)

1. $I \rightarrow$ Злоумышленнику: HDR_I, SA_I.

   1'. Злоумышленник ("I") $\rightarrow R$: HDR_I, SA_I.

   2'. $R \rightarrow$ Злоумышленнику ("I"): HDR_R, SA_R.

2. Злоумышленник $\rightarrow I$: HDR_R, SA_R.

3. $I \rightarrow$ Злоумышленнику: HDR_I, g^x, N_I.

   3'. Злоумышленник ("I") $\rightarrow R$: HDR_I, g^x, N_I.

   4'. $R \rightarrow$ Злоумышленнику ("I"): HDR_R, g^y, N_R.

4. Злоумышленник $\rightarrow I$: HDR_R, g^y, N_R.

5. $I \rightarrow$ Злоумышленнику: HDR_I, $\{ID_I, \mathrm{Cert}_I, \mathrm{Sig}_I\}_{g^{xy}}$.

   5'. Злоумышленник ("I") $\rightarrow R$: HDR_I, $\{ID_I, \mathrm{Cert}_I, \mathrm{Sig}_I\}_{g^{xy}}$.

   6'. $R \rightarrow$ Злоумышленнику ("I"): HDR_R, $\{ID_R, \mathrm{Cert}_R, \mathrm{Sig}_R\}_{g^{xy}}$

6. Конец связи.

ПОСЛЕДСТВИЯ:

Пользователь R полностью обманут и ошибочно полагает, что общается и обменивается ключами с пользователем I, в то время как пользователь I считает, что вступил в контакт со Злоумышленником, досрочно прервав выполнение протокола. Пользователь R ни о чем не подозревает и может получить отказ от пользователя I, оставаясь в режиме ожидания ответа на свой запрос.

Эта атака стала возможной из-за того, что подписанное сообщение в протоколе содержит только имя отправителя и может использоваться для обмана пользователя, не являющегося адресатом этого сообщения. Если бы подписанное сообщение содержало имена как отправителя, так и получателя, сообщение стало бы очень конкретным и не смогло бы применяться для других целей.

Мы еще не раз увидим, что отсутствие имен в сообщении открывает возможность для атаки.

12.2.3.4 Агрессивный режим первой фазы протокола IKE, основанного на цифровой подписи

Агрессивный режим первой фазы протокола IKE, основанного на цифровой подписи, представляет собой усеченную форму основного режима: в нем не ис-

пользуется шифрование, и происходит обмен тремя сообщениями, а не шестью. Используя те же обозначения, что и в протоколе 12.1, этот режим можно записать следующим образом.

1. $I \rightarrow R$: HDR_I, SA_I, g^x, N_I, ID_I.
2. $R \rightarrow I$: HDR_R, SA_R, g^y, N_R, ID_R, Cert_R, Sig_R.
3. $I \rightarrow R$: HDR_R, Cert_I, Sig_I.

На первый взгляд этот режим очень напоминает протокол STS, предназначенный только для аутентификации (протокол 11.7), так как в нем не используется шифрование. Однако при ближайшем рассмотрении обнаруживается существенное отличие: в протоколе STS, предназначенном только для аутентификации, подписанные сообщения не обрабатывались с помощью сеансового ключа, в то время как в агрессивном режиме подписанные сообщения поступают на вход псевдослучайной функции, использующей сеансовый ключ g^{xy} в качестве начального значения. Таким образом, в этом режиме подписи могут верифицироваться только пользователями, владеющими общим сеансовым ключом. Это позволяет предотвратить атаку с помощью замены сертификата и подписи (атака 11.2).

И все же агрессивный режим не обеспечивает взаимную аутентификацию. Он уязвим для атаки с помощью отказа в обслуживании, которая, по существу, совпадает с атакой Лоува на протокол STS (атака 11.3). Теперь пользователь I оказывается полностью обманутым. Он полагает, что общается и делит общий ключ с пользователем R, в то время как пользователь R так не думает. Читатели могут сами описать конструкцию такой атаки (упражнение 12.6).

Следует также заметить, что если схема цифровой подписи, использованная в этом режиме, допускает восстановление сообщения, Злоумышленник может достичь большего. Например, по подписанному сообщению Злоумышленник может вычислить значение $\mathrm{prf}_2(N_I \mid N_R \mid g^{xy})$ и воспользоваться эти материалом для создания собственной подписи со своим сертификатом и именем. Следовательно, он может организовать атаку с помощью замены сертификата и подписи, аналогичную атаке 11.2. Такая атака является весьма эффективной, поскольку оба сеанса связи, которые Злоумышленник организует между пользователями I и R, успешно завершаются и оба пользователя ничего плохого не подозревают. Отметим, что некоторые схемы цифровой подписи позволяют восстановить сообщение (например, схема [220], которая даже была принята в качестве стандарта). Таким образом, два общающихся партнера не должны проводить переговоры, используя схемы цифровой подписи, допускающие восстановление сообщений.

Отсутствие шифрования не позволяет обеспечить “правдоподобное отрицание”, обсуждаемое в разделе 12.2.4. Если это свойство не требуется, исправить недостаток агрессивного режима очень просто: в обе подписи следует включить имена обоих конечных пользователей так, чтобы подписанное сообщение было невозможно применить в другом контексте.

Методы обеспечения взаимной аутентификации с возможностью правдоподобного отрицания обсуждаются в разделе 12.2.4.

12.2.3.5 Другие свойства протоколов IPSec и IKE

Анализ безопасности протоколов IPSec и IKE проводился в нескольких работах.

Используя свой автоматизированный анализатор протоколов NRL (см. раздел 17.5.2 [193, 194]), Мидоуз обнаружил, что быстрый режим (вторая фаза протокола IKE) уязвим для атаки с помощью отражения сообщений [195].

Ferguson (Фергюсон) и Шнайер (Schneier) провели полное криптографическое исследование протокола IPSec в работе [108].

Белловин обнаружил, что в некоторых ситуациях протокол IKE не обеспечивает защиту целостности данных в зашифрованных текстах [27]. Как известно, секретность сообщений невозможна без защиты целостности данных (см. раздел 11.7.8). В дальнейших главах (главы 14–17) будет показано, что большинство алгоритмов шифрования не обеспечивают секретность, если зашифрованные тексты не гарантируют целостности данных. Однако, по всей видимости, эта опасность осталась незамеченной большинством исследователей. Возможно, это объясняется высокой сложностью спецификаций протокола IPSec.

12.2.4 Возможность правдоподобного отрицания в протоколе IKE

Во время написания книги была опубликована спецификация второй версии протокола IKE (IKEv2) [158]. Первая фаза протокола IKEv2 представляет собой объединение различных режимов первой фазы протокола IKE. Однако в современной спецификации [158] для обеспечения аутентификации используются цифровые подписи. Бойд, Мао и Патерсон (Paterson) продемонстрировали, что первая фаза протокола IKEv2 имеет те же самые недостатки, что и протокол IKE, и по-прежнему уязвима для атаки 12.2.3.4 [56].

Новым свойством протокола IKEv2 является “возможность правдоподобного отрицания” [139]. Впервые это свойство было продемонстрировано в протоколе SIGMA, предложенном Кравчиком [171], а также Канетти (Canetti) и Кравчиком [67]. Оно позволяет пользователю правдоподобно отрицать наличие контакта с партнером. Такое свойство весьма желательно на IP-уровне, поскольку оно обеспечивает сохранение тайны, например, анонимность, которая является непременным атрибутом соединений на высоком уровне. Нарушение анонимности на IP-уровне может повлечь нарушение анонимности на уровне приложений. Например, если протокол не позволяет отрицать имя пользователя, связанное с IP-адресом, анонимность на уровне приложений полностью уничтожается.

Возможность “правдоподобного отрицания”, присущая протоколу SIGMA, обеспечивается следующими двумя сообщениями протокола 12.1.

$I \rightarrow R$: s, ID_I, Sig_I(“1”, s, g^x, g^y,), MAC(g^{xy}, “1”, s, ID_I).
$R \rightarrow I$: s, ID_R, Sig_R(“0”, s, g^y, g^x,), MAC(g^{xy}, “0”, s, ID_R).

Здесь s — идентификатор сеанса. Обе стороны могут верифицировать подписи, а затем применять общий сеансовый ключ для верификации соответствующего кода аутентификации сообщений (MAC), чтобы убедиться, что контакт установлен с подлинным партнером. После уничтожения сеансового ключа партнеры не могут доказать третьей стороне, что между ними существовала связь.

Очевидно, что эта конструкция небезупречна и уязвима для атаки 12.2.3.4. Канетти и Кравчик предусмотрели возможность менее интересной атаки, в которой Злоумышленник просто блокировал последнее сообщение, направленное пользователю I. Они предложили метод для предотвращения этой атаки, который заключался в том, что пользователь I должен был послать заключительное подтверждение пользователю R [67]. Поскольку теперь пользователь R (как правило, сервер) ожидает заключительное сообщение, атаку с помощью блокирования последнего сообщения легко распознать. Не получив подтверждения, сервер R просто восстановит предыдущее состояние и освободит выделенные ресурсы. В этом случае протокол становится менее уязвимым для атаки с помощью отказа в обслуживании. Заключительное подтверждение может иметь полезный побочный эффект, устраняя недостаток аутентификации (в зависимости от криптографической формулировки сообщения). Однако совершенно очевидно, что такой способ исправления протокола не совсем желателен, так как он увеличивает трафик и усложняет протокол.

Поскольку возможность правдоподобного отрицания — весьма ценное свойство, при корректировке протокола его желательно сохранить. Представим протокол SIGMA в следующем виде.

$I \rightarrow R$: s, ID_I, Sig_I(“1”, s, g^x, g^y,), MAC(g^{xy}, “1”, s, ID_I, ID_R).
$R \rightarrow I$: s, ID_R, Sig_R(“0”, s, g^y, g^x,), MAC(g^{xy}, “0”, s, ID_R, ID_I).

Итак, пользователи, как и прежде, не обязаны явно указывать свои имена в сообщениях, однако для правдоподобного отрицания необходимо, чтобы они *явно* верифицировали свои имена, содержащиеся в кодах аутентификации сообщений.

Отметим, что возможность отрицания связи (denying-of-connection-feature) не является безупречной, поскольку партнер (назовем его “предателем”), сохранивший сеансовый ключ g^{xy}, в дальнейшем может выдать третьей стороне доказательства своего контакта с аутентифицированным пользователем. Очевидно, что это вполне возможно, поскольку предатель может использовать для верификации те же самые операции, которые выполнялись при аутентификации сторон. Вот почему возможность отрицания сопровождается эпитетом “правдоподобное” (plausible).

В разделе 13.3.5 будут описаны новые практичные криптографические конструкции, которые гарантируют аутентификацию с возможностью полного отрицания.

12.2.5 Критика протоколов IPSec и IKE

Основными недостатками протоколов IPSec и IKE являются их высокая сложность и запутанность. Они обладают слишком многими возможностями и предоставляют слишком гибкие механизмы. Одну и ту же операцию в рамках этого протокола можно выполнить разными способами. Кауфман (Kaufman) подсчитал количество передач криптографических сообщений в рамках IKE: одна обязательная и 806 399 возможных [197]. Из-за высокой сложности протокола его спецификация становится чрезвычайно запутанной. В ней трудно разобраться даже экспертам. Вследствие этого ее недостатки могут остаться незамеченными.

Фергюсон и Шнайер считают, что эта сложность является следствием коллективной разработки протокола [108]. Как известно, комитеты включают в протоколы различные свойства, возможности и дополнительные механизмы, стремясь достичь компромисса между разными группировками. Если даже обычным (функциональным) стандартам дополнительная сложность наносит существенный вред, то на стандарты систем безопасности она и вовсе оказывает разрушающее воздействие.

Пагубные последствия высокой сложности этих протоколов и слишком большого количества возможностей, заложенных в них, проявляются не столько в том, что их трудно понять или правильно запрограммировать, сколько в опасности самих механизмов, предусмотренных в протоколах. В разделе 12.2.3.4 описан типичный сценарий, в котором Злоумышленник организует эффективную атаку на агрессивный режим протокола IKE, основанного на цифровой подписи.

Рассмотрим еще один пример, взятый из работы [12]. В этой статье, опубликованной в марте 2000 года, протоколы IPSec и IKE описываются с разных точек зрения, начиная с общей концепции защиты сетей и заканчивая детальным описанием протоколов. В приведенной ниже цитате описывается возможность аутентификации с помощью инкапсулированных зашифрованных данных (ESP). Напомним, что ESP — это зашифрованная порция информации, содержащая секретные данные, передаваемые в IP-пакете (см. раздел 12.2.2.2).

> Поле аутентификации в зашифрованном тексте ESP содержит контрольное значение ICV (integrity check value), позволяющее проверить целостность данных. Оно, по существу, представляет собой цифровую подпись, вычисленную по остальной части зашифрованного текста ESP, которая остается после удаления из него поля аутентификации. Его длина зависит от применяемого алгоритма аутентификации. Ес-

ли в зашифрованном тексте ESP алгоритм аутентификации не указан, поле аутентификации можно пропустить.

Итак, в протоколе предусмотрена возможность вообще игнорировать защиту целостности данных, содержащихся в зашифрованном тексте. Как было показано в разделе 11.7.8 и еще не раз будет продемонстрировано в последующих главах, шифрование без защиты целостности данных (т.е. то, что в цитате называется "аутентификацией") весьма опасно. Большинство алгоритмов шифрования не могут обеспечить секретность, если не гарантирована целостность данных. Итак, даже через четыре года недостатки протокола IPSec, обнаруженные Белловиным в 1996 году (см. раздел 12.2.3.5), еще не были устранены (напомним, что цитируемая нами статья опубликована в марте 2000!). Единственной разумной причиной этого может быть лишь слишком высокая сложность протокола.

Айелло (Aiello) с соавторами [10] критиковали протокол IKE за сложность вычислений и обмена сообщениями. Он полагает, что протокол IKE уязвим для атаки с помощью отказа в обслуживании: Злоумышленник и его сообщники, распределенные в Internet, могут послать много запросов, содержащих огромное количество строк, которые должен запомнить сервер ("cookies"). Айелло с соавторами предложили протокол JFK ("Just Fast Keing" — "Моментальное кодирование"), который должен был стать преемником протокола IKE. Блейз (Blaze) привел следующий шутливый аргумент, обосновывающий название протокола JFK.

> Как известно, президента Эйзенхауэра звали Ike (Айк), а его сторонники использовали каламбур "I like Ike". Мы не любим протокол IKE, потому что являемся сторонниками его преемника. По этой причине мы изобрели название "Just Fast Keing", первые буквы которого являются инициалами Джона Кеннеди — преемника Эйзенхауэра на посту президента. Мы никогда не будем обсуждать наш протокол в Далласе. Если когда-либо группа IETF снова соберется в Далласе[7], мы ни словом не обмолвимся о нашем протоколе.

12.3 Протокол удаленной регистрации SSH

Протокол SSH (Secure Shell — оболочка безопасности) [304–308] — это набор протоколов аутентификации с открытым ключом, позволяющий пользователю, работающему на клиентской машине, безопасно регистрироваться на удаленном сервере через ненадежную сеть, выполнять команды на удаленном сервере и перемещать файлы с одного компьютера на другой. Этот протокол фактически стал промышленным стандартом и широко применяется на компьютерах, работающих под управлением систем UNIX или LINUX. Клиентская часть протокола может

[7] 34-я конференция IETF состоялась в декабре 1995 года в г. Даллас, штат Техас.

использоваться на компьютерах, работающих под управлением разнообразных операционных систем. Причина, по которой этот протокол работает в основном на серверах под управлением операционной системы UNIX, заключается в том, что открытая архитектура таких серверов позволяет организовывать интерактивные сеансы связи с удаленными пользователями.

Основная идея протокола SSH состоит в том, что пользователь, работающий на клиентской машине, должен загрузить с удаленного сервера открытый ключ и установить с его помощью защищенный канал, используя криптографический мандат. Представим себе, что криптографическим мандатом пользователя является его пароль: он может шифроваться с помощью полученного открытого ключа и передаваться на сервер. Этот протокол представляет собой большой шаг вперед в обеспечении безопасности по сравнению в протоколами аутентификации, в которых используется обычный пароль.

12.3.1 Архитектура протокола SSH

Протокол SSH выполняется между двумя ненадежными компьютерами, работающими в незащищенной сети. Один из них называется удаленным сервером (головным компьютером), а другой — клиентом. Пользователь, работающий на клиентском компьютере, должен зарегистрироваться на удаленном сервере, используя протокол SSH.

Набор протоколов SSH состоит из трех компонентов.

- Протокол транспортного уровня SSH (SSH Transport Layer Protocol) [308], обеспечивающий аутентификацию сервера. Этот протокол использует открытый ключ. Исходной информацией для этого протокола как со стороны сервера, так и со стороны клиента, является пара открытых ключей, называемая "ключом головного компьютера" ("host key"). Результатом протокола является взаимно аутентифицированный защищенный канал, гарантирующий секретность и целостность данных, *направленный от сервера к клиенту*. Этот протокол обычно выполняется на основе протокола управления передачей данных (Transport Control Protocol — TCP) и протокола Internet (Internet Protocol — IP), однако может использоваться поверх любых других надежных потоков данных.
- Протокол аутентификации пользователя SSH (SSH User Authentication Protocol) [305]. Этот протокол выполняется по каналу односторонней аутентификации, установленному протоколом транспортного уровня SSH. Он поддерживает работу различных протоколов односторонней аутентификации для того, чтобы выполнить аутентификацию сущности в направлении *от клиента к серверу*. Для того чтобы аутентификация в этом направлении стала возможной, удаленный сервер должен иметь априорную информацию о криптографическом мандате пользователя, т.е. пользователь должен быть изве-

стен серверу. Эти протоколы могут применять либо открытый ключ, либо пароль. Например, они могут быть созданы на основе протокола аутентификации с помощью простого пароля (протокол 11.3). Результатом выполнения протокола аутентификации пользователя SSH является взаимно аутентифицированный защищенный канал между сервером и пользователем.

- Протокол связи SSH (SSH Connection Protocol) [306]. Этот протокол выполняется по взаимно аутентифицированному защищенному каналу, установленному двумя предыдущими протоколами. Он обеспечивает работу защищенного канала и разделяет его на несколько защищенных логических каналов, используя стандартные методы организации интерактивных сеансов.

Очевидно, что протокол связи SSH не является протоколом аутентификации и выходит за рамки наших интересов, а протокол аутентификации пользователя SSH можно рассматривать как совокупность стандартных протоколов односторонней аутентификации, описанный в главе 11 (см. также раздел 12.3.4). Итак, необходимо описать лишь протокол транспортного уровня SSH.

12.3.2 Протокол транспортного уровня SSH

В основу новой версии протокола транспортного уровня SSH [307, 308] положен протокол обмена ключами Диффи–Хеллмана, обеспечивающий одностороннюю аутентификацию в направлении от сервера к клиенту на основе цифровой подписи параметров ключа.

12.3.2.1 Пары ключей сервера

Каждый сервер владеет парой, состоящей из открытого и закрытого ключа. Сервер может иметь несколько пар ключей и применять несколько разных алгоритмов. При этом сервер должен использовать как минимум одну пару ключей, применяя требуемый алгоритм с открытым ключом. В будущем [307] в качестве алгоритма с открытым ключом предлагается алгоритм DSS (Digital Signature Standard, см. раздел 10.4.8.2). Однако в текущей версии протокола по умолчанию используется алгоритм RSA (раздел 10.4.2).

Для обмена используется пара серверных ключей (открытый, закрытый): закрытый ключ используется сервером для подписания параметров ключа, а открытый ключ применяется клиентом для верификации того, что связь установлена с подлинным сервером. Для этого клиент должен иметь априорную информацию об открытом ключе сервера.

Протокол SSH поддерживает две модели открытого ключа сервера.

- Клиент поддерживает локальную базу данных, в которой имени каждого сервера поставлена в соответствие открытая часть серверного ключа. Этот метод не требует централизованного управления инфраструктурой открытого ключа, описанной в главе 13. Эта модель имеет один недостаток: в неко-

торых случаях пользователю бывает сложно поддерживать базу данных, содержащую записи (имя_сервера, открытый_ключ_сервера). Пример практичного метода, с помощью которого пользователь может получить аутентифицированную копию открытого ключа сервера, описан в разделе 12.3.2.2.

- Пара (имя_сервера, открытый_ключ_сервера) сертифицируется службой сертификации (certification authority — CA) с помощью методов, описанных в главе 13. Для того чтобы верифицировать корректность всех открытых ключей сервера, клиенту достаточно знать лишь открытый ключ службы сертификации.

Вторая модель упрощает поддержку базы данных, поскольку в идеале клиенту достаточно хранить единственный открытый ключ службы сертификации в тайне (здесь секретность подразумевает целостность данных). С другой стороны, каждый открытый ключ сервера перед аутентификацией должен быть сертифицирован службой сертификации. Кроме того, приходится возлагать большую долю ответственности на центральную инфраструктуру.

Поскольку в настоящее время в Internet еще не существует широко развернутой инфраструктуры с открытым ключом (Public Key Infrastructure — PKI, глава 13), первая модель оказывается намного полезнее, обеспечивая намного более высокую секретность, чем все предыдущие решения (например, сеансовые команды операционной системы UNIX: rlogin, rsh, rftp и т.п.).

12.3.2.2 Практичные методы аутентификации открытого ключа сервера

Один из практичных способов аутентификации заключается в том, что пользователь должен иметь аутентифицированную копию открытого ключа сервера еще до старта протокола обмена ключами. Например, если пользователь путешествует, он может носить с собой дискету, на которой записан открытый ключ сервера. В текущей версии протокола SHH [304], предназначенной для клиентских машин, работающих под управлением операционной системы UNIX, открытый ключ сервера хранится в файле $HOME/.shh/known_hosts. Пользователь должен обеспечить физическую защиту открытого ключа сервера (например, хранить его на дискете во время путешествия) и гарантировать целостность данных. Если клиентский компьютер работает под управлением операционной системы Windows, открытый ключ сервера может существовать только в оперативной памяти компьютера и должен загружаться с сервера в режиме реального времени через незащищенный канал связи вместе со своим “слепком”.

Другой практичный способ аутентификации копии открытого ключа сервера, загруженной через незащищенную линию связи, — аутентифицировать его по телефону. Сначала открытый ключ сервера загружается пользователем на клиентский компьютер через незащищенную линию связи. Затем пользователю демон-

стрируется шестнадцатеричный "слепок" ключа, имеющий следующий вид.

$$\text{"слепок"}\ (\text{ключ сервера}) = H(\text{ключ сервера}),$$

где H — согласованная функция хэширования, например, SHA-1. Если для аутентификации открытого ключа сервера используется хэш-функция SHA-1, его "слепок" состоит из 160 бит. Следовательно, этот "слепок" можно продиктовать по телефону, назвав 40 шестнадцатеричных цифр. Итак, пользователь может позвонить на удаленный сервер и сверить "слепок" ключа у системного администратора. Если копия ключа, вычисленная на клиентской машине, совпадет со "слепком", хранящимся у системного администратора, ключ считается аутентичным.

В данном случае пользователь, работающий на клиентской машине, и администратор, управляющий удаленным сервером, должны знать голос абонента. Следовательно, такую безопасность нельзя назвать полной. Однако, пока инфраструктура PKI еще не получила широкого распространения в Internet, это — неплохое решение.

12.3.2.3 Протокол обмена ключами

Обмен ключами всегда инициируется клиентом, а сервер прослушивает конкретный порт, ожидая сеанса связи. При этом сервер может обслуживать несколько клиентов одновременно.

В новой версии протокола SSH [307, 308] для согласования сеансового ключа применяется протокол обмена ключами Диффи–Хеллмана (раздел 8.3). При описании протокола используются следующие обозначения:

- C: клиент;
- S: сервер;
- p: большое безопасное простое число;
- g: порождающий элемент подгруппы G_q поля $GF(p)$;
- q: порядок подгруппы G_p;
- V_C, V_S: версии протокола, принадлежащие клиенту и серверу соответственно;
- K_S: открытый ключ сервера S;
- I_C, I_S: первоначальное сообщение протокола обмена ключами, модифицированное перед началом протокола со стороны клиента и сервера соответственно.

Протокол обмена ключами выглядит следующим образом.

1. Клиент C генерирует случайное число $x(1 < x < q)$, вычисляет значение

$$e \leftarrow g^x (\mathrm{mod}\, p)$$

и отсылает его серверу S.

2. Сервер S генерирует случайное число $y(0 < y < q)$, вычисляет значение

$$f \leftarrow g^y (\mathrm{mod}\, p),$$

получает число e, вычисляет значения

$$K \leftarrow e^y (\mathrm{mod}\, p),$$
$$H \leftarrow hash(V_C \parallel V_S \parallel I_C \parallel I_S \parallel K_S \parallel e \parallel f \parallel K \parallel),$$
$$s \leftarrow \mathrm{Sig}_S(H)$$

и отсылает число $K_S \parallel f \parallel s$ клиенту C.

3. Клиент C убеждается, что число K_S действительно является ключом сервера S (используя любой подходящий метод, например, с помощью сертификата или проверенной локальной базы или метода, описанного в разделе 12.3.2.2). Затем клиент C находит числа

$$K \leftarrow f^x (\mathrm{mod}\, p),$$
$$H \leftarrow hash(V_C \parallel V_S \parallel I_C \parallel I_S \parallel K_S \parallel e \parallel f \parallel K \parallel)$$

и проверяет подпись s по числу H. Если аутентификация прошла успешно, клиент C принимает обмен ключами.

После обмена ключами все сообщения, которыми обмениваются обе стороны, будут зашифрованы с помощью согласованного сеансового ключа K. Затем обе стороны приступают к выполнению протокола аутентификации пользователя SSH [305]. После этого клиент может послать запрос на выполнение протокола связи SSH [306].

12.3.3 Стратегия SSH

Одна из целей протокола SSH — повысить защиту информации в Internet с помощью современных методов. Разрешение клиенту использовать "любой подходящий метод" (например, описанный в разделе 12.3.2.2) для верификации открытого ключа сервера ясно демонстрирует стратегию SSH, направленную на быстрое распространение и поддержку обратной совместимости.

Как только инфраструктура с открытым ключом получит широкое распространение в Internet, необходимость в повышенной секретности отпадет, однако безопасность Internet с такой инфраструктурой намного выше, чем без нее. Легкость применения и быстрое распространение являются неоспоримым преимуществом стратегии SSH. По этой причине она получает все большую популярность и широко используется на серверах, работающих под управлением операционных систем UNIX и Linux.

Анализ протокола SSH демонстрирует, что криптография с открытым ключом существенно облегчает решение проблемы. Ключ сервера в незащищенной среде (например, на клиентской машине или в линии связи, соединяющей сервер с клиентом) существует только в открытой форме. Это намного облегчает управление ключами. Если бы протокол использовал методы криптографии с секретным ключом, проблема стала бы значительно сложнее.

12.3.4 Предостережения

В заключение приведем несколько предостережений, касающихся криптографического мандата, используемого в протоколе аутентификации пользователя. Этот мандат может быть основан на применении открытого ключа, пароля или секретного аппаратного средства идентификации. Он применяется в протоколе, выполняемом на клиентской машине, которая рассматривается как часть незащищенной среды.

В текущей версии протокола SSH [304] криптографический мандат представляет собой закрытый ключ, соответствующий открытому ключу пользователя, шифруется с помощью его пароля и хранится на клиентской машине в файле `$HOME/.ssh/identity` (если клиентская машина работает под управлением операционных систем UNIX или Linux). Этот файл считывается во время выполнения протокола на клиентской машине, предлагающего пользователю ввести пароль. Естественно, пользователь должен убедиться, что часть протокола, выполняемая на клиентской машине, является подлинной. Чтобы минимизировать риск применения закрытого ключа, скомпрометированного в ходе автономной атаки, в конце работы пользователь должен удалить файл `$HOME/.ssh/identity`, содержащий зашифрованный закрытый ключ.

Наиболее надежным считается механизм, основанный на применении секретного аппаратного средства идентификации. Он использует небольшое устройство, в окошке которого отображается несколько цифр. В ходе синхронизации устройства с головным компьютером эти цифры изменяются и считаются паролем пользователя, известным серверу. Разумеется, поскольку это устройство невелико, пользователь должен тщательно беречь его и немедленно сообщать о потере.

12.4 Протокол Kerberos и его реализация в операционной системе Windows 2000

Предположим, что Алиса работает в транснациональной компании, и в ее распоряжении имеются различные информационные источники и службы. Например, со своего "домашнего сервера" Алиса может получить доступ к обычным сетевым службам (например, World Wide Web, электронной почте и т.п.). На "сервере

проекта" Алиса и члены ее команды распоряжаются информацией, относящейся к их работе. На "сервере кадровых ресурсов" Алиса может управлять информацией о кадрах, например, определять, какой процент ее заработной платы в следующем месяце следует инвестировать в покупку акций. Если Алиса работает менеджером, ей необходимо отслеживать записи в базе данных. На "сервере интеллектуальной собственности" Алиса может работать со своим изобретением. На "сервере финансовых затрат" Алиса может планировать затраты на поездки. Нетрудно вообразить множество других услуг.

В корпоративной среде пользователи (сотрудники или клиенты) обычно имеют доступ к информации, распределенной по сети. Эти услуги предоставляются разными подразделениями. В результате разные информационные серверы могут функционировать в разных географических регионах. С точки зрения сетевых технологий эти серверы представляют собой разные **сетевые домены** (network domains). Для секретного использования этих доменов (все примеры, перечисленные в предыдущем абзаце, связаны с использованием конфиденциальной информации) пользователь должен иметь разные мандаты, которые должны аутентифицироваться сервером. Однако было бы нереально требовать от пользователя хранить несколько криптографических мандатов, представляющих собой пароли или смарт-карты.

Для решения проблемы аутентификации сетевых пользователей предназначен протокол Kerberos [168, 202]. Его основная идея заключается в использовании доверенной третьей стороны, предоставляющей пользователю доступ к серверу с помощью общего сеансового ключа, разделенного между пользователем и сервером. Идея протокола Kerberos была выдвинута Нидхемом и Шредером [213] и воплощена в протоколе аутентификации Нидхема–Шредера (протокол 2.4). Поскольку исходный вариант протокола Нидхема–Шредера имеет недостатки (раздел 2.6.4.2), в основу протокола Kerberos был положен вариант протокола Нидхема–Шредера с использованием временной метки.

Представим теперь, что в протоколе 2.4 Алиса является пользователем, разделившим долговременный секретный ключ с доверенной третьей стороной (Трентом). Кроме того, Боб в рамках этого протокола играет роль сервера, также разделяющего общий долговременный секретный ключ с доверенной третьей стороной. Если Алиса хочет воспользоваться услугами Боба, она инициирует протокол связи с Трентом и запрашивает у него криптографический мандат для доступа к Бобу. Трент может предоставлять мандаты ("ticket service"), генерируя сеансовый ключ, разделенный между Алисой и Бобом. Трент скрытно доставляет этот ключ, спрятанный внутри двух мандатов, шифруя их долговременными секретными ключами, которые он разделяет с Алисой и Бобом. Именно в этом заключается основная идея протокола Kerberos.

Протокол Kerberos (основанный на версии 5 [168]) положен в основу аутентификации пользователей операционной системы Windows 2000, широко применяющейся в корпоративных сетях.

Протокол Kerberos создан в рамках проекта Athena в Массачусетском технологическом институте (Massachusetts Institute of Technology — MIT) для обеспечения безопасности в сетевых средах. Институт MIT опубликовал пятую версию Kerberos в виде открытого кода, который можно загрузить с Web-сайта института MIT <http://web.mit.edu/kerberos/www/>. Однако из-за ограничений на экспорт криптографических продуктов, введенных правительством США, на момент написания книги протокол Kerberos был доступен только гражданам США, живущим в США, и гражданам Канады, проживающим в Канаде.

Протокол Kerberos Version 5 ненамного сложнее протокола аутентификации Нидхема–Шредера на основе временно́й метки.

12.4.1 Архитектура однократной регистрации

Протокол аутентификации Kerberos состоит из трех частичных протоколов, называемых **обменами.**[8]

1. Обмен сообщениями со службой аутентификация (Authentication Service Exchange — AS Exchange). Этот обмен выполняется между клиентом C и сервером аутентификации AS.
2. Обмен сообщениями со службой выдачи мандата (Ticket-Granting Service Exchange — TGS Exchange). Этот обмен осуществляется между клиентом C и службой выдачи мандатов TGS после выполнения аутентификации.
3. Обмен сообщениями со службой клиент-серверной аутентификации приложений (Client/Server — AP Exchange Authentication Application Exchange). Этот обмен осуществляется между клиентом C и сервером приложений S после выдачи мандата.

Каждый из этих трех обменов представляет собой протокол обмена двумя сообщениями. Эти обмены связаны между собой так, как показано на рис. 12.4.1.1.

В трех обменах протокола Kerberos участвуют пять пользователей, играющих следующие роли.

- U: пользователь (человек), выполняющий протокол на своей клиентской машине. Таким образом, пользователь U появляется в протоколе только в виде сообщения. Для работы в системе Kerberos каждый пользователь должен помнить свой пароль, представляющий собой мандат однократной регистрации (single-signon credential).

[8] На самом деле протокол Kerberos состоит из намного большего числа частичных протоколов, предназначенных для решения специализированных задач, например, для обмена паролями, обновления мандатов, обработки ошибок и тому подобное, однако к аутентификации относятся только три указанных частичных протокола.

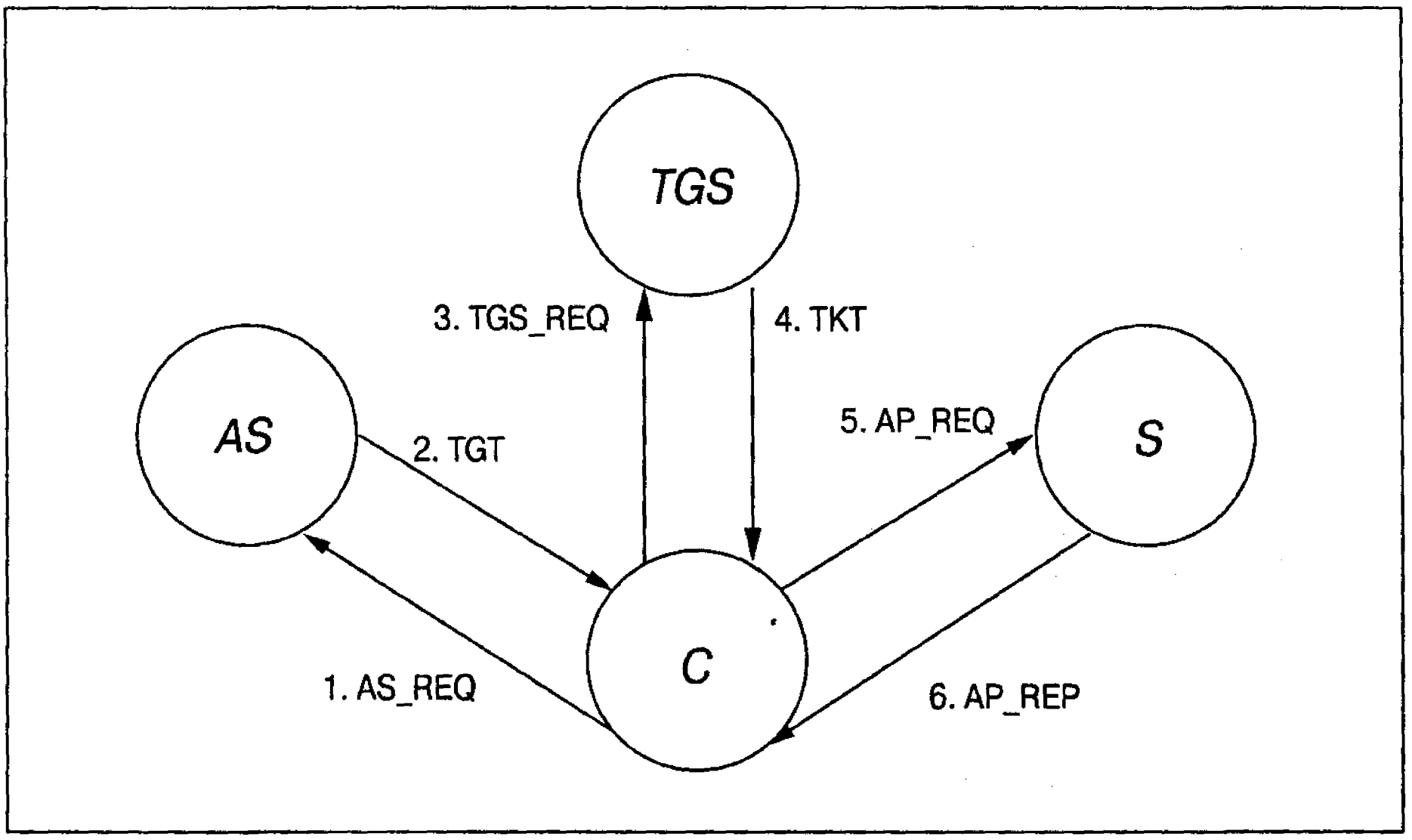


Рис. 12.4. Обмены в рамках протокола Kerberos

- C: клиент (процесс), предоставляющий пользователю сетевые услуги. Для того чтобы запустить процесс C на клиентской машине в рамках аутентификации, пользователь U должен иметь мандат системы Kerberos. Этот мандат вводится пользователем U в виде пароля.
- S: сервер приложений (процесс), предоставляющий ресурсы клиенту C. В рамках клиент-серверной аутентификации приложений он получает от клиента C запрос на выполнение приложения (application request – AP_REQ). В свою очередь сервер приложений пересылает ответ (application reply – AP_REP), открывающий клиенту C доступ к приложениям.
- Ответ AP_REP содержит мандат клиента C (ticket – TKT), который в свою очередь содержит сеансовый ключ приложения $K_{C,S}$, временно разделенный между клиентом C и сервером S.
- KDC: центр распределения ключей (Key Distribution Center). Это коллективное название двух серверов аутентификации.
  - AS: сервер аутентификации (Authentication Server). В рамках аутентификации он получает от клиента C исходный текст запроса на аутентификацию (Authentication Service Request — AS_REQ) и возвращает клиенту мандат на получение мандата (ticket granting ticket — TGT), который в дальнейшем клиент C может использовать для получения мандата в рамках протокола TGS Exchange.

Сервер аутентификации разделяет пароль с каждым обслуживаемым пользователем. Этот пароль устанавливается с помощью средств однократной регистрации за пределами протокола Kerberos.

Мандат на получение мандата, предоставляемый клиенту C, состоит из двух частей. Одна часть предназначена для клиента и шифруется с помощью ключа, построенного на основе пароля однократной регистрации. Другая часть предназначена для службы выдачи мандатов TGS и шифруется с помощью долговременного ключа, разделенного между сервером аутентификации и сервером TGS. Обе части мандата на получение мандата содержат сеансовый ключ мандата $K_{C,TGS}$, разделенный между клиентом C и службой выдачи мандатов.

— TGS: служба выдачи мандатов (Ticket Granting Server — TGS). Она получает запрос на получение мандата (ticket granting request — TGS_REQ), содержащий мандат на получение мандата TGT, принадлежащий клиенту C. В ответ она возвращает мандат TKT, который используется клиентом C в процессе аутентификации приложений при обмене с сервером S.

  Как и мандат TGT, мандат TKT состоит из двух частей. Одна из его частей предназначена для клиента C и шифруется с помощью сеансового ключа $K_{C,TGS}$, распределенного между клиентом C и сервером TGS. Другая часть предназначена для сервера приложений S и шифруется долговременным ключом $K_{S,TGS}$, разделенным между серверами S и TGS.

  Обе части мандата TKT содержат новый сеансовый ключ приложения $K_{C,S}$, разделяемый между клиентом C и сервером S. Сеансовый ключ приложения (application session key) представляет собой криптографический мандат клиента C, который используется для аутентификации приложений при последующем обмене с сервером S для получения доступа к его ресурсам.

12.4.1.1 Зачем центр распределения ключей KDS разделяется на две части: сервер приложений AS и службу выдачи мандатов TGS?

В будущем мы убедимся, что роли серверов AS и TGS очень похожи, поэтому они объединяются под общим названием: центр распределения ключей (KDS).

Причина, по которой центр распределения ключей разделяется на две части, заключается в том, что сеть, в которой он работает, может быть очень большой. В таких сетях серверы приложений и службы выдачи мандатов принадлежат разным сетевым доменам. Таким образом, даже если фиксированный пользователь U должен пройти однократную регистрацию на фиксированном сервере аутенти-

фикации AS, он может обслуживаться многими службами выдачи мандатов и, следовательно, многими серверами приложений.

12.4.2 Обмены протокола Kerberos

Опишем каждый из трех обменов, образующих протокол Kerberos. Чтобы упростить изложение, будем рассматривать только необходимые протокольные сообщения. Полное описание всех протокольных сообщений, включая огромное количество необязательных сообщений, читатель найдет в работе [168].

12.4.2.1 Обмен сообщениями со службой аутентификации

Этот обмен происходит только между клиентом C и сервером аутентификации AS.

1. AS_REQ $C \rightarrow AS$: U, TGS, Срок_действия_1, N_1.
2. TGT $AS \rightarrow C$: U, $T_{C,TGS}, TGT_C$.

Здесь

$$T_{C,TGS} = \{U, C, TGS, K_{C,TGS}, \text{Начало, Конец}\}_{K_{AS,TGS}},$$

$$TGT_C = \{TGS, K_{C,TGS}, \text{Начало, Конец}, N_1\}_{K_U}.$$

Пользователь U генерирует сообщение 1. Используя исходное сообщение AS_REQ, клиент C информирует сервер аутентификации AS о том, что пользователь U желает установить связь со службой выдачи мандатов TGS. В запрос включается информация о сроке действия мандата и одноразовое случайное число N_1 (идентификатор "свежести").

В ответ сервер аутентификации AS генерирует новый сеансовый ключ $K_{C,\,TGS}$, разделяемый между клиентом C и сервером TGS. Затем он шифруется с помощью сеансового ключа внутри мандата на получение мандата TGT и возвращается клиенту C в виде сообщения 2.

Одной частью мандата TGT является мандат $T_{C,TGS}$, предназначенный для службы выдачи мандатов. Он шифруется с помощью долговременного ключа $K_{AS,\,TGS}$ и разделяется между сервером аутентификации и службой выдачи мандатов. Другой частью мандата TGT является мандат T_C, шифрующийся с помощью пароля пользователя K_U.

Получив сообщение 2, клиент C может расшифровать мандат T_C, предложив пользователю U ввести пароль K_U. Если проверка прошла успешно, клиент C получает сеансовый ключ $K_{C,TGS}$ и мандат $T_{C,TGS}$. Теперь клиент C имеет мандат на получение мандата, предназначенный для предъявления серверу TGS.

12.4.2.2 Обмен сообщениями со службой выдачи мандатов

Формат обмена сообщениями со службой выдачи мандатов похож на формат обмена сообщениями с сервером аутентификации, за исключением того, что запрос клиента TGS_REQ теперь содержит **аутентификатор** (autheticator), сопровождающий исходный текст сообщения.

3. TGS_REQ $C \rightarrow TGS$: S, Срок_действия_2, N_2, $T_{C,TGS}$, $A_{C,TGS}$.
4. TKT $TGS \rightarrow C : U, T_{C,S}, TKT_C$,

Здесь

$$T_{C,S} = \{U, C, S, K_{C,S}, \text{Начало, Конец}\}_{K_{S,TGS}},$$

$$TKT_C = \{S, K_{C,S}, \text{Начало, Конец}, N_2\}_{K_{C,TGS}},$$

$$A_{C,TGS} = \{C, \text{Время\_клиента}\}_{K_{C,TGS}}.$$

Предназначение этой пары сообщений и действия участников протокола объясняются так же, как и при описании обмена сообщениями со службой аутентификации. Единственный дополнительный элемент — аутентификатор $A_{C,TGS}$. Он предназначен для того, чтобы продемонстрировать службе выдачи мандатов TGS, что клиент C в момент Время_клиента применил сеансовый ключ $K_{C,TGS}$. Мандат TGS должен сверить локальное время и подтвердить, что разница между ним и временем клиента не превышает допустимой величины.

12.4.2.3 Обмен сообщениями с сервером приложений

В ходе обмена сообщениями с сервером приложений клиент C использует вновь полученный сеансовый ключ приложения $K_{C,S}$ и мандат $T_{C,S}$, чтобы получить доступ к ресурсам сервера приложений S.

5. AP_REQ $C \rightarrow S$: $T_{C,S}$, $A_{C,S}$.
6. AP_REP $S \rightarrow C$: $A_{S,C}$.

Здесь

$$A_{C,S} = \{C, \text{Время\_клиента1}\}_{K_{C,S}},$$

$$A_{S,S} = \{\text{Время\_клиента1}\}_{K_{C,S}}.$$

Смысл этих сообщений очевиден.

12.4.3 Предостережения

Необходимо сделать два предостережения.

Во-первых, во время расшифровки сообщений в ходе протокола Kerberos необходимо выполнять дополнительную проверку.

Когда участник протокола расшифровывает мандат, необходимо провести проверку идентификатора "свежести" и имен пользователей. Однако, что не совсем очевидно, необходимо также убедиться в целостности данных. Важность верификации целостности данных уже не раз упоминалась в предыдущих главах (например, в разделе 11.7.8) и будет исследована в разделе 17.2.1.

Это предупреждение относится ко всем видам шифрования в ходе обмена сообщениями протокола Kerberos.

Второе предупреждение касается аутентификатора.

На первый взгляд, имя аутентификатора, его положение и применение (в качестве приложения к мандату) свидетельствуют о том, что он играет роль кода аутентификации сообщений (MAC, раздел 10.3), гарантирующего защиту целостности данных в мандате (например, может показаться, что аутентификатор $A_{C,TGS}$ защищает мандат $T_{C,TGS}$). На самом деле защита отсутствует.

Итак, необходимо предусмотреть отдельный механизм для защиты целостности данных (например, с помощью кода аутентификации сообщения MAC). При этом следует иметь в виду следующее: для создания аутентификатора нельзя применять шифрование. Для того чтобы предотвратить преднамеренное искажение времени клиента в аутентификаторе, необходимо защитить блок шифра в самом аутентификаторе!

Это предостережение относится ко всем аутентификаторам, применяемым в протоколе Kerberos.

12.5 Протоколы SSL и TLS

Для обеспечения безопасности в сети World Wide Web используется протокол защищенных сокетов (Secure Sockets Layer Protocol — SSL) [111, 136]. Термин "сокет" обозначает стандартный канал связи, соединяющий процессы, протекающие в сети (например, клиентов и сервер). Протокол защищенных сокетов выполняется в рамках протоколов на уровне приложений, например, в рамках протокола передачи гипертекстовых файлов (Hypertext Transfer Protocol — HTTP), облегченного протокола службы каталогов (Lightweight Directory Access Protocol — LDAP) или протокола доступа к сообщениям в сети Internet (Internet Messaging Access Protocol — IMAP), а также поверх протоколов сетевого уровня, таких как протокол управления передачей данных (Transport Control Protocol — TCP) и Internet-протокола (Internet Protocol — IP). Если соединения на уровне сокетов защищены (например, в смысле секретности и целостности данных), эта защита распространяется на все соединения в рамках протоколов на уровне приложений.

Протокол SSL обычно применяется для защиты сообщений при передаче через Internet. Этот протокол был разработан корпорацией Netscape Communications как составная часть Web-браузера и Web-сервера. Позднее он был одобрен корпо-

рацией Microsoft и другими разработчиками клиент-серверных приложений для Internet и был фактическим стандартом вплоть до появления протокола транспортного уровня (Transport Layer Security — TLS) [95], разработанного проблемной группой проектирования Internet (Internet Engineering Task Force — IETF).

Протокол SSL основан на протоколе TLS и не очень сильно отличается от него. Однако, поскольку протокол TLS является преемником протокола SSL и стандартом безопасности в World Wide Web, в дальнейшем при описании протокола защиты данных в Internet мы будем иметь в виду именно протокол TLS.

12.5.1 Архитектура протокола TLS

Протокол TLS состоит из двух протоколов: **протокола записи TLS** (TLS Record Protocol) и **протокола квитирования TLS** (TLS Handshake Protocol). Второй протокол выполняется поверх первого.

Протокол записи TLS обеспечивает безопасную инкапсуляцию канала связи для применения в рамках протокола более высокого уровня приложений. Этот протокол запускается поверх протоколов TCP и IP и обеспечивает надежный сеанс связи. Он распределяет данные по блокам, сжимает их при необходимости, применяет код аутентификации сообщения (HMAC, см. раздел 10.3.2) для защиты целостности данных, шифрует сообщение с помощью симметричного алгоритма и передает результат адресату. Адресат получает зашифрованные блоки данных, расшифровывает их, верифицирует код MAC, разархивирует их при необходимости, собирает блоки в одно целое и доставляет результат на более высокий уровень приложений.

Ключи для симметричного шифрования и код HMAC генерируются для каждого сеанса связи отдельно и основаны на секрете, согласованном с протоколом квитирования TLS.

Протокол квитирования TLS позволяет серверу и клиенту аутентифицировать друг друга, согласовывать криптографические алгоритмы и ключи, устанавливая таким образом безопасный сеанс связи с протоколом записи TLS для защиты соединений на более высоком уровне приложений.

Из этого описания ясно, что протокол записи TLS не является протоколом аутентификации, хотя он и обеспечивает секретное соединение. Следовательно, достаточно описать протокол квитирования TLS.

12.5.2 Протокол квитирования TLS

Протокол квитирования TLS изменяет параметры своего состояния в процессе выполнения на клиентской машине и сервере. Такое соединение называется **сеансом** (session), в котором участники выполняют следующие шаги.

- Обмениваются приветствиями, согласовывая алгоритм, пересылают друг другу случайные числа и проверяют возможность возобновления сеанса.
- Обмениваются необходимыми криптографическими параметрами, позволяющими клиенту и серверу согласовать секрет (так называемый "главный секрет").
- Обмениваются сертификатами и криптографической информацией, позволяющей клиенту и серверу аутентифицировать друг друга.
- Генерируют сеансовые секреты на основе главного секрета, обмениваясь случайными числами.
- Убеждаются, что их партнер вычислил те же самые параметры безопасности, и квитирование выполнено успешно и не подверглось атаке.
- Установленный защищенный канал передается протоколу записи TSL для обработки на более высоком уровне приложений.

Эти этапы реализуются с помощью четырех обменов, описанных ниже. Для того чтобы лучше изложить основную идею протокола, мы остановимся на упрощенной версии протокола квитирования TLS, опуская необязательные элементы. В описании протокола символ C обозначает клиента (т.е. клиентскую сторону Web-браузера), а символ S — Web-сервер. Необязательные сообщения сопровождаются звездочкой.

1. $C \rightarrow S$: ClientHello;
2. $S \rightarrow C$: ServerHello,
   ServerSertification*,
   ServerKeyExchange*,
   CertificateRequest*,
   ServerHelloDone;
3. $C \rightarrow S$: ClientCetificate;
   ClientKeyExchange,
   CertificateVerify*,
   ClientFinished;
4. $S \rightarrow C$: ServerFinished.

Этот протокол можно выполнять, опуская все необязательные сообщения, а также сообщение ClientKeyExchange. В этом случае клиент может возобновить существующий сеанс.

Рассмотрим обмен сообщениями, возникающий при выполнении протокола квитирования TLS.

12.5.2.1 Обмен приветствиями

Сеанс начинается с того момента, когда клиент посылает сообщение ClientHello, на которое сервер должен ответить сообщением ServerHello. В противном

случае связь прерывается. Эти два сообщения заполняют следующие поля: protocol_version ("версия_протокола"), random ("случайные_числа"), session_id ("номер_сеанса"), cipher_suites ("элементы_шифра") и compression_message ("методы_сжатия").

Поле protocol_version обладает свойством обратной совместимости: сервер и клиент могут использовать это поле для того, чтобы информировать своего партнера о версии используемого протокола.

Поле random содержит одноразовые случайные числа (идентификаторы "свежести"). Партнеры генерируют эти числа для последующего обмена. Кроме того, это поле содержит локальное время установления связи каждой из сторон.

Поле session_id идентифицирует текущее сеансовое соединение. В начале нового сеанса связи поле ClientHello.session_id должно оставаться пустым. В этом случае сервер генерирует новый номер сеанса, записывает его в поле ServerHello.session_id и сохраняет в локальной кэш-памяти. Если поле ClientHello.session_id содержит какое-то число (например, когда клиент желает возобновить текущий сеанс), сервер должен попытаться найти номер сеанса в локальной кэш-памяти и возобновить указанный сеанс.

Рассмотрим теперь поле cipher_suites. Поле ClientHello.cipher_suites представляет собой список криптографических опций, поддерживаемых на клиентской машине и упорядоченных в соответствии с приоритетами клиента. Клиент может предложить серверу широкий круг криптографический алгоритмов, как симметричных, так и с открытым ключом, кодов аутентификации сообщений и функций хэширования. Для каждой криптографической операции сервер выбирает единственную схему и сообщает о ней клиенту, используя поле ClientHello.cipher_suites.

12.5.2.2 Сертификат сервера и материал для обмена ключами

После обмена приветствиями сервер может послать свой сертификат, подлежащий аутентификации. Если сообщение ServerCertificate не является пустым, оно содержит список сертификатов X.509.v3 (см. раздел 13.2). Сертификат X.509 содержит информацию об имени и открытом ключе владельца сертификата, а также об источнике сертификата (см. пример 13.1). Получив список сертификатов, клиент может выбрать алгоритм с открытым ключом, поддерживаемый клиентской машиной.

Вслед за сообщением ServerCertificate пересылается сообщение ServerKeyExchange. Оно содержит элементы открытого ключа, соответствующего списку сертификата в сообщении ServerCertificate. Элементы ключа Диффи–Хеллмана включаются в тройку (p, g, g^y), где p — простой модуль, g — порождающий элемент большой группы по модулю p, а y — целое число, записанное в кэш-памяти сервера и связанное с полем "номер_сеанса".

В дальнейшем сервер, обеспечивающий неанонимные услуги, с помощью сообщения CertificateRequest может запросить у клиента сертификат, согласованный с полем ClientHello.cipher_suite.

После этого сервер посылает сообщение ServerHelloDone, означающее завершение обмена приветствиями, и переходит в режим ожидания.

12.5.2.3 Ответ клиента

Получив сообщение CertificateRequest, клиент должен в ответ отослать либо сообщение ClientCertificate, либо предупреждение NoCertificate.

Затем клиент пересылает серверу сообщение ClientKeyExchange. Содержание этого сообщения зависит от выбранного алгоритма с открытым ключом, согласованным путем обмена сообщениями ClientHello и ServerHello.

Если для обмена ключами выбран алгоритм RSA, клиент генерирует "главный секрет" (число, состоящее из 48 бит) и шифрует его с помощью сертифицированного открытого ключа RSA, принадлежащего серверу и содержащегося в сообщении ServerCertificate.

Если клиент способен подписывать сертификаты, он должен отослать серверу сообщение CertificateVerify, снабженное цифровой подписью. Это позволяет серверу явно верифицировать сертификат клиента.

12.5.2.4 Обмен заключительными сообщениями

На этом этапе клиент посылает серверу сообщение ClientFinished, содержащее код HMAC, зашифрованный с помощью "главного секрета". Это дает серверу возможность подтвердить квитирование, выполненное клиентом.

В ответ сервер пересылает клиенту сообщение ServerFinished, которое также содержит код HMAC, позволяющий клиенту подтвердить квитирование, выполненное сервером.

Таким образом, квитирование считается завершенным, а клиент и сервер могут переходить к обмену данных на уровне приложений.

12.5.3 Выполнение протокола квитирования TLS

Рассмотрим обычное течение протокола квитирования TLS.

Поскольку в ходе выполнения этого протокола клиент предпочитает оставаться анонимным и не аутентифицироваться сервером, он выбирает для шифрования алгоритм RSA, а для вычисления кода HMAC — алгоритм SHA-1. В результате происходит односторонняя аутентификация — сервер доказывает свою подлинность клиенту. Итогом протокола становится установка одностороннего аутентифицированного канала связи от сервера к клиенту.

Такой ход протокола TLS является типичным для средств электронной коммерции в WWW, например, при покупке книг в электронных магазинах. Уста-

Протокол 12.2. Выполнение протокола квитирования TLS

1. $C \rightarrow S$: ClientHello.protocol_version="TLS Version 1.0",
   ClientHello.random=T_C, N_C,
   ClientHello.session_id="NULL",
   ClientHello.crypto_suite="RSA: encryption, SHA-1: HMAC".
   ClientHello.compression_method="NULL";
2. $S \rightarrow C$: ServerHello.protocol_version="TLS Version 1.0",
   ServerHello.random=T_S, N_S,
   ServerHello.session_id="xyz123",
   ServerHello.crypto_suite="RSA: encryption, SHA-1: HMAC",
   ServerHello.compression_method="NULL";
   ServerCertificate=point_to(сертификат сервера),
   ServerHelloDone;
3. $C \rightarrow S$: ClientKeyExchange=point_to(RSA_Encryption(master_secret)),
   ClientFinished=SHA_1(master_secret $\|$ C $\|$, N_C, N_S, ...);
4. $S \rightarrow C$: ServerFinished=SHA_1(master_secret $\|$ C $\|$, N_S, N_C, ...).

новленный канал гарантирует клиенту, что его инструкции будет получать только аутентифицированный сервер. Это очень важно, поскольку эти инструкции могут содержать конфиденциальную информацию о клиенте, например, данные о банковской карточке, название книги и адрес доставки.

12.5.4 Атака на приложение TLS с помощью обходного канала

В ходе **атаки с помощью обходного канала** (side channel attack) Злоумышленник пытается найти скрытую информацию, непреднамеренно разглашенную пользователем. Одной из разновидностей атаки с помощью обходного канала является **атака на основе временно́го анализа** (timing analysis attack). В данном случае Злоумышленник наблюдает и анализирует реакцию пользователя на получаемые от него запросы. Первые примеры атаки с помощью обходного канала и атаки на основе временно́го анализа были изобретены Кочером (Kocher) [167]. Он атаковал систему, выполнявшую возведение в степень по модулю. Как известно, эта операция выполняется в ходе подписания сообщений, при расшифровке сообщений с помощью алгоритма RSA, а также при создании эфемерных ключей в семействе схем цифровой подписи Эль-Гамаля в рамках протокола обмена ключами Диффи–Хеллмана. Атака направлена на раскрытие секретного показателя степени. Возведение в степень использует метод возведения в квадрат с последующим умножением и побитовой обработкой показателя (см. алгоритм 4.3).

Каждый единичный бит в показателе возводится в квадрат и умножается, а каждый нулевой бит — только возводится в квадрат. Целью атаки является измерение временно́й разницы между этими двумя вариантами. В случае успеха Злоумышленник сможет определить биты показателя один за другим.

Недавно Кэнвел с соавторами [68] разработал атаку с помощью обходного канала (на основе временно́го анализа) на линии связи между сервером и клиентом, защищенные протоколом TLS/SSL. Как правило, целью такой атаки является пароль пользователя для доступа к серверу электронной почты IMAP. В ходе атаки атакованный пароль пересылается с клиентской машины на сервер электронной почты, в то время как линия связи между клиентом и сервером защищена протоколом TLS. Сообщения, передаваемые по защищенной линии связи, шифруются с помощью стойкого сеансового ключа в соответствии с протоколом TLS (см. протокол 12.2). Шифрование сеанса связи осуществляется с помощью стойкого блочного шифра (например, тройного алгоритма DES) в режиме CBC (см. раздел 7.8.2).

Атака на основе временно́го анализа использует атаку с помощью взрывающегося оракула, изобретенную Воденэ для взлома стандартной схемы CBC с заполнением (CDC padding scheme) [294], описанной в разделе 7.8.2.1. Вкратце эта атака выглядит следующим образом. Допустим, что C — это блок зашифрованного текста в режиме CBC, шифрующий пароль и перехваченный Злоумышленником. В ходе атаки Воденэ на стандартную схему CBC с заполнением Злоумышленник посылает оракулу расшифровки следующую информацию:

$$r, C,$$

где r — блок(и) случайных чисел. Затем Злоумышленник ждет от оракула расшифровки ответа, правильно или неправильно выполнено заполнение. Если оракул отвечает, что заполнение было выполнено правильно, последний байт исходного текста, зашифрованного с помощью блока C, оказывается раскрытым (если блок C используется для шифрования паролей, Злоумышленнику станет известен последний символ пароля). Теперь мы можем описать атаку на основе временно́го анализа, позволяющую взломать линию связи, защищенную паролем TLS.

В ходе этой атаки Злоумышленник посылает серверу электронной почты информацию r, S, заявляя, что он является владельцем атакуемого пароля, зашифрованного с помощью шифра C. Получив сообщение r, S, сервер выполняет расшифровку в режиме CBC и проверяет правильность заполнения. Если заполнение было выполнено правильно (вероятность этого события равна 2^{-8} (см. раздел 7.8.2.1)), он проверяет целостность данных, заново вычисляя код аутентификации сообщений (см. раздел 10.3.3). Если заполнение было выполнено неправильно, проверку целостности данных выполнять необязательно. В любом случае ошибка зашифровывается с помощью стойкого сеансового ключа TLS и возвращается на клиентскую машину.

На первый взгляд, Злоумышленник, не знающий стойкого сеансового ключа, не имеет доступа к услугам оракула, т.е. сервер электронной почты, зашифровавший и передавший ошибку, не является оракулом расшифровки.

Однако, если число r является случайным, а схема заполнения в режиме CBC — правильной, то в подавляющем числе случаев целостность данных будет нарушена. Следовательно, атакованный сервер электронной почты на самом деле имеет только две возможности:

1) отослать обратно сообщение $\{\text{“неправильное заполнение”}\}_K$ с вероятностью $\approx 1 - 2^{-8}$ или
2) отослать обратно сообщение $\{\text{“неправильный код MAC”}\}_K$ с вероятностью $\approx 2^{-8}$.

Второй вариант означает, что заполнение было выполнено правильно, и Злоумышленник узнал последний байт исходного сообщения, зашифрованного шифром C.

Теперь наступает очередь применить временно́й анализ! Для достаточно большого числа r (состоящего из нескольких блоков) сервер во втором случае должен заново вычислить длинный код MAC в режиме CBC, в то время как в первом варианте такие вычисления не выполняются. Кэнвел и его соавторы [68] обнаружили, что на стандартных серверах разница во времени между получением сообщения и ответом составляет несколько миллисекунд. Следовательно, в рамках временно́го анализа сервер играет роль оракула расшифровки. Обратите внимание на то, что обработка ошибок в данном случае означает, что оракул расшифровки весьма надежен и никогда не “взорвется”!

Изменив число r вручную, т.е. не применяя шифр C, Злоумышленник может раскрыть пароль байт за байтом, начиная с конца. Способ изменения числа r читатели могут изобрести сами (см. подсказку к упражнению 12.12). Если шифр C шифрует пароль, состоящий из 8 байт, для раскрытия всего пароля понадобится $8 \times 2^8 = 2048$ попыток доступа к серверу электронной почты.

Эта атака весьма эффективна, хотя она успешнее работает в локальных сетях (LAN), в которых клиент и сервер находятся в одной сети, так что разность между временами получения запроса и ответом можно вычислить точнее. Кроме того, эта атака демонстрирует, что услуги оракула иногда можно получить косвенным образом, с помощью обходного канала. И наконец, она свидетельствует о том, что обработка ошибок должна проводиться очень аккуратно.

Предотвратить эту атаку можно, если перед отправкой ответа сервер будет делать паузы, имеющие случайную продолжительность.

12.6 Резюме

В главе описаны четыре протокола аутентификации, пригодные для применения в реальных приложениях. К ним относятся протокол IKE, принятый органи-

зацией IETF в качестве стандартного протокола IPSec, протокол SSH, ставший фактическим стандартом аутентификации для защиты интерактивных сеансов, протокол Kerberos — промышленный стандарт операционных систем семейства Windows для защиты информации в корпоративных сетях и информационных хранилищах, а также протокол TLS (SSL), ставший фактическим стандартом в области защиты данных в World Wide Web.

Несмотря на то что все протоколы описаны в очень упрощенном виде, мы убедились, что они достаточно сложны с технической точки зрения. Эти сложности возникают в результате требований, предъявляемых к алгоритмам реальными приложениями. В частности, они должны обеспечивать согласование алгоритмов и параметров, легко распространяться на широкий круг систем, обладать свойством обратной совместимости, быль удобными в работе и т.д. Для протоколов IPSec и IKE необходимость повышенной безопасности данных вынуждает создавать очень сложные системы. В главе показано, что разработка протоколов аутентификации в реальных приложениях сопряжена не только с решением проблем безопасности, но и с преодолением инженерных трудностей. Недостаточное внимание к решению технических проблем может вызвать серьезные последствия для безопасности данных.

В главе показано, что протоколы аутентификации, предназначенные для реальных систем, чрезвычайно уязвимы. По этой причине мы не завершаем обсуждение этой темы и вернемся к ней в главе 17, посвященной методам формального анализа.

Упражнения

12.1. Предположим, что соединения в Internet не защищены протоколом IPSec. Каким образом Злоумышленник может манипулировать сообщениями, передаваемыми через Internet (например, имитировать автора, перенаправлять сообщение и т.п.)?

12.2. Какую роль играет заголовок аутентификации (AH) в протоколе IPSec?

12.3. Как связаны между собой протоколы IPSec и IKE?

12.4. Назовите два способа криптографической защиты IP-пакета.

12.5. В упражнении 11.15 рассмотрены способы исправления несущественного дефекта протокола STS без ущерба для анонимности пользователей. Предложите свой способ исправления несущественного дефекта первой фазы основного режима протокола IKE с помощью цифровой подписи без ущерба для возможности отказа от авторства.

12.6. Продемонстрируйте "совершенную атаку с помощью отказа в обслуживании" на первую фазу агрессивного режима протокола IKE на основе цифровой подписи.

Подсказка: эта атака очень похожа на атаку 11.3.

12.7. Протоколы EKE и SSH шифруют пароли с помощью асимметричных алгоритмов. Однако между ними существует значительная разница. В чем она заключается?

12.8. Как пользователь может аутентифицировать сервер в рамках протокола SSH?

12.9. Почему в протоколе Kerberos каждому клиенту должно соответствовать три разных вида серверов?

12.10. Чем протокол Kerberos удобен для применения в корпоративных сетях?

12.11. Протоколы TLS (SSL) широко применяются в электронной коммерции, осуществляемой через Internet. Действительно ли эти протоколы подходят для таких приложений? Если нет, объясните почему?

Подсказка: эти протоколы не поддерживают авторизацию оплат с возможностью отказа.

12.12. В разделе 12.5.4 описана атака на основе временно́го анализа, позволяющая распознать последний байт сообщения, зашифрованного с помощью блочного шифра в режиме CBC, использующего стандартную схему заполнения. Как распознать остальные байты?

Подсказка: проанализируйте стандартную схему заполнения исходного текста в режиме CBC, описанную в разделе 7.8.2.1. Для того чтобы распознать остальные байты, следует рассмотреть следующий вариант "правильного заполнения": два завершающих байта ("два заполнителя") равны '02' || '02', и модифицировать последний байт числа r так, чтобы вероятность этого варианта была максимальной.

Глава 13

Аутентификация в криптографии с открытым ключом

13.1 Введение

Процедура генерации ключа в криптографии с открытым ключом обязательно содержит операцию

$$\text{открытый\_ключ} = F(\text{закрытый\_ключ}). \tag{13.1.1}$$

Здесь F — эффективная и однонаправленная функция, отображающая пространство закрытых ключей в пространство открытых ключей. Вследствие однонаправленности функции F, обеспечивающей перемешивание сообщений, часть открытого ключа, вычисленного по закрытому, всегда выглядит случайной.

Поскольку часть любого открытого ключа выглядит случайной, необходимо, чтобы он содержал информацию об имени пользователя, которую можно было бы верифицировать. Очевидно, чтобы послать секретное сообщение, зашифрованное открытым ключом, отправитель должен удостовериться, что применяемый внешне случайный открытый ключ действительно принадлежит подлинному получателю. Аналогично, чтобы идентифицировать источник сообщения с помощью цифровой подписи, получатель должен убедиться, что открытый ключ, используемый при верификации подписи, действительно принадлежит подлинному отправителю.

Для того чтобы применять криптографию с открытым ключом в реальных приложениях, необходим механизм, позволяющий легко распознавать связь между открытым ключом и именем пользователя. Такой механизм, как правило, реализуется с помощью аутентификации: он позволяет аутентифицировать владельца открытого ключа.

13.1.1 Структурная схема главы

В главе описываются два способа аутентификации, применяемой в криптографии с открытым ключом: **инфраструктура сертификации открытого ключа**

(public key certification infrastructure — PKI) (раздел 13.2) и **личностная криптография с открытым ключом** (identity-based public-key cryptography) (раздел 13.3).

13.2 Системы аутентификации с помощью службы каталогов

Если два пользователя часто общаются друг с другом, необходимо, чтобы они могли легко узнавать открытый ключ партнера: сначала они могут скрытно обмениваться своими открытыми ключами, например, при личной встрече, а затем также скрытно хранить их. Однако такой простой метод **управления ключами** (key management) трудно применить в системах с большим количеством пользователей. В открытых сетях пользователи могут быть совершенно незнакомыми друг с другом, причем контакты между ними в большинстве случаев оказываются *одноразовыми*. "Простой" метод управления ключами предъявляет к пользователям совершенно нереальные требования и заставляет их хранить огромное количество открытых ключей. Более того, этот метод сводит на нет преимущества самой криптографии с открытым ключом.

В разделе 2.4 описан интерактивный метод управления секретными ключами с помощью доверенного посредника. Этот метод включает в себя регистрацию ключа, аутентификацию и хранение имени пользователя в каталоге. Для того чтобы прибегнуть к услугам посредника, каждый пользователь должен сначала установить однозначную и долговременную связь с доверенным сервером (сервером аутентификации), разделив с ним долговременный секретный ключ. Когда два пользователя захотят вступить в контакт друг с другом, они могут запустить протокол аутентификации и установить защищенный канал связи. Таким образом, каждый конечный пользователь должен управлять только одним секретным ключом, разделенным с сервером аутентификации. Методы управления ключами и способы аутентификации, описанные в главе 2, предназначены для создания протоколов аутентификации, основанных на применении криптосистем с секретными ключами. (Это относится и к протоколу аутентификации открытых ключей Нидхема–Шредера, рассмотренному в разделе 2.6.6, поскольку он пользуется услугами интерактивного доверенного посредника, связь с которым осуществляется в секретном режиме.)

Систему управления секретными ключами можно без труда распространить на открытые ключи. В этом случае она называется **службой сертификации открытых ключей** (public-key certification service), а доверенный сервер называется **органом сертификации** (certification authority — CA). Орган сертификации — это особый пользователь, пользующийся доверием не только всех пользователей обслуживаемого им домена, но и пользователей более широкого круга. (Термин "доверие" будет уточнен позднее.) Для каждого конечного пользователя в домене

орган сертификации выпускает **сертификат открытого ключа** (public-key certificate), представляющий собой структурированную запись, поля которой содержат уникальную информацию об имени пользователя и параметрах его открытого ключа. Сертификат сопровождается цифровой подписью органа сертификации. Таким образом осуществляется связь между именем владельца и его открытым ключом. Пользователь, проверивший подлинность сертификата другого пользователя, должен доверять ему, если он доверяет органу сертификации. В этом случае пользователь, выполняющий верификацию, устанавливает секретный **канал для обмена ключами** (key channel), соединяющий владельца сертифицированного открытого ключа с системой. Термин "сертификат открытого ключа" впервые был предложен Конфельдером (Kohnfelder) в работе [169].

Канал для обмена открытыми ключами, использующий механизм сертификации, часто называют каналом, основанным на каталогах (directory-based channel), а саму сертификацию — службой каталогов.

Обратите внимание на то, что в отличие от доверенного посредника, используемого сервером аутентификации в протоколах аутентификации секретных ключей (раздел 2.4), требования к органу сертификации намного ниже. Он должен лишь выполнять аутентификацию сообщений, которую можно провести, не раскрывая никаких секретов, поскольку для проверки подлинности цифровой подписи органа сертификации нужен только открытый ключ самого органа сертификации. Следовательно, эту услугу можно осуществлять автономно (off-line), и орган сертификации не обязан быть участником протокола связи между конечными пользователями. Важной особенностью автономной обработки является то, что ее легко применять в больших системах. Очевидно, открытый ключ органа сертификации, применяемый для проверки подлинности сертификатов, выпускаемых органом сертификации, в свою очередь должен быть сертифицирован другим органом сертификации и т.д.

Поля данных в сертификате должны содержать информацию об органе сертификации и параметрах его открытого ключа. Кроме того, в них должна быть записана дополнительная информация, например, описание алгоритма, применяемого для верификации цифровой подписи органа сертификации, данных, используемых сертифицированным открытым ключом, срок действия, условия применения и т.п. Неформальное описание сертификата открытого ключа приведено в примере 13.1.

Пример 13.1 (Сертификат открытого ключа).

```
certificate ::=
    {
     issuer name;
     issuer information;
     subject name;
     subject information;
```

```
    validity period;
   }
issuer information ::=
   {
    issuer public key;
    signature algorithm identifier;
    hash function identifier
   }
subject information ::=
   {
    subject public key;
    public key algorithm identifier
   }
validity period ::=
   {
    start date;
    finish date;
   }
```

□

13.2.1 Выпуск сертификатов

Выпуская сертификат, орган сертификации должен проверить подлинность пользователя, приславшего запрос. Процесс верификации, естественно, связан с определенными некриптографическими средствами идентификации, которые выполняются, например, при открытии счета в банке. Пользователь должен также доказать, что ему известен закрытый компонент сертифицируемого открытого ключа. Это доказательство может быть предъявлено либо в виде цифровой подписи на запросе, подлинность которого проверяется с помощью открытого ключа, либо с помощью протокола доказательства с нулевым разглашением, выполняемого между пользователем и органом сертификации. В некоторых приложениях закрытый компонент открытого ключа должен иметь заданную структуру. В таких приложениях для проверки структуры открытого ключа применяется протокол доказательства с нулевым разглашением. Несколько примеров таких протоколов будут рассмотрены в последующих главах.

13.2.2 Аннулирование сертификатов

Иногда сертификат необходимо аннулировать, например, если закрытый ключ пользователя был скомпрометирован или пользователь сменил имя.

Если для сертификации используется служба каталогов, корневой орган сертификации должен поддерживать список аннулированных сертификатов, доступный

в интерактивном режиме. В качестве альтернативы корневой орган сертификации может разослать пользователям системы "Δ-список аннулированных сертификатов", содержащий лишь недавно аннулированные сертификаты. Получив этот список, пользователи системы смогут обновить свои локальные копии списка аннулированных сертификатов.

Аннулирование сертификата должно сопровождаться временно́й меткой. Подписи пользователя, поставленные до этой даты, должны считаться подлинными, даже если верификация подписи произошла после аннулирования сертификата.

13.2.3 Примеры систем аутентификации открытого ключа

Рассмотрим несколько примеров систем аутентификации открытых ключей с помощью службы каталогов.

13.2.3.1 Инфраструктура сертификации открытого ключа X.509

Стандартная инфраструктура сертификации открытого ключа X.509 [159] лежит в основе **дерева информационных каталогов** (directory information tree — DIT). В этой иерархии каждый узел представляет пользователя, чей сертификат открытого ключа был выпущен органом сертификации, которому соответствует ближайший родительский узел. Листья этого дерева изображают конечных пользователей. Остальные узлы иерархии представляют собой органы сертификации, принадлежащие разным уровням и доменам. Например, на уровне страны органами сертификации могут быть домены промышленных или правительственных организаций. Каждый из этих доменов может содержать множество поддоменов (например, домен системы образования может содержать поддомены разных университетов). Корень дерева называется **корневым органом сертификации** (root CA). Он должен быть хорошо известен всем пользователям системы. Корневой орган сертификации должен сертифицировать свой собственный открытый ключ. Поскольку каждый орган сертификации потенциально может обслуживать большой домен, содержащий другие органы сертификации и конечных пользователей, глубина дерева информационных каталогов не обязательно должна быть большой. Два конечных пользователя могут установить скрытый канал связи друг с другом, обратившись к органу сертификации, расположенному в ближайшем родительском узле.

13.2.3.2 Система PGP "Паутина доверия"

Другая система аутентификации открытых ключей, получившая широкую популярность, называется системой PGP "Паутина доверия" ("Web of Trust"), или "Кольцо для ключей" ("Key-ring"). Аббревиатура PGP означает "Pretty Good Privacy" — набор алгоритмов и программ для высоконадежного шифрования элек-

тронных сообщений с использованием открытых ключей, разработанный Циммерманом (Zimmermann) [312]. Эта модель аутентификации масштабируется без помощи иерархических структур. В системе PGP "Паутина доверия" любой пользователь может играть роль органа сертификации по отношению к любым пользователям системы, выпуская "сертификат ключа", представляющий собой пару ⟨имя, ключ⟩. Очевидно, что такой способ сертификации образует паутину. Любой отдельный орган сертификации может вызывать лишь частичное доверие либо не вызывать доверия вообще. Теоретически большое количество подписей должно вызывать доверие к сертификату ⟨имя, ключ⟩, поскольку не все подписи могут быть поддельными. Таким образом, если Алиса хочет установить аутентичность ключа Боба, она должна запросить все его сертификаты. Если некоторые из них "известны" Алисе, она может частично аутентифицировать открытый ключ Боба. Если Алиса хочет повысить уровень своего доверия к Бобу, она должна потребовать от него дополнительные сертификаты.

13.2.3.3 Простая инфраструктура открытых ключей (SPKI)

Систему сертификации открытых ключей X.509 можно представить в виде глобальной интерактивной телефонной книги. Каждому пользователю соответствует отдельная запись. Значит, имя субъекта (subject name) в сертификате пользователя (см. пример 13.1) должно быть уникальным. Такая система аутентификации на ранних этапах развития криптографии с открытым ключом была вполне адекватной: соединения были защищены от прослушивания, а получатели секретной информации должны были однозначно идентифицировать друг друга.

Начиная с 1990-х годов, криптография с открытым ключом получила широкое распространение в области электронной коммерции и обеспечения удаленного доступа (см. предисловие). Эллисон (Ellison) и его соавторы [103] считают, что в новых приложениях использование уникального имени, связанного с ключом, уже не обеспечивает требуемого уровня безопасности. Получив сертификат открытого ключа, приложение должно определить, имеет ли удаленный пользователь право доступа к ресурсам. Для того чтобы ответить на этот вопрос, одного имени пользователя недостаточно. Следовательно, в сертификате открытого ключа, кроме имени пользователя, должна содержаться информация о его правах.

Эллисон с соавторами также полагают, что система X.509 вряд ли вообще пригодна для решения этой задачи. Набор записей (например, списки сотрудников, клиентов, контактов и т.п.) представляет ценность лишь для его владельца и не должен распространяться по всему миру, образуя поддеревья системы X.509. Чтобы довести ситуацию до абсурда, авторы предложили читателям вообразить, что ЦРУ (Центральное разведывательное управление США) разместило в системе X.509 список своих агентов. Идея об уникальном имени, позволяющем однозначно идентифицировать пользователя, также неудачна. Она требует от пользователей немалой дисциплины, причем многие уже существующие участники рынка име-

ют имена, которые совершенно не подчиняются правилам системы X.509. Таким образом, идея, лежащая в основе X.509, входит в противоречие с существующим положением дел.

Эллисон с соавторами предложили новую систему сертификации открытых ключей с помощью службы каталогов, получившую название **SPKI** (Simple Public Key Infrastructure — Простая инфраструктура открытых ключей). [103]. Она, как и система сертификации открытых ключей X.509, также имеет древовидную структуру. Однако соглашение об именах в этой системе предполагает, что сертификат содержит обычное имя пользователя и хэшированное значение открытого ключа. Например, в системе SPKI запись

```
(name (hash sha1 |TLCgPLF1GTzgUbcaYLW8kGTEnUK=|)
      jim therese)
```

представляет собой имя пользователя Jim Therese. Применение алгоритма хэширования SHA-1 к открытому ключу делает имя уникальным, даже если в мире существует множество людей, которых зовут Jim Therese.

Этот способ образования имен был предложен Ривестом (Rivest) и Лампсоном (Lampson) при разработке системы **SDSI** [245] (A Simple Distributed Security Insfrastructure — Простая распределенная инфраструктура безопасности). Система SDSI предназначена для локализации правил образования имен. В частности, это подразумевает децентрализацию систем аутентификации и авторизации. Таким образом, система SPKI является разновидностью системы SDSI.

В систему SPKI входят записи об авторизации пользователя и делегировании его прав. Запись об авторизации содержит описание авторизации, связанное с открытым ключом. Таким образом, сертификат может явно продемонстрировать приложению, является ли отправитель запроса авторизованным пользователем и имеет ли он право на выполнение определенных действий. Информация о делегировании описывает право пользователя передавать авторизацию другому лицу. Можно сказать, что система SPKI расширяет систему аутентификации X.509, дополняя ее возможностью авторизации и делегирования прав. В основе схемы авторизации в системе SPKI лежит применение S-выражений, напоминающих выражения в языке программирования LISP. S-выражения были предложены Ривестом в работе [244]. Например, S-выражение

```
(object document (attributes (name *.doc) (loc Belgium))
        (op read) (principals (users OrgEU)))
```

позволяет всем авторизованным пользователям системы OrgEU читать объекты типа document, имеющие расширение doc и расположенные в Бельгии.

Альтернативный вариант, предполагающий авторизацию и правила аутентификации пользователей, был предложен в работе [40] и получил название Ро-

liceMaker. Эта система предназначена для описания роли владельца сертификата и составления правил для каждой роли.

13.2.4 Протоколы, связанные с инфраструктурой аутентификации открытого ключа X.509

Существует несколько протоколов, связанных с инфраструктурой аутентификации открытого ключа X.509.

- Протокол управления сертификатами (Certificate Management Protocol — CMP) [7, 208] поддерживает интерактивные соединения между компонентами инфраструктуры открытых ключей (Public Key Infrastructure — PKI). Например, этот протокол можно использовать для связи между органом сертификации и клиентской системой, с которой ассоциирована пара ключей, а также между двумя органами сертификации, обеспечивающими перекрестную взаимную сертификацию. Эти связи необходимы, например, когда сущность (пользователь или орган сертификации) должна доказать свое владение закрытым ключом при сертификации ключа или его обновлении.
- Интерактивный протокол управления статусом сертификатов (Online Certificate Status Protocol — OCSP) [207] позволяет приложениям определять состояние идентифицируемого сертификата (т.е. проверять, не аннулирован ли он). Этот протокол можно использовать для оперативной проверки информации об аннулированных сертификатах и получения дополнительной информации о статусе сертификата. Клиент протокола OCSP посылает запрос о статусе сертификата и ожидает ответа.
- Протокол инфраструктуры Internet X.509 с открытым ключом и временно́й меткой (Internet X.509 Public-Key Infrastructure Time Stamp Protocol) [6] содержит запрос, посылаемый службе меток времени (Time Stamping Authority — TSA), и полученный ответ. Этот протокол предъявляет к операциям, выполняемым службой TSA, некоторые дополнительные требования, касающиеся ответов на полученные запросы. Служба, обеспечивающая невозможность отказа от авторства, должна обеспечивать возможность устанавливать существование данных до указанного времени. Этот протокол можно использовать в качестве строительного блока при создании более сложных конструкций.
- Оперативные протоколы инфраструктуры Internet X.509 с открытым ключом (Internet X.509 Public-Key Infrastructure Operational Protocol): FTP и HTTP [140]. Эта спецификация протоколов предусматривает применение протоколов передачи файлов (File Transfer Protocol — FTP) и гипертекстовых файлов (Hypertext Transfer Protocol — HTTP) для получения сертификатов и списков

аннулированных сертификатов (certificate revocation list — CRL) из хранилищ системы PKI.

Эти протоколы были разработаны группой PKIX Working Group (Public-Key Infrastructure X.509 Working Group), входящей в организацию IETF, в качестве стандартов. Детальное описание этих протоколов выходит за рамки нашей книги. Читатели, заинтересовавшиеся этой темой, могут загрузить описание протоколов с Web-сайта группы PKIX Working Group

`http://www.ietf.org/html.charters/pkix-charter.html`.

13.3 Системы аутентификации открытых ключей без помощи службы каталогов

Процедура формирования ключей, описанная формулой (13.1.1), как правило, создает случайные открытые ключи. Следовательно, необходимо установить однозначное соответствие между открытым ключом и именем его владельца. Как показано выше, такую связь можно установить с помощью иерархической системы сертификации открытых ключей (например, системы X.509). Однако необходимость создавать и поддерживать иерархическую структуру значительно усложняет и удорожает систему PKI. Следовательно, стандартную систему аутентификации открытых ключей желательно упростить.

Естественно предположить, что если бы открытые ключи не выглядели как случайный набор цифр, сложность и стоимость организации и поддержки системы аутентификации открытых ключей можно было бы снизить. Представьте себе, что открытый ключ пользователя был бы очевидным образом связан с его именем, местом работы, электронным и почтовым адресом, так что, по существу, аутентификация стала бы излишней. Именно по такому принципу действуют обычные почтовые службы.

Первая работа в этом направлении была выполнена Шамиром (Shamir) [260]. Он смог существенно упростить систему аутентификации ключей: по существу, он воспроизвел работу обычной почтовой системы. Это применение криптосистемы с открытым ключом весьма нетипично. Генерация ключа в этой системе выглядит следующим образом.

$$\text{закрытый\_ключ} = F(\text{главный\_ключ}, \text{открытый\_ключ}) \qquad (13.3.1)$$

Эта формула противоположна формуле создания ключа (13.3.1) в криптографии с открытым ключом в обычном смысле. Разумеется, для того, чтобы сохранить найденный закрытый ключ в тайне, вычисления должны быть секретными: они должны выполняться привилегированным пользователем — доверенным органом (trusted authority — TA). Для этого доверенный орган должен владеть секрет-

ным главным ключом. В таком случае открытый ключ поступает на вход процедуры генерирования ключа, причем в качестве открытого ключа может использоваться любая строка битов! Шамир предложил использовать в качестве открытого ключа имя пользователя, поскольку это существенно упрощает аутентификацию. Благодаря этому криптосистема Шамира называется **личностной криптографией с открытым ключом** (identity-based public-key cryptography).

Очевидно, что процедура генерации ключа (13.3.1) должна выполняться доверенным органом для всех пользователей системы. По существу, эта процедура является аутентификацией: закрытый ключ, создаваемый доверенным органом для пользователя, в сочетании с его идентификатором, используемым в качестве открытого ключа, обеспечивает признание открытого ключа всеми пользователями системы. Перед созданием закрытого ключа доверенный орган должен тщательно проверить информацию, удостоверяющую пользователя. Эта проверка должна включать в себя определенные физические (некриптографические) средства идентификации. Кроме того, доверенный орган должен убедиться, что информация, удостоверяющая пользователя, позволяет однозначно идентифицировать его. Аналогичную проверку должен проводить и орган сертификации, прежде чем выдавать пользователю сертификат открытого ключа (раздел 13.2.1).

Пользователи, для которых доверенный орган генерирует закрытые ключи, должны доверять ему полностью и безусловно: они не должны беспокоиться, что доверенный орган может читать всю их закрытую переписку или фальсифицировать их подписи. Следовательно, личностная криптография пригодна лишь для приложений, в которых пользователи могут безусловно доверять друг другу. Предположим, что в некоей организации каждый сотрудник полностью владеет информацией, которой владеют и обмениваются другие пользователи. В этом случае сотрудники могут сами играть роль доверенного органа. Однако возможна ситуация, когда доверенный орган представляет собой набор сущностей, коллективно вычисляющих закрытый ключ (13.3.1) для пользователя. Это открывает возможность для массового вмешательства в частную переписку. В таких системах должен применяться принцип коллективного доверия, описанный в разделе 13.3.7.1.

Если в качестве открытого ключа используется уникальная информация, идентифицирующая пользователя, в криптографической системе, основанной на идентификации, не нужен канал для обмена ключами, изображенный на рис. 7.1 и 10.1. Более того, ключи ke на рис. 7.1 и kv на рис. 10.1 можно заменить строкой, содержащей очевидную информацию, например, глобально распознаваемое имя.

13.3.1 Личностная схема цифровой подписи Шамира

В личностной схеме цифровой подписи Шамира применяются четыре алгоритма.

- Setup: этот алгоритм выполняется доверенным органом (с этого момента мы будем называть его Трентом) и предназначен для формирования глобальных параметров системы и главного ключа.
- User-key-generate: этот алгоритм также выполняется Трентом. На его вход поступает главный ключ и произвольная строка битов $\mathsf{id} \in \{0,1\}^*$, а его результатом является закрытый ключ, соответствующий идентификатору id. Этот алгоритм является конкретизацией формулы (13.3.1).
- Sign: алгоритм генерации подписи. На вход этого алгоритма поступает сообщение и закрытый ключ подписывающего лица, а его результатом является цифровая подпись.
- Verify: алгоритм верификации подписи. На вход этого алгоритма поступает пара "сообщение-подпись" и идентификатор id, а его результатом является значение True или False.

Личностная схема цифровой подписи Шамира представлена в алгоритме 13.1.

Если результатом верификации подписи является значение True, значит, Алиса владеет как числом $\mathsf{ID} \cdot t^{h(t\|M)}$, так и его *единственным* корнем e-й степени по модулю N, равным числу s. Единственность корня гарантируется условием $\gcd(e, \phi(N)) = 1$.

Вычисление значения $\mathsf{ID} \cdot t^{h(t\|M)}$ не обязательно должно быть сложным. Например, можно выбрать случайное число t, вычислить сначала значение $h(t \parallel M)$, а затем $t^{h(t\|M)} (\mathrm{mod}\, M)$ и умножить его на идентификатор ID. Однако, поскольку благодаря применению криптографической функции хэширования число $\mathsf{ID} \cdot t^{h(t\|M)} (\mathrm{mod}\, N)$,является *распознаваемым*, вычисление корня e-й степени должно представлять собой трудноразрешимую задачу. Следовательно, предполагается, что Алиса должна знать корень e-й степени числа ID (закрытый ключ, сгенерированный Трентом) и применять его при создании цифровой подписи.

Однако мы не приводим формальные и строгие аргументы, гарантирующие невозможность фальсификации в личностной схеме цифровой подписи Шамира. Поскольку устойчивость фальсификации зависит от вычисления числа $\mathsf{ID} \cdot t^{h(t\|M)} (\mathrm{mod}\, N)$ и его корня e-й степени по модулю N, очевидно, она определяется особенностями применяемой функции хэширования (в дополнение к проблеме RSA). Как и в других схемах цифровой подписи, для строгого доказательства стойкости личностной схемы цифровой подписи Шамира необходимо создать формальную модель функции хэширования h. Эта модель приводится в следующей главе.

Алгоритм 13.1. Личностная схема цифровой подписи Шамира

Установка системных параметров

Трент устанавливает следующие параметры.

1. N: произведение двух больших простых множителей.
2. e: целое число, удовлетворяющее условию $\gcd(e, \phi(N)) = 1$.
   (* (N, e) — открытые параметры, предназначенные для системных пользователей. *)
3. d: целое число, удовлетворяющее условию $ed \equiv 1 (\mathrm{mod}\, \phi(N))$.
   (* Число d является главным ключом Трента. *)
4. $h : \{0,1\}^* \mapsto \mathbb{Z}_{\phi(N)}$.
   (* Число h — стойкая однонаправленная функция хэширования. *)

Трент хранит число d как закрытый системный ключ (главный ключ) и обнародует системные параметры (N, e, h).

Генерация ключа пользователя

Пусть ID — однозначно распознаваемый идентификатор Алисы. Выполнив физическую идентификацию Алисы и убедившись в уникальности ее идентификатора ID, Трент выполняет операцию

$$g \leftarrow \mathsf{ID}^d (\mathrm{mod}\, N).$$

Генерация подписи

Для подписи сообщения $M \in \{0,1\}^*$ Алиса выбирает число $r \in_U \mathbb{Z}_N^*$ и выполняет следующие операции.

$$t \leftarrow r^e (\mathrm{mod}\, N),$$
$$s \leftarrow g \cdot r^h (t \parallel M) (\mathrm{mod}\, N).$$

Подписью является пара (s, t).

Верификация подписи

Получив сообщение M и подпись (s, t), Боб использует идентификатор Алисы ID для проверки подлинности подписи.

$$\mathsf{Verify}(\mathsf{ID}, s, t, M) = \mathsf{True}, \text{ если } s^e \equiv \mathsf{ID} \cdot t^{h(t \| M)} (\mathrm{mod}\, N).$$

13.3.2 Преимущества личностной схемы цифровой подписи Шамира

В обычной криптографии с открытым ключом применяется следующая схема верификации подписи. Для того чтобы верифицировать подпись Алисы, Боб

должен отдельно удостовериться в подлинности ее открытого ключа, например, проверив его сертификат. В частности, Боб должен убедиться, что канал для обмена ключами между ним и Алисой установлен успешно (см. рис. 10.1).

В личностной схеме цифровой подписи это делать не обязательно. Если алгоритм верификации возвращает значение True, Боб одновременно приходит к двум выводам.

- Подпись была создана Алисой с помощью ее закрытого ключа, основанного на ее идентификаторе ID.
- Ее идентификатор ID был сертифицирован Трентом. Именно это позволило Алисе создать свою подпись.

Возможность одновременно верифицировать эти факты представляет собой несомненное преимущество личностной схемы цифровой подписи. Возможность избежать передачи сертификата от пользователя, подписавшего сообщение, к пользователю, верифицирующему подпись, позволяет не перегружать канал связи. Благодаря этой особенности личностная криптография с открытым ключом получила второе название: **неинтерактивная криптография с открытым ключом** (non-interactive public key cryptography).

В заключение напомним, что Трент может подделать подпись любого пользователя! Следовательно, личностная схема цифровой подписи Шамира непригодна для применения в открытых системах. Она больше подходит для закрытых систем, в которых Трент имеет полное право обладать всей информацией. К сожалению, это требование является слишком жестким.

Личностную схему цифровой подписи, на которую не распространялось бы указанное выше требование, создать еще не удалось. Кроме того, до сих пор никто не смог разработать личностную схему цифровой подписи, допускающую неинтерактивное аннулирование ключей. Это совершенно необходимо, если ключ был скомпрометирован!

Однако, даже если бы эти две задачи были решены, личностная схема цифровой подписи все равно имела бы довольно ограниченное применение. В оставшейся части главы будет показано, что одну из этих задач можно решить с помощью личностной схемы шифрования, если не оказывать Тренту абсолютного доверия.

13.3.3 Самосертифицированные открытые ключи

Пусть (s, P) — пара, состоящая из закрытого и открытого ключей соответственно. Система аутентификации открытого ключа предоставляет гарантию G, связывающую ключ P с пользователем I.

В системе аутентификации открытых ключей, использующей базу данных (например, в системе X.509), гарантия G имеет вид цифровой подписи (I, P), вычисляемой и доставляемой органом сертификации CA. Система аутентификации имеет четыре атрибута: (s, I, P, G). Три из них (I, P, G) являются открытыми

и должны располагаться в открытом каталоге. Если пользователю необходима аутентичная копия открытого ключа пользователя I, он получает открытую тройку (I, P, G), проверяет гарантию G с помощью открытого ключа органа сертификации CA и выполняет аутентификацию ключа P.

В личностной системе аутентификации (например, в схеме Шамира) открытым ключом является не что иное, как само имя пользователя I. Следовательно, $P = I$ и система аутентификации имеет всего два атрибута: (s, I). Как показано в разделе 13.3.1, если пользователю необходимо аутентифицировать открытый ключ пользователя I, он должен верифицировать подпись. Если алгоритм верификации возвращает значение True, открытый ключ пользователя I является аутентичным. Следовательно, гарантией G является сам закрытый ключ, т.е. $G = s$.

Жиро (Girault) предложил схему аутентификации открытого ключа, обладающую преимуществами обеих упомянутых схем [121, 122]. В схеме Жиро гарантией является открытый ключ, т.е. $G = P$, который, таким образом, сам себя сертифицирует, а каждый пользователь обладает тремя атрибутами: (s, P, I). В схеме Жиро пользователь сам может выбирать свой закрытый ключ.

13.3.3.1 Схема Жиро

В схеме Жиро по-прежнему необходим доверенный посредник Трент, устанавливающий параметры системы и помогающий отдельным пользователям определять свои атрибуты.

13.3.3.2 Параметры закрытого ключа

Трент генерирует следующие параметры ключа RSA.

1. Открытый модуль $N = PQ$, где P, Q — большие простые числа, имеющие примерно одинаковую величину, например, $|P| = |Q| = 512$.
2. Открытый показатель степени e, взаимно простой с числом $\phi(N) = (P - 1)(Q - 1)$.
3. Закрытый показатель степени d, удовлетворяющий условию $ed \equiv 1 (\mathrm{mod}\, \phi(N))$.
4. Открытый элемент $g \in \mathbb{Z}_N^*$, имеющий максимальный мультипликативный порядок по модулю N. Трент может вычислить элемент g_P, являющийся порождающим по модулю P, и элемент g_Q, являющийся порождающим по модулю Q, а затем сконструировать число g с помощью китайской теоремы об остатках (теорема 6.7 в разделе 6.2.3).

Трент оглашает параметры открытого ключа (N, e, g), а компонент закрытого ключа d хранит в тайне.

13.3.3.3 Параметры ключа пользователя

Алиса случайным образом выбирает закрытый ключ s_A, представляющий собой 160-битовое целое число, вычисляет значение

$$v \leftarrow g^{-s_A} (\mathrm{mod}\, N)$$

и передает его Тренту. Затем она доказывает Тренту, что знает число s_A, не раскрывая его и используя простой протокол, описанный в разделе 13.3.3.4. Кроме того, Алиса посылает Тренту свой идентификатор I_A.

Трент создает открытый ключ Алисы в виде простой RSA-подписи числа $v - I_A$:

$$P_A \leftarrow (v - I_A)^d (\mathrm{mod}\, N).$$

Трент посылает Алисе число P_A как часть ее открытого ключа. Итак, выполняется следующее уравнение.

$$I_A \equiv P_A^e - v (\mathrm{mod}\, N). \tag{13.3.2}$$

На первый взгляд, поскольку оба числа P_A и v являются случайными элементами группы $\mathbb{Z}_N^*$, создать уравнение (13.3.2) не составляет большого труда. Например, Алиса может случайным образом извлечь число P_A, а затем, используя числа P_A^e, I_A и уравнение (13.3.2), вычислить значение v. Однако в этом случае Алиса не сможет вычислить дискретный логарифм числа v по основанию g и модулю N.

Именно способность Алисы демонстрировать знание дискретного логарифма числа v по основанию g и модулю N, т.е. числа $-s_A$, гарантирует, что параметр P_A был сгенерирован Трентом. Простейший способ продемонстрировать это знание — применить вариант протокола обмена ключами Диффи–Хеллмана, описанный в разделе 13.3.3.4.

13.3.3.4 Протокол обмена ключами

Пусть (s_A, P_A, I_A) — параметры открытого ключа Алисы, а (s_B, P_B, I_B) — параметры открытого ключа Боба. Они могут просто обменяться аутентичными ключами, приняв следующее соглашение.

$$K_{AB} \equiv (P_A^e + I_A)^{s_B} \equiv (P_B^e + I_B)^{s_A} \equiv g^{-s_A s_B} (\mathrm{mod}\, N).$$

В рамках этого соглашения Алиса вычисляет значение $(P_B^e + I_B)^{s_A} (\mathrm{mod}\, N)$, а Боб — $(P_A^e + I_A)^{s_B} (\mathrm{mod}\, N)$. Следовательно, это соглашение действительно является элементом протокола обмена ключами Диффи–Хеллмана. Если две стороны могут согласовать один и тот же ключ, значит, они знакомы и могут доказать друг другу свою подлинность.

Жиро также предложил протокол личностной идентификации и личностную схему цифровой подписи, являющуюся частью схемы цифровой подписи Эль-Гамаля [122].

13.3.3.5 Обсуждение

Самосертифицирующиеся открытые ключи Жиро обладают одним преимуществом личностной схемы Шамира: они не нуждаются в верификации дополнительного сертификата ключа, выданного его владельцу доверенным посредником. Верификация выполняется неявно и одновременно с проверкой криптографических способностей владельца ключа.

Однако проверяющая сторона должна, кроме имени пользователя I, знать его открытый ключ P, причем верификатор не может определить этот ключ по имени пользователя. Это означает, что верификатор должен обратиться к владельцу ключа и попросить его переслать открытый ключ еще до его применения. Таким образом, в протоколе возникает дополнительный этап. Следовательно, систему самосертифицирующихся открытых ключей Жиро нельзя считать *неинтерактивной* криптографией с открытым ключом (см. раздел 13.3.2). Это свойство является недостатком системы Жиро.

13.3.4 Личностная криптография с открытым ключом на "слабых" эллиптических кривых

Личностная криптосистема с открытым ключом, предложенная Шамиром, представляет собой схему цифровой подписи. Она основана на предположении, что существуют личностные системы шифрования. После того как Шамир впервые сформулировал эту проблему в 1984 году, было предложено несколько личностных криптосистем шифрования [51, 78, 141, 191, 251, 287, 289].

Сакаи (Sakai), Огиши (Ohgishi) и Касахара (Kasahara) [251], а также Жё (Joux) [154] независимо друг от друга выдвинули идею использовать особое свойство **функции пересчета пар** (pairing-mapping function), применяемой в абелевых группах, состоящих из точек эллиптической кривой (см. раздел 5.5). Работа Сакаи, Огиши и Касахары [251] представляет собой замечательное приложение результатов криптоанализа, описанных выше (см. раздел 13.3.4.1). Она перевела схему Шамира из разряда теоретических разработок в мир реальных приложений. Независимо от Сакаи, Огиши и Касахары в работе [154] Жё использовал тот же прием для создания другого приложения: однораундового трехстороннего разделения ключей Диффи–Хеллмана (Жё назвал его трехсторонним протоколом Диффи–Хеллмана).

Работы Сакаи, Огиши и Касахары [251], а также Жё [154] подтвердили результаты криптоанализа, выполненного Менезесом (Menezes), Окамото (Okamoto) и Ванстоуном (Vanstone) [197]. Эти работы возродили интерес к личностной криптографии.

Особое свойство, использованное Сакаи и его соавторами [251], а также Жё [154], заключается в следующем. В группе, состоящей из точек "слабой" эллип-

тической кривой, допускающей эффективное вычисление функции пересчета пар, **задача принятия решений Диффи–Хеллмана** (DDH problem — decisional Diffie–Hellman problem) оказывается легко разрешимой, а то время как ее вычислительный аналог — нет. Рассмотрим сначала "слабые" эллиптические кривые и связанную с ними легкую задачу принятия решений Диффи–Хеллмана, а затем схемы согласования ключей с помощью пересчета пар.

13.3.4.1 Класс слабых эллиптических кривых

Менезес, Окамото и Ванстоун [197] показали, что для специального класса эллиптических кривых, определенных на конечных полях $\mathbb{F}_q$, существует эффективный алгоритм, отображающий пары, состоящие из точек кривой $E(\mathbb{F}_q)$, в невырожденный (т.е. не равный мультипликативной единице) элемент конечного поля $\mu \in \mathbb{F}_{q^\ell}$. Эти особые кривые называются **суперсингулярными** (supersingular curves). Эти кривые удовлетворяют следующему условию: число t из уравнения (5.5.6) является кратным характеристике поля $\mathbb{F}_q$ (мы ограничимся рассмотрением характеристик поля, превышающих число 3).

В таком случае (когда число t из уравнения (5.5.6) является кратным характеристике поля $\mathbb{F}_q$) для некоторого большого простого числа $\alpha \mid \#E(\mathbb{F}_q)$ кривая, определенная на расширении поля $\mathbb{F}_q$, т.е. $E\left(\mathbb{F}_{q^\ell}\right)$, содержит много точек порядка α, т.е. точек P, удовлетворяющих условию $[\alpha]P = \mathcal{O}$ (см. раздел 5.5). Группа $E\left(\mathbb{F}_{q^\ell}\right)$ является нециклической, и, следовательно, многие из этих точек порядка α являются линейно-зависимыми. Точки P и Q, имеющие простой порядок α, называются **линейно независимыми**, если $P \neq [a]Q$ и $Q \neq [b]P$ для любых целых чисел a и b. (Иначе говоря, $P \notin \langle Q \rangle$ и $Q \notin \langle P \rangle$.) Для данного простого числа α расширение поля $\mathbb{F}_{q^\ell}$ имеет единственную подгруппу порядка α (поскольку группа $\mathbb{F}^*_{q^\ell}$ является циклической (см. теорему 5.2)).

Менезес, Окамото и Ванстоун [197] показали, что подгруппа поля $E\left(\mathbb{F}_{q^\ell}\right)$ и подгруппа расширения поля $\mathbb{F}_{q^\ell}$, имеющие одинаковый порядок α, являются изоморфными. Более того, расширение поля является небольшим: $\ell \leqslant 6$! Этот факт очень важен! Мы вернемся к нему чуть позже.

Изоморфизм, использованный Менезесом, Окамото и Ванстоуном, называется **спариванием Вейля** (Weil pairing). В соответствие двум точкам подгруппы поля $E\left(\mathbb{F}_{q^\ell}\right)$ порядка α этот изоморфизм ставит элемент расширения поля $\mathbb{F}_{q^\ell}$. Поскольку это отображение является изоморфизмом между двумя группами порядка α, одну из отображаемых точек, образующих пару, можно считать фиксированной константой, а вторую — переменной. Пусть $X \in E\left(\mathbb{F}_{q^\ell}\right)$ точка фиксированного порядка α (фиксированная константа). Спаривание Вейля обозначается как

$$e_X(P, Q),$$

где либо $P \in \langle X \rangle$, либо $Q \in \langle X \rangle$. Число $e_X(P, Q)$ является корнем единицы с номером α, т.е. элементом порядка α поля $\mathbb{F}_{q^\ell}$, тогда и только тогда, когда числа

P и Q имеют простой порядок α и являются линейно независимыми (например, либо $P \in \langle X \rangle$, либо $Q \in \langle X \rangle$).

Обозначим через G_1 подгруппу порядка α поля $E\left(\mathbb{F}_{q^\ell}\right)$, элементы которой (кроме элемента $\mathcal{O}$) являются линейно независимыми от "фиксированной константы X порядка α". (Такая подгруппа существует, если группа $E\left(\mathbb{F}_{q^\ell}\right)$ не является циклической и, следовательно, суперсингулярной.) Через $G_2 = \langle e_X(P, X) \rangle$ обозначим подгруппу порядка α поля $\mathbb{F}_{q^\ell}$, порожденную элементом $e_X\langle P, X \rangle$. Теперь нам известно, что $\#G_1 = \#G_2 = \alpha$. Эти две подгруппы изоморфны относительно спаривания Вейля. Обратите внимание на то, что даже если группа G_1 является аддитивной группой точек эллиптической кривой, а группа G_2 — мультипликативной подгруппой конечного поля, с точки зрения изоморфизма между ними нет существенной разницы.

Спаривание Вейля обладает следующими свойствами.

Свойство 13.1 (Свойства спаривания Вейля). Пусть $(G_1, +)$ и $(G_2, \cdot)$ — две изоморфные абелевы группы простого порядка относительно спаривания Вейля e_X. Спаривание обладает следующими свойствами.

Тождественность. Для всех $P \in G_1$

$$e_X(P, P) = 1_{G_2}.$$

Билинейность. Для всех $P, Q \in G_1$

$$e_X(P + Q, R) = e_X(P, R)e_X(Q, R), e_X(R, P + Q) = e_X(R, P)e_X(R, Q).$$

Невырожденность. Для всех $P \in G_1$, таких что $P \neq \mathcal{O}$ (поскольку точки P и X являются линейно независимыми)

$$e_X(P, X) \neq 1_{G_2}, e_X(X, P) \neq 1_{G_2}.$$

Практическая эффективность. Для всех $P \in G_1$ и $R \in \langle X \rangle$, числа $e_X(P, R)$ и $e_X(R, P)$ допускают эффективное вычисление.

Из билинейности следует, что

$$e_X([n]P, X) = e_X(P, X)^n = e_X(P, [n]X).$$

Из свойства невырожденности и условия $\alpha \nmid n$ следует, что результатом спаривания не может быть единица группы G_2.

Благодаря свойствам спаривания Вейля задача дискретного логарифмирования на эллиптической кривой (задача ECDLP, описанная в разделе 5.5.3) значительно упрощается и сводится к задаче вычисления дискретного логарифма в конечном поле (MOV reduction). Для того чтобы свести задачу ECDLP к задаче вычисления

дискретного логарифма в конечном поле, используя пару точек $(P, [n]P)$, можно вычислить пары Вейля $\xi = e_X(P, X)$ и $\eta = e_X([n]P, X)$ и учесть, что

$$\eta = e_X([n]P, X) = e_X(P, X)^n = \xi^n. \tag{13.3.3}$$

Следовательно, пара точек (ξ, η) из группы G_2 образует задачу дискретного логарифмирования в мультипликативной группе G_2, а значит, в конечном поле $\mathbb{F}_{q^\ell}$. Для этой задачи существует субэкспоненциальный алгоритм решения, сложность которого задается выражением sub_exp(q^ℓ). Напомним, что $\ell \leqslant 6$. Таким образом, достигается существенное уменьшение сложности: широко известная экспоненциальная задача $O\left(\sqrt{\alpha}\right) \approx O\left(\sqrt{q}\right)$ сводится к субэкспоненциальной: sub_exp(q^ℓ), где $\ell \leqslant 6$.

По мере развития компьютерной техники число q должно увеличиваться. Одновременно с ним должны увеличиваться параметры суперсингулярной кривой и размер конечного поля. Иначе говоря, преимущества эллиптических кривых в криптографии становятся все менее весомыми. После появления работы Менезеса, Окамото и Ванстоуна [197] стало общепризнанным, что суперсингулярные эллиптические кривые являются слабыми и не должны использоваться в криптографии.

Почему в заглавии раздела термин "слабые" взят в кавычки? Недавно было открыто новое полезное свойство суперсингулярных эллиптических кривых, шокировавшее научное сообщество. Однако для начала опишем задачу принятия решений Диффи–Хеллмана.

13.3.4.2 Задача принятия решений Диффи–Хеллмана

В определении 8.1 (раздел 8.4) была сформулирована вычислительная задача Диффи–Хеллмана (CDH problem — computational Diffie–Hellman problem). Ее вариант, касающийся принятия решений, описывается следующим определением.

Определение 13.1 (Задача принятия решений Диффи–Хеллмана (задача DDH)).
ВВОД desc(G): описание абелевой группы G;
$(g, g^a, g^b, g^c) \in G^4$, где g — порождающий элемент группы G.
ВЫВОД ДА, если $ab \equiv c (\mathrm{mod}\, \#G)$.

Задача принятия решений Диффи–Хеллмана не может быть сложнее, чем ее вычислительный аналог. Если существует алгоритм решения вычислительной задачи Диффи–Хеллмана (т.е. некий оракул, решающий эту задачу), то, получив четверку (g, g^a, g^b, g^c), он по первым трем элементам может найти число g^{ab} и ответ к задаче CDH, проверив, равен ли результат вычисления числу g^c.

Однако в общем случае никакие другие зависимости между этими двумя задачами не известны. Более того, до сих пор нет ни одного алгоритма решения задачи

DDH. Трудность решения задачи DDH общепризнанна, а сама задача стала образцом неразрешимой проблемы и лежит в основе многих криптографических систем [20, 58, 84, 209, 283].

Для специального случая суперсингулярных эллиптических кривых недавно был открыт следующий факт: задача DDH является легко разрешимой. Это доказали Жё и Нгуен [155]. Чтобы объяснить этот факт, необходимо сначала представить задачи DDH, CDH и DL в аддитивном виде, поскольку группы точек эллиптической кривой являются аддитивными.

Задача дискретного логарифмирования (DL) в группе $(G, +)$

ВВОД Два элемента $P, Q \in G$, где P — порождающий элемент группы.

ВЫВОД Целое число a, удовлетворяющее условию $Q = aP$.

Вычислительная задача Диффи–Хеллмана (CDH) в группе $(G, +)$

ВВОД Три элемента $P, aP, bP \in G$, где P — порождающий элемент группы.

ВЫВОД Элемент $(ab)P \in G$.

Задача принятия решения Диффи–Хеллмана (DDH) в группе $(G, +)$

ВВОД Четыре элемента $P, aP, bP, cP \in G$, где P — порождающий элемент группы.

ВЫВОД ДА, тогда и только тогда, когда $c \equiv ab (\mathrm{mod}\, \#G)$.

13.3.4.3 "Слабые" кривые, допускающие простое решение задачи принятия решений Диффи–Хеллмана

Свойство тождественности спаривания Вейля (свойство 13.1) является довольно неудобным. Оно означает, что для точек $P, Q \in G_1$ (а также точки $Q = [a]P$, где a — некоторое целое число) спаривание приводит к следующему результату: $e_X(P, Q) = e_X(P, P) = 1^a = 1$. Для того чтобы отображение не приводило к вырожденному результату, необходимо найти другую точку X порядка α, линейно независимую от элементов группы G_1. Это сильное ограничение, не позволяющее непосредственно применять спаривание Вейля в криптографических приложениях.

Ферхойль (Verheul) [296] разработал метод, который он назвал "деформирующим отображением" координат точек, лежащих на кривой. Отображение выполняется в поле $\mathbb{F}_q$ и обозначается через $\mathbb{F}(P)$. Если точка P лежит в поле $E(\mathbb{F}_{q^\ell})$, точка $\mathbb{F}(P)$ остается в поле $E(\mathbb{F}_{q^\ell})$ того же порядка (кручения), поскольку порядок точки P превышает число 3. Еще более важно то, что точки P и $\mathbb{F}(P)$ являются линейно независимыми. При деформирующем отображении спаривание Вейля приобретает следующий вид.

$$e(P, P) = e_X(P, \mathbb{F}(P)).$$

Очевидно, что $e(P,P) \neq 1$, поскольку точки P и $\mathbb{F}(P)$ линейно независимы. Более того, для точек $P, Q \in G_1$ спаривание Вейля обладает следующим свойством симметрии.

$$e(P,Q) = e(P,[n]P) = e(P,P)^n = e([n]P,P) = e(Q,P). \qquad (13.3.4)$$

Рассмотрим теперь задачу DDH в группе G_1. Для того чтобы определить, является ли четверка $(P,[a]P,[b]P,[c]P)$ четверкой задачи DDH, вычислим функцию спаривания: $\xi = e(P,[c]P)$ и $\eta = e([a]P,[b]P)$. Учитывая, что

$$\xi = e(P,P)^c, \eta = e(P,P)^{ab}$$

и $e(P,P) \neq 1$, получаем, что указанная четверка описывает задачу DH тогда и только тогда, когда $\xi = \eta$, т.е. если и только если $ab \equiv c (\mathrm{mod}\, \#G_1)$. Поскольку функция спаривания Вейля допускает эффективное вычисление, задача принятия решений становится легкоразрешимой.

Это важное наблюдение, сделанное Жё и Нгуеном [155], расширило возможности применения суперсингулярных эллиптических кривых в криптографии. Одним из таких приложений стала личностная криптография. Она основана на свойствах суперсингулярных кривых и предположении о неразрешимости вычислительной задачи Диффи–Хеллмана и задачи о вычислении дискретного логарифма.

Факт (задача DDH является легкоразрешимой)

Задачу DDH в группах точек суперсингулярных кривых можно эффективно решить с помощью алгоритма спаривания Вейля.

Предположение (задачи CDH и DL остаются трудноразрешимыми)

При правильном выборе размера эллиптической кривой задача CDH (а значит, и задача DL) в группах точек суперсингулярной кривой является трудноразрешимой. Для кривой $E\left(\mathbb{F}_{q^\ell}\right)$ сложность задается выражением $\mathrm{sub\_exp}(q^\ell)$, где $\ell \leqslant 6$.

Предположение о том, что "задачи CDH и DL остаются трудноразрешимыми" содержит оценку сложности решения задачи для кривой $E\left(\mathbb{F}_{q^\ell}\right)$. Эта оценка следует из возможности упрощения задачи дискретного логарифмирования и существования субэкспоненциального алгоритма решения задачи о вычислении дискретного логарифма в конечных полях. Таким образом, по сравнению с обычными эллиптическими кривыми при работе с суперсингулярными кривыми необходимо выбирать большой параметр безопасности (размер кривой). Этот параметр должен быть настолько большим, чтобы величина $\mathrm{sub\_exp}(q^\ell)$ была практически недостижимой. Итак, для того, чтобы получить возможность применять недавно открытое свойство суперсингулярных эллиптических кривых в криптографии (см. разделы 13.3.5–13.3.7), необходимо пожертвовать эффективностью, которая присуща эллиптическим кривым, определенным в конечных полях.

Существует два метода спаривания: модифицированное спаривание Вейля, рассмотренное выше, и спаривание Тэйта (Tate). Второй метод более эффективен. Детали этих алгоритмов выходят за рамки нашей книги. Заинтересованный читатель найдет их в книгах [51, 116]. Оставшаяся часть главы посвящена спариванию Вейля.

13.3.5 Личностная неинтерактивная система разделения ключей Сакаи, Огиши и Касахары

Как и в личностной схеме цифровой подписи Шамира, для установки параметров в системе разделения ключей Сакаи, Огиши и Касахары [252] (SOK key sharing system) необходим доверенный посредник (Трент).

Система разделения ключей SOK обладает следующими свойствами.

- **Установка системных параметров.** Этот алгоритм выполняется Трентом для установки глобальных системных параметров и вычисления главного ключа.
- **Генерация ключа пользователя.** Этот алгоритм выполняется Трентом. Получая на вход главный ключ и произвольную строку битов $\mathsf{id} \in \{0,1\}^*$, алгоритм вычисляет закрытый ключ, соответствующий строке id. Алгоритм является конкретизацией операции (13.3.1).
- **Алгоритм разделения ключей.** Этот алгоритм выполняется двумя конечными пользователями. Алгоритм получает закрытый и открытый (id) ключи каждого пользователя и вычисляет секретный ключ, разделенный между обоими пользователями.

Эти три алгоритма разбиваются на следующие этапы.

Установка параметров системы

Перед открытием центра генерации ключей Трент устанавливает параметры системы, выполняя следующие операции.

1. Генерируются две группы $(G_1, +)$ и $(G_2, \cdot)$, имеющие простой порядок p, и выполняется модифицированное спаривание Вейля[1] $e : (G_1, +)^2 \mapsto (G_2, \cdot)$. Выбирается произвольный порождающий элемент $P \in G_1$.
2. Случайным образом извлекается число $\ell \in_U \mathbb{Z}_p$ и устанавливается параметр $P_{pub} \leftarrow [\ell]P$. Число ℓ играет роль главного ключа.
3. Выбирается стойкая криптографическая функция хэширования $f : \{0,1\}^* \mapsto G_1$. Эта хэш-функция представляет собой отображение идентификатора пользователя id в элемент группы G_1.

[1] В исходном варианте схемы разделения ключей SOK использовалось немодифицированное спаривание Вейля. Этот вариант менее удобен, чем описанный в книге.

Трент оглашает системные параметры и их описания

$$(\mathrm{desq}(G_1), \mathrm{desq}(G_2), e, P, P_{pub}, f)$$

и сохраняет число ℓ в качестве системного закрытого ключа. Поскольку Трент считается известным всем пользователям, обнародованные параметры системы также становятся известными всем пользователям (например, эти параметры могут быть зашиты в каждое устройство, использующее данную схему).

Обратите внимание на то, что секретность главного ключа ℓ защищена сложностью решения задачи DLP в группе G_1.

Теперь Трент открывает центр генерирования ключей.

Генерация ключей пользователей

Пусть ID_A — однозначно определяемый идентификатор Алисы. Будем предполагать, что ID_A обладает достаточной избыточностью, так что любой другой пользователь системы не может иметь такой же идентификатор. Выполнив физическую идентификацию и убедившись в уникальности идентификатора ID_A, Трент генерирует ключ, выполняя следующие операции.

1. Вычисляется $P_{\mathsf{ID}_A} \leftarrow f(\mathsf{ID}_A)$ — элемент группы G_1 и личностный открытый ключ Алисы.
2. Устанавливается закрытый ключ $S_{\mathsf{ID}_A} \leftarrow [\ell]P_{\mathsf{ID}_A}$.

Поскольку значение P_{ID_A} хэшируется, оно выглядит как случайное число. Однако идентификатор ID_A обладает достаточной избыточностью и является прообразом значения P_{ID_A} относительно криптографической функции хэширования f. Следовательно, элемент P_{ID_A} является *распознаваемым*. Таким образом, не имеет существенного значения, что считать открытым ключом Алисы — идентификатор ID_A или число P_{ID_A}.

Следует отметить, что закрытый ключ Алисы защищен сложностью решения задачи CDH в группе G_1. Это объясняется тем, что число P_{ID_A} должно генерироваться элементом P (порождающим элементом группы G_1), и его можно обозначить следующим образом: $P_{\mathsf{ID}_A} = [a]P$, где $a < p$. Таким образом, вычисление по значениям $P, P_{pub}(= [\ell]P), P_{\mathsf{ID}_A}(= [a]P)$ секретного ключа

$$S_{\mathsf{ID}_A} = [\ell]\, P_{\mathsf{ID}_A} = [\ell a]P$$

эквивалентно решению задачи CDH в группе G_1.

Алгоритм разделения ключей

Идентификаторы Алисы и Боба, ID_A и ID_B, известны обоим партнерам. Следовательно, им известны и соответствующие открытые ключи $P_A = f(\mathsf{ID}_A)$ и $P_B = = f(\mathsf{ID}_B)$.

Алиса может сгенерировать общий ключ $K_{AB} \in (G_2, \cdot)$ с помощью следующей операции

$$K_{AB} \leftarrow e(S_{\mathsf{ID}_A}, P_{\mathsf{ID}_B}).$$

В свою очередь Боб может сгенерировать общий ключ $K_{BA} \in (G_2, \cdot)$, выполнив операцию

$$K_{BA} \leftarrow e(S_{\mathsf{ID}_B}, P_{\mathsf{ID}_A}).$$

Учитывая свойство билинейности 13.1, получаем

$$K_{AB} = e(S_{\mathsf{ID}_A}, P_{\mathsf{ID}_B}) = e([\ell]P_{\mathsf{ID}_A}, P_{\mathsf{ID}_B}) = e(P_{\mathsf{ID}_A}, P_{\mathsf{ID}_B})^{\ell}.$$

Аналогично

$$K_{BA} = e(P_{\mathsf{ID}_B}, P_{\mathsf{ID}_A})^{\ell}.$$

Из свойства симметричности модифицированного спаривания Вейля (13.3.4) следует, что

$$K_{AB} = K_{BA}.$$

Итак, Алиса и Боб действительно разделяют общий секрет, не взаимодействуя друг с другом.

Постороннему пользователю (не Алисе, Бобу или Тренту) практически невозможно найти число K_{AB}, зная открытые числа $(P, P_{\mathsf{ID}_A}, P_{\mathsf{ID}_B}, P_{pub})$. Эта задача называется **билинейной задачей Диффи–Хеллмана** (bilinear Diffie–Hellman problem) [51]. По существу, она эквивалентна задаче CDH.

Когда Боб получает сообщение, аутентифицированное с помощью ключа K_{AB}, он точно знает, что автором сообщения является Алиса. Однако Алиса, доказывая свое авторство *назначенному верификатору*, Бобу, может отрицать свое участие в контакте, отвечая на запрос третьей стороны, поскольку Боб обладает теми же самыми криптографическими возможностями для создания сообщения. Рассмотрим сценарий, в котором Алиса и Боб являются шпионами. Вступая в контакт, они должны аутентифицировать друг друга. Однако Алиса является двойным агентом и подозревает, что Боб также ведет нечестную игру. Следовательно, схема аутентификации должна допускать возможность отрицания своего участия в контакте. Система разделения ключей SOK полностью соответствует этому требованию. Она использует открытые ключи, т.е. для аутентификации не требуется интерактивная связь с доверенным посредником (см. главу 2).

Более серьезный сценарий, описывающий применение системы разделения ключей SOK, — протокол IKE. Этот протокол имеет режим аутентификации, допускающий "правдоподобное отрицание" (см. раздел 12.2.4). Система разделения ключей SOK обеспечивает возможность полного отрицания авторства в рамках протокола IKE, сохраняя преимущества криптосистемы с открытым ключом.

13.3.6 Трехсторонний протокол согласования ключей Диффи–Хеллмана

Жё [154] применил метод спаривания и разработал протокол трехстороннего согласования ключей с помощью весьма простого способа. Он назвал свой метод "трехсторонним протоколом согласования ключей Диффи–Хеллмана". В этом протоколе снова использовался метод спаривания Вейля, поэтому он был не слишком удобен для реальных приложений (в нем требовалось генерировать линейно независимые точки). Для решения этой проблемы был предложен протокол, основанный на модифицированном спаривании Вейля.

Предположим, что Алиса генерирует компоненты своего ключа P_A.

$$P_A \leftarrow [a]P,$$

где $P \in G_1$ — точка на суперсингулярной эллиптической кривой, имеющая простой порядок α, а число $a < \alpha$ — целое. Аналогично предположим, что Боб и Чарли вычисляют параметры своих ключей.

$$P_B \leftarrow [b]P, P_C \leftarrow [c]P,$$

где $b < \alpha$ и $c < \alpha$. Целые числа a, b и c являются секретными ключами Алисы, Боба и Чарли соответственно.

Каждый их трех партнеров помещает свой параметр ключа P_A, P_B или P_C в открытый каталог. Сделав это, они становятся обладателями общего ключа

$$e(P_B, P_C)^a = e(P_A, P_C)^b = e(P_A, P_B)^c = e(P, P)^{abc}.$$

Алиса вычисляет общий ключ, возведя в степень образ первой пары, Боб — второй, а Чарли — третьей.

Если бы в протоколе не применялся метод спаривания, согласование ключа не удалось бы провести за один сеанс.

Разумеется, как и протокол обмена ключами Диффи–Хеллмана, эта схема не обладает свойством аутентификации.

13.3.7 Личностная криптосистема Бонэ и Франклина

Поскольку между двумя пользователями можно разделить общий ключ, используя лишь их идентификаторы, для шифрования также достаточно знать имена пользователей. Бонэ (Boneh) и Франклин (Franklin) [51] применили метод спаривания и создали первую прикладную личностную криптосистему с открытым ключом, удовлетворяющую требованиям, сформулированным Шамиром.

В личностной криптосистеме Бонэ и Франклина используются четыре алгоритма.

- **Установка параметров системы.** Этот алгоритм выполняется Трентом. Он предназначен для генерации глобальных параметров системы и главного ключа.
- **Генерация ключа пользователя.** Этот алгоритм выполняется Трентом. На его вход поступает главный ключ и произвольная строка битов $\mathsf{id} \in \{0,1\}^*$. Результатом работы алгоритма является закрытый ключ, соответствующий строке id. Этот алгоритм является конкретизацией операции (13.3.1).
- **Шифрование.** Это — вероятностный алгоритм. Он шифрует сообщения с помощью открытого ключа id.
- **Расшифровка.** На вход этого алгоритма поступает зашифрованный текст и закрытый ключ, а результатом его работы является исходный текст.

Личностная криптосистема Бонэ и Франклина описывается алгоритмом 13.3.7.1.

Алгоритм 13.2. Личностная криптосистема Бонэ и Франклина

Установка системных параметров

Трент выполняет следующие инструкции.

1. Сгенерировать две группы $(G_1, +)$ и $(G_2, \cdot)$, имеющие простой порядок p, и вычислить отображение $e : (G_1, +)^2 \mapsto (G_2, \cdot)$. Выбрать произвольный порождающий элемент $P \in G_1$.
2. Извлечь случайное число $s \in_U \mathbb{Z}_p$ и установить параметр $P_{pub} \leftarrow [s]P$, используя s в качестве главного ключа.
3. Выбрать криптографически стойкую функцию хэширования $F : \{0,1\}^* \mapsto G_1$, предназначенную для отображения идентификатора пользователя id в элемент группы G_1.
4. Выбрать криптографически стойкую функцию хэширования $H : G_2 \mapsto \{0,1\}^n$. В этом случае пространством исходных текстов $\mathcal{M}$ является множество $\{0,1\}^n$.

Трент сохраняет число s в качестве закрытого ключа системы (главного ключа) и оглашает открытые системные параметры, а также их описание.

$$(G_1, G_2, e, n, P, P_{pub}, F, H).$$

Генерация ключа пользователя

Пусть ID — однозначно распознаваемый идентификатор Алисы. Выполнив физическую идентификацию Алисы и убедившись в уникальности ее идентификатора ID, Трент выполняет следующие операции.

1. Вычисляет величину $Q_{\mathsf{ID}} \leftarrow F(\mathsf{ID})$, являющуюся элементом группы G_1 и личностным открытым ключом Алисы.
2. Вычисляет закрытый ключ Алисы $d_{\mathsf{ID}} \leftarrow [s]Q_{\mathsf{ID}}$.

Шифрование

Чтобы послать Алисе секретное сообщение, Боб должен сначала получить системные параметры $(G_1, G_2, e, n, P, P_{pub}, F, H)$. Используя эти параметры, Боб вычисляет величину

$$Q_{\mathsf{ID}} \leftarrow F(\mathsf{ID}).$$

Допустим, что сообщение состоит из блоков по n бит. Для того чтобы зашифровать сообщение $M \in \{0,1\}^n$, Боб извлекает случайное число $r \in_U \mathbb{Z}_p$ и выполняет следующие операции.

$$g_{\mathsf{ID}} \leftarrow e(Q_{\mathsf{ID}}, [r]P_{pub}) \in G_2,$$
$$C \leftarrow ([r]P, M \oplus H(g_{\mathsf{ID}})).$$

Зашифрованный текст представляет собой пару $C = ([r]P, M \oplus H(g_{\mathsf{ID}}))$.

Расшифровка

Пусть $C = (U, V) \in \mathbb{C}$ — текст, зашифрованный с помощью открытого ключа Алисы ID. Для того чтобы расшифровать текст C, используя свой закрытый ключ $d_{\mathsf{ID}} \in G_1$, Алиса вычисляет величину

$$V \oplus H(e(d_{\mathsf{ID}}, U)).$$

Покажем, что система, описанная алгоритмом 13.2, является криптографической.

$$\begin{aligned} e(d_{\mathsf{ID}}, U) &= e([s]Q_{\mathsf{ID}}, [r]P) = \\ &= e(Q_{\mathsf{ID}}, [r]P)^s = \\ &= e(Q_{\mathsf{ID}}, [rs]P) = \\ &= e(Q_{\mathsf{ID}}, [r]P_{pub}) = \\ &= g_{\mathsf{ID}}. \end{aligned}$$

Следовательно, значение, к которому Алиса применяет функцию хэширования H в ходе расшифровки, фактически является величиной g_{ID}, т.е. именно тем числом, к которому Боб применял функцию хэширования H во время шифрования. Следовательно,

$$V \oplus H(e(d_{\mathsf{ID}}, U)) = M \oplus H(g_{\mathsf{ID}}) \oplus H(g_{\mathsf{ID}}) = M,$$

поскольку побитовая операция XOR является обратной самой себе.

Бонэ и Франклин привели формальное доказательство стойкости личностной схемы шифрования в строгом смысле. Иначе говоря, они доказали, что их схема шифрования является стойкой по отношению к атаке на основе адаптивно

подобранного открытого текста. Непосредственное применение функции хэширования в стиле схемы Эль-Гамаля означает, что доказательство основано на так называемой "модели случайного оракула". Поскольку формальные методы и понятие сильной стойкости, а также модель случайного оракула рассматриваются в части V, доказательство Бонэ и Франклина здесь не приводится.

13.3.7.1 Расширение схемы для открытой системы

Следует отметить, что Трент может расшифровывать любой текст, посланный любому пользователю системы! Следовательно, схема Бонэ и Франклина непригодна для использования в открытых системах. Однако исходную схему можно расширить так, чтобы она удовлетворяла требованиям открытых систем. Рассмотрим упрощенный вариант метода, предложенного Бонэ и Франклином.

Основная идея, естественно, заключается в использовании нескольких доверенных посредников. Однако это имеет смысл, только если такая модификация не приводит к резкому увеличению количества идентификаторов пользователей системы или размеров зашифрованного текста. Вот как можно решить эту задачу. Рассмотрим вариант с двумя доверенными посредниками. Его можно тривиальным образом распространить на случай с многими посредниками.

Установка параметров системы. Допустим, что параметры системы $(G_1, G_2, e, n, P, F, H)$ идентичны параметрам, указанным в разделе 13.3.7. Пусть, далее

$$P_1 \leftarrow [s_1]P,$$
$$P_2 \leftarrow [s_2]P,$$

причем тройка (P, P_1, P_2) играет роль пары (P, P_{pub}), упомянутой в разделе 13.3.7, т.е. числа s_1 и s_2 являются главными ключами доверенных посредников TA_1 и TA_2 соответственно.

Таким образом, набор $(G_1, G_2, e, n, P, P_1, P_2, F, H)$ содержит открытые системные параметры. Эти параметры можно "зашить" в приложения.

Генерация ключа пользователя. Пусть ID — идентификатор Алисы. Для $i = 1, 2$ генерация ключа, выполняемая посредником TA_i, осуществляется следующим образом.

1. Вычисляется величина $Q_{\mathsf{ID}} \leftarrow F(\mathsf{ID})$, которая одновременно является элементом группы G_1 и уникальным личностным открытым ключом Алисы.
2. Вычисляется компонент закрытого ключа Алисы $d_{\mathsf{ID}}^{(i)} \leftarrow [s_i]\, Q_{\mathsf{ID}}$.

Закрытый ключ Алисы представляет собой сумму двух компонентов.

$$d_{\mathsf{ID}} \leftarrow d_{\mathsf{ID}}^{(1)} + d_{\mathsf{ID}}^{(2)}.$$

Если два посредника не состоят в сговоре, им неизвестен закрытый ключ Алисы.

Обратите внимание на то, что Алиса имеет единственный открытый ключ — ID.

Шифрование. Для того чтобы послать Алисе секретное сообщение, Боб должен сначала получить системные параметры $(G_1, G_2, e, n, P, P_1, P_2, F, H)$. Используя эти параметры, Боб вычисляет величину

$$Q_{\mathsf{ID}} = F(\mathsf{ID}).$$

Допустим, что сообщение разбито на блоки, состоящие из n бит. Для того чтобы зашифровать сообщение $M \in \{0,1\}^n$, Боб извлекает случайное число $r \in_U \mathbf{Z}_p$ и выполняет следующие операции.

$$g_{\mathsf{ID}} \leftarrow e(Q_{\mathsf{ID}}, [r](P_1 + P_2)),$$
$$C \leftarrow ([r]P, M \oplus H(g_{\mathsf{ID}})).$$

Зашифрованный текст представляет собой пару C. Таким образом, зашифрованный текст является парой, составленной из точки, принадлежащей группе G_1, и блока, принадлежащего множеству $\{0,1\}^n$. Иначе говоря, пространством зашифрованных текстов $\mathcal{C}$ является пара $G_1 \times \{0,1\}^n$.

Расшифровка. Пусть $C = (U, V) \in \mathcal{C}$ — текст, зашифрованный с помощью открытого ключа Алисы ID. Для того чтобы расшифровать его, используя свой закрытый ключ $d_{\mathsf{ID}} \in G_1$, Алиса вычисляет сумму

$$V \oplus H(e(d_{\mathsf{ID}}, U)).$$

Следует отметить, что

$$\begin{aligned} e(d_{\mathsf{ID}}, U) &= e([s_1]Q_{\mathsf{ID}} + [s_2]Q_{\mathsf{ID}}, [r]P) = \\ &= e([s_1]Q_{\mathsf{ID}}, [r]P)e([s_2]Q_{\mathsf{ID}}, [r]P) = \\ &= e(Q_{\mathsf{ID}}, [rs_1]P)e(Q_{\mathsf{ID}}, [rs_2]P) = \\ &= e(Q_{\mathsf{ID}}, [r](P_1 + P_2)) = \\ &= g_{\mathsf{ID}}. \end{aligned}$$

Итак, Алиса вычисляет величину g_{ID} и может расшифровать сообщение.

$$V \oplus H(e(d_{\mathsf{ID}}, U)) = M \oplus H(g_{\mathsf{ID}}) \oplus H(g_{\mathsf{ID}}) = M,$$

поскольку побитовая операция XOR является обратной самой себе.

Замечания

- По сравнению с вариантом, в котором предусмотрен лишь один посредник, объем вычислений, связанных с шифрованием и расшифровкой, удваивается. Однако ни количество идентификаторов Алисы, ни размер зашифрованных текстов не увеличиваются.

- К расшифровке приводит лишь сговор между посредниками. По отдельности они не в состоянии расшифровать зашифрованный текст. Чем больше в системе посредников, тем меньше вероятность сговора между ними. Очевидно, что увеличение количества посредников не приводит к увеличению количества идентификаторов Алисы или размера зашифрованных текстов. Однако объем вычислений, связанных с шифрованием и расшифровкой сообщений, растет прямо пропорционально количеству посредников.
- Для расшифровки сообщения конечному пользователю необходима помощь всех посредников. Если хотя бы один из посредников пользуется абсолютным доверием, перлюстрация сообщения становится невозможной. Итак, расширенная схема Бонэ–Франклина становится пригодной для использования в открытых системах.

13.3.8 Неинтерактивность: аутентификация без организации канала для обмена ключами

Когда Боб посылает Алисе секретное сообщение, используя обычные средства криптографии, он должен сначала организовать канал для обмена ключами (см. рис. 7.1). В обычной криптографии с открытым ключом канал для обмена ключами может использовать службу каталогов: например, отправитель может верифицировать сертификат открытого ключа адресата. Итак, отправитель должен сначала послать адресату запрос на получение сертификата его открытого ключа. Поскольку в открытых системах пользователи не могут запомнить связь между открытым ключом и его владельцем, необходимо, чтобы отправитель и адресат организовали канал для обмена ключами и лишь затем посылали секретные сообщения, зашифрованные открытым ключом получателя.

В личностных криптосистемах с открытым ключом организация канала для обмена ключами не только не нужна, но и невозможна! Даже если Боб попросит Алису прислать ее личностный открытый ключ, он не сможет проверить, действительно ли полученный идентификатор является открытым ключом Алисы. Все что Бобу необходимо сделать, — считать полученный идентификатор подлинным открытым ключом Алисы и применять его для шифрования своих сообщений. Если Алиса сможет расшифровать сообщения, присланные ей Бобом, значит, ее идентификатор действительно является ее открытым ключом. Следовательно, поскольку идентификатор однозначно определяет пользователя (это должны гарантировать один или несколько посредников), отправителю нет необходимости связываться с получателем перед шифрованием. Вот почему личностные криптосистемы с открытым ключом называются **неинтерактивными криптосистемами с открытым ключом**.

Неинтерактивность личностной схемы шифрования особенно ярко проявляется в системе разделения ключей SOK (см. раздел 13.3.5). Зарегистрировав свои

личностные открытые ключи, Алиса и Боб сразу получают в свое распоряжение защищенный канал для обмена ключами, даже не вступая в контакт: защита канала поддерживается двумя закрытыми ключами. Алиса и Боб не обязаны запускать протокол для того, чтобы установить защищенный канал связи! Неинтерактивность такой процедуры согласования ключей резко контрастирует с протоколом Жиро, основанным на использовании самосертифицирующихся ключей (раздел 13.3.3.4). Протокол Жиро не является неинтерактивным, поскольку два его участника должны обменяться своими открытыми ключами перед тем, как согласовать общий закрытый ключ.

Невозможность прямо подтвердить, что открытый ключ принадлежит подлинному пользователю, т.е. отсутствие открыто верифицируемого сертификата ключа, позволяет решить так называемую "шпионскую задачу" ("spy problem"). Эта задача рассматривается в следующей главе после описания **протоколов с нулевым разглашением** (zero-knowledge protocol).

13.3.9 Нерешенные задачи

Рассмотрим процедуру генерации ключа пользователя.

$$\text{закрытый\_ключ} = F(\text{главный\_ключ}, \text{открытый\_ключ}).$$

В этом методе для вычисления закрытого ключа пользователь должен представить открытый ключ по своему усмотрению. Опасность заключается в том, что пользователем может оказаться Злоумышленник. Однако доверенный посредник должен безусловно обслуживать любого пользователя и возвращать ему закрытый ключ.

Для того чтобы криптосистема была личностной (ID-based), т.е. неинтерактивной или не использующей сертификаты, функция F должна быть **детерминированной** (deterministic). Таким образом, процедура генерации ключа пользователя не имеет случайных входных данных. Иначе говоря, закрытый ключ каждого пользователя однозначно определяется главным ключом. Такой вид вычислений таит в себе потенциальную опасность (с точки зрения криптоанализа главного ключа). Опасность стала очевидной после критики, которой Голдвассер и Микали подвергли детерминированные функции с секретом, используемые в модели Диффи–Хеллмана [125]. В стандартных приложениях криптографии с открытым ключом эти вычисления не используются (например, доверенный посредник добавляет случайный аргумент).

Таким образом, несомненный интерес вызывают личностные криптосистемы с открытым ключом и вероятностным закрытым ключом.

Второй нерешенной проблемой является разработка личностной криптосистемы, допускающей неинтерактивное аннулирование идентификаторов. Это анну-

пирование становится необходимым, если закрытый ключ конечного пользователя был скомпрометирован.

13.4 Резюме

В главе рассмотрены системы аутентификации в криптографии с открытым ключом. К этим системам относятся схемы сертификации с помощью службы каталогов и иерархически организованных органов сертификации, неиерархические системы аутентификации ("паутина доверия") и методы личностной криптографии, в которых открытый ключ является неслучайным и не требующим засвидетельствования (self-authenticated).

Недавние достижения в области личностной криптографии с открытым ключом не только обеспечили практичную и удобную аутентификацию (распознавание) открытых ключей, но и открыли возможность для создания новых средств аутентификации: аутентификацию с помощью открытых ключей без применения сертификатов. Система аутентификации без сертификатов обладает рядом интересных и полезных свойств. Она описывается в следующей главе.

Упражнения

13.1. Почему открытый ключ, созданный по закрытому ключу с помощью операции (13.1.1), должен быть сертифицирован?

13.2. Конкретизируйте функцию F, входящую в формулу (13.1.1), для криптографических систем RSA, Рабина и Эль-Гамаля.

13.3. Приводит ли аннулирование сертификата открытого ключа к отзыву цифровой подписи, использованной до момента аннуляции?

13.4. Допустим, что закрытый ключ был создан на основе открытого ключа, как показано в формуле (13.3.1). Почему в этом случае открытый ключ не нуждается в сертификации?

13.5. Почему личностная криптография называется неинтерактивной криптографией с открытым ключом?

13.6. Является ли неинтерактивной криптосистема Жиро с открытыми ключами, не требующими сертификации?

13.7. Пользователь, вовлеченный в протокол обмена ключами криптосистемы Жиро (раздел 13.3.3.4), может отрицать свое участие. Почему?

13.8. Почему суперсингулярные эллиптические кривые, применяемые в криптографии, должны иметь намного больший параметр безопасности, чем обычные эллиптические кривые?

13.9. В отличие от протокола согласования ключей Диффи–Хеллмана (протокол 8.1), протокол обмена ключами в криптосистеме Жиро (раздел 13.3.3.4) и системе SOK (раздел 13.3.5) неуязвим для атаки "человек посередине". Почему?

13.10. Из формулы (13.3.4) следует, что модифицированное спаривание Вейля является симметричным. Может ли обычное спаривание Вейля оказаться симметричным для двух линейно независимых точек?

Подсказка: примените свойство билинейности и тождественности к выражению $e(P + X, P + X)$.

13.11. Систему SOK (раздел 13.3.5) можно рассматривать как неинтерактивный вариант протокола обмена ключами Диффи–Хеллмана. Назовите интерактивный прототип личностной криптосистемы Бонэ–Франклина (алгоритм 13.3.7.1).

Подсказка: обычные криптосистемы с открытым ключом являются интерактивными, т.е отправитель сначала должен получить от адресата открытый ключ и лишь после этого создать и отправить зашифрованный текст.

Часть V

Методы формального доказательства стойкости

Основной причиной отказов вычислительных систем и сетей является их сложность. Криптографические системы (алгоритмы и протоколы), как правило, представляют собой сложные системы, и, следовательно, являются уязвимыми для атак. В то время как обычные вычислительные системы и сети работают в дружественном окружении (пользователи внимательно следят за входными данными программ, чтобы не допустить краха системы), криптографические системы функционируют во враждебной среде. Кроме возможных обычных сбоев, криптографическим системам угрожают преднамеренные атаки, организованные не только внешними интервентами, но и вполне законными пользователями (инсайдерами). Довольно часто криптографические системы, даже если их разрабатывали эксперты, оказываются уязвимыми. Скрытый дефект протокола Нидхема–Шредера, который очень долго не замечали, представляет собой яркий пример ненадежности экспертных оценок (см. главу 2).

Формальные методы системного анализа используют систематические процедуры, строгость которых обеспечивается математическими средствами. Эти процедуры позволяют либо разрабатывать системы, обладающие заранее намеченными свойствами, либо проверять уже существующие системы, чтобы выявить возможные скрытые ошибки. Уязвимость криптографических систем общепризнанна, поэтому в настоящее время практически все исследователи считают, что эти системы должны разрабатываться и анализироваться с помощью формальных методов.

Часть V состоит из четырех глав, посвященных формальным методам разработки и анализа криптографических систем. В главе 14 приводятся формальные определения строгих (т.е. пригодных для реальных приложений) понятий стойкости криптографии с открытым ключом. Это позволяет разрабатывать прикладные криптографические системы, которые намного надежнее учебных. Глава 15 посвящена двум важным криптосистемам с открытым ключом. В главе 16 формулируется понятие сильной стойкости цифровых подписей и описываются методы доказательства сильной стойкости нескольких цифровых схем. В главе 17 вновь рассматриваются протоколы аутентификации, в частности, излагаются формальные методы доказательства их корректности.

Глава 14

Определения формальной и сильной стойкости криптосистем с открытым ключом

14.1 Введение

Секретность — основа криптографии. В главе заново рассматриваются понятия стойкости алгоритмов шифрования с открытым ключом. Позднее мы убедимся, что понятия стойкости, ориентированные на обеспечение секретности, носят более общий характер, чем другие определения.

До сих пор мы убеждали себя в стойкости криптосистем с открытым ключом, используя слабое понятие стойкости, сформулированное в разделе 8.2 (свойство 8.2): атакующий считается пассивным и может взламывать криптосистему только по принципу "все или ничего". Это — типично учебное понятие. Оно непригодно для практического применения.

На практике гораздо вероятнее, что злоумышленник является активным: он может модифицировать зашифрованный текст или каким-то образом вычислить открытый текст и послать результат ничего не подозревающему пользователю для получения *услуг оракула* (oracle service), описанных в разделах 7.8.2.1 и 8.9. Следовательно, понятие стойкости, связанное с пассивным злоумышленником, является недостаточно сильным. Необходимо быть готовым к атаке, организованной активным и коварным Злоумышленником (мощь Злоумышленника описана в разделе 2.3).

Более того, во многих приложениях исходный текст может содержать *априорную* информацию, которую легко угадать. Например, такой *априорной* информацией может быть диапазон зарплат, имя одного из нескольких кандидатов в протоколе голосования, одна из нескольких возможных инструкций и даже однобитовая информация об открытом тексте (например, инструкция ПОКУПАТЬ/ПРОДАВАТЬ, адресованная биржевому брокеру или ОРЕЛ/РЕШКА в про-

токоле "Подбрасывание монеты по телефону" (протокол 1.1). Для того чтобы угадать априорную информацию об исходном тексте, зашифрованном с помощью однонаправленной функции с секретом и алгоритма шифрования с открытым ключом, Злоумышленник может просто заново зашифровать угаданный исходный текст и проверить, соответствует ли полученный результат анализируемому зашифрованному тексту. Следовательно, с точки зрения безопасности принцип "все или ничего" является недостаточно сильным.

Для того чтобы предотвратить атаки, имеющие несколько уровней угрозы, необходимо более строгое понятие стойкости. Во-первых, необходимо правильно *формализовать* задачу. В теории криптографических систем с формально доказуемой стойкостью для моделирования различных атак и сценариев были предложены разнообразные атакующие игры. В этих играх участвуют Злоумышленник и оракул. Правила игры позволяют Злоумышленнику получать от оракула криптографическую помощь по первому требованию. Такую помощь можно рассматривать как "курс обучения" Злоумышленника. Криптосистема считается стойкой по отношению к формальной игровой атаке, если даже после соответствующего "курса обучения" Злоумышленник не может ее взломать.

Второй аспект формального понятия стойкости заключается в строгом измерении успехов Злоумышленника. В теории доказательства формальной стойкости безопасность криптографической системы имеет количественную оценку, связывающую безопасность криптосистемы с определенной неразрешимой задачей из теории вычислительной сложности. Стандартный способ доказательства высокой стойкости системы предусматривает выражение уровня успехов Злоумышленника при попытке взлома криптосистемы в виде определенных величин, измеряющих, *как быстро* и *как часто* Злоумышленнику удается решить одну из трудноразрешимых задач. Это достигается с помощью эффективного математического преобразования или серии преобразований, позволяющих свести предполагаемую успешную атаку к решению одной из трудноразрешимых задач. Поскольку мы убеждены, что решение таких задач выполняется *очень медленно* и *очень редко* приводит к успеху, это позволяет утверждать, что Злоумышленник терпит *неудачу*, пытаясь взломать атакуемую криптосистему.

Вследствие разнообразия сценариев атак и широкого выбора трудноразрешимых задач в теории криптографических систем с формально доказуемой стойкостью используется много жаргонных выражений. С их помощью мы опишем разные свойства стойкости и требования, предъявляемые к системам. Вот несколько примеров.

- Криптосистема X является *семантически стойкой* (semantically secure) по отношению к пассивному соглядатаю, обладающему неограниченными вычислительными ресурсами, однако *уязвима* (malleable) к *атаке на основе подобранного зашифрованного текста* (chosen-ciphertext attack), которую

атакующий может выполнить *во время ленча* (at lunchtime) с помощью калькулятора.

- Схема цифровой подписи Y является стойкой по отношению к фальсификациям с помощью *атаки на основе адаптивно подобранного сообщения* (adaptive chosen-message attack). Формальное доказательство ее стойкости *сводит* задачу успешной фальсификации к задаче дискретного логарифмирования. Эта редукция выполняется в рамках *модели случайного оракула* (random oracle model), в которой фальсификатор может *разветвиться* (be forked) со значимой вероятностью.
- Схема шифрования подписи Z является стойкой в смысле неразличимости (indistinguishablity) шифрования, подвергающегося *атаке на основе адаптивно подобранного зашифрованного текста* (adaptive chosen-ciphertext attack), и невозможности подделать подпись с помощью атаки на основе адаптивно подобранного сообщения, независимо от того, когда осуществляется атака: *за завтраком*, *в полночь* (at midnight) или *пополуночи* (in the small hours). Формальное доказательство стойкости связано с факторизацией целого числа.
- Протокол электронного аукциона П является стойким, позволяя покупателю *отрицать* свое участие в протоколе, а победителю — оставаться неузнанным. Формальное доказательство стойкости протокола связано со *стандартным предположением о неразрешимости задачи* (standard intractability assumption) — задачей принятия решений Диффи–Хеллмана.

В ходе исследования формально доказуемой стойкости в этой и следующих трех главах каждое использованное жаргонное выражение будет подробно разъясняться. После этих объяснений приведенные выше утверждения станут намного понятнее.

14.1.1 Структурная схема главы

Раздел 14.2 начинается с введения в основную тему этой главы: формальное доказательство стойкости с помощью формального моделирования атак и точного измерения их последствий. Формальное доказательство продемонстрирует неадекватность принципа “все или ничего”, введенного в главе 8. Более сильное понятие “семантической стойкости”, означающее сокрытие любой частичной информации о сообщении, будет сформулировано в разделе 14.3. Неадекватность понятия семантической стойкости будет показана в разделе 14.4. Это доказательство приводит к новым понятиям: “стойкости относительно подобранного зашифрованного текста” (“chosen-ciphertext security”), “стойкости относительно адаптивно подобранного зашифрованного текста” (“adaptive chosen-ciphertext security”) и “неуязвимости” (“non-malleability”). Эти понятия и связи между ними будут изучены в разделе 14.5.

14.2 Формальное доказательство стойкости

Абстрактно говоря, "формально доказуемая стойкость" криптографической системы представляет собой *меру* устойчивости системы к атакам. Система называется стойкой, если Злоумышленник не может взламывать ее *слишком часто* или *слишком быстро*. Следовательно, для измерения стойкости системы необходимо оценивать вероятность взлома и объем вычислительных затрат.

Чтобы конкретизировать наши рассуждения, рассмотрим "игровую атаку", которая разыгрывается между Злоумышленником и "оракулом". Роль оракула исполняет наивный пользователь криптографической системы, не способный отказаться от взаимодействия со Злоумышленником. Эта игра позволит продемонстрировать вычислительные аспекты, связанные с понятием стойкости. Кроме того, она является первым шагом на пути усиления понятия стойкости криптосистем по сравнению со слабым понятием, введенным в разделе 8.2 (свойство 8.2).

Допустим, что Злоумышленник организовывает атаку, а $\mathcal{O}$ — оракул, принимающий участие в игре. Целью Злоумышленника является взлом криптосистемы. Следовательно, в результате игры с криптосистемой должно случиться "нечто ужасное", что позволит Злоумышленнику преодолеть защиту.

Для описания криптосистемы будем использовать определение 7.1 из раздела 7.2, т.е. будем считать, что она состоит из алгоритма шифрования $\mathcal{E}$, пространства исходных текстов $\mathcal{M}$ и пространства зашифрованных текстов $\mathcal{C}$. Однако следует сделать одно важное замечание: алгоритм шифрования $\mathcal{E}$ считается вероятностным, т.е. содержит внутреннюю случайную операцию, имеющую некое распределение вероятностей. Таким образом, зашифрованный текст представляет собой случайную величину, имеющую то же самое распределение. Например, если исходное сообщение дважды шифруется с помощью одного и того же ключа шифрования, с большой вероятностью возникнут два разных зашифрованных текста (поскольку алгоритм шифрования является однозначным преобразованием).

Игровая атака описывается протоколом 14.1.

В этой игре оракул $\mathcal{O}$ предлагает Злоумышленнику ответить на следующий вопрос.

Результатом какого из экспериментов, $\mathcal{E}_{kl}(m_0)$ или $\mathcal{E}_{kl}(m_1)$, является оклик c^*?

Будем считать, что Злоумышленник применяет вероятностный полиномиальный алгоритм распознавания, описанный в определении 4.14 (раздел 4.7). Этот алгоритм считается вероятностным не только потому, что результат работы оракула $\mathcal{O}$ является случайным, но и потому, что Злоумышленник является *полиномиально ограниченным* и, следовательно, может использовать вероятностный полиномиальный алгоритм (PPT), если будет уверен, что такой алгоритм более эффективен, чем детерминированный. (Как правило, это предположение выполняется (см. главу 4).) Обозначим через Adv преимущество Злоумышленника, с которым он может распознавать различие между зашифрованными текстами. По определе-

Протокол 14.1. Атака на основе неразличимых подобранных исходных текстов

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ:

а) Злоумышленник и оракул $\mathcal{O}$ действуют в заданной криптосистеме, состоящей из алгоритма шифрования $\mathcal{E}$, пространства исходных сообщений $\mathcal{M}$ и пространства зашифрованных текстов $\mathcal{C}$;

б) оракул $\mathcal{O}$ владеет фиксированным ключом шифрования ke.

1. Злоумышленник подбирает разные сообщения $m_0, m_1 \in \mathcal{M}$ и посылает их оракулу $\mathcal{O}$.
   (* Сообщения m_0 и m_1 называются **подобранным исходным текстом** (chosen plaintext messages). Злоумышленник пока находится на **этапе поиска** ("find-stage") сообщений m_0 и m_1. Разумеется, он готовит их таким образом, чтобы легко распознать результат их шифрования. *)

2. Если эти сообщения имеют разную длину, оракул $\mathcal{O}$ дополняет более короткое из них, чтобы их длины стали одинаковыми.
   (* Например, для дополнения сообщений оракул может использовать фиктивную строку 0^d, где d — разность между длинами сообщений. *)
   Оракул $\mathcal{O}$ подбрасывает идеальную монету $b \in_U \{0,1\}$ и выполняет следующую операцию шифрования

$$c^* = \begin{cases} \mathcal{E}_{ke}(m_0), & \text{если } b = 0, \\ \mathcal{E}_{ke}(m_1), & \text{если } b = 1. \end{cases}$$

   Оракул $\mathcal{O}$ посылает Злоумышленнику сообщение $c^* \in \mathcal{C}$.
   (* Зашифрованное сообщение c^* называется *зашифрованным текстом окли-ка* (challenge ciphertext). В теории доказуемой стойкости зашифрованный текст, сопровождаемый звездочкой, всегда считается окликом. *)
   (* Следует помнить, что зашифрованный текст c^* является случайной величиной, зависящей от двух случайных входных величин: идеальной монеты b и внутренней случайной операции алгоритма $\mathcal{E}$. *)

3. Получив зашифрованный текст c^*, Злоумышленник должен угадать результат подбрасывания монеты, послав оракулу 0 или 1.
   (* Теперь Злоумышленник находится на **этапе осмысленного угадывания** ("guess-stage" for an educated guess), пытаясь угадать результат жеребьевки, проведенной оракулом $\mathcal{O}$. Ответы, отличающиеся от 0 и 1, не допускаются. *)

нию 4.14 преимуществом называется разность между вероятностями, с которыми Злоумышленник может распознавать эксперименты $\mathcal{E}_{ke}(m_0)$ и $\mathcal{E}_{ke}(m_1)$.

$$\begin{aligned}\text{Adv} = |\,&\text{Prob}[0 \leftarrow \text{Злоумышленник}(c^* = \mathcal{E}_{ke}(m_0))]- \\ &- \text{Prob}[0 \leftarrow \text{Злоумышленник}(c^* = \mathcal{E}_{ke}(m_1))]|.\end{aligned} \tag{14.2.1}$$

Вероятностное пространство должно содержать случайный выбор, сделанный оракулом $\mathcal{O}$ и Злоумышленником, а также внутреннюю случайную операцию алгоритма шифрования. Отметим, что ответ Злоумышленника зависит не только от зашифрованного текста оклика c^*, но и от двух подобранных исходных сообщений (m_0, m_1). Только поэтому его ответ можно считать результатом "осмысленного угадывания" ("educated guess"). Однако для простоты изложения мы не будем указывать исходные сообщения (m_0, m_1) во входной информации, поступающей к Злоумышленнику.

Следует заметить, что Злоумышленник имеет дополнительную возможность для того, чтобы улучшить результаты своего угадывания: оракул $\mathcal{O}$ подбрасывает идеальную монету. Формула вычисления преимущества (14.2.1) не указывает явно на то, как Злоумышленник может воспользоваться этим фактом, хотя *неявно* известно, что каждая из вероятностей, входящих в формулу (14.2.1), никогда не превысит $\frac{1}{2}$, например, вероятность события $c^* = \mathcal{E}_{ke}(m_0)$ составляет ровно $\frac{1}{2}$. Необходимо явно указать, как именно Злоумышленник использует дополнительный ключ в формуле вычисления преимущества. Применяя условную вероятность (определение 3.3 из раздела 3.4.1) и замечая, что вероятность обоих результатов при подбрасывании идеальной монеты равна $\frac{1}{2}$, можно переписать формулу (14.2.1) в следующем виде.

$$\begin{aligned}\text{Adv} = \Big|&\frac{1}{2}\,\text{Prob}[0 \leftarrow \text{Злоумышленник}(c^*) \mid c^* = \mathcal{E}_{ke}(m_0)]- \\ &-\frac{1}{2}\,\text{Prob}[0 \leftarrow \text{Злоумышленник}(c^*) \mid c^* = \mathcal{E}_{ke}(m_1)]\Big|.\end{aligned} \tag{14.2.2}$$

По правилам игры Злоумышленнику не разрешается давать ответы, отличающиеся от 0 и 1, следовательно, неправильный ответ является событием, дополнительным по отношению к правильному ответу. Применяя свойство 5 (раздел 3.3), получаем следующую формулу.

$$\begin{aligned}\text{Adv} = \Big|&\frac{1}{2}\,\text{Prob}[0 \leftarrow \text{Злоумышленник}(c^*) \mid c^* = \mathcal{E}_{ke}(m_0)]- \\ &-\frac{1}{2}(1 - \text{Prob}[0 \leftarrow \text{Злоумышленник}(c^*) \mid c^* = \mathcal{E}_{ke}(m_0)])\Big|.\end{aligned}$$

Иначе говоря,

$$\text{Prob}[0 \leftarrow \text{Злоумышленник}(c^*) \mid c^* = \mathcal{E}_{ke}(m_0)] = \frac{1}{2} \pm \text{Adv}\,. \tag{14.2.3}$$

Формула (14.2.3) часто применяется для выражения преимущества алгоритма при угадывании результатов подбрасывания идеальной монеты (вероятность $\frac{1}{2}$). Итак, если Adv = 0, то случайный ответ Злоумышленника имеет точно такое же распределение, как и результаты подбрасывания идеальной монеты. Однако следует учитывать, что преимущество всегда больше нуля. Очевидно, что формула (14.2.3) также справедлива, если в результатах жеребьевки все нули заменить единицами.

Из формулы (14.2.3) следует, что преимущество Злоумышленника никогда не превышает $\frac{1}{2}$, поскольку вероятность всегда лежит в интервале [0,1]. Действительно, если оракул $\mathcal{O}$ шифрует оба исходных текста с вероятностью $\frac{1}{2}$, преимущество Adv, заданное формулой (14.2.1), представляет собой разность вероятностей *совместных* событий и никогда не превышает $\frac{1}{2}$. Читатель может заметить, что формула (14.2.3) выглядит так, будто оракул $\mathcal{O}$ подбрасывает *несимметричную* монету, например, оракул с вероятностью $\frac{1}{4}$ шифрует сообщение m_0 и с вероятностью $\frac{1}{4}$ — сообщение m_1. Подсказка: замените число $\frac{1}{2}$ в формуле (14.2.2) соответствующими смещенными вероятностями и посмотрите, как изменится формула (14.2.3). Впоследствии будет показано, что преимущество Злоумышленника может превышать $\frac{1}{2}$, *если* оракул $\mathcal{O}$ подбрасывает несимметричную монету.

Будем говорить, что исследуемая криптосистема устойчива по отношению к игровой атаке, описанной протоколом 14.1, если эксперимент $\mathcal{E}_{kd}(m_0)$ неотличим от эксперимента $\mathcal{E}_{kd}(m_1)$. В соответствии с определением 4.15 (раздел 4.7) это означает, что при любом значимом преимуществе Adv не должно существовать ни одного полиномиального алгоритма распознавания. Иначе говоря, преимущество, с которым Злоумышленник может различать результаты шифрования, должно быть пренебрежимо малой величиной по сравнению с параметром безопасности схемы шифрования, которым, как правило, является длина ключа шифрования. Можно считать, что преимущество любого полиномиального алгоритма распознавания, применяемого Злоумышленником, задается медленно растущей функцией, зависящей от объема вычислительных ресурсов. Здесь термин "медленно растущая" означает, что, даже если Злоумышленник резко увеличит объем своих вычислительных ресурсов, преимущество увеличится крайне незначительно. Именно это мы имели в виду, когда в начале главы указывали, что Злоумышленник не должен взламывать криптосистему слишком часто или слишком быстро.

Из аргумента, сопровождающего определение **полиномиально неразличимых ансамблей** (определение 4.15), следует, что введенное выше понятие стойкости можно назвать **полиномиальной неразличимостью шифрования** (polinomial inditinguishability of encryption). Более того, поскольку оба исходных текста, подобранных Злоумышленником, являются неразличимыми, точное название формулируется так: **стойкость к атаке на основе полиномиально неразличимых подобранных исходных текстов** (security against polynomially indinguihable

chosen-plaintext attack, IND-SPA), или сокращенно: **стойкость IND-CPA** (security IND-CPA).

Несмотря на то что в рамках протокола 14.1 Злоумышленник свободен в выборе исходных сообщений, основную трудность при взломе криптосистемы по принципу "все или ничего" представляет однобитовый ответ на вопрос о подобранном зашифрованном тексте: "Какое сообщение содержится в зашифрованном тексте: m_0 или m_1?". Действительно, все учебные алгоритмы шифрования с открытым ключом, рассмотренные до сих пор, уязвимы для атаки IND-CPA. Легко проверить, что это относится и к криптосистемам RSA и Рабина, поскольку они являются детерминированными и, следовательно, позволяют Злоумышленику восстановить сообщения m_0 и m_1 путем повторного шифрования. В разделе 14.3.5 будет показано, что криптосистема Эль-Гамаля, предусматривающая вероятностный алгоритм шифрования, также уязвима для атаки IND-CPA.

По мере роста вычислительных ресурсов и упрощения способов атаки стойкость криптосистемы следует увеличивать. Методы повышения стойкости криптосистем излагаются в следующей главе.

14.3 Семантическая стойкость — введение в теорию доказуемой стойкости

Понятие стойкости по отношению к атаке IND-CPA впервые было предложено Голдвассером и Микали [125]. Они назвали это понятие **семантической стойкостью** (semantic security). Оно заключается в том, что зашифрованный текст не допускает никакой утечки полезной информации об исходном тексте (если не считать полезной информацией длину самого исходного текста) ни одному взломщику, обладающему полиномиально ограниченными вычислительными ресурсами. Голдвассер и Микали обнаружили, что во многих приложениях сообщения могут содержать *априорную* информацию, полезную для организации атак. Например, зашифрованный текст может содержать только одну простую инструкцию, скажем, "ПОКУПАТЬ" или "ПРОДАВАТЬ", либо имя одного из нескольких кандидатов при голосовании. Голдвассер и Микали указали на то, что криптосистемы с открытым ключом, основанные на непосредственном применении однонаправленных функций с секретом, как правило, очень слабо скрывают такие сообщения. В дальнейшем будет показано, что их критика относится и к системам, описанным в главе 8.

Назрела необходимость ввести более строгое понятие стойкости. Отказ протокола "мысленного покера" является хорошей иллюстрацией слабости криптосистем с открытым ключом, основанных на применении однонаправленных функций с открытым ключом. Для начала рассмотрим протокол мысленного покера, предложенный Шамиром, Ривестом и Адлеманом [261].

14.3.1 Протокол мысленного покера SRA

Предположим, что Алиса живет в Нью-Йорке, а Боб — в Лондоне. Они никогда не встречались, но желают сыграть в покер, оставаясь в своих городах. Это стало возможным благодаря изобретателям алгоритма RSA — Шамиру, Ривесту и Адлеману, предложившим "протокол мысленного покера SRA" [261].

В мысленный покер играют по тем же правилам, что и в обычный, однако карты кодируются в сообщениях, поэтому в мысленный покер можно играть с помощью средств связи. Для того чтобы сыграть в покер, Алиса и Боб сначала должны честно раздать карты. Слово "честно" означает, что при раздаче карт Алиса и Боб должны придерживаться четырех правил.

1. Все карты должны раздаваться с одинаковой вероятностью (т.е. равномерно), причем одна и та же карта не должна оказаться у двух разных игроков одновременно.
2. Алиса и Боб должны знать карты, которыми они владеют, а другие игроки не должны знать, какие карты находятся на руках их партнеров.
3. Алиса и Боб должны подозреваться в мошенничестве и нарушении правил протокола.
4. Алиса и Боб должны иметь возможность проверять, честно ли была сыграна предыдущая игра.

Идея, лежащая в основе мысленного покера, заключается в применении шифра, обладающего свойством коммутативности. Применяя такой шифр, Алиса и Боб могут дважды шифровать сообщение, используя свои секретные ключи, и должны дважды расшифровывать его. Пусть

$$C = E_X(M) \text{ и } M = D_X(C)$$

обозначают алгоритмы шифрования и расшифровки, выполняемые пользователем X. Коммутативное свойство шифра состоит в том, что для любого сообщения M выполняются следующие уравнения.

$$M = D_A(D_B(E_A(E_B(M)))) = D_B(D_A(E_B(E_A(M)))). \qquad (14.3.1)$$

Иначе говоря, исходное сообщение можно восстановить путем двойной расшифровки независимо от порядка шифрования.

Для простоты предположим, что Алиса и Боб решили раздать по одной карте, используя колоду, состоящую из трех карт. Способ честной раздачи карт описывается протоколом 14.2. Совершенно очевидно, что его можно распространить на любое количество карт.

Протокол 14.2. Протокол честной раздачи карт при игре в мысленный покер

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ:

Алиса и Боб согласовали коммутативный шифр, обладающий свойством (14.3.1), и сгенерировали свои секретные ключи шифрования.

Колода состоит из трех карт: M_1, M_2 и M_3.

ЦЕЛЬ: Честная раздача по одной карте каждому игроку по правилам 1–4.

1. Алиса шифрует три карты следующим образом: $C_i = E_A(M_i)$, где $i = 1, 2, 3$. Она посылает Бобу эти три зашифрованных текста в случайном порядке.
   (* Передача зашифрованных текстов в случайном порядке эквивалентна перетасовке колоды. *)
2. Боб случайным образом извлекает один из зашифрованных текстов. Обозначим его через C. Затем он дважды шифрует его как $CC = E_B(C)$, случайным образом извлекает другой текст, обозначенный как C', и отсылает зашифрованные тексты CC и C' Алисе.
   (* Символы CC обозначают карту, принадлежащую Бобу, а символ C' — карту, выданную Алисе. Оставшаяся зашифрованная карта игнорируется. *)
3. Алиса расшифровывает тексты CC и C'. Результат расшифровки текста C' обозначает ее карту, а расшифровка текста CC, обозначаемая как C'', — карту Боба. Текст C'' возвращается Бобу.
4. Боб расшифровывает текст C'' и выясняет, какую карту он получил.
   (* Теперь можно приступать к игре. *)

14.3.2 Анализ стойкости с точки зрения учебной криптографии

Предположим на время, что в криптосистеме, описанной протоколом 14.2, обе операции однократного и двойного шифрования являются достаточно стойкими. Говоря, что эта криптосистема является достаточно стойкой, мы имеем в виду, что обладая заданным исходным текстом (соответственно зашифрованным текстом), но не зная подлинного ключа шифрования (соответственно расшифровки), полиномиально ограниченный Злоумышленник не может создать правильный зашифрованный текст (соответственно не может восстановить исходное сообщение). Такое понятие стойкости полностью соответствует принципу "все или ничего", на котором построена учебная криптография (см. раздел 8.2). Проведем анализ стойкости протокола 14.2, придерживаясь принципов учебной криптографии и учитывая правила честной игры в мысленный покер.

После выполнения протокола 14.2 складывается следующая ситуация.

- Поскольку на первом шаге Алиса тасует колоду, оба игрока с одинаковой вероятностью получают одну из карт, принадлежащих множеству $\{M_1, M_2, M_3\}$. Обратите внимание на то, что Алиса стремится хорошо перетасовать колоду, чтобы Боб не получил преимущества. Итак, первое условие честной раздачи выполнено.
- Каждый из двух игроков после двойной расшифровки узнает свою карту, но не знает, какая карта досталась сопернику. Таким образом, второе правило честной раздачи также выполнено.
- Очевидно, что в протоколе учтена возможность мошенничества со стороны игроков. Следовательно, третье правило честной раздачи также выполнено.

Выполнение четвертого правила зависит от того, допускает ли криптосистема, использованная в протоколе, честную верификацию оконченной игры. Шамир и его соавторы предложили применить один из вариантов криптосистемы RSA (см. раздел 8.5), в котором обе стороны до завершения игры сохраняют в тайне показатели степени, использованные при шифровании и расшифровке, а после ее окончания раскрывают их для проверки.

Пусть N — общий модуль в криптосистеме RSA. Алиса и Боб знают разложение числа N на множители. Обозначим через (e_A, d_A) показатели степеней, использованные при шифровании и расшифровке Алисой, а через (e_B, d_B) — показатели, применяемые Бобом. Знание разложения числа N на простые множители позволяет Алисе (Бобу) найти число d_A по заданному числу e_A (число d_B — по заданному числу e_B) с помощью сравнения

$$e_X d_X \equiv 1 (\mathrm{mod}\, \phi(N)), \tag{14.3.2}$$

где символ X обозначает игрока A или B. Для пользователя X получаем следующие формулы.

$$E_X(M) = M^{ex} \,(\mathrm{mod}\, N),$$
$$D_X(C) = C^{dx} \,(\mathrm{mod}\, N).$$

Поскольку группа RSA является коммутативной, легко видеть, что условие (14.3.1) выполняется. До завершения игры обе стороны сохраняют свои показатели степеней в тайне. Таким образом, никто не может создать зашифрованный текст, идентичный тексту, созданному партнером. Это не позволяет одному партнеру раскрыть карты другого партнера, получив зашифрованный текст. Кроме того, никто не может расшифровать текст, созданный другим партнером. Таким образом, криптосистема действительно является "достаточно стойкой".

Очевидно, что после завершения игры обе партии могут раскрыть свои показатели степеней и убедиться, что шифрование и расшифровка были выполнены правильно. Итак, четвертое правило честной раздачи карт выполнено.

В проведенном анализе использовалось не вполне адекватное и разумное понятие стойкости: "достаточная стойкость" криптосистемы означает, что Злоумышленник не способен создать правильный зашифрованный текст по заданному исходному тексту, не зная правильного ключа шифрования, либо расшифровать текст, не зная правильного ключа расшифровки. Неадекватность и неразумность такого понятия стойкости совершенно очевидны. Липтон (Lipton) [178] доказал, что протокол 14.2, основанный на оригинальной криптосистеме RSA, не обеспечивает честной раздачи карт при игре в мысленный покер. Это происходит потому, что криптосистема RSA не позволяет скрыть определенную *априорную* информацию об исходных сообщениях. В данном случае *априорная* информация такова: исходное сообщение является квадратичным вычетом. Как показано в разделе 6.5, число a представляет собой квадратичный вычет по модулю n, если выполняется условие $\gcd(a, N) = 1$ и существует число $x < N$, такое что

$$x^2 \equiv a (\mathrm{mod}\, N).$$

Следует заметить, что, поскольку число $\phi(N)$ является четным, показатель степени e, используемый при шифровании, и показатель степени d, применяемый при расшифровке, удовлетворяющие уравнению (14.3.2), должны быть нечетными. Следовательно, исходный текст M является квадратичным вычетом по модулю N, т.е. $M \in \mathrm{QR}_N$ тогда и только тогда, когда соответствующий зашифрованный текст удовлетворяет условию $C \in \mathrm{QR}_N$, поскольку

$$C \equiv M^e \equiv (x^2)^e \equiv (x^e)^2 (\mathrm{mod}\, N)$$

для некоторого числа $x < N$. Иначе говоря, шифрование по алгоритму RSA не может отменить тот факт, что исходное сообщение является квадратичным вычетом. Следовательно, зная разложение числа N, легко выбрать зашифрованный текст $C \in \mathrm{QR}_N$: сначала применить операцию деления числа C по модулю, равному каждому простому множителю числа N, затем вычислить символ Лежандра, используя алгоритм 6.2.

Следовательно, если одни исходные тексты принадлежат множеству QR_N, а другие — нет, то партнер, знающий трюк Липтона, получит нечестное преимущество в игре: он точно знает, какие карты не были вообще ни разу зашифрованы.

Итак, игра в мысленный покер не является стойкой. Точнее говоря, протокол 14.1 не является стойким по отношению к атаке IND-CPA.

14.3.3 Вероятностное шифрование Голдвассера и Микали

Протокол 14.2 можно защитить от атаки Липтона. Например, все карты можно выбирать из множества QR_N. Однако Голдвассер и Микали предложили решить

гораздо более сложную проблему, введя более строгое понятие стойкости — семантическую стойкость.

Свойство 14.1 (Семантическая стойкость). Все элементы открытого текста, которые можно эффективно вычислить по заданному зашифрованному тексту, можно эффективно вычислить и без него.

Голдвассер и Микали предложили схему **вероятностного шифрования** (probabilistic encryption), обладающую этим свойством. Изобретенная ими схема называется криптосистемой GM. Она шифрует все сообщение бит за битом, причем вся сложность, связанная с поиском отдельного зашифрованного бита в тексте c, заключается в проверке, принадлежит число c множеству QR_N или множеству $J_N(1)\backslash\mathrm{QR}_N$, где $J_N(1) = \left\{x \mid x \in \mathbb{Z}_N^*, \left(\frac{x}{N}\right) = 1\right\}$.

Криптосистема GM описывается алгоритмом 14.1.

Покажем, что система, определенная алгоритмом 14.1, является криптографической, т.е., выполняя расшифровку, Алиса действительно восстанавливает подлинное исходное сообщение, отправленное Бобом.

Анализируя алгоритм шифрования, легко видеть, что нулевой бит исходного сообщения шифруется числом, принадлежащим множеству QR_N.

Если бит исходного текста равен 1, соответствующий зашифрованный текст имеет вид $c = yx^2$. Учитывая, что $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$, и мультипликативное свойство символа Лежандра (теорема 6.16 из раздела 6.5.2), получаем следующее.

$$\left(\frac{c}{p}\right) = \left(\frac{yx^2}{p}\right) = \left(\frac{y}{p}\right)\left(\frac{x^2}{p}\right) = (-1) \times 1 = -1$$

и

$$\left(\frac{c}{q}\right) = \left(\frac{yx^2}{q}\right) = \left(\frac{y}{q}\right)\left(\frac{x^2}{q}\right) = (-1) \times 1 = -1.$$

Следовательно,

$$\left(\frac{c}{N}\right) = \left(\frac{c}{p}\right)\left(\frac{c}{q}\right) = (-1) \times (-1) = 1.$$

Иначе говоря, единица в исходном тексте превращается в зашифрованный текст, принадлежащий множеству $J_N(1)\backslash\mathrm{QR}_N$.

Алгоритм расшифровки работает правильно, поскольку, зная числа p и q, Алиса может определить, какому множеству принадлежит зашифрованный текст $c_i : \mathrm{QR}_N$ или $J_N(1)\backslash\mathrm{QR}_N$, и правильно восстановить исходное сообщение бит за битом.

Нетрудно убедиться, что для шифрования сообщения, состоящего из ℓ бит, необходимо выполнить $O(\ell(\log_2 N)^2)$ побитовых операций. Это выражение представляет собой оценку временно́й сложности алгоритма. Степень расширения

Алгоритм 14.1. Вероятностная криптосистема Голдвассера и Микали

Генерация ключа

Чтобы установить параметры ключа, Алиса должна выполнить следующие операции.

1. Выбрать два случайных простых числа p и q, удовлетворяющих условию $|p| = |q| = k$.
   (* Например, применив алгоритм 4.7 в строке 1^k. *)
2. Вычислить значение $N = pq$.
3. Извлечь случайное целое число y, удовлетворяющее условию $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$.
   (* Таким образом, $y \in J(N) \backslash \mathrm{QR}_N$. *)
4. Огласить пару (N, y) в качестве открытого ключа, а пару (p, q) сохранить в тайне как закрытый ключ.

Шифрование

Чтобы послать Алисе строку $m = b_1 b_2 \ldots b_\ell$, Боб выполняет следующие операции.

for $(i = 1, 2, \ldots, \ell)$
{
$x \leftarrow_U \mathbb{Z}_N^*$;
if $(b_i == 0) c_i \leftarrow x^2 (\mathrm{mod}\, N)$;
else $c_i \leftarrow y x^2 (\mathrm{mod}\, N)$;
}

Боб посылает Алисе сообщение: $E_N(m) \leftarrow (c_1, c_2, \ldots, c_\ell)$.

Расшифровка

Получив кортеж $(c_1, c_2, \ldots, c_\ell)$, Алиса выполняет следующие операции.

for $(i = 1, 2, \ldots, \ell)$
{
if $(c_i \in \mathrm{QR}_N) b_i \leftarrow 0$;
else $b_i \leftarrow 1$;
}

set $m \leftarrow (b_1, b_2, \ldots, b_\ell)$.

этого алгоритма шифрования равна $\log_2 N$: одному биту исходного текста соответствуют $\log_2 N$ бит зашифрованного текста.

Поскольку для вычисления символа Лежандра по модулю p и по модулю q при условии, что $|p| = |q| = k$, необходимо выполнить $O_B(k^2)$ побитовых операций (см. обсуждение алгоритма 6.2 для вычисления символа Якоби), для расшифровки кортежа $(c_1, c_2, \ldots, c_\ell)$ требуются $O_B(\log_2 N)^2)$ побитовых операций. Это выражение представляет собой оценку временно́й сложности расшифровки.

Побитовое шифрование, применяемое в криптосистеме GM, свидетельствует о ее крайней неэффективности.

14.3.4 Стойкость криптосистемы GM

Алгоритм шифрования в криптосистеме GM можно рассматривать как безошибочный рандомизированный алгоритм: случайные операции в алгоритме шифрования не могут исказить зашифрованный текст и обладают при этом следующим важным свойством.

Нулевые биты в исходном тексте равномерно распределяются по множеству QR_N, а единичные — по множеству $J_N(1)\backslash\mathrm{QR}_N$.

Оба распределения являются равномерными, поскольку для нулевого бита, содержащегося в исходном тексте, возведение в квадрат означает отображение группы $\mathbb{Z}_N^*$ на множество QR_N, а для единичного бита умножение элемента множества QR_N на число y является отображением из множества QR_N на множество $J_N(1)\backslash\mathrm{QR}_N$. Итак, извлечение случайного числа $x \in_U \mathbb{Z}_N^*$ в ходе шифрования означает либо извлечение случайного числа, равномерно распределенного по множеству QR_N, если бит в исходном тексте равен нулю, либо извлечение случайного числа, равномерно распределенного по множеству $J_N(1)\backslash\mathrm{QR}_N$, если бит в исходном тексте равен единице.

Чтобы формализовать эту процедуру, отметим, что основная трудность в криптосистеме GM связана с решением задачи о распознавании квадратичных вычетов (QR-problem), описанной в определении 6.2 (раздел 6.5.1). Задача QR является общепризнанной трудноразрешимой задачей теории чисел.

Предположение 14.1 (Предположение о квадратичных вычетах (предположение QR)). *Пусть $\mathcal{IG}$ — генератор целых чисел, на вход которого поступает строка 1^k, время работы полиномиально зависит от числа k, а результатом работы является модуль $N = pq$, состоящий из $2k$ бит, где числа p и q — k-битовые равномерно распределенные нечетные простые числа.*

Генератор $\mathcal{IG}$ удовлетворяет условию о квадратичных вычетах (QR), *если для всех достаточно больших чисел k и модуля $N \leftarrow \mathcal{IG}(1^k)$ ансамбли QR_N и $J_N(1)\backslash\mathrm{QR}_N$ являются полиномиально неразличимыми в смысле определения 4.14 из раздела 4.7.*

Очевидно, что открытый ключ N необходимо выбирать на пределе возможностей решения задачи о распознавании квадратичных вычетов, поскольку Злоумышленник может знать разложение числа N на простые множители и применять алгоритм расшифровки из криптосистемы GM для решения задачи QR. Следовательно, криптосистема GM основана на предположении, что атакующий использует полиномиально ограниченный алгоритм. По этой причине семантиче-

скую стойкость алгоритмов шифрования часто называют **полиномиальной неразличимостью шифрования** (polinomial indistinguishability of encryption).

Если предположение о квадратичных вычетах действительно выполняется, то алгоритм шифрования в криптосистеме GM равномерно распределяет биты исходного текста по пространству зашифрованных текстов $J_N(1)$. Равномерное распределение зашифрованных текстов означает, что попытка угадать исходный текст по соответствующему тексту является совершенно бессмысленной. Именно в этом заключается суть понятия семантической стойкости, предложенного Голдвассером и Микали.

Определение 14.1 (Семантическая стойкость, стойкость по отношению к атаке на основе неразличимых подобранных исходных текстов (стойкость к атаке IND-CPA)). *Криптосистема с параметром безопасности k называется семантически стойкой (стойкой к атаке IND-CPA), если после игровой атаки, описанной в протоколе 14.1, организованной полиномиально ограниченным Злоумышленником, преимущество, заданное формулой (14.2.3), является пренебрежимо малой величиной по сравнению с числом k.*

Итак, сформулируем основной результат о стойкости криптосистемы GM.

Теорема 14.1. *Пусть k — размер двух простых множителей модуля N в криптосистеме RSA. Криптосистема GM с параметром безопасности k является семантически стойкой (стойкой к атаке IND-CPA) тогда и только тогда, когда выполняется предположение о квадратичных вычетах.* □

14.3.5 Семантически стойкий вариант криптосистемы Эль-Гамаля

Как и криптосистема RSA, криптосистема Эль-Гамаля, описанная алгоритмом 8.3, не скрывает того, что исходный текст является квадратичным вычетом, поскольку алгоритм установки открытых параметров (p, g) организован так, чтобы число g было порождающим элементом всей группы $\mathbb{Z}_p^*$. В этом случае тот факт, что исходный текст является квадратичным вычетом, может отразиться на зашифрованном тексте.

Пример 14.1. Допустим, что оракул $\mathcal{O}$ устанавливает компоненты открытого ключа (p, g, y) для криптосистемы Эль-Гамаля, описанной алгоритмом 8.3. По критерию Эйлера (теорема 6.13 из раздела 6.5.1) $g \in \mathrm{QNR}_p$ (т.е. число g является квадратичным невычетом по модулю p).

Предположим, что Злоумышленник организовал атаку IND-CPA. Он должен послать сообщения $m_0 \in \mathrm{QR}_p$ и $m_1 \in \mathrm{QNR}_p$, применяя алгоритм 6.2. Для Злоумышленника это не сложно. Пусть (c_1^*, c_2^*) — пара зашифрованных окликов, воз-

вращенных оракулом $\mathcal{O}$. Тогда

$$c_2^* = \begin{cases} y^k m_0 (\mathrm{mod}\, p) & \text{с вероятностью 50\%,} \\ y^k m_1 (\mathrm{mod}\, p) & \text{с вероятностью 50\%.} \end{cases}$$

Теперь Злоумышленник может распознать исходный текст, проверив, являются ли квадратичными вычетами числа y, c_1^* и c_2^*. Необходимо рассмотреть несколько случаев.

Сначала проанализируем вариант, когда $y \in \mathrm{QR}_p$. Этот случай весьма прост. Вследствие мультипликативного свойства символа Лежандра (теорема 6.16.2 из раздела 6.5.2) исходный текст равен m_0 тогда и только тогда, когда $c_2^* \in \mathrm{QR}_p$.

Вариант $y \in \mathrm{QNR}_p$ распадается на два очень простых частных случая. В первом случае, если $c_1^* \in \mathrm{QR}_p$, то $y^k \in \mathrm{QR}_p$ (поскольку число k является четным), и решение принимается точно так же, как в предыдущем абзаце. Второй случай, когда $c_1^* \in \mathrm{QNR}_p$, читатели могут разобрать самостоятельно (учтите, что число k теперь является нечетным). □

Как всегда, осознание проблемы позволяет ее более или менее легко решить. Если ограничить криптосистему множеством квадратичных вычетов по модулю p, то атака, описанная в примере 14.1, станет невозможной.

Алгоритм 14.2. Семантически стойкий вариант криптосистемы Эль-Гамаля

Установка открытых параметров

Пусть G — абелева группа, имеющая следующее описание.

1. Найти случайное простое число q, удовлетворяющие условию $|q| = k$.
2. Проверить простоту числа $p = 2q + 1$. Если число p — простое, вернуться к п. 1.
3. Извлечь случайный порождающий элемент $h \in \mathbb{Z}_p^*$. Установить параметр $g = h^2 (\mathrm{mod}\, p)$.
4. Пусть $\mathrm{desc}(G)$ удовлетворяет условию $G = \langle g \rangle$.
   (* Группа порождается элементом g (см. определение 5.10 в разделе 5.2.3). *)
5. Пусть (p, g) — открытые параметры криптосистемы Эль-Гамаля.
6. Пусть G — пространство исходных текстов.

(* Остальная часть совпадает с алгоритмом 8.3. *)

Прежде всего следует отметить, что алгоритм 14.2 обязательно завершит работу, поскольку существует огромное число простых чисел p, таких что число $(p-1)/2$ также является простым (например, 7, 11, 23, 39, 47 и т.д.). Такие числа называются **безопасными простыми числами** (safe prime).

По малой теореме Ферма $\mathrm{ord}_p(g) = q$ — простое число. Следовательно, группа $G = \langle g \rangle$ имеет большой порядок. Это необходимое условие для того, чтобы выполнялось предположение о невозможности дискретного логарифмирования (предположение 8.2).

Более того, из критерия Эйлера (теорема 6.13 из раздела 6.5.1) следует, что $g \in \mathrm{QR}_p$ и, следовательно, $G = \mathrm{QR}_p$ (теорема 5.2 из раздела 5.2.3). Таким образом, для подобранных исходных сообщений m_0, $m_1 \in \mathrm{QR}_p$ числа g, y, c_1^* и c_2^* являются квадратичными вычетами по модулю p. Следовательно, атака на основе квадратичных вычетов, продемонстрированная в примере 14.1, становится невозможной, поскольку для всех квадратичных вычетов проверка завершается положительным ответом.

Требование $G = \mathrm{QR}_p$ не может создать сложностей при шифровании и расшифровке сообщений. Например, для любого сообщения $m < p$, если $m \in \mathrm{QR}_p$, шифрование закончено, а если $m \notin \mathrm{QR}_p$, то $-m = p - m \in G$, поскольку

$$(-1)^{(p-1)/2} = (-1)^q = (-1)^{\text{нечетное число}} = -1 (\mathrm{mod}\, p),$$
$$(-m)^{(p-1)/2} = (-1)^{(p-1)/2} m^{(p-1)/2} = (-1)(-1) = 1 (\mathrm{mod}\, p),$$

а значит, по критерию Эйлера

$$-m \in \mathrm{QR}_p = G.$$

В исправленном варианте криптосистемы Эль-Гамаля перед Злоумышленником стоит несколько задач принятия решений. Получив пару зашифрованных окликов (c_1^*, c_2^*) в ответ на зашифрованные сообщения m_0 и m_1, он может вычислить следующую величину.

$$c_2^*/m_0 = \begin{cases} y^k \equiv g^{xk} (\mathrm{mod}\, p) & \text{с вероятностью 50\%,} \\ y^k (m_1/m_0) (\mathrm{mod}\, p) & \text{с вероятностью 50\%.} \end{cases}$$

Обратите внимание на то, что в первом случае кортеж

$$(g, y, c_1^*, c_2^*/m_0) = (g, g^x, g^k, g^{xk}) (\mathrm{mod}\, p)$$

представляет собой четверку Диффи–Хеллмана, а во втором случае — нет. Итак, Злоумышленник должен ответить на следующие вопросы.

Является ли кортеж $(g, y, c_1^*, c_2^*/m_0) (\mathrm{mod}\, p)$ четверкой Диффи–Хеллмана?

Является ли кортеж $(g, y, c_1^*, c_2^*/m_1) (\mathrm{mod}\, p)$ четверкой Диффи–Хеллмана?

Иначе говоря, игра IND-CPA действительно вынуждает Злоумышленника решать задачу принятия решений Диффи–Хеллмана (DDH Question) в группе G (см. определение 13.1 в разделе 13.3.4.3).

Если Злоумышленник способен правильно решить задачу принятия решений Диффи–Хеллмана в группе G, то по заданной зашифрованной паре окликов он может восстановить исходный текст, т.е. правильно угадать результаты жеребьевки, проведенной оракулом $\mathcal{O}$. В противном случае, поскольку $(g, y, c_1^*, c_2^*/m_0)(\mathrm{mod}\, p)$ и $(g, y, c_1^*, c_2^*/m_1)(\mathrm{mod}\, p)$ являются случайными кортежами, порожденными элементом g, если бы Злоумышленник был способен правильно восстановить исходный текст, он смог бы решить задачу DDH в группе $G = \langle g \rangle$. Итак, стойкость криптосистемы Эль-Гамаля, использующей открытые параметры, установленные с помощью алгоритма 14.2, к атаке IND-CPA эквивалентна сложности решения задачи DDH в группе G (теорема 14.2).

В *общем случае*, для абелевой группы (в том числе, и для группы G, определенной в алгоритме 14.2) ни одного эффективного алгоритма решения задачи DDH не существует. Сложность решения задачи DDH считается эталоном неразрешимости. Подробное описание этой проблемы читатели могут найти в обзорной работе Бонэ [47].

Предположение 14.2 (Предположение о неразрешимости задачи принятия решений Диффи–Хеллмана в конечных полях.) *Пусть $\mathcal{IG}$ — генератор целых чисел, на вход которого поступает строка 1^k, время работы полиномиально зависит от числа k, а результатом работы является 1) описание $\mathrm{desc}(G)$ абелевой группы G над конечным полем, где $|\#G| = k$; и 2) порождающий элемент группы $g \in G$.*

Генератор $\mathcal{IG}$ удовлетворяет условию о неразрешимости задачи принятия решений Диффи–Хеллмана (DDH), если для всех достаточно больших чисел k и $(\mathrm{desc}(G), g) \leftarrow \mathcal{IG}(1^k)$, множества (g, g^a, g^b, g^{ab}) и (g, g^a, g^b, g^c) являются полиномиально неразличимыми в смысле определения 4.14 из раздела 4.7.

Следует отметить, что предположение о неразрешимости задачи DDH касается только групп над конечными полями, а не абелевых групп вообще, поскольку задачу DDH легко решить в группах точек суперсингулярных эллиптических кривых (см. раздел 13.3.4.3).

Итак, для исправленной криптосистемы Эль-Гамаля справедлив следующий результат.

Теорема 14.2. *Криптосистема Эль-Гамаля, использующая открытые параметры, установленные с помощью алгоритма 14.2, является стойкой по отношению к атаке IND-CPA тогда и только тогда, когда выполняется предположение о неразрешимости задачи DDH.* □

14.3.6 Семантически стойкие криптосистемы, основанные на битах Рабина

После публикации работы Голдвассера и Микали о семантически стойких криптосистемах, Блюм и Микали [46], Яо [303], а также Блюм и Голдвассер [45] предложили несколько семантически стойких схем шифрования с открытым ключом и модификаций криптосистемы GM.

Основная идея этих усовершенствований заключалась в использовании генератора **криптографически стойких псевдослучайных битов** (cryptographically strong pseudo-random bits — CSPRB). Такой генератор представляет собой программу, на вход которой поступает начальное k-битовое число, а результатом является k^t-битовое число, где параметр $t > 1$ фиксирован. Считается, что генератор порождает псевдослучайные числа высокого качества, т.е. если начальное k-битовое число совершенно неизвестно, то итоговое k^t-битовое число абсолютно невозможно отличить от истинно случайного числа, имеющего такую же длину, если проверка осуществляется с помощью статистических критериев, на выполнение которых тратится время, полиномиально зависящее от аргумента k.

Итак, на вход генератора CSPRB поступает k-битовый открытый ключ шифрования s. Результатом работы генератора является строка pr, состоящая из ℓ бит. Отправитель шифрует сообщение m, применяя к нему и строке pr операцию исключающего ИЛИ, и отсылает получателю пару

$$(c_1, c_2) = (\mathcal{E}_{pk}(s), m \oplus ps). \tag{14.3.3}$$

Подлинный получатель (т.е. владелец открытого ключа pk) может расшифровать текст c_1 и получить начальное значение s. Это позволяет ему заново создать псевдослучайную ℓ-битовую строку pr с помощью генератора CSPRB и восстановить сообщение m, содержащееся в тексте c_2, применив к нему оператор исключающего ИЛИ.

Схема шифрования, основанная на применении генератора CSPRB, намного повысила эффективность побитового шифрования. Теперь ℓ-битовый исходный текст шифруется с помощью $\ell + k$, а не ℓk бит. Временна́я и пространственная сложность схемы побитового шифрования сравнима со сложностью криптосистем RSA, Рабина и Эль-Гамаля.

14.3.6.1 Семантическая стойкость схемы шифрования с помощью CSPRB-генератора

Если начальное число s представляет собой k-битовую строку, имеющую равномерное распределение, а детерминированный блочный алгоритм шифрования $\mathcal{E}_{pk}$ (с длиной ключа k) представляет собой перестановки в пространстве сообщений, то первый блок зашифрованного текста c_1 в формуле (14.3.3) образуется путем перестановки битов в равномерно распределенном случайном числе и,

следовательно, также имеет равномерное распределение. Итак, он не позволяет Злоумышленнику извлечь ни *априорную*, ни *апостериорную* информацию об исходном тексте.

Алгоритм шифрования RSA представляет собой перестановку в пространстве сообщений. Алгоритм шифрования Рабина можно представить в виде перестановки во множестве QR_N, если N — число Блюма (см. теорему 6.19.4 в разделе 6.7). Итак, эти алгоритмы можно считать подходящими кандидатами на роль алгоритма $\mathcal{E}_{pk}$.

Далее, благодаря силе генератора CSPRB псевдослучайная строка ps, сгенерированная на основе начального числа s, играет роль внутренней случайной операции в схеме шифрования. Следовательно, применение оператора исключающего ИЛИ к сообщению m и строке ps является семантически стойким.

В эффективной схеме шифрования Блюма–Голдвассера, основанной на применении генератора CSPRB (BG cryptosystem [45]), блок c_1 в формуле (14.3.3) равен $s^{2^i} (\mathrm{mod}\, N)$, где $i = \left\lfloor \frac{\log_2 m}{\log_2 \log_2 m} \right\rfloor + 1$, а псевдослучайная строка битов ps генерируется по числу s с помощью датчика псевдослучайных чисел BBS (9.3.1) блок за блоком: каждый блок состоит из $\log_2 \log_2 N$ младших значащих битов элемента, представляющего собой степень 2^j числа s по модулю $N (j = 1, 2, \ldots, i-1)$. Следует учесть, что первый элемент в паре зашифрованного текста, по существу, является результатом шифрования строки s с помощью алгоритма Рабина.

Поскольку задача одновременного извлечения $\log_2 \log_2 N$ младших значащих битов исходного текста, зашифрованного по схеме Рабина, эквивалентна решению задачи о разложении числа N на простые множители (см. замечание 9.1. в разделе 9.3.1), семантическая стойкость криптосистемы BG эквивалентна сложности факторизации модуля N.

14.4 Неадекватность семантической стойкости

Понятие семантической стойкости (определение 14.1 и свойство 14.1) формализует интуитивное предположение, что любой полиномиально ограниченный Злоумышленник не должен иметь возможность извлекать *априорную* информацию об исходном тексте из зашифрованного текста. Однако это гарантирует секретность, только если атакущий ведет себя пассивно, т.е. просто перлюстрирует переписку.

В разделах 8.6 и 8.14 указано, что многие криптосистемы с открытым ключом особенно уязвимы для **атак на основе подобранного зашифрованного текста** (chosen-ciphertext attack) (CCA и CCA2). В ходе атак CCA и CCA2 Злоумышленник может получать помощь при расшифровке, т.е. иметь определенный уровень доступа к "блоку расшифровки" и читать некоторые зашифрованные тексты, даже не владея ключом расшифровки. Такую помощь мы назвали "обучением крипто-

анализу". Атаки на основе подобранного зашифрованного текста, особенно атака CCA2, часто встречаются в реальных приложениях. Например, некоторые протоколы требуют, чтобы пользователь выполнял операцию расшифровки случайного оклика, т.е. участвовал в работе механизма "оклик-отзыв". В других случаях получатель зашифрованного электронного письма может раскрыть исходные сообщения в ходе последующего открытого обсуждения.

Криптосистемы с открытым ключом особенно уязвимы для атак CCA и CCA2 из-за алгебраических свойств, лежащих в их основе. Злоумышленник может использовать эти свойства и создать зашифрованный текст путем сложных вычислений. Если Злоумышленник получает помощь при расшифровке, то в его распоряжение могут попасть сообщения, которые должны были быть недоступными.

В примере 8.9 продемонстрирована уязвимость криптосистемы Эль-Гамаля к атаке CCA2. Очевидно, что эта атака опасна и для семантически стойких криптосистем, даже если они используют генератор CSPRB. В ходе таких атак блок c_2 в формуле (14.3.3) заменяется выражением $c_2' = r \oplus c_2$, где r — строка, состоящая из ℓ бит, играющая роль случайного числа r в примере 8.9.

Продемонстрируем уязвимость криптосистемы GM для атаки CCA2.

Пример 14.2. Пусть Злоумышленник имеет ограниченный доступ к блоку расшифровки, принадлежащему Алисе. Это условие вполне "разумно": если результат расшифровки текста, полученного от Злоумышленника, выглядит как случайный набор чисел, то Алиса должна вернуть Злоумышленнику исходный текст.

Предположим, что зашифрованный текст $C = (c_1, c_2, \ldots, c_\ell)$ содержит исходный текст $B = (b_1, b_2, \ldots, b_\ell)$, отправленный Алисой кому-нибудь (за исключением Злоумышленника!). Злоумышленник перехватывает сообщение C и желает восстановить текст B. Для этого он посылает Алисе следующий "хитроумный зашифрованный текст".

$$C' = (yc_1, yc_2, \ldots, yc_\ell) (\mathrm{mod}\, N). \qquad (14.4.1)$$

В ходе этой атаки Злоумышленник использует следующее алгебраическое свойство.

$$ab (\mathrm{mod}\, N) \in \mathrm{QR}_N \text{тогда и только тогда, когда}$$
$$\begin{cases} a \in \mathrm{QR}_N & \text{и } b \in \mathrm{QR}_N, \\ a \in J_N(1) \backslash \mathrm{QR}_N & \text{и } b \in J_N(1) \backslash \mathrm{QR}_N. \end{cases}$$

Это свойство непосредственно следует из критерия Эйлера (теорема 6.13 в разделе 6.5.1).

Итак, поскольку $y \in J_N(1) \backslash \mathrm{QR}_N$, можно убедиться, что число y шифрует единичный бит. Затем, применяя атаку "путем умножения на y" ("multiplying-y attack") к шифрованному тексту (14.4.1), Злоумышленник переключает биты

$b_i, i = 1, 2, \ldots, \ell$, т.е. в результате расшифровки Алиса получит исходный текст

$$B' = (b'_1, b'_2, \ldots, b'_\ell),$$

где b'_i — дополнение к биту $b_i, i = 1, 2, \ldots, \ell$.

Алисе этот результат расшифровки покажется набором случайных чисел, поэтому она вернет текст B' Злоумышленнику, который, в свою очередь, определит по нему текст B.

Используя вместо вектора $(y, y, \ldots, y)$ множитель $Y = (y_1, y_2, \ldots, y_\ell)$, где Y — результат шифрования ℓ-битового кортежа $Z = (z_1, z_2, \ldots, z_\ell) \in_U \{0,1\}^\ell$ с помощью алгоритма GM и открытого ключа Алисы, Злоумышленник может сделать текст B' не просто случайным, но и *равномерно* распределенным. Легко проверить, что

$$B = \left(b'_1 \oplus z_1, b'_2 \oplus z_2, \ldots, b'_\ell \oplus z_\ell\right). \qquad \square$$

В ходе этой атаки Алиса предоставляет Злоумышленнику "услуги оракула", помогая ему расшифровывать текст. Обратите внимание на то, что услуги оракула могут быть неявными. В примере 14.1 показано, что Алисе не следует отвечать на запросы Злоумышленника.

Пример 14.3. Допустим, что Алиса теперь не должна возвращать Злоумышленнику сообщения, которые выглядят как набор случайных цифр. Получив зашифрованное сообщение $(c_1, c_2, \ldots, c_\ell)$, посланное Бобом Алисе, Злоумышленник попрежнему может распознать исходный текст бит за битом.

Например, для того, чтобы определить, какую цифру, 0 или 1, шифрует блок c_1, Злоумышленник может послать Алисе зашифрованный вопрос, требующий однозначного ответа (ДА или НЕТ). Первую половину вопроса Злоумышленник может зашифровать, как обычно, а вторую — используя блок c_1 вместо множителя y в алгоритме 14.1.

Если $c_1 \in \mathrm{QR}_N$, Алиса только первую половину сообщения расшифрует правильно, а вторая половина вопроса будет содержать одни нули. Следовательно, она будет вынуждена спросить у Злоумышленника, почему он послал неполное сообщение. Тогда Злоумышленник узнает, что блок c_1 шифрует нуль. С другой стороны, если Алиса правильно отвечает на вопрос, Злоумышленник знает, что блок c_1 является невычетом, и, следовательно, шифрует единицу.

Следует отметить, что в этой атаке Злоумышленник может даже сопровождать цифровой подписью все свои сообщения, направленные Алисе, чтобы убедить ее в своем авторстве. Иначе говоря, Злоумышленника невозможно обвинить в нарушении правил! $\square$

Анализ двух описанных выше атак показывает, что криптосистема GM безнадежно уязвима для активных атак, а семантическая стойкость является слабой.

14.5 За рамками семантической стойкости

Эволюция принципа "все или ничего" (свойство 8.2 в разделе 8.2) до семантической стойкости (определение 14.1) представляет собой лишь первый шаг на пути усиления понятия стойкости криптосистем.

В разделе 14.4 было показано, что понятие семантической стойкости является недостаточно сильным для реальных приложений, в которых пользователи могут прибегать к услугам оракулов расшифровки. Действительно, трудно ожидать, что наивный пользователь реальных криптографических систем будет проявлять бдительность и отказываться отвечать на подозрительные запросы. Следовательно, необходимо предложить понятие более сильной стойкости.

Рассмотрим **атаку на основе неразличимых подобранных зашифрованных текстов** (indistinguishable chosen-ciphertext attack — IND-CCA). В рамках этой модели Злоумышленнику еще легче взломать атакованную криптосистему: кроме помощи при шифровании, предоставляемой в рамках протокола 14.1, Злоумышленник может прибегать к услугам оракула расшифровки. Формальное описание атаки IND-CCA основано на игре Наора (Naor) и Юнга (Yung) [210]. Эта игра называется **атакой во время ленча** (lunchtime attack) или **атакой на основе индифферентных подобранных зашифрованных текстов** (indifferent chosen-ciphertext attack).

14.5.1 Стойкость к атакам на основе подобранных зашифрованных текстов

Атака "во время ленча" описывает реальный сценарий, в котором Злоумышленник в отсутствие остальных сотрудников организации (например, во время ленча) посылает службе расшифровки сообщений запросы, надеясь таким образом пройти "курс обучения криптоанализу". Поскольку ленч проходит быстро, у Злоумышленника недостаточно времени для подготовки зашифрованных текстов так, чтобы ответы на запросы оказались связанными с определенной функцией. Следовательно, все шифрованные запросы, которые он посылает во время ленча, Злоумышленник должен приготовить заранее.

Этот реальный сценарий можно описать с помощью игры, в которой участвуют те же персонажи, что и в игре IND-CPA (протокол 14.1): Злоумышленник (обиженный сотрудник организации) и оракул $\mathcal{O}$ — механизм расшифровки (и шифрования), обслуживающий организацию. Эта игра называется **атакой на основе неразличимых подобранных зашифрованных текстов** (IND-CCA).

На первый взгляд, атака во время ленча не слишком реалистична. Где найти такого простака, который согласился бы отвечать на запросы Злоумышленника? Ответить на этот вопрос можно следующим образом.

Протокол 14.3. "Атака во время ленча" (атака на основе неадаптивно подобранных неразличимых зашифрованных текстов)

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ:

а) Как и в протоколе 14.1, Злоумышленник и оракул $\mathcal{O}$ используют криптосистему $\mathcal{E}$, для которой оракул $\mathcal{O}$ имеет фиксированный ключ шифрования;

б) Злоумышленник заранее приготовил несколько зашифрованных сообщений.

1. Злоумышленник посылает оракулу $\mathcal{O}$ заранее подготовленное зашифрованное сообщение $c \in \mathcal{C}$.

2. Оракул $\mathcal{O}$ расшифровывает сообщение $\dot{c}$ и возвращает результат расшифровки Злоумышленнику.
   (* Сообщение c называется **подобранным зашифрованным текстом** (chosen-ciphertext) или **индифферентным подобранным шифрованным текстом** (indifferent chosen-ciphertext). Считается, что возврат результата расшифровки позволяет Злоумышленнику пройти "курс обучения криптоанализу". Злоумышленник может попросить об этой услуге, повторяя запросы столько, сколько захочет. Для ускорения процесса обучения он может даже использовать специальную программу. *)

3. Пройдя полный "курс обучения расшифровке", Злоумышленник предлагает оракулу $\mathcal{O}$ сыграть в игру CPA (протокол 14.1).
   (* В этом варианте игры CPA подобранные исходные сообщения m_0 и m_1 называются **адаптивно подобранными зашифрованными текстами** (adaptive chosen-plaintext). Иначе говоря, эти два сообщения могут зависеть от предыстории "курса обучения расшифровке", пройденного на этапах 1 и 2. Этап поиска начинается с первого пункта протокола и заканчивается получением зашифрованного оклика $c^* \in \mathcal{C}$, который с одинаковой вероятностью может шифровать как сообщение m_0, так и сообщение m_1 из пространства $\mathcal{M}$. *)
   (* Резонно предположить, что Злоумышленник может вычислять адаптивно подобранные исходные сообщения, несмотря на ограниченное время атаки, поскольку создать исходное сообщение намного легче, чем зашифрованный текст. *)
   (* Итак, ленч завершился. Теперь Злоумышленник должен угадать результаты жеребьевки оракула и послать число 0 или 1. Однако, несмотря на то, что игра закончена, Злоумышленник остается на этапе угадывания, пока не ответит на вопросы оракула. *)

- Довольно часто во многих прикладных криптосистемах (в частности, в криптографических протоколах) пользователь (участник протокола), получивший запрос, должен выполнить расшифровку, используя свой закрытый ключ, и отослать результат обратно. Так работает механизм оклика-отзыва (см. главы 2 и 11).
- На практике многие пользователи безнадежно наивны и неспособны проявлять бдительность, чтобы предотвратить попытки взлома. Можно даже сказать, что самые строгие и стойкие криптосистемы разрабатываются именно для наивных пользователей.
- Злоумышленник может внедрять запросы на расшифровку внутри нормальных и невинных сообщений, причем ответы он может получать так же неявно. Примеры 9.2 и 14.3 — яркое свидетельство этому. Такие атаки трудно отличить от обычных сеансов связи. Нёвозможно запретить пользователям отвечать на зашифрованные запросы, поскольку это ограничивает возможности для обмена секретными сообщениями.
- Злоумышленник может даже использовать обходной канал, как, например, в атаке на основе временно́го анализа (см. раздел 12.5.4), в ходе которой Злоумышленник измеряет разницу между временны́ми задержками.

Правильное отношение к Злоумышленнику — встретить его лицом к лицу и предоставить ему возможность пройти "курс обучения криптоанализу" по его требованию. Курс обучения может включать как шифрование, так и расшифровки целых блоков или отдельных битов. Стратегия такова: разрабатывать настолько стойкие криптосистемы, чтобы "курс обучения криптоанализу" не давал Злоумышленнику никаких преимуществ!

Повторяя рассуждения, приведенные в разделе 14.2, касающиеся оценки преимущества Злоумышленника при взломе криптосистемы в ходе атаки CPA (протокол 14.1), можно оценить преимущество Злоумышленника в ходе атаки во время ленча.

$$\text{Prob}[1 \leftarrow \text{Злоумышленник}(c^*, m_0, m_1, \text{Hist-CCA}) \mid c^* = \mathcal{E}_{kd}(m_1)] = \\ = \frac{1}{2} + \text{Adv}\,. \tag{14.5.1}$$

Итак, мы готовы сформулировать новое понятие стойкости, более сильное, чем понятие стойкости к атаке INA-CPA.

Определение 14.2 (Стойкость к атаке на основе неразличимых подобранных зашифрованных текстов (IND-CCA Security)). *Криптосистема с параметром безопасности k называется стойкой к атаке на основе неразличимых подобранных зашифрованных текстов (стойкой IND-CCA), если в результате атаки, описанной в протоколе 14.3, участником которой является произвольный полиномиально ограниченный Злоумышленник, преимущество* Adv, *заданное формулой (14.5.1), является пренебрежимо малой величиной по сравнению с параметром k.*

Поскольку в атаке во время ленча предполагается, что Злоумышленник может получить помощь при расшифровке сообщений (пройти "курс обучения криптоанализу"), новая атака должна упростить задачу криптоанализа. Следовательно, следует ожидать, что некоторые криптосистемы, считавшиеся стойкими к атаке IND-CCA, теперь станут уязвимыми.

Стойкость IND-CCA ни одной из криптосистем, описанных ранее в этой главе, не была доказана. Увы, среди них есть даже совершенно *нестойкие*! В частности, к ним относится криптосистема Блюма–Голдвассера, основанная на применении эффективного CSPRB-генератора (раздел 14.3.6).

Пример 14.4. Для того чтобы организовать атаку на криптосистему BG во время ленча, Злоумышленник должен подобрать запрос в виде зашифрованного текста (c, m), где c — квадратичный вычет по модулю N, а $|m| = \lfloor \log_2 \log_2 N \rfloor$. Анализируя криптосистему BG, описанную в разделе 14.3.6, легко определить, что в ответ на свой запрос Злоумышленник получит следующую расшифровку.

$m \oplus$ "$\lfloor \log_2 \log_2 N \rfloor$ младших битов квадратного корня числа c по модулю N".

Применяя к ответу побитовую операцию XOR, Злоумышленник получит $\lfloor \log_2 \log_2 N \rfloor$ младших битов квадратного корня числа c по модулю N. Напомним, что число c представляет собой модуль N, являющийся квадратичным вычетом. Учитывая замечание 9.1 (раздел 9.3.1), приходим к выводу, что знание $\lfloor \log_2 \log_2 N \rfloor$ младших битов квадратного корня числа c по модулю N дает Злоумышленнику возможность разложить модуль N на простые множители за полиномиальное время! □

Пример 14.4 демонстрирует, что курс криптоанализа, который проходит Злоумышленник, резко увеличивает вероятность взлома криптосистемы. Последствия взлома являются чрезвычайно тяжкими, поскольку Злоумышленник может раскрыть не просто отдельный шифрованный текст, но и разрушить всю криптосистему в целом.

Таким образом, нами абсолютно точно доказана нестойкость криптосистемы BG к атаке IND-CCA. Это напоминает нестойкость криптосистемы Рабина в рамках подхода "все или ничего" (теорема 8.2 в разделе 8.11).

Наор и Юнг предложили криптосистему, обладающую доказуемой стойкостью к атаке IND-CCA [210]. В этой криптосистеме исходное сообщение побитово шифруется в виде двух текстов с помощью двух разных ключей. Алгоритм шифрования содержит процедуру неинтерактивного доказательства с нулевым разглашением (non-interactive zero-knowledge — NIZK), позволяющую отправителю исходного сообщения доказать, что два зашифрованных текста действительно содержат один и тот же бит, зашифрованный разными открытыми ключами. (Считается, что алгоритм шифрования, на вход которого поступает исходный тест

и случайное число, представляет собой NP-задачу (см. раздел 4.8.1)). Это доказательство проверяется во время расшифровки (например, оракулом $\mathcal{O}$, принимающим участие в атаке во время ленча). Применение процедуры верификации во время расшифровки основано на предположении, что пара зашифрованных сообщений уже известна отправителю (например, Злоумышленнику в ходе атаки во время ленча). Таким образом, обслуживание Злоумышленника во время ленча не дает ему никаких новых знаний, облегчающих криптоанализ. Вследствие довольно высокой стоимости процедуры неинтерактивного доказательства с нулевым разглашением (верификации) и побитового шифрования (расшифровки) криптосистема Наора и Юнга не получила практического применения.

Атака во время ленча является довольно ограниченной моделью, поскольку помощь, получаемая Злоумышленником при расшифровке, доступна лишь в течение короткого периода. Это выглядит так, будто "блок расшифровки" по окончании ленча каждый раз отключается или Злоумышленник не может ничего поделать даже во время ленча на следующий день. Это совершенно нереалистичный сценарий. На практике наивные пользователи всегда остаются наивными, и Злоумышленник способен нанести ответный удар, даже если для этого придется использовать обеденный перерыв на следующий день! Таким образом, стойкость к атаке IND-CCA не является сильной.

14.5.2 Стойкость к атаке на основе адаптивно подобранных зашифрованных текстов

Рассмотрим **атаку на основе неразличимых адаптивно подобранных зашифрованных текстов** (IND-CCA2), предложенную Раковым (Rackoff) и Симоном (Simon) [241].

В этой атаке перед Злоумышленником стоит более простая задача. В атаке во время ленча (см. протокол 14.3) помощь в расшифровке ("курс обучения криптоанализу") прекращалась, как только Злоумышленник посылал пару адаптивно подобранных исходных сообщений. Иначе говоря, атака во время ленча прекращалась сразу после завершения игры IND-CCA (т.е. протокола 14.1).

В новой модели это нереалистичное условие было снято. Теперь Злоумышленник может получать помощь при расшифровке сообщений как до, так и после атаки, организованной во время ленча. Таким образом, новый сценарий можно рассматривать как атаку во время очень долгого перерыва, например, ночью. По этой причине эту атаку называют **атакой пополуночи** (small-hours attack). Это название отражает реальный сценарий, в рамках которого Злоумышленник, например, обиженный сотрудник или конкурирующая фирма, остается на ночь и вступает в контакт с механизмом расшифровки сообщений. Следует заметить, что **атака пополуночи** отличается от **полуночной атаки** (midnight attack), кото-

рая в литературе часто упоминается как синоним атаки во время ленча. Почему-то считается, что охранники должны ужинать в полночь.

Поскольку теперь Злоумышленник не ограничен во времени, он может организовать более сложную и интересную игру с механизмом расшифровки. Кроме того, что он мог делать в ходе атаки во время ленча (или в полночь), т.е. создавать адаптивно подобранные запросы, содержащие исходный текст, используя информацию, полученную в рамках курса обучения криптоанализу, и получать зашифрованный оклик, Злоумышленник теперь может посылать *адаптивно подобранные зашифрованные сообщения после* получения оклика. Следовательно, адаптивно подобранные зашифрованные сообщения могут быть каким-то образом связанными с окликом или соответствующим исходным текстом. Разумеется, механизм расшифровки достаточно разумен, чтобы не расшифровывать оклик, посланный Злоумышленнику! Это единственное и вполне обоснованное ограничение. Без него Злоумышленник мог бы просто попросить блок расшифровки расшифровать оклик и не организовывать сложную игру! С другой стороны, блок расшифровки достаточно глуп, чтобы расшифровать текст, связанный с окликом! Блок не распознает любое изменение оклика, например, умножение на два или добавление единицы, и расшифровывает измененное сообщение!

Повторяя рассуждения, приведенные в разделе 14.2, можно получить оценки преимущества Злоумышленника при взломе криптосистемы в ходе атаки полулуночи. Эта формула очень напоминает формулу (14.2.3), за исключением того, что Злоумышленник теперь вводит всю предысторию о двух предшествующих курсах обучения криптоанализу, один — для атаки CCA, а второй — для расширенной атаки CCA. Обозначим эту предысторию как Hist-CCA2. Тогда формула для преимущества Злоумышленника примет следующий вид.

$$\begin{aligned}\text{Prob}[1 \leftarrow \text{Злоумышленник}(c^*, m_0, m_1, \text{Hist-CCA2}) \mid c^* = \mathcal{E}_{ke}(m_1)] = \\ = \frac{1}{2} + \text{Adv}\,.\end{aligned} \tag{14.5.2}$$

Теперь мы можем сформулировать новое, более строгое понятие стойкости.

Определение 14.3 (Стойкость к атаке на основе неразличимых адаптивно подобранных зашифрованных текстов (стойкость IND-CCA2)). *Криптосистема с параметром безопасности k называется стойкой к атаке на основе неразличимых адаптивно подобранных зашифрованных текстов (стойкой к атаке IND-CCA2), если в результате атаки, описанной в протоколе 14.4, участником которой является произвольный полиномиально ограниченный Злоумышленник, преимущество* Adv, *заданное формулой (14.5.2), является пренебрежимо малой величиной по сравнению с параметром k.*

Протокол 14.4. "Атака пополуночи" (атака на основе адаптивно подобранных неразличимых зашифрованных текстов)

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ:

Как и в протоколе 14.1, Злоумышленник и оракул $\mathcal{O}$ используют криптосистему $\mathcal{E}$, для которой оракул $\mathcal{O}$ имеет фиксированный ключ шифрования.

1. Злоумышленник и оракул $\mathcal{O}$ разыгрывают атаку во время ленча (протокол 14.3).
   (* В этой разновидности атаки во время ленча "этап поиска" совпадает с этапом поиска в протоколе 14.3. Он завершается тем, что Злоумышленник получает зашифрованный оклик $c^* \in \mathcal{C}$, который с одинаковой вероятностью может шифровать как сообщение m_0, так и сообщение m_1 из пространства $\mathcal{M}$. Однако теперь Злоумышленник может продлить "этап угадывания". *)
2. Злоумышленник вычисляет зашифрованный текст $c' \in \mathcal{C}$ и посылает его оракулу $\mathcal{O}$ для расшифровки.
   (* Зашифрованный текст c' называется **адаптивно подобранным зашифрованным текстом** (adaptive chosen-plaintext) или **зашифрованным текстом, подобранным после получения оклика** (post-challenge chosen ciphertext). В отличие от него, зашифрованный текст, подобранный в ходе атаки во время ленча (протокол 14.3), называется **зашифрованным текстом, подобранным до получения оклика** (pre-challenge chosen ciphertext). Второй этап представляет собой "*расширенный* курс обучения криптоанализу", который может повторяться столько, сколько будет угодно Злоумышленнику. *)
   (* Предполагается, что $c' \neq c^*$, т.е. Злоумышленник не может посылать зашифрованный оклик обратно для расшифровки. *)
3. Пройдя "*расширенный* курс обучения криптоанализу", Злоумышленник должен угадать результаты жеребьевки оракула и послать число 0 или 1.

Атаки на основе неразличимых зашифрованных текстов проиллюстрированы на рис. 14.1.

Поскольку в ходе атаки пополуночи Злоумышленник проходит "расширенный курс обучения криптоанализу", задача взлома по сравнению с атакой во время ленча должна упроститься. Таким образом, следует ожидать, что криптосистемы, стойкие к атаке IND-CCA, могут оказаться уязвимыми для атаки IND-CCA2. Фактически, за исключением алгоритма RSA-OAEP (см. алгоритм 10.6), ни одна из криптосистем, описанных в книге, до сих пор не обладает доказанной стойкостью к атаке IND-CCA2. Мы продемонстрировали большое количество примеров

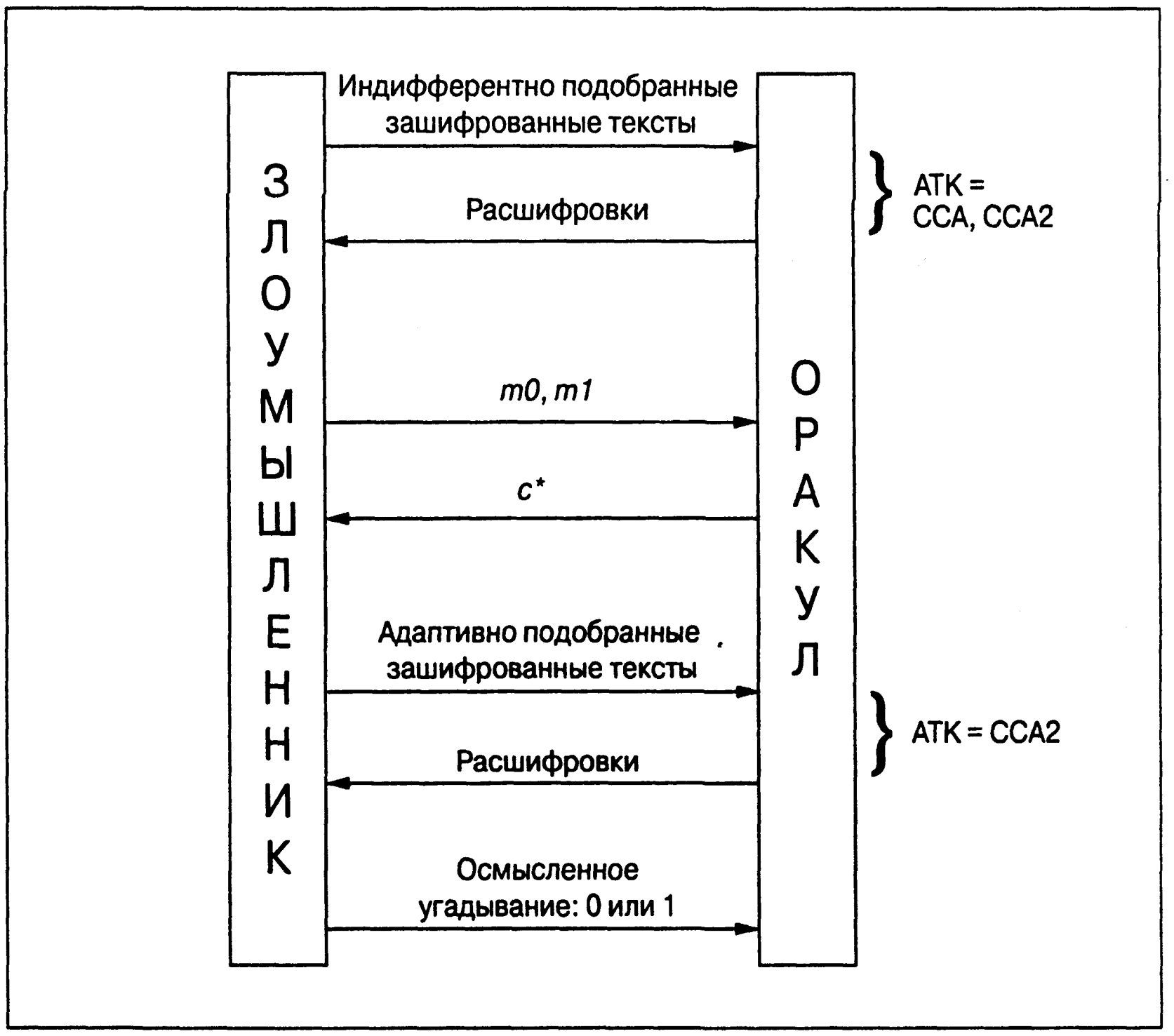


Рис. 14.1. Атаки на основе неразличимых зашифрованных текстов

криптосистем, уязвимых как к атаке CCA2, т.е. по принципу "все или ничего", так и к атаке IND-CCA2 (см. примеры 8.5, 8.7, 8.9, 9.2, 14.2 и 14.3).

Введя понятие стойкости к атаке IND-CCA2, Раков и Симон предложили криптосистемы, основанные на неинтерактивном доказательстве с нулевым разглашением (доказательство NIZK). Однако они рассмотрели доказательство NIZK с *конкретным средством доказательства* (specific prover). В их криптосистеме парой, состоящей из открытого и закрытого ключа, обладал не только получатель, но и отправитель сообщения. Более того, открытый ключ отправителя сертифицировался с помощью инфраструктуры сертификации открытых ключей (см. раздел 13.2). Для того чтобы зашифровать сообщение, отправитель использовал не только открытый ключ получателя, что вполне привычно, но и свой собственный закрытый ключ, позволяющий создать доказательство NIZK так, чтобы получатель зашифрованного текста мог проверить его, используя открытый ключ отправителя. Передача доказательства NIZK означала, что исходный текст был создан именно данным отправителем, а значит, возвращение исходного текста не дает ему никакой новой информации, позволяющей взломать криптосистему. Криптосистема Ракова и Симона также выполняет все операции побитово.

14.5.3 Строгая криптография

Строгая криптография (non-malleable cryptography — NM cryptography) [100] — это криптография с открытым ключом, основанная на более сильных понятиях стойкости. Строгость является важным требованием, означающим, что Злоумышленник не должен иметь возможности легко модифицировать исходное сообщение, изменяя соответствующий зашифрованный текст. Долев (Dolev) с коллегами очень хорошо обосновали необходимость этого требования на примере аукциона.

Предположим, что муниципалитет пригласил строительные компании принять участие в конкурсе на получение права построить новую школу. Городская администрация, широко применяющая электронные средства связи, обнародовала свой открытый ключ E, используемый для шифрования заявок, и открыла свой почтовый ящик `e-gov@bid.for.it.gov`. Компания A предложила 1 500 000 долл., послав на почтовый ящик сообщение $\mathcal{E}(1\,500\,000)$. Однако это электронное сообщение было перехвачено Злоумышленником, главой компании CheapSub, специализирующейся на заключении дешевых строительных контрактов. Если алгоритм шифрования является уязвимым, Злоумышленник может каким-либо образом трансформировать сообщение $\mathcal{E}(1\,500\,000)$ в $\mathcal{E}(15\,000\,000)$. Таким образом, шансы Злоумышленника на заключение контакта возрастают.

Простейшим примером уязвимого алгоритма шифрования является одноразовое заполнение (one-time pad). В части III показано, что все основные и популярные функции шифрования с открытым ключом уязвимы.

В отличие от разнообразных атак на защиту неразличимости (IND), основанных на задачах принятия решений, в основе гибких атак лежат вычислительные задачи.

Поскольку в ходе этой атаки целью Злоумышленника, получившего оклик c^*, не является обучение, ему совершенно не обязательно знать параметр α. Однако для того, чтобы атака оказалась успешной, Злоумышленник должен вычислить "разумное" отношение R, связывающее расшифровки сообщений c^* и c'.

Успех Злоумышленника выражается через преимущество. В работе [100] авторы для оценки преимущества использовали идею моделирования с нулевым разглашением.[1] Во-первых, Злоумышленник, выполняющий PPT-алгоритм, получает оклик $c^* = \mathcal{E}_{pk}(\alpha)$ и с определенной вероятностью вычисляет пару $(\mathcal{E}_{pk}(\beta), R)$. Во-вторых, *имитатор* (simulator) ZK-Sim, представляющий собой PPT-алгоритм, не получает оклик c^*, но, как и Злоумышленник, вычисляет зашифрованный текст $\bar{c}$ с определенной вероятностью. (Имитатор даже игнорирует алгоритм шифрования и открытый ключ!) Преимущество Злоумышленника, организующего гибкую

[1] Доказательства с нулевым разглашением и полиномиальное моделирование таких доказательств будут рассмотрены в следующей главе.

Протокол 14.5. Гибкая атака в режиме подбора исходных текстов

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ:

Как и в протоколе 14.1, Злоумышленник и оракул $\mathcal{O}$ используют криптосистему $\mathcal{E}$, для которой оракул $\mathcal{O}$ имеет фиксированный ключ шифрования pk.

Злоумышленник и оракул $\mathcal{O}$ разыгрывают следующую атаку.

1. Злоумышленник посылает оракулу $\mathcal{O}$ вектор v, содержащий большое количество исходных текстов, и $\mathrm{desc}(v)$ — описание распределения исходных текстов v.
2. Оракул $\mathcal{O}$ создает правильный зашифрованный оклик $c^* = \mathcal{E}_{pk}(\alpha)$, где параметр α извлекается из генеральной совокупности исходных сообщений, подчиняющихся распределению текстов в векторе v. Оракул $\mathcal{O}$ посылает сообщение c^* Злоумышленнику.
3. Получив сообщение c^*, Злоумышленник должен вычислить "разумное" отношение R и другой корректный зашифрованный текст $c' = \mathcal{E}_{pk}(\beta)$, удовлетворяющий условию $R(\alpha, \beta) = 1$.

атаку, оценивается следующим образом.

$$NM - Adv = \mathrm{Prob}[(\mathcal{E}_{pk}(\beta), R) \leftarrow \text{Злоумышленник}(\mathcal{E}_{pk}(\alpha), pk, \mathrm{desc}(v))] - \\ - \mathrm{Prob}[(\bar{c}, R) \leftarrow \text{ZK-Sim}]. \tag{14.5.3}$$

Криптосистема $\mathcal{E}_{pk}()$ с параметром безопасности k называется строгой, если для всех полиномиально вычислимых отношений R и полиномиально ограниченных злоумышленников значение функции NM-Adv является пренебрежимо малой величиной по сравнению с числом k. В работе [100] это свойство называется "**семантической стойкостью к атаке на основе подобранных открытых текстов с учетом отношений**" ("semantic security with respect to relations under chosen-plaintext attack"). По этой причине оно сокращенно называется **NM-CPA**.

Свойство 14.2 (Стойкость NM-CPA). Преимущество Злоумышленника в ходе гибкой атаки на криптосистему NM-CPA на основе имеющегося зашифрованного текста по сравнению с имитацией этой атаки без такого текста увеличивается незначительно.

Поскольку обладание зашифрованным текстом не облегчает задачу взлома, Злоумышленнику следует пройти "курс обучения криптоанализу"! Как и атаки IND-CCA и IND-CCA2, гибкая атака также может быть облегченным вариантом атаки во время ленча и атаки пополуночи. В ходе гибкой атаки, организованной по

примеру атаки во время ленча, Злоумышленник может посылать оракулу $\mathcal{O}$ подобранные зашифрованные тексты для расшифровки, но эта возможность исчезает после запроса на получение оклика c^*. В ходе гибкой атаки, организованной по примеру атаки пополуночи, после получения оклика доступ к блоку расшифровки не прекращается. Разумеется, Злоумышленник по-прежнему не имеет права пересылать оклик c^* оракулу $\mathcal{O}$ для расшифровки.

Таким образом, мы сформулировали два понятия стойкости: **NM-CCA** и **NM-CCA2**.

Поскольку задачи, лежащие в основе NM-стойкости, носят вычислительный характер, мы не приводим формальных доказательств. Читатели, которых заинтересовала эта тема, могут найти подробную информацию в работе [100]. Естественно ожидать, что формализация понятия NM-стойкости связана с понятиями, выходящими за рамки IND-стойкости. Например, в отличие от задач принятия решений, в которых размер пространства исходных сообщений не важен (он может быть равен двум, как в криптосистеме Голдвассера-Микали), в теории NM-стойкости пространство исходных сообщений должно быть достаточно большим, чтобы вычисление отношения R оказалось нетривиальной задачей.

В работе [19] авторы предложили немного другой способ формализации понятия NM-стойкости, основанный на игровых атаках, аналогичных IND-атакам.

Несмотря на это, изложенных выше сведений вполне достаточно для того, чтобы понять идею NM-стойкости. Легко видеть, что большинство учебных алгоритмов шифрования, использующих однонаправленную функцию с секретом, весьма уязвимо. Как показано в главе 9, все однонаправленные функции с секретом, лежащие в основе популярных криптосистем с открытым ключом, можно инвертировать, используя частичную информацию, полученную от оракула (например, от "оракула четности"). Именно эти принципы инверсии позволяют организовывать гибкие атаки на неизвестные исходные сообщения. Например, в алгоритме RSA выполняется равенство $c = m^e (\mathrm{mod}\, N)$. Благодаря этому Злоумышленник знает, что неизвестный исходный текст m можно продублировать, если умножить его на $2^e (\mathrm{mod}\, N)$.

Долев с соавторами разработали схему шифрования с открытым ключом, обладающую доказуемой стойкостью к атаке NM-CCA2 [100]. В этой схеме для побитового шифрования исходного сообщения используется большое количество пар, состоящих из открытых и закрытых ключей. Кроме того, шифрование каждого бита исходного текста содержит доказательство NIZK.

14.5.4 Связь между неразличимостью и строгостью

Несомненно, понятие строгой стойкости весьма важно. Однако вследствие того, что задачи, лежащие в основе этого понятия, носят вычислительный характер, формальное доказательство строгой стойкости является довольно сложным.

Следовательно, разработка строгой криптосистемы и доказательство ее стойкости представляют собой нетривиальную задачу.

К счастью, исследователи обнаружили большое количество важных зависимостей между строгой стойкостью и неразличимостью. Более того, оказалось, что понятие стойкости CCA2 эквивалентно понятию неразличимости. Поскольку формальное доказательство стойкости к атаке IND-CCA2 хорошо известно, с его помощью можно доказать стойкость криптосистемы к атаке NM-CCA2.

Формальное доказательство этих зависимостей можно получить, сконструировав **полиномиальный алгоритм редукции** (polynomial-time reduction algorithm). Применительно к атакам на криптосистемы это означает, что целевую атаку (Target Attack) можно свести к исходной (Source Attack). Если алгоритм редукции является полиномиальным, то целевую атаку можно реализовать с помощью исходной, причем стоимость организации целевой атаки является полиномиально ограниченной.

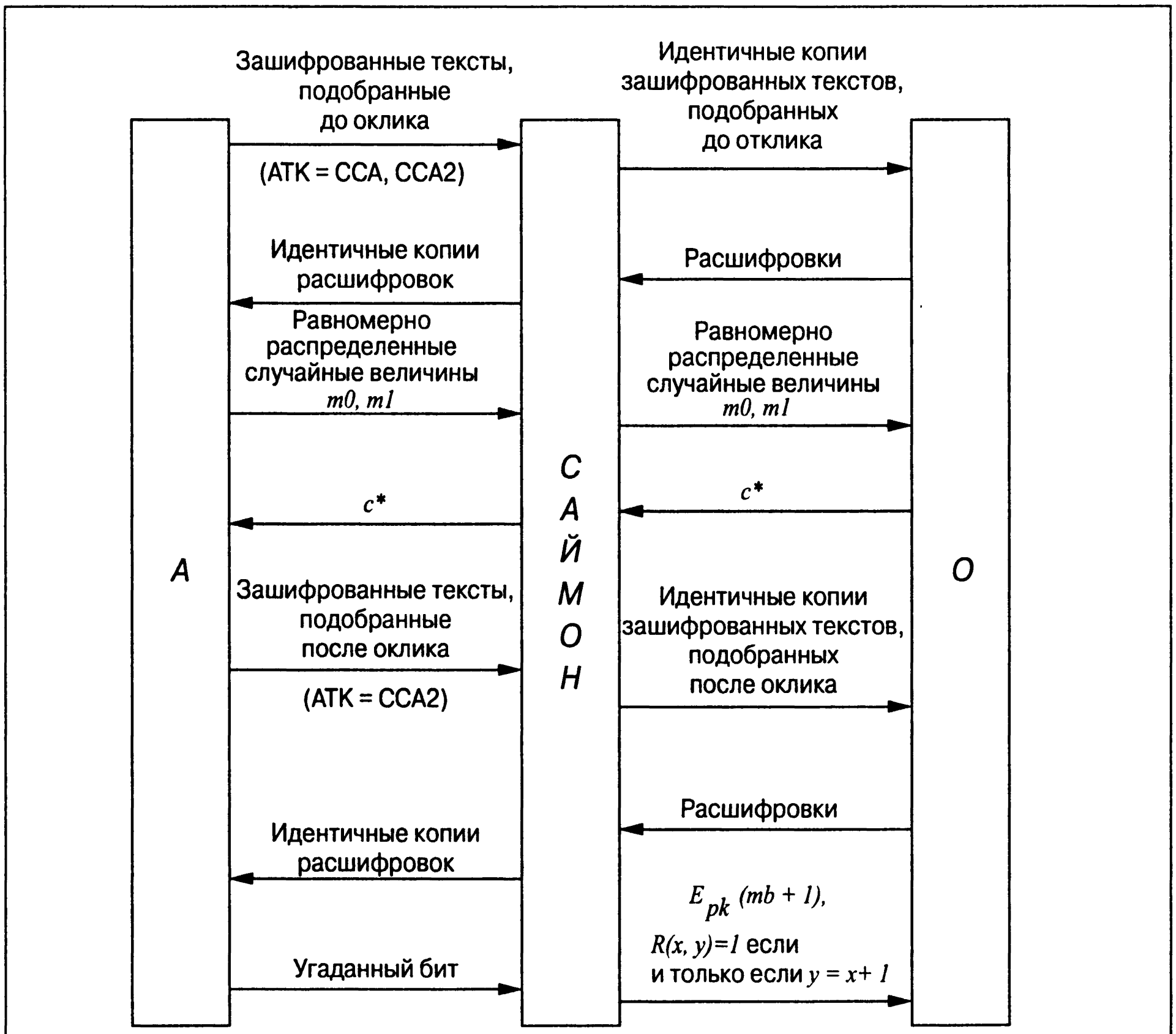
Поскольку любая атака на криптосистему всегда основана на некоторых предположениях и сопровождается определенными ограничениями (например, в атаке CCA2 атакующий должен пройти курс обучения криптоанализу до и после получения оклика), алгоритм редукции также должен удовлетворять этим условиям. Как правило, для выполнения редукции мы будем использовать специального агента, **Саймона-имитатора** (Simon Simulator). Предполагается, что Саймон удовлетворяет всем условиям и ограничениям, предъявляемым к рабочему окружению при организации атаки.

Иногда, после взаимодействия со Злоумышленником Саймон сам становится атакующим, управляя сразу двумя играми между атакующим и оракулом шифрования-расшифровки — целевой и исходной атаками. С одной стороны, Саймон участвует в исходной игре с атакующим, имитируя рабочее окружение (т.е. представляясь атакующему оракулом шифрования-расшифровки). С другой стороны, Саймон участвует в целевой игре с оракулом шифрования-расшифровки, играя роль атакующего. В такой ситуации можно считать, что атакующий в исходной атаке обучает Саймона методам организации целевой атаки. Схема управления атаками изображена на рис. 14.2 и 14.5.4.1.

14.5.4.1 Из строгой стойкости следует неразличимость

Обозначим атаки CPA, CCA и CCA2 через ATK. Покажем, что из стойкости криптосистемы с открытым ключом к атакам NM-ATK следует стойкость к атакам IND-ATK.

Теорема 14.3. *Если криптосистема с открытым ключом является стойкой к атакам NM-ATK, то она также является стойкой к атакам IND-ATK.*



A: организатор атаки IND-ATK

Саймон:

по отношению к атакующему A имитирует оракула шифрования-расшифровки O в атаке IND-ATK
по отношению к оракулу O в атаке NM-ATK играет роль атакующего

O: оракул шифрования-расшифровки в атаке NM-ATK

Рис. 14.2. Редукция атаки NM к атаке IND

Доказательство. Докажем эту теорему методом от противного, показав, что из уязвимости криптосистемы с открытым ключом $\mathcal{E}_{pk}$ для атак IND-ATK следует уязвимость к атакам NM-ATK.

Предположим, что криптосистема $\mathcal{E}_{pk}$ является уязвимой к атакам IND-ATK. Тогда существует полиномиально ограниченный атакующий $\mathcal{A}$, способный взломать криптосистему $\mathcal{E}_{pk}$ с помощью атаки IND-ATK с преимуществом $\mathrm{Adv}(\mathcal{A})$, которое не является пренебрежимо малой величиной. Допустим, что Саймону-

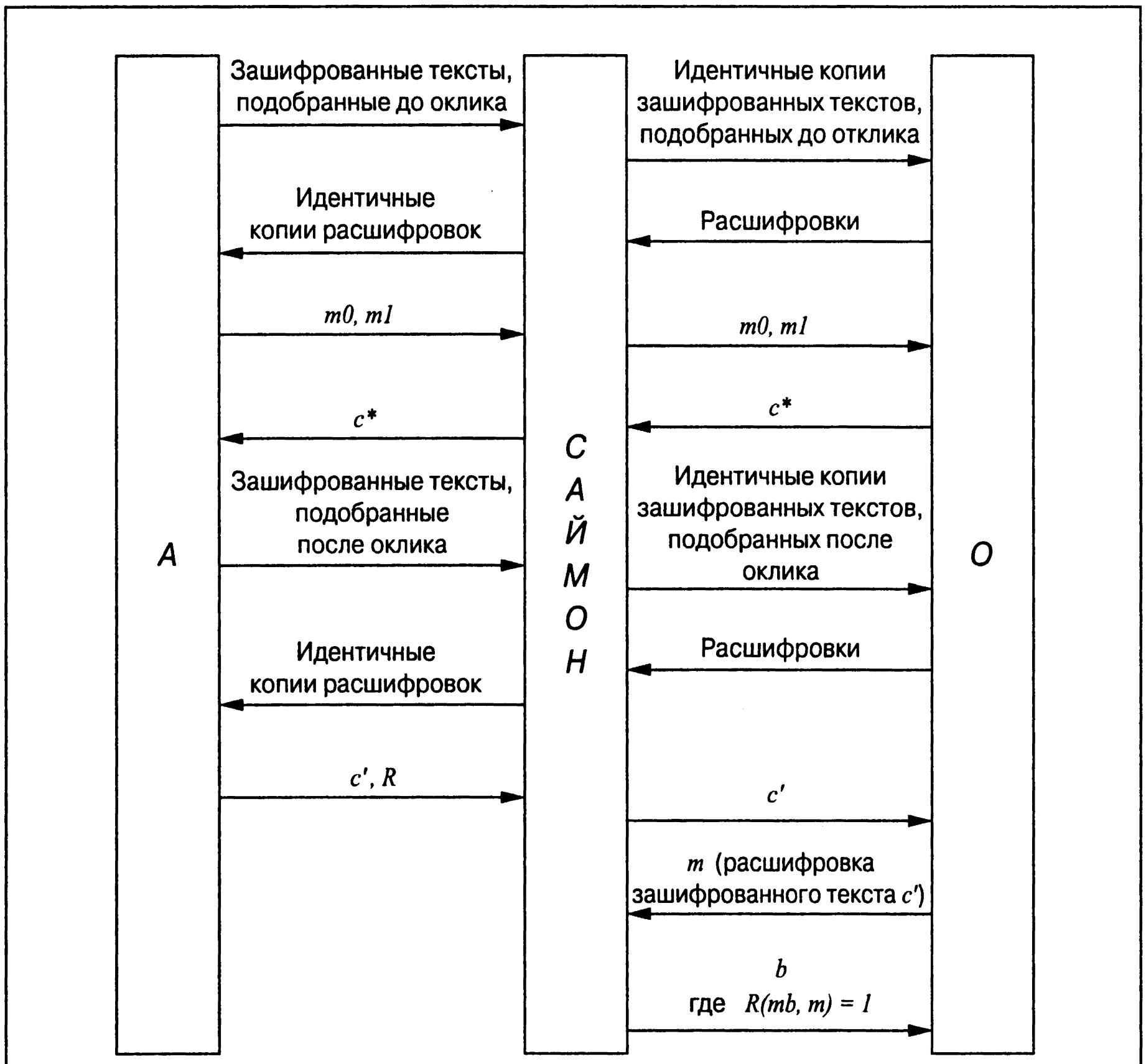


Рис. 14.3. Редукция атаки IND-CCA2 к атаке NM-CCA2

имитатору и атакующему $\mathcal{A}$ удалось провести редукцию и взломать криптосистему $\mathcal{E}_{pk}$ с помощью атаки NM-ATK.

Саймон одновременно управляет двумя атаками. Одна из этих атак протекает в режиме IND-ATK (т.е. в соответствии с одним из протоколов 14.1, 14.3 или 14.4). Саймон играет в ней роль оракула $\mathcal{O}$, взаимодействующего с атакующим, т.е. со Злоумышленником. Другая игра проходит в режиме NM-ATK (т.е. по протоколу 14.5). В ней Саймон играет роль Злоумышленника, взаимодействующе-

го с оракулом шифрования (и оракулом расшифровки, если ATK$\in\{$CCA, CCA2$\}$). На рис. 14.2 показан наиболее общий процесс редукции, когда ATK = CCA2. В остальных атаках некоторые взаимодействия между участниками игры могут быть пропущены.

Следует заметить, что в атаке на уязвимую криптосистему (см. правую часть взаимодействий, изображенных на рис. 14.2) распределение подобранных зашифрованных текстов является равномерным. Следовательно, оракул $\mathcal{O}$ должен зашифровывать случайные подобранные тексты.

Атакующий $\mathcal{A}$ должен "осмысленно" угадать бит $b\in\{0,1\}$. После этого Саймон может свободно вычислить зашифрованный текст $c'=\mathcal{E}_{pk}(m_b+1)$ и отношение $R(x,y)=1$ тогда и только тогда, когда $y=x+1$ для всех чисел x из пространства исходных текстов. Очевидно, что, если атакующий $\mathcal{A}$ использует полиномиально ограниченный алгоритм, Саймон может вычислить правильный результат за полиномиальное время.

Поскольку атакующий $\mathcal{A}$ угадывает бит b с преимуществом $\mathrm{Adv}(\mathcal{A})$, получаем следующую оценку.

$$\text{NM-Adv(Simon)} = \mathrm{Adv}(\mathcal{A}) - \mathrm{Prob}[(\bar{c}, R) \leftarrow \text{ZK-Sim}].$$

Обратите внимание на то, что имитатор ZK-Sim не имеет доступа к зашифрованному оклику c^* и, следовательно, не может взаимодействовать с атакующим $\mathcal{A}$. Таким образом, для результирующего зашифрованного текста $\bar{c}$ вероятность $\mathrm{Prob}[(\bar{c}, R) \leftarrow \text{ZK-Sim}]$ должна быть пренебрежимо малой. Следовательно, преимущество NM-Adv(Sim) не является пренебрежимо малой величиной, что и требовалось доказать. □

Напомним, что мы продемонстрировали множество атак в режиме IND-ATK на самые разнообразные криптосистемы. Из теоремы 14.5.4.2 следует, что все они уязвимы для атак NM-ATK.

Как известно, существуют криптосистемы, стойкие к атакам IND-CPA (соответственно IND-CCA), но уязвимые к атакам NM-CPA (соответственно NM-CCA) [19].

Среди зависимостей между понятиями стойкости к атакам в режимах NM и IND, проанализированных в работе [19], наиболее важным является тот факт, что из неразличимости следует строгая стойкость к атаке на основе адаптивно подобранных зашифрованных текстов.

14.5.4.2 Неразличимость эквивалентна строгой стойкости к атаке на основе адаптивно подобранного зашифрованного текста

Для ситуации, когда ATK = CCA2, справедлива следующая теорема.

Теорема 14.4. *Криптосистема с открытым ключом является стойкой к атаке NM-CCA2 тогда и только тогда, когда она является стойкой к атаке IND-CCA2.*

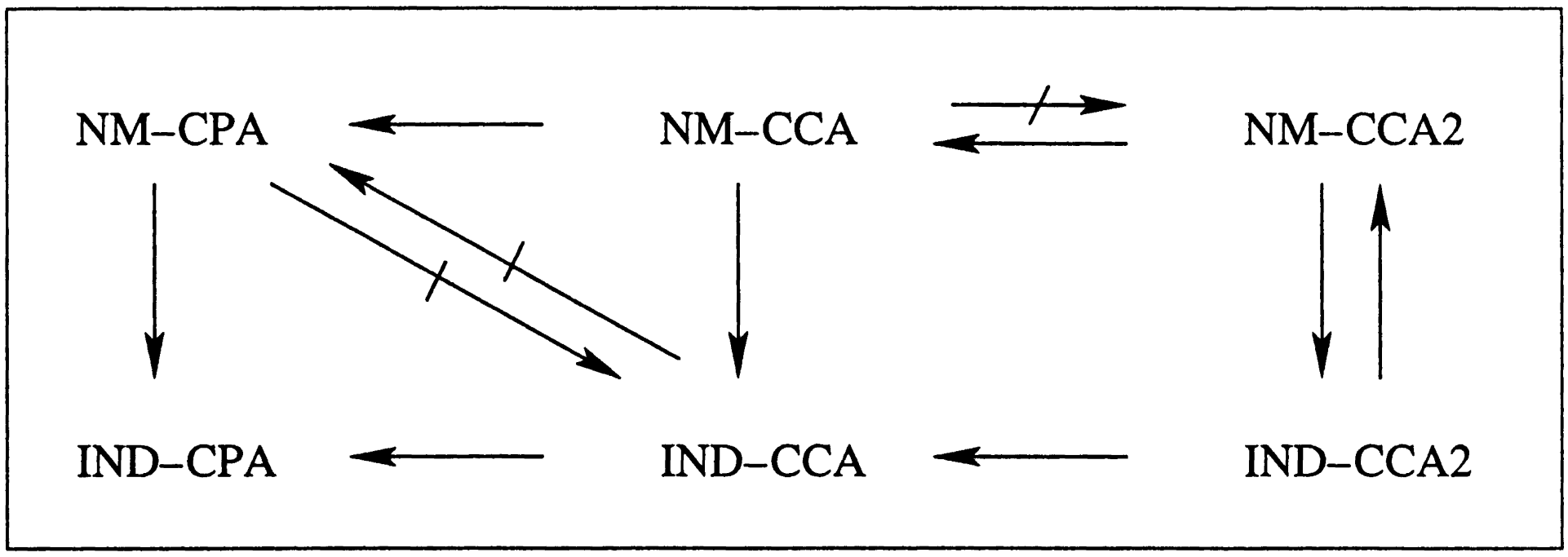


Рис. 14.4. Отношения между понятиями стойкости криптосистем с открытым ключом

Доказательство. В теореме 14.5.4.2 доказан следующий факт: NM-CCA2 ⇒ IND-CCA2. Следовательно, осталось доказать обратное: IND-CCA2 ⇒ NM-CCA2. Покажем, что, если криптосистема $\mathcal{E}_{pk}$ является уязвимой для атаки NM-CCA2, она также уязвима для атаки IND-CCA2.

Допустим, что криптосистема $\mathcal{E}_{pk}$ является уязвимой для атаки NM-CCA2. Тогда существует полиномиально ограниченный атакующий $\mathcal{A}$, способный взломать криптосистему $\mathcal{E}_{pk}$ с помощью атаки NM-CCA2 с преимуществом $\mathrm{Adv}(\mathcal{A})$, которое не является пренебрежимо малой величиной. Допустим, что Саймону-имитатору и атакующему $\mathcal{A}$ удалось провести редукцию и взломать криптосистему $\mathcal{E}_{pk}$ с помощью атаки NM-CCA2.

Процесс управления атаками показан на рис. 14.5.4.1. Заметим, что редукция стала возможной именно благодаря тому, что зашифрованный текст c', вычисленный атакующим $\mathcal{A}$, отличается от зашифрованного оклика c^*. Следовательно, Саймон одновременно управляет двумя атаками. Одна из этих атак протекает в режиме NM-CCA2. Саймон играет в ней роль Злоумышленника, посылающего оракулу $\mathcal{O}$ зашифрованный текст c', подобранный после оклика. Получив результат расшифровки, Саймон может проверить отношение между исходными текстами (заданное атакующим $\mathcal{A}$) и определить бит b.

Легко видеть, что, поскольку атакующий $\mathcal{A}$ является полиномиально ограниченным, Саймон угадает бит за полиномиальное время со значимым преимуществом, так как преимущество атакующего $\mathcal{A}$ также не является пренебрежимо малой величиной. □

На рис. 14.4 продемонстрированы все известные взаимосвязи между введенными понятиями стойкости. Если между двумя понятиями нет отношения следования, линия между ними не проводится. Детали этих взаимосвязей описаны в работе [19].

Из теоремы 14.4 следует, что в схемах шифрования с открытым ключом достаточно рассмотреть стойкость к атаке IND-CCA2. Этот вид стойкости легче доказать, чем стойкость к атаке NM-CCA2. Кроме того, вследствие эквивалентности стойкости к атакам IND-CCA2 и NM-CCA2 принято считать, что именно понятие стойкости к атаке IND-CCA2 является основным при исследовании алгоритмов шифрования с открытым ключом.

В следующей главе будут рассмотрены две прикладные криптосистемы, стойкие по отношению к атаке IND-CCA2.

14.6 Резюме

В главе рассмотрены различные понятия стойкости криптосистем с открытым ключом, упорядоченные по мере возрастания строгости.

Изучение начинается с протокола, в котором используется обычный учебный алгоритм. Показана его слабость и непригодность для практического применения. Затем вводится более строгое понятие стойкости: семантическая стойкость, или неразличимое шифрование при пассивной атаке. Продемонстрированы недостатки семантической стойкости и необходимость разработать понятие более сильной стойкости. В заключение изложено наиболее строгое понятие стойкости криптосистем с открытым ключом: неразличимое шифрование при атаке на основе адаптивно подобранного зашифрованного текста (IND-CCA2), которое пригодно для применения на практике. Продемонстрирована стойкость алгоритмов шифрования к различным сценариям атак: строгая стойкость и связанное с ней понятие стойкости к атаке IND-CCA2.

В настоящее время понятие стойкости к атаке IND-CCA2 является стандартным требованием, предъявляемым к криптосистемам с открытым ключом. Все новые схемы шифрования с открытым ключом должны обладать этим свойством. В следующей главе будут рассмотрены две прикладные криптосистемы, обладающие формально доказуемой стойкостью к атаке IND-CCA2.

Упражнения

14.1. Может ли учебный вариант алгоритма RSA скрывать знак символа Якоби исходного сообщения?

14.2. Стоек ли учебный вариант схемы шифрования RSA (Рабина) при шифровании, например, зарплаты? Стоек ли учебный вариант схемы шифрования Эль-Гамаля, если число, выражающее величину зарплаты, не принадлежит множеству $\langle g \rangle$?

14.3. Предположим, что в атаке на основе подобранного открытого текста (протокол 14.1) оракул $\mathcal{O}$ подбрасывает неидеальную монету, у которой вероят-

ность ОРЛА равна 2/3. Выведите формулу, позволяющую вычислить преимущество Злоумышленника по аналогии с формулой (14.2.3).

14.4. Должен ли Злоумышленник организовывать атаку на основе подобранного открытого текста для взлома алгоритма шифрования с открытым ключом?

14.5. Что такое семантическая стойкость? Для какой атаки является неуязвимой криптосистема с открытым ключом, обладающая семантической стойкостью: 1) если атакующий является пассивным и полиномиально ограниченным; 2) если атакующий является пассивным и полиномиально неограниченным; 3) если атакующий является активным и полиномиально ограниченным?

14.6. Семантическая стойкость означает сокрытие любой частичной информации об исходных сообщениях. Почему эта стойкость недостаточно сильна для применения на практике?

14.7. Предположим, что схема шифрования Рабина (алгоритм 8.2) подвергается атаке во время ленча (протокол 14.3). Каких результатов может ожидать атакующий?

Подсказка: в результате атаки во время ленча атакующий может *адаптивно* подбирать открытые сообщения и получать помощь при расшифровке.

14.8. Курсы обучения криптоанализу (помощь при шифровании и расшифровке) — весьма эффективное средство, позволяющее Злоумышленнику взламывать все учебные криптографические алгоритмы. Почему Злоумышленник, как правило, имеет доступ к этой услуге?

14.9. Что такое стойкость к атаке IND-CCA2? Какие атаки относятся к этой разновидности?

14.10. В чем заключается важность эквивалентности понятий стойкости к атакам IND-CCA2 и NM-CCA2?

Подсказка: опишите трудности, с которыми сопряжено применение понятия стойкости к атакам NM.

14.11. Предположим, что в ходе атаки пополуночи (протокол 14.4) Злоумышленник только посылает зашифрованные тексты, созданные с помощью заранее заданной схемы шифрования. Покажите, что в этом случае атака пополуночи вырождается в атаку во время ленча. Почему эта атака не вырождается в атаку IND-CPA?

Глава 15

Доказуемо стойкие и эффективные криптосистемы с открытым ключом

15.1 Введение

В предыдущей главе показано, что первые решения задачи о стойкости криптосистем с открытым ключом к атаке IND-CCA2 (соответственно NM-CCA2) основывались на неинтерактивных доказательствах с нулевым разглашением (NIZK). Такие доказательства демонстрируют получателю зашифрованного текста, что его автору уже известен соответствующий открытый текст, поскольку он в состоянии доказать утверждение о своем NP-членстве[1].

> "Зашифрованный текст c принадлежит языку L, определенному алгоритмом шифрования $\mathcal{E}$ при открытом ключе pk, а его автор обладает вспомогательной информацией (т.е. свидетельством NP-задачи), позволяющей доказать его членство."

Здесь под "вспомогательной информацией" ("auxiliary input"), позволяющей доказать членство автора, понимается соответствующий зашифрованный текст и случайное число, поступающее на вход алгоритма шифрования $\mathcal{E}$. (Случайное число необходимо для обеспечения семантической стойкости схемы шифрования.) Если верификация доказательства завершается успешно, получатель, который вольно или невольно предоставляет услуги по расшифровке сообщений, может быть уверен, что даже если автором зашифрованного текста c является Злоумышленник, возвращение ему соответствующего открытого текста не добавит ничего нового к тому, что он уже знает. Следовательно, такая помощь в расшифровке не повышает вероятность взлома криптосистемы.

Эта идея вполне разумная, но слишком дорогостоящая. Общепринятый метод неинтерактивного доказательства с нулевым разглашением заключается в том, что

---

[1] Взаимосвязи между утверждением об NP-членстве и доказательством с нулевым разглашением изучаются в главе 18.

доказывающая сторона (автор зашифрованного текста) и проверяющая сторона (получатель сообщения) владеют общей случайной строкой, вызывающей *обоюдное доверие* (mutually trusted). Это требование намного жестче, чем необходимо. Поскольку основным достоинством криптосистем с открытым ключом является тот факт, что двум сторонам не обязательно разделять между собой секретную информацию перед началом безопасных переговоров, доказуемая стойкость схем шифрования с открытым ключом не должна аннулировать этот преимущество![2]

Доказуемая стойкость должна означать лишь, что Злоумышленник не может взламывать систему *слишком быстро* или *слишком часто*. Таким образом, доказуемая стойкость следует из оценок вероятности и вычислительной сложности взлома. В контексте доказуемой стойкости требование гарантий, что Злоумышленнику должен быть известен соответствующий открытый текст, является завышенным, а доказательство NIZK — излишним и чрезмерным. Фактически ни одна из предыдущих схем шифрования с открытым ключом, обладающих свойством доказуемой стойкости к атаке IND-CCA2 и основанных на доказательстве NIZK, не является достаточно эффективной для практического применения.

Существует множество практически эффективных и доказуемо стойких схем шифрования с открытым ключом и схем цифровой подписи. Как правило, эти схемы основаны на учебных прототипах и используют механизм проверки целостности данных. Здесь под учебными прототипами подразумеваются алгоритмы с открытым ключом, использующие однонаправленные функции с секретом, например, функции RSA, Рабина и Эль-Гамаля (раздел 8.14). Механизм проверки целостности данных позволяет установить вероятность и вычислительную сложность взлома схемы шифрования.

Стоимость схемы шифрования, усиленной таким образом, ненамного превышает стоимость ее учебного прототипа.

15.1.1 Структурная схема главы

В главе описываются две эффективные схемы шифрования с открытым ключом, обладающие свойством доказуемой стойкости к атаке IND-CCA2 — **схема оптимального асимметричного шифрования с заполнением** (Optimal Asymmetric Encryption Padding — OAEP) [24, 114, 270] (раздел 15.2) и **криптосистема с открытым ключом Крамера–Шоупа** (Cramer–Shoup public-key cryptosystem) [84] (раздел 15.3). Затем приводится обзор семейства так называемых **гибридных криптосистем** (hybrid cryptosystem), представляющих собой комбинацию алгоритмов шифрования с открытым и секретным ключами. Показано, что они являются эффективными и обладают свойством доказуемой стойкости к атаке IND-CCA2

[2] В разделе 13.3 показано, что криптосистемы с открытым ключом можно сделать безопасными, даже если обе стороны не разделяют между собой открытую информацию.

(раздел 15.4). Глава завершается обзором литературы, посвященной прикладным и доказуемо стойким криптосистемам с открытым ключом (раздел 15.5).

15.2 Схема оптимального асимметричного шифрования с заполнением

Схема оптимального асимметричного шифрования с заполнением (OAEP) была изобретена Белларе и Роджуэем [24]. Она представляет собой **метод рандомизированного заполнения сообщений** (randomized message padding technique), осуществляющий легко обратимое отображение из пространства исходных сообщений в область **односторонних перестановок с секретом** (one-way trapdoor permutation — OWTP). Наиболее известными примерами таких перестановок являются функции RSA и Рабина [3]. Это отображение использует две криптографические функции хэширования и получает на вход исходное открытое сообщение, случайное число и строку нулей, обеспечивающих распознаваемость сообщений (см. рис. 15.1). Подробные инструкции по применению схемы RSA-OAEP (т.е. по вычислению перестановок OWTP с помощью функции RSA) изложены в алгоритме 10.6. Следует отметить, что, поскольку в схеме RSA-OAEP, описанной алгоритмом 10.6, процедура шифрования содержит дополнительный этап проверки, целое число, полученное в результате дополнения числа $s \parallel t$, всегда меньше модуля N в схеме RSA.

Схему шифрования с открытым ключом OAEP можно представить в виде последовательных преобразований.

$$\text{Исходный текст} \overset{\text{OAEP}}{\mapsto} \text{Область OWTP} \overset{\text{OWTP}}{\mapsto} \text{Зашифрованный текст.} \quad (15.2.1)$$

Рассмотрим три аспекта идеи, лежащей в основе этих преобразований.

Перемешивание разных алгебраических структур. Как указано в разделе 8.6, математическая функция, используемая в учебном алгоритме с открытым ключом, как правило, обладает хорошими и широко известными алгебраическими свойствами. Эти свойства она наследует от алгебраической структуры, определяющей перестановку OWTP (например, аксиомы группы или поля обеспечивают очень хорошие алгебраические свойства (см. определения 5.1 и 5.13)). Большое количество атак на учебные алгоритмы шифрования с открытым ключом, продемонстрированные до сих пор (см. алгоритм 14.1 и примеры 14.2 и 14.3), предполагали один и тот же образ действий Злоумышленника: манипулирование зашифрованным текстом, позволяющее

[3] В разделе 14.3.6.1 показано, почему алгоритм шифрования Рабина является перестановкой OWTP и как его применять.

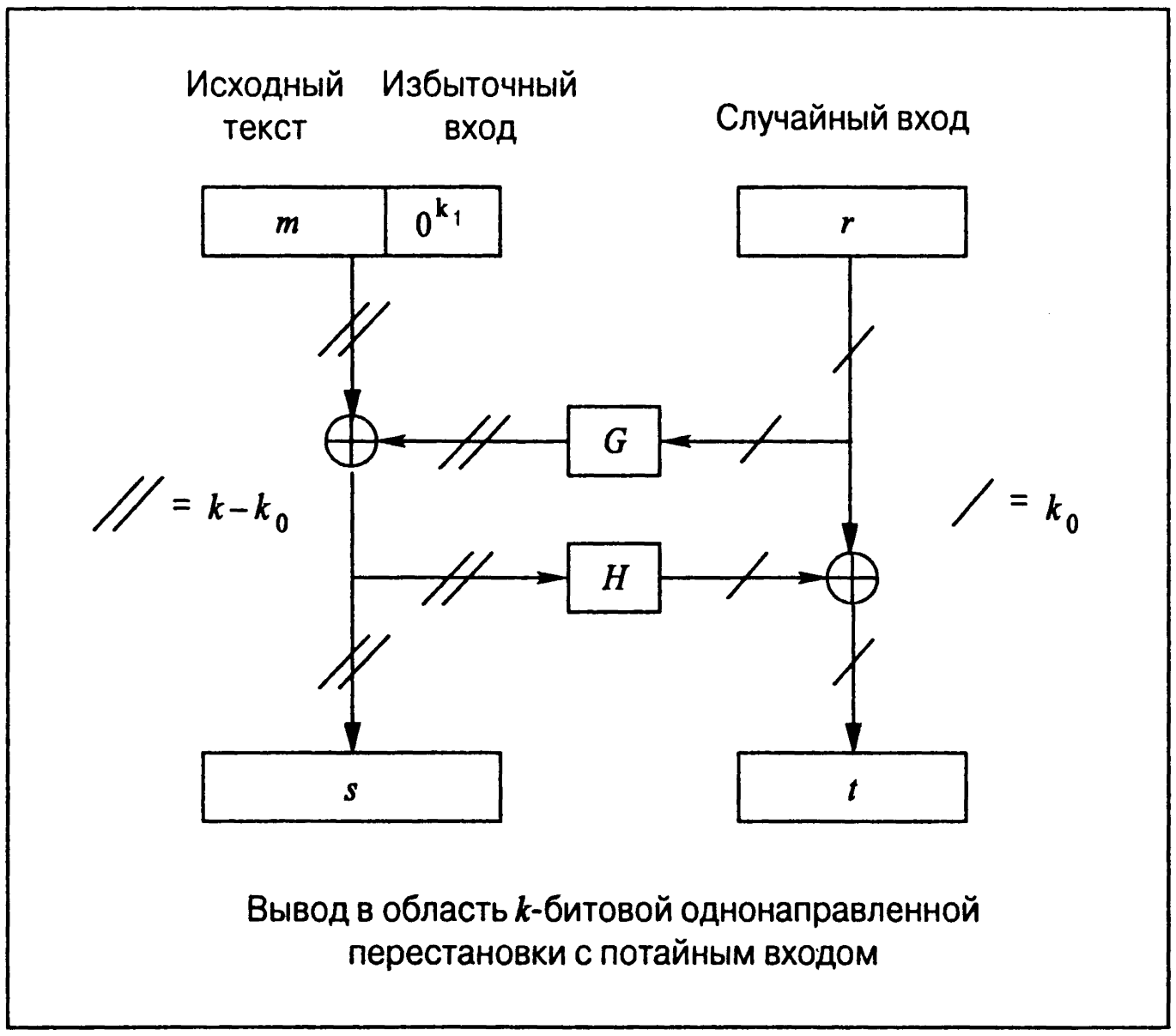


Рис. 15.1. Схема оптимального асимметричного шифрования с заполнением

целенаправленно изменять исходный текст на основе заранее известных алгебраических свойств перестановок OWTP.

В отличие от этих схем, преобразование OAEP использует сетевые криптографические функции хэширования с хорошо известной симметричной алгоритмической структурой. Действительно, как показано на рис. 15.2 (ср. с рис. 7.2), конструкцию преобразования OAEP можно рассматривать как двухраундовый шифр Файстеля (Feistel), в котором в первом раунде используется функция хэширования G, а во втором — вместо функции "S-блока", применяемой в шифре Файстеля, используется функция хэширования H. Тем не менее обе функции S-блока не снабжены ключами, а два половинчатых блока имеют разные размеры.

Эти две структуры, т.е. структура OWTP, лежащая в основе популярных криптосистем с открытым ключом, и структура шифра Файстеля, использованная в схеме OAEP, имеют совершенно разные алгебраические свойства. Очевидно, что первая структура имеет блочные алгебраические свойства в пространстве высокой размерности, а вторая — побитовые алгебраические свойства в пространстве размерности 2. Таким образом, есть основания надеяться, что комбинированное преобразование (15.2.1) должно поставить перед Злоумышленником непреодолимые препятствия и предотвратить це-

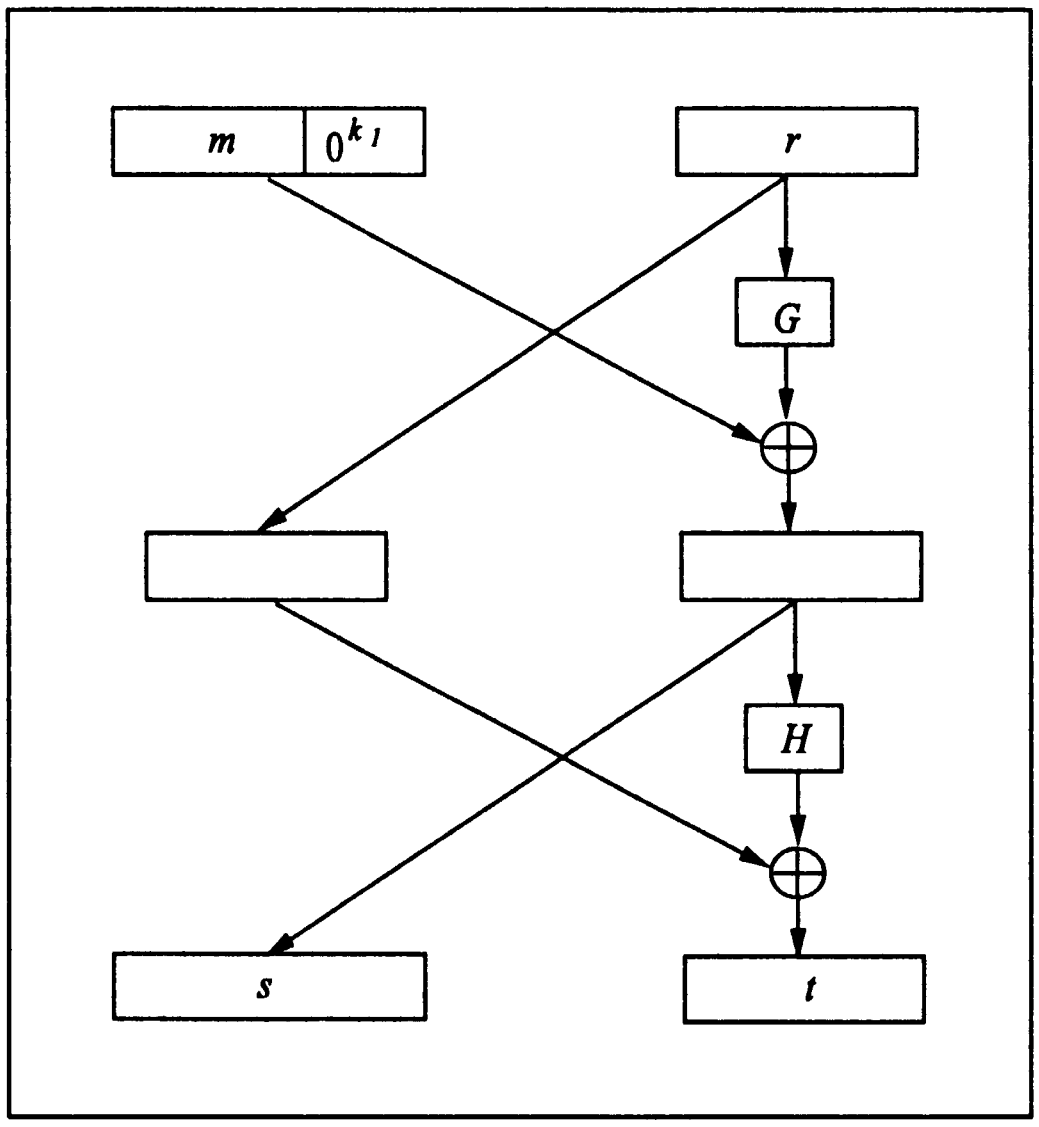


Рис. 15.2. Схема оптимального асимметричного шифрования с заполнением как двухраундовый шифр Файстеля

ленаправленную модификацию исходного текста с помощью манипулирования соответствующим текстом.

Рандомизация исходного текста. Как показано в разделе 9, если исходный текст, поступающий на вход учебной криптографической функции с открытым ключом, представляет собой случайную величину, функция обеспечивает полное сокрытие информации об исходном тексте, даже на уровне отдельного бита. Схема заполнения, такая как OAEP, получает на вход случайное число, гарантирующее случайность результата заполнения, т.е. обеспечивает случайность входа схемы OWTP. Таким образом, последовательная комбинация рандомизированной схемы заполнения со схемой OWTP создает побитовую стойкость примитивной криптографической функции с открытым ключом (см. главу 9).

Защита целостности данных. Мы много раз убеждались, что основным недостатком, присущим учебным криптоалгоритмам, является их чрезвычайная уязвимость к активным атакам. Обратимая функция (расшифровка RSA и сеть Файстеля) и добавленная избыточность 0^{k_1} допускают применение механизма проверки целостности данных (см. раздел 10.5). Таким образом, активная атака становится невозможной.

Эти три наблюдения имеют смысл, только если тексты, полученные путем рандомизированного заполнения, обладают достаточно хорошим случайным распределением в пространстве входных данных схемы OWTP.

Формальное доказательство схемы шифрования OAEP основано на мощном методе, называемом **моделью случайного оракула** (random oracle model). Такое доказательство предполагает, что функции хэширования, использованные в конструкции схемы OAEP, носят совершенно случайный характер. Эти функции называются **случайными оракулами** (random oracle) (их свойства описаны в разделе 10.3.1.2). Если функции хэширования, использованные в схеме заполнения, являются случайными оракулами, результат заполнения (т.е. случайные входные данные схемы OWTP) должен иметь равномерное распределение. Следовательно, интуитивно ясно, что этот способ приводит к результатам, полученным в главе 9.

Точнее говоря, целью доказательства стойкости схемы OAEP-OWTP, основанного на модели случайного оракула (ROM-based), является создание эффективного преобразования (называемого **редукцией** (reduction)) преимущества атаки на схему шифрования OAEP-OWTP в аналогичное преимущество инвертирования соответствующей перестановки OWTP (с точностью до полинома). Например, если перестановка OWTP является функцией RSA, инверсия решает задачу RSA или нарушает предположение о ее неразрешимости (определение 8.4, предположение 8.3 в разделе 8.7). Поскольку считается, что эффективного алгоритма инвертирования перестановки OWTP не существует, создание эффективного преобразования редукции приводит к противоречию. Доказательство, построенное таким образом, называется **сведением к противоречию** (reduction to contradiction).

15.2.1 Доказательство стойкости с помощью модели случайного оракула

Понятие **случайного оракула** введено в разделе 10.3.1.2. Случайный оракул представляет собой мощную гипотетическую детерминированную функцию, эффективно вычисляющую равномерно распределенные случайные величины.

Белларе и Роджуэй использовали эти свойства для доказательства стойкости схемы шифрования OAEP [24]. Их доказательство называется **моделью случайного оракула** (ROM) [22].

В модели ROM предполагается существование не только случайного оракула, но и специального агента, Саймона-имитатора, с которым мы уже встречались в разделе 14.5.4. Предполагается, что Саймон-имитатор может *имитировать любого случайного оракула* (даже Злоумышленника). Итак, если кто-либо захочет применить случайного оракула, скажем, оракула G, к числу a, он автоматически пошлет Саймону так называемый **запрос случайному оракулу** (random oracle

query) и получит от него результат $G(a)$. Саймон всегда честно выполняет любой запрос и возвращает его результат.

Благодаря этому правилу Саймон может точно имитировать поведение случайного оракула. Попробуем разобраться, как это ему удается.

Для оракула G, например, Саймон поддерживает G-список, содержащий все пары $(a, G(a))$, где хранятся все предшествующие запросы a. Имитация выполняется довольно просто: получив запрос a, Саймон проверяет, хранится ли он в списке. Если есть, Саймон просто возвращает значение $G(a)$ (вот почему результат запроса называется *детерминированным*), в противном случае Саймон извлекает из генеральной совокупности равномерно распределенных чисел новую величину $G(a)$, возвращает ее в качестве ответа на запрос (вот почему он называется *равномерным*) и заносит пару $(a, G(a))$ в список, упорядоченный по первому элементу пары. Дополнительный алгоритм сортировки списка применять необязательно, поскольку каждый список инициализируется как пустой и растет по мере выполнения запросов. При получении очередного запроса поиск элемента в списке, состоящем из N записей, можно выполнить за $\log N$ единиц времени (см. алгоритм 4.4), т.е. за полиномиально ограниченное время (вот почему имитатор называется *эффективным*).

Лемма 15.1. *Случайный оракул точно имитируется полиномиально ограниченным алгоритмом.* □

В схеме шифрования с открытым ключом, использующей случайных оракулов (например, в схеме OWTP-OAEP), этот способ имитации позволяет Саймону построить однозначное отображение исходного текста в зашифрованный текст (например, в схеме OWTP-OAEP Саймон создает отображение, заданное формулой (15.2.1)). Если атакующий, например, Злоумышленник, хочет создать *корректный* подобранный зашифрованный текст c, применяя OWTP-перестановку f, он должен обратиться к Саймону за расшифровкой. Следовательно, Саймон способен расшифровать сообщение c, даже не владея тайной информацией, позволяющей инвертировать функцию f. Это довольно просто, поскольку Саймон поддерживает список пар, состоящих из исходных и зашифрованных текстов. Действительно, если зашифрованный текст является корректным, он должен содержаться в списке случайного оракула.

Следовательно, кроме случайных оракулов, Саймон может *имитировать* оракулов расшифровки[4], например, оракула $\mathcal{O}$ в протоколах 14.3. и 14.4. Это еще одна причина, по которой наш специальный агент называется Саймоном-имитатором.

[4]Не следует путать оракула расшифровки и случайного оракула. Это совершенно разные понятия. Оракул расшифровки вполне реален, например, его роль может играть наивный пользователь, вынужденный расшифровывать сообщения Злоумышленника, а случайный оракул представляет собой гипотетическую функцию.

Имитируемая возможность расшифровки позволяет Саймону организовать "курс обучения криптоанализу" для Злоумышленника в рамках атаки IND-ATK (напомним, что сокращение ATK означает CPA, CCA или CCA2).

Если "курс обучения" выполняется точно, Злоумышленник должен успешно взламывать схему шифрования со значимым преимуществом и за относительно короткое время (полиномиальное). Иначе говоря, при атаке IND-ATK Злоумышленник должен связать один из двух подобранных зашифрованных текстов с зашифрованным окликом. Затем Симон, контролирующий случайных оракулов, также успешно завершит инвертирование криптографической функции, примененной к зашифрованному оклику: пару (исходный текст, зашифрованный оклик) можно найти в списке имитируемых случайных оракулов. Детали этого приема описаны в разделе 15.2.3.1. В следующей главе будет также описан трюк, позволяющий доказать стойкость некоторых цифровых схем с помощью модели случайных оракулов.

Все это хорошо, однако действует только в идеальном мире, в котором существуют случайные оракулы!

Несмотря на это, благодаря точной имитации поведения случайных оракулов с помощью криптографических функций хэширования схема шифрования OAEP оказалась вполне пригодной для практического применения. Хотя предположение об имитации случайных оракулов является недоказанным, его широко применяют на практике. В конце концов, любая схема OWTP, лежащая в основе популярных криптосистем с открытым ключом, основывается на недоказанных, но общепринятых предположениях.

Голдрайх (Goldreich) [65] считает, что методы доказательства стойкости, основанные на модели случайного оракула, представляют собой полезные испытательные схемы. Криптографические схемы, не прошедшие испытания, должны быть отвергнуты. В настоящее время считается общепризнанным, что современные криптографические схемы должны быть основаны на модели случайных оракулов.

Если функции хэширования (или псевдослучайные функции), используемые в реальных криптографических схемах или протоколах, не имеют очевидных недостатков, то доказательство их стойкости с помощью идеализированной версии считается корректным, особенно если целью является доказательство необоснованного предположения о полиномиальной неразличимости. Такое рассуждение называется **доказательством стойкости**, основанным на модели случайных оракулов (ROM-based security proof).

Перейдем к описанию доказательства, основанного на модели случайных оракулов. В следующей главе будут рассмотрены некоторые схемы цифровой подписи, стойкость которых доказывается с помощью модели случайных оракулов.

15.2.2 Схема RSA-OAEP

В схеме **RSA-OAEP** перестановка OWTP является результатом применения функции шифрования RSA. Отметим, что в оставшейся части главы все примеры, иллюстрирующие применение схемы RSA-OAEP, используют реализацию перестановки OWTP с помощью функции шифрования Рабина (раздел 14.3.6.1).

Поскольку в схеме шифрования RSA-OAEP сначала используются две функции хэширования, допускающие эффективное вычисление, а затем — функция RSA (см. алгоритм 10.6), эффективность этой схемы очень высока и сравнима с учебной схемой RSA. Эта схема обладает очень широкой полосой пропускания для восстановления сообщений. Если предположить, что модуль RSA состоит из 2048 бит (причина такого выбора будет указана в разделе 15.2.5) и $k_0 = k_1 = 160$ (так что числа 2^{-k_0} и 2^{-k_1} являются пренебрежимо малыми), то исходное сообщение может иметь длину $|M| = |N| - k_0 - k_1 = 2048 - 320 = 1728$ бит, т.е. длина исходного сообщения, зашифрованного в схеме RSA-OAEP, достигает 84% длины модуля.

Это свойство является очень важным с практической точки зрения. Благодаря ему схема шифрования RSA получила широкое признание со стороны организаций, осуществляющих международную стандартизацию (PKCS#1, IEEE, P1363). Она также стала основой для общеизвестного протокола SET, применяемого в электронной коммерции [259].

Итак, схема шифрования RSA-OAEP оказалась чрезвычайно успешной. Однако при доказательстве ее стойкости успех стал следствием неудачи.

Для того чтобы понять, *как* применяется схема RSA-OAEP, достаточно изучить алгоритм 10.6 и перейти к разделу 15.3. Дальнейшее изложение предназначено для читателя, которого интересует, *почему* схема RSA-OAEP является стойкой. Подчеркнем, что это изложение основано на интуитивных рассуждениях.

15.2.3 Трюк в доказательстве стойкости схемы RSA-OAEP

Оригинальное доказательство стойкости схемы шифрования f-OAEP, основанное на модели случайного оракула [24], базируется на сведении атаки IND-CCA2 на схему f-OAEP к задаче инвертирования OWTP-функции f с помощью скрытой (trapdoor) информации. Недавно Шоуп (Shoup) обнаружил пробел в этом доказательстве [270]. Более того, он заметил, что если функция f является OWTP-перестановкой, то само существование доказательства стойкости функции f-OAEP к атаке IND-CCA2 становится маловероятным. К счастью, довольно быстро опасность утратить весьма успешный алгоритм шифрования с открытым ключом была устранена! Более тщательный анализ, выполненный Фуджисаки

(Fujisaki) и его соавторами [114], позволил найти правильный способ применения функции RSA в схеме OAEP.

Рассмотрим эту драматическую ситуацию. Начнем с изучения оригинального доказательства стойкости, приведенного Белларе и Роджуэем. Затем будет приведено замечание Шоупа, демонстрирующее дефект в этом доказательстве. В заключение излагается работа Фуджисаки и его соавторов, спасающая положение. (Шоуп также разработал специальный вариант спасения, который мы рассмотрим в свое время.)

15.2.3.1 Редукция, основанная на модели случайного оракула

Предположим, что атакующий $\mathcal{A}$, которые выполняет алгоритм PPT, имеет значимое преимущество взлома схемы f-OAEP с помощью атаки IND-CCA2. Сконструируем алгоритм, позволяющий специальному агенту Саймону-имитатору использовать атакующего $\mathcal{A}$ для инвертирования функции f-OWTP со значимым преимуществом. Этот алгоритм должен быть эффективным (т.е. полиномиальным). Таким образом, Саймон эффективно "сводит" задачу инвертирования функции f к способности атакующего $\mathcal{A}$ взломать схему f-OAEP. По этой причине алгоритм, используемый Саймоном, называется **полиномиальной редукцией** (polynomial-time reduction). Поскольку алгоритмы, которые выполняют атакующий $\mathcal{A}$ и Саймон, являются полиномиальными, инвертирование функции f также выполняется за полиномиальное время. Считается, что инвертирование функции f невозможно выполнить за полиномиальное время. Это противоречит предположению о том, что атакующий $\mathcal{A}$ способен организовать успешную атаку IND-CCA2 на функцию f-OAEP (см. раздел 15.2.5). Такое доказательство стойкости называется **редукционным** (reductionist proof).

Рассмотрим процесс редукции.

Допустим, что Саймон имеет описание функции OWTP f и точку c^* из области равномерно распределенных значений функции f. Саймон желает вычислить значение $f^{-1}(c^*)$, используя организатора атаки IND-CCA2 $\mathcal{A}$. Следует заметить, что точка c^* должна быть случайной: если это условие не выполняется, результат алгоритма, выполняемого Саймоном, будет бесполезным.

Схематичное описание алгоритма редукции

Процесс редукции показан на рис. 15.2.3.2. Как видим, Саймон имеет доступ ко всем линиям, связывающим атакующего $\mathcal{A}$ с внешним миром, так что атакующий $\mathcal{A}$ может общаться только с Саймоном.

- Сначала Саймон посылает атакующему $\mathcal{A}$ описание алгоритма шифрования f-OAEP.
- Затем Саймон начинает с атакующим $\mathcal{A}$ игру IND-CCA2 (т.е. запускает протокол 14.4). В этой игре Саймон имитирует оракула расшифровки $\mathcal{O}$. Как будет показано далее, в рамках модели случайного оракула Саймон

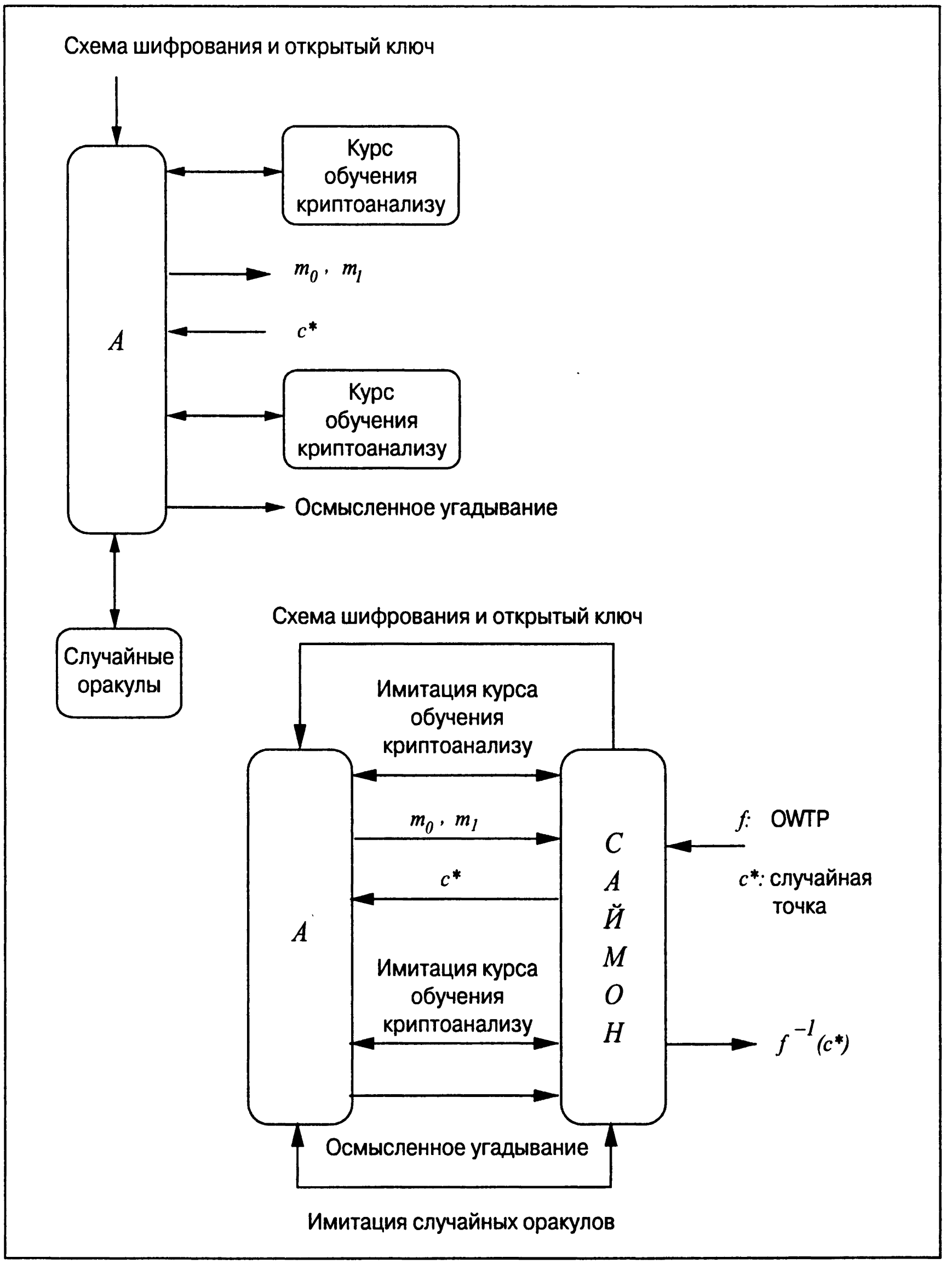


Рис. 15.3. Редукция задачи инвертирования однонаправленной функции с секретом f к атаке на схему f-OAEP

действительно может имитировать оракула расшифровки $\mathcal{O}$, не вызывая подозрений у атакующего $\mathcal{A}$.

- Саймон предоставляет атакующему $\mathcal{A}$ поддельные услуги, имитируя случайных оракулов G и H, задействованных в схеме OAEP (см. рис. 15.1).

Как указано в разделе 15.2.1, как только атакующему понадобится помощь оракулов G и/или H (например, при подборе зашифрованного текста), он пошлет запрос Саймону и получит соответствующие ответы.

Чрезвычайно важным обстоятельством является тот факт, что имитирование, выполняемое Саймоном, должно быть точным, чтобы атакующий $\mathcal{A}$ ничего не заподозрил. Только в этом случае атакующий получит полный объем знаний и облегчит свою задачу. Атака IND-CCA2, разыгранная между Саймоном и атакующим $\mathcal{A}$, развивается следующим образом.

1. На "этапе поиска" Симон получает для расшифровки индифферентно подобранные зашифрованные тексты, посланные атакующим $\mathcal{A}$. Эти тексты могут быть выбраны атакующим $\mathcal{A}$ по своему усмотрению, однако, если он хочет, чтобы зашифрованные тексты были подобраны правильно, т.е. с помощью случайных оракулов, запросы следует направлять Саймону. (Как показано на рис. 15.2.3.2, Саймон контролирует все каналы, связывающие атакующего $\mathcal{A}$ с внешним миром.) Методы имитации, применяемые Саймоном, будут описаны позднее.
2. Поскольку Саймон получает от атакующего $\mathcal{A}$ подобранные тексты, подлежащие расшифровке, он должен отвечать, имитируя блок расшифровки (оракула $\mathcal{O}$). Детали будут описаны в свое время.
3. Атакующий $\mathcal{A}$ завершает "этап поиска", посылая Саймону пару подобранных исходных текстов m_0, m_1. Получив эти тексты, Саймон "подбрасывает монету" $b \in_U \{0, 1\}$ и посылает атакующему $\mathcal{A}$ зашифрованный оклик c^*, представляющий собой сообщение m_b, зашифрованное с помощью *имитированного* алгоритма шифрования f-OAEP. Саймон считает, что зашифрованный текст c^* шифрует сообщение m_b.
4. Теперь атакующий $\mathcal{A}$ находится на "этапе угадывания". Следовательно, он может посылать адаптивно подобранные зашифрованные тексты, проходя "расширенный курс обучения криптоанализу". Саймон реагирует так, как показано в п. 2. Если атакующий $\mathcal{A}$ посылает запросы случайным оракулам, пытаясь создать адаптивно подобранные зашифрованные тексты, Саймон должен отвечать на них так, как показано в п. 1.

   В заключение атакующий $\mathcal{A}$ должен переслать результат осмысленного угадывания бита b. На этом атака завершается.

Как уже несколько раз указывалось, атакующий $\mathcal{A}$ не должен возвращать для расшифровки оклик c^*. Саймон не должен иметь возможности расшифровать сообщение c^*, поскольку именно этого ждет от него атакующий.

Имитация случайных оракулов. Саймон имитирует двух случайных оракулов G и H, используя преобразование OAEP. Для этого Саймон поддерживает два списка: G- и H-список, которые в начальный момент являются пустыми.

G-оракул. Предположим, что атакующий $\mathcal{A}$ создал запрос g, предназначенный для оракула G. Если запрос g уже хранится в списке, Саймон возвращает атакующему $\mathcal{A}$ результат $G(g)$. В противном случае запрос g является новым. В этом случае Саймон формирует случайную строку $G(g)$, имеющую равномерное распределение и длину k_0, возвращает атакующему $\mathcal{A}$ число $G(g)$ и заносит в G-список новую пару $(g, G(g))$.

Если запрос возник на "этапе угадывания", Саймон пытается инвертировать функцию f в точке c^*. Для каждой пары $(g, G(g))$, принадлежащей G-списку, и каждой пары $(h, H(h))$, принадлежащей списку H-списку, Саймон вычисляет значение $w = h \parallel (g \oplus H(h))$ и проверяет равенство $c^* = f(w)$. Если это равенство выполняется для некоторой строки, то значение $f^{-1}(c^*)$ считается искомым.

H-оракул. Предположим, что атакующий $\mathcal{A}$ создал запрос h, предназначенный для оракула H. Если запрос h уже хранится в списке, Саймон возвращает атакующему $\mathcal{A}$ результат $H(h)$. В противном случае запрос h является новым. В этом случае Саймон формирует случайную строку $H(h)$, имеющую равномерное распределение и длину $k - k_0$, возвращает атакующему $\mathcal{A}$ число $H(h)$ и заносит в H-список новую пару $(h, H(h))$.

Если запрос возник на "этапе угадывания", Саймон пытается инвертировать функцию f в точке c^*, как показано выше.

Имитация оракула расшифровки. Симон имитирует блок расшифровки (оракула $\mathcal{O}$) следующим образом. Получив от атакующего $\mathcal{A}$ зашифрованный текст c, Симон проверяет G- и H-списки в поисках пар $(g, G(g))$ и $(h, H(h))$. Для каждой пары, найденной в списке, Симон вычисляет значения

$$w = h \parallel (g \oplus H(h)),$$
$$v = G(g) \oplus h,$$

проверяет равенство

$$c \stackrel{?}{=} f(w)$$

и отвечает на вопрос:

"Содержит ли число vk_1 дополнительных нулей?"

Если на оба вопроса Саймон отвечает утвердительно, он возвращает атакующему $An = k - k_0 - k_1$ старших бит числа v. В противном случае Саймон возвращает атакующему $\mathcal{A}$ ОТКАЗ.

Поскольку атакующий $\mathcal{A}$ является полиномиально ограниченным, количество запросов случайным оракулам и запросов на расшифровку также ограничено полиномом. Следовательно, Саймон может завершить имитацию за полиномиальное время.

15.2.3.2 Точность имитации

Как уже указывалось, для того, чтобы атака была успешной, имитация должна быть точной!

Прежде всего, как следует из леммы 15.1, два случайных оракула имитируются точно. Проверим теперь, насколько точно Саймон имитирует блок расшифровки.

Предположим, что Саймон получил подобранный зашифрованный текст c (подобранный до или после оклика, т.е. индифферентно или адаптивно). Саймон очень точно имитирует блок расшифровки. Допустим, что величины

$$s \parallel t = f^{-1}(c), \tag{15.2.2}$$

$$r = t \oplus H(s), \tag{15.2.3}$$

$$m \parallel 0^{k_1} = s \oplus G(r) \tag{15.2.4}$$

определены корректным зашифрованным текстом c. С этого момента под термином "корректное значение" подразумевается величина, вычисленная реальным оракулом расшифровки $\mathcal{O}$ в ответ на полученный зашифрованный текст c.

Если корректный текст s, определенный по формуле (15.2.2), не является запросом оракулу H, то корректного значения $H(s)$ не существует. Следовательно, вероятность того, что величина r в каждом запросе оракулу G является корректной, равна 2^{-k_0}. Таким образом, ни в запросе оракулу H, ни в запросе оракулу G нет корректных значений (вероятность противоположного события равна 2^{-k_0}). Значит, вероятность того, что в числах s и $G(r)$ содержатся k_1 нулевых младших разрядов, равна 2^{-k_1}, поскольку для этого необходимо, чтобы запросы s и $G(r)$ содержали k_1 идентичных младших бита. Это невозможно, поскольку число s не существует, а запрос $G(r)$ является равномерно распределенным. Заметим, что этот анализ учитывает случай, когда число r является корректным и не содержится в запросе оракулу G: вероятность ошибки в этом случае равна 2^{-k_1}.

Итак, мы пришли к следующим выводам.

- Если оба числа s и r содержатся в запросах соответствующим случайным оракулам, то имитация расшифровки позволяет правильно сконструировать значение $f^{-1}(c)$ и расшифровать сообщение c.
- Если число s и/или r не содержится в запросах соответствующим случайным оракулам, то правильная имитация расшифровки должна возвращать отказ с вероятностью ошибки, равной $2^{-k_0} + 2^{-k_1}$.

Отметим, что во втором случае вероятность ошибки имеет *статистический* смысл, т.е. не зависит от мощности атакующего $\mathcal{A}$.

Итак, мы доказали, что схема f-OAEP является доказуемо стойкой к атаке IND-CCA (т.е. к атаке во время ленча или на основе индифферентно подобранного зашифрованного текста). Это утверждение справедливо, поскольку в атаке IND-CCA блок расшифровки включается только на "этапе поиска", а мы ранее

показали, что вероятность неправильной имитации расшифровки на этом этапе крайне незначительна.

Следует подчеркнуть, что все приведенные выше рассуждения основаны лишь на однонаправленности функции f.

15.2.3.3 Неполнота

Шоуп обнаружил, что оригинальное доказательство стойкости схемы OAEP к атаке IND-CCA2 содержит пробел [270]. Прежде чем приступить к дальнейшим разъяснениям, подчеркнем, что на самом деле никакого дефекта в конструкции схемы OAEP *нет*. Просто формальное доказательство, приведенное в разделе 15.2.3.2, не является полным. Этот недостаток проявляется в следующем.

> Имитация расшифровки, выполняемая Саймоном, является статистически точной, поскольку число s^*, определяемое по оклику c^* в формуле (15.2.5), не содержится в запросе случайному оракулу H. Однако статистическая точность исчезает, если число s^* содержится в запросе, причем оценка вероятности этого события в доказательстве отсутствует.

Для того чтобы объяснить неполноту доказательства, рассмотрим разные величины, определяемые по зашифрованному оклику c^*. Допустим, что

$$s^* \parallel t^* = f^{-1}(c^*), \tag{15.2.5}$$

$$r^* = t^* \oplus H(s^*), \tag{15.2.6}$$

$$m_b \parallel 0^{k_1} = s^* \oplus G(r^*). \tag{15.2.7}$$

Величины (s^*, r^*, m_b) определяются по зашифрованному оклику c^*, где число b является результатом жеребьевки с помощью подбрасывания идеальной монеты, выполняемой Саймоном.

Представим себе, что число s^* содержится в запросе случайному оракулу H. Разумеется, это событие является весьма маловероятным, поскольку на "этапе поиска" зашифрованный оклик c^* еще не вычислен. Вот почему в разделе 15.2.3.2 мы пришли к выводу, что доказательство стойкости схемы f-OAEP к атаке IND-CCA является *корректным*. Однако существует *вероятность*, что атакующий $\mathcal{A}$ угадает число s^*, получив оклик c^* (т.е. на "этапе угадывания").

Чему равна вероятность этого события? Это никому не известно. Определенно можно утверждать лишь, что, если атакующий $\mathcal{A}$ является достаточно мощным, эта вероятность не равна нулю. В противном случае почему мы должны предполагать, что атакующий $\mathcal{A}$ вообще способен угадать бит b? (Несмотря на это, можно оценить условную вероятность угадывания числа s^* в запросе. Эта вероятность не равна нулю. Соответствующая оценка приводится в разделе 15.2.3.4.)

Поскольку атакующий $\mathcal{A}$ может угадать число s^*, атака содержит противоречие, которое легко обнаружить. Зафиксируем значение r^*, вычисленное по формуле (15.2.6). Тогда атакующий $\mathcal{A}$ может послать его в следующем запросе. Вероятность того, что для равномерно распределенной случайной величины $G(r^*)$, возвращенной Саймоном, число $G(r^*) \oplus s^*$ совпадет с каким-либо из подобранных исходных текстов, весьма мала. Следовательно, в этот момент атакующий $\mathcal{A}$ воскликнет: "Прекратите дурачиться!". Он должен воскликнуть именно так, поскольку зашифрованный оклик c^* не соответствует ни одному из подобранных исходных текстов.

Разумеется, эта "легкость" слишком дорого стоит: атакующий $\mathcal{A}$ вынужден раскрыть Саймону оба числа s^* и $t^* = r^* \oplus H(s^*)$ и помочь ему, таким образом, инвертировать функцию f в точке $c^* = f(s^* \parallel t^*)$!

Шоуп нашел этой задаче лучшее применение. Он обнаружил, что для некоторой функции f, например, для перестановки OWTP, способности угадать запрос s^* по заданному зашифрованному тексту c^* вполне достаточно для создания корректного зашифрованного текста без запроса числа r^* у случайного оракула G (фактически вообще без взаимодействия с оракулом G на протяжении всей атаки). Более того, поскольку корректный зашифрованный текст, построенный таким образом, является результатом уязвимости другого корректного зашифрованного текста, схема f-OAEP является стойкой к атаке NM-CCA2 и по теореме 14.4 (раздел 14.5.4.2) стойкой к атаке IND-CCA2. Однако метод редукции, описанный в разделе 15.2.3.1, не помогает Саймону инвертировать функцию f, поскольку G-список может быть даже пустым (т.е. атакующий $\mathcal{A}$ еще не посылал ни одного запроса случайному оракулу G).

Шоуп построил в качестве контрпримера k-битовую OWTP-функцию f. Он предположил, что эта перестановка является "xor-гибкой": по заданным числам $f(w_1)$, w_2 со значительным преимуществом можно найти число $f(w_1 \oplus w_2)$. Отметим, что это является вполне разумным предположением. В доказательстве стойкости схемы f-OAEP, описанном в разделе 15.2.3.1, мы требовали лишь, чтобы функция f была однонаправленной, и она не обязана была быть стойкой. Помимо всего, как показано в предыдущей главе, учебные алгоритмы шифрования с открытым ключом, лежащие в основе схем OWTP, являются уязвимыми. Именно по этой причине возник метод OAEP.

Чтобы прояснить ситуацию, допустим, что функция f вообще не скрывает $k - k_0$ старших бит. Тогда эту функцию f можно представить в виде

$$f(s \parallel t) = s \parallel f_0(t),$$

где функция f_0 является "xor-гибкой" $(k - k_0)$-битовой перестановкой OWTP. Следовательно, по заданным числам $f_0(t_1)$, t_2 со значительным преимуществом можно вычислить значение $f_0(t_1 \oplus t_2)$. Эта функция f остается OWTP-перестановкой, обладающей свойством однонаправленности и параметром безопасности k_0.

Рассмотрим теперь схему шифрования f-OWTP, использующую функцию f. Напомним, что если c^* — зашифрованный оклик, то в схеме f-OAEP ему соответствуют величины s^*, t^*, r^* и $m_b \parallel 0^{k_1}$.

Поскольку атакующий $\mathcal{A}$ представляет собой черный ящик, корректный зашифрованный текст можно легко получить с помощью модификации другого корректного зашифрованного текста. Получив зашифрованный оклик c^*, атакующий $\mathcal{A}$ представляет зашифрованный текст в виде $c^* = s^* \parallel f_0(t^*)$. Затем он выбирает произвольное ненулевое сообщение $\Delta \in \{0,1\}^{k-k_0-k_1}$ и вычисляет следующее значение.

$$s = s^* \oplus (\Delta \parallel 0^{k_1}), v = f_0(t^* \oplus H(s^*) \oplus H(s)), c = s \parallel v.$$

Очевидно, что для того, чтобы создать новый зашифрованный текст c на основе зашифрованного оклика c^*, атакующему $\mathcal{A}$ необходимо лишь послать оракулу H запросы s^* и s.

Докажем теперь, что число c является корректной величиной $m_b \oplus \Delta$, зашифрованной по схеме f-OAEP, поскольку зашифрованный текст c^* является числом m_b, корректно зашифрованным по схеме f-OAEP. По числу $t = t^* \oplus H(s^*) \oplus H(s)$ можно найти величину r.

$$r = H(s) \oplus t = t^* \oplus H(s^*) = r^*. \tag{15.2.8}$$

Очевидно, что формула (15.2.8) выполняется даже тогда, когда атакующий $\mathcal{A}$ не посылает оракулу G запрос $r = r^*$ (вполне вероятно, что ему не известно число r^*, так как он может не знать число t^*).

Если бы в этой игре участвовали атакующий $\mathcal{A}$ и реальный оракул расшифровки $\mathcal{O}$, то оракул $\mathcal{O}$ мог бы восстановить число r, выполнив следующие вычисления.

$$f^{-1}(c) = s \parallel t,$$
$$r = H(s) \oplus t.$$

Однако, учитывая формулу (15.2.8), на самом деле оракул $\mathcal{O}$ может найти число r^*. Итак, применяя функцию хэширования G, оракул $\mathcal{O}$ может вычислить следующие значения.

$$G(r) \oplus s = G(r^*) \oplus s^* \oplus (\Delta \parallel 0^{k_1}) = (m_b \oplus \Delta) \parallel 0^{k_1}.$$

Выполняя поиск младших k_1 нулевых разрядов, оракул $\mathcal{O}$ в результате правильной расшифровки сообщения c должен вернуть число $m_b \oplus \Delta$. Получив исходный текст $m_b \oplus \Delta$, атакующий $\mathcal{A}$ может легко найти число m_b и взломать таким образом схему f-OAEP с помощью атаки IND-CCA2.

Однако, поскольку атака разыгрывается между атакующим $\mathcal{A}$ и Саймоном, а не реальным оракулом расшифровки, G-список, поддерживаемый Саймоном, является пустым, и Саймон должен немедленно вернуть атакующему отрицательный ответ. Вот теперь атакующий $\mathcal{A}$ действительно очень громко воскликнет:

"ПРЕКРАТИТЕ ДУРАЧИТЬСЯ!"

15.2.3.4 Вероятность того, что атакующий $\mathcal{A}$ пошлет запрос s^* на "этапе угадывания"

Мы оставили в стороне одну маленькую тонкость, связанную с оригинальным доказательством Белларе–Роджуэя: вероятность того, что атакующий $\mathcal{A}$ пошлет запрос s^* на "этапе угадывания" при условии, что он может правильно определить бит оклика. Поскольку эта часть доказательства связана с оценкой вероятности, ее можно безболезненно пропустить. (На самом деле это доказательство элементарно, и для его понимания достаточно знать правила, установленные в главе 3.)

Для начала предположим, что атакующий $\mathcal{A}$ может правильно угадать бит b в оклике с преимуществом Adv после долгого обучения адаптивному подбору зашифрованного текста.

В ходе обучения Саймон может ошибочно отвергнуть корректный зашифрованный текст, содержащийся в запросе на расшифровку. Это — неприятное событие, демонстрирующее низкое качество обучения атакующего $\mathcal{A}$. По этой причине обозначим это событие символами DBad. В разделе 15.2.3.2 анализ процедуры, выполняемой Саймоном, показал, что имитация расшифровки является точной: вероятность ошибки равна $2^{-k_0} + 2^{-k_1}$. Таким образом,

$$\text{Prob}[\mathsf{DBad}] \approx 2^{-k_0} + 2^{-k_1}. \tag{15.2.9}$$

Пусть символы AskG (соответственно AskH) обозначают событие, которое заключается в том, что число r^* (соответственно s^*) содержится в G-списке (соответственно в H-списке). Эти два события являются нежелательными, поскольку они сообщают атакующему, что оклик c^* на самом деле не связан с подобранными исходными сообщениями m_0 и m_1. Определим событие Bad следующим образом.

$$\mathsf{Bad} = \mathsf{AskG} \cup \mathsf{AskH} \cup \mathsf{DBad}.$$

Обозначим через $\mathcal{A}$wins событие, которое заключается в том, что атакующий $\mathcal{A}$ правильно угадывает бит оклика b. Очевидно, что в отсутствие события Bad бит оклика b не зависит от зашифрованного текста c^*, поскольку все случайные величины, которые генерируются случайными оракулами, имеют равномерное распределение. Следовательно,

$$\text{Prob}[\mathcal{A}\mathsf{wins} \mid \overline{\mathsf{Bad}}] = \frac{1}{2}. \tag{15.2.10}$$

Применяя правило вычисления условной вероятности (определение 3.3 в разделе 3.4.1), выразим формулу (15.2.10) иначе.

$$\mathrm{Prob}[\mathcal{A}\mathsf{wins} \mid \overline{\mathsf{Bad}}] = \frac{\mathrm{Prob}\left[\mathcal{A}\mathsf{wins} \cap \overline{\mathsf{Bad}}\right]}{\mathrm{Prob}\left[\overline{\mathsf{Bad}}\right]} = \frac{1}{2}.$$

$$\mathrm{Prob}[\mathcal{A}\mathsf{wins} \cap \overline{\mathsf{Bad}}] = \frac{1}{2}(1 - \mathrm{Prob}[\mathsf{Bad}]). \tag{15.2.11}$$

Необходимо отметить, что при событии $\overline{\mathsf{Bad}}$ (т.е. в отсутствие события Bad) имитируемые случайные оракулы и имитируемый блок расшифровки работают идеально. Итак,

$$\mathrm{Prob}[\mathcal{A}\mathsf{wins}] = \frac{1}{2} + \mathrm{Adv}\,. \tag{15.2.12}$$

С другой стороны (теорема 3.1 из раздела 3.4.3),

$$\mathrm{Prob}[\mathcal{A}\mathsf{wins}] = \mathrm{Prob}[\mathcal{A}\mathsf{wins} \cap \overline{\mathsf{Bad}}] + \mathrm{Prob}[\mathcal{A}\mathsf{wins} \cap \mathsf{Bad}]. \tag{15.2.13}$$

Сопоставив формулы (15.2.12) и (15.2.13), получаем

$$\mathrm{Prob}[\mathcal{A}\mathsf{wins} \cap \overline{\mathsf{Bad}}] + \mathrm{Prob}[\mathcal{A}\mathsf{wins} \cap \mathsf{Bad}] = \frac{1}{2} + \mathrm{Adv}\,.$$

$$\mathrm{Prob}[\mathcal{A}\mathsf{wins} \cap \overline{\mathsf{Bad}}] + \mathrm{Prob}[\mathsf{Bad}] \geqslant \frac{1}{2} + \mathrm{Adv}\,. \tag{15.2.14}$$

Учитывая формулу (15.2.11), неравенство (15.2.14) можно переписать следующим образом.

$$\frac{1}{2}(1 - \mathrm{Prob}[\mathsf{Bad}]) + \mathrm{Prob}[\mathsf{Bad}] \geqslant \frac{1}{2} + \mathrm{Adv},$$

т.е.

$$\mathrm{Prob}[\mathsf{Bad}] \geqslant \mathrm{Adv}\,. \tag{15.2.15}$$

Поскольку $\mathsf{Bad} = \mathsf{AskG} \cup \mathsf{AskH} \cup \mathsf{DBad}$, получаем

$$\mathrm{Prob}[\mathsf{Bad}] \leqslant \mathrm{Prob}[\mathsf{AskG} \cup \mathsf{AskH}] + \mathrm{Prob}[\mathsf{DBad}] = \tag{15.2.16}$$

$$= \mathrm{Prob}[\mathsf{AskH}] + \mathrm{Prob}[\mathsf{AskG} \cap \overline{\mathsf{AskH}}] + \mathrm{Prob}[\mathsf{DBad}] \leqslant \tag{15.2.17}$$

$$\leqslant \mathrm{Prob}[\mathsf{AskH}] + \mathrm{Prob}[\mathsf{AskG} \mid \overline{\mathsf{AskH}}] + \mathrm{Prob}[\mathsf{DBad}]. \tag{15.2.18}$$

Формула (15.2.6) справедлива по первому правилу сложения вероятностей, формула (15.2.7) вытекает из утверждений, доказанных в примере 3.3, а формула (15.2.18) следует из определения условной вероятности и того факта, что вероятность никогда не превышает единицу.

В заключение отметим, что равномерное распределение случайных величин, генерируемых оракулом H, и условное событие $\mathsf{AskG} \mid \overline{\mathsf{AskH}}$ (т.е. число r^* содержится в запросе, а число s^* — нет) могут возникать с вероятностью 2^{-k_0}. Кроме

того, из формулы (15.12.9) следует, что вероятность события DBad также равна $2^{-k_0} + 2^{-k_1}$. Из неравенств (15.2.15)–(15.2.18) явствует, что

$$\text{Prob}[\mathsf{AskH}] \geqslant \text{Adv} - (2^{-k_0+1} + 2^{-k_1}).$$

Следовательно, если преимущество Adv не является пренебрежимо малым по сравнению с параметром k, вероятность Prob[AskH] также является существенной.

Итак, совершенно ясно, что если атакующий способен взломать криптосистему с помощью атаки IND-CCA2, а криптосистема RSA-OAEP реализована с помощью случайных оракулов, то атакующий может частично инвертировать функцию RSA, найдя число s^* с тем же самым преимуществом. **Частичная инверсия** (partial inversion) функции RSA может привести к полной инверсии. Рассмотрим ситуацию, в которой это возможно.

15.2.4 Исправленный вариант схемы RSA-OAEP

Математические выкладки, приведенные в разделе 15.2.3.4, играют важную роль в процессе исправления схемы RSA-OAEP. Однако первая попытка исправить схему не использовала эти рассуждения.

15.2.4.1 Первая попытка Шоупа

К счастью, свойство гибкости функции RSA не похоже на свойство гибкости шифра Файстеля, основанного на OAEP-преобразовании (см. рис. 15.2 и обсуждение различий между алгебраическими свойствами этих двух структур). Ирония судьбы заключается в том, что значительная разница между алгебраическими свойствами этих двух структур позволила Шоупу доказать, что схема RSA-OAEP является стойкой к атаке IND-CCA2 при условии, что показатель степени в схеме шифрования RSA чрезвычайно мал: если N — это модуль RSA, то в доказательстве Шоупа требуется, чтобы выполнялось следующее неравенство.

$$k_0 \leqslant (\log_2 N)/e. \tag{15.2.19}$$

Попробуем разобраться, зачем нужно это условие.

Напомним, что в результате анализа мы пришли к выводу, что если число s^*, определенное зашифрованным окликом c^* в формуле (15.2.5), не содержится в запросе, посланном оракулу H, то редукция является статистически корректной. Единственная ситуация, когда редукция оказывается некорректной, возникает тогда, когда число s^* содержится в запросе, посланном оракулу H. В этой ситуации поведение Саймона становится неопределенным.

Шоуп обнаружил, что, если число s^* представляет собой строку, состоящую из $k-k_0$ бит, и содержится в H-списке, Саймон может решить следующее уравнение:

$$\left(X + 2^{k_0} I(s^*)\right)^e \equiv c (\text{mod}\, N), \tag{15.2.20}$$

где $I(x)$ — целое число, заданное строкой x. Если $X < N^{1/e}$, это уравнение можно решить за время, полиномиально зависящее от числа N, используя алгоритм Копперсмита [82]. Если число X сравнимо с числом 2^{k_0}, и справедливо ограничение (15.2.19), то условие $X < N^{1/e}$ выполняется.

Итак, если Саймон получит для расшифровки текст c и не сможет его расшифровать, используя метод, описанный в разделе 15.2.3.1, он должен попытаться решить уравнение (15.2.20) относительно величины X, используя каждый элемент из своего H-списка. Если все попытки окончатся неудачей, Саймон должен отвергнуть зашифрованный текст c. В противном случае решением уравнения является величина $X = I(t^*)$. Зная числа s^* и t^*, Саймон должен расшифровать текст c, как обычно.

Таким образом, для схемы RSA-OAEP, показатель степени которой удовлетворяет условию (15.2.20), запрос s^*, посланный оракулу H, помогает Саймону инвертировать число c^*. Вопрос заключается в следующем: какова величина числа c^*, удовлетворяющего условию (15.2.20)? Для стандартного параметра безопасности, применяемого в схеме RSA-OAEP, должны выполняться условия $N > 2^{1024}$ и $k_0 = 160$ (для того, чтобы число 2^{-k_0} было пренебрежимо малым). Следовательно, $e \leqslant \frac{1024}{160} = 6{,}4$. Итак, в стандартном варианте $e = 3$ или $e = 5$ — единственно возможные варианты показателя степени (число e должно быть взаимно простым с числом $\phi(N)$, которое является четным). Несмотря на то, что эти малые значения показателей степени позволяют обеспечить доказуемую стойкость схемы RSA-OAEP, после появления работы Копперсмита считается, что применять их для шифрования не следует.

Поскольку стандартные установки параметра безопасности k_0 близки к нижней границе, а число N невозможно резко увеличить, маловероятно, что применение метода редукции к более высоким значениям показателя e приведет к успеху.

15.2.4.2 Полное решение, полученное Фуджисаки и его соавторами

Вскоре после появления работы Шоупа Фуджисаки и его соавторы [114] обнаружили способ инвертировать функцию RSA, не налагая на показатель степени никаких ограничений.

В схеме RSA-OAEP раскрытие Саймону числа s^*, представляющего собой значительную часть прообраза зашифрованного текста c^*, является слишком избыточным. Поскольку число s^* состоит из $k - k_0$ бит и $k > 2k_0$, более половины бит (причем старших) прообраза зашифрованного текста c^* оказываются раскрытыми. Зная такую крупную часть прообраза, Фуджисаки с соавторами применили метод "бриллиантовой решетки", позволяющий найти число $T = I(t^*)$ из уравнения

$$\left(2^{k_0} I(s^*) + T\right)^e \equiv c^* (\mathrm{mod}\, N), \tag{15.2.21}$$

где число e может быть произвольно большим. Напомним, что алгоритм 9.1 инвертирует функцию RSA, $\log_2 N$ раз применяя однобитовый оракул. Применим

точно такой же принцип, учитывая, что атакующий $\mathcal{A}$ фактически является оракулом RSA, позволяющим ответить на вопрос: "Правда ли, что число s^* содержит больше половины битов прообраза зашифрованного текста c^*?". Используя алгоритм Фуджисаки, Саймон может дважды применить атакующий алгоритм $\mathcal{A}$ и получить два связанных между собой блока (блоки "больше-половины-или-нет") частичной информации о прообразе. Эти два блока можно использовать в формуле (15.2.21) для поиска двух неизвестных целых чисел, которые меньше $\sqrt{N}$. Одним из этих двух чисел является число $T(t^*)$. Таким образом, Саймон инвертирует функцию RSA.

Поскольку Саймон применяет атакующий алгоритм $\mathcal{A}$ дважды, он должен дважды разыграть с ним атаку, основанную на редукции: один раз Саймон посылает алгоритму $\mathcal{A}$ число c^*, а второй раз — число $\bar{c}^* = c^*\alpha^e \pmod N$, где α — случайное число, принадлежащее группе $\mathbb{Z}_N^*$. Соответствующие числа s^* и $\bar{s}^*$ будут записаны в H- и $\bar{H}$-списке соответственно. Пусть $q = max(\#(H\text{-список}), \#(\bar{H}\text{-список}))$. Работая с этими списками, Саймон будет хранить не более q^2 пар $(t, \bar{t})$. Одна из этих пар позволит Саймону решить два корректных уравнения, заданных формулой (15.2.21), и таким образом инвертировать функцию RSA, если число α было выбрано правильно (вероятность противоположного события пренебрежимо мала, так как в схеме RSA-OAEP $k \gg 2k_0$). Поскольку решить два уравнения, заданных формулой (15.2.21), можно за $O_B((\log_2 N)^3)$ единиц времени, оценка времени, необходимого для инвертирования функции RSA, имеет следующий вид.

$$2\tau + q^2 \times O_B((\log_2 N)^3) \tag{15.2.22}$$

Здесь число τ — время, необходимое атакующему $\mathcal{A}$ для проведения атаки IND-CCA2 на схему RSA-OAEP.

Схема RSA-OAEP имеет еще два варианта дополнения параметра безопасности: PKCS#1 версии 2 и выше [230] и SET [259]. В этих вариантах известная порция данных s^* размещается в разных местах исходного текста. Благодаря достаточно большому размеру числа s^* (значительно превышающему половину размера блока), для извлечения корня достаточно не более двух раз вызвать атакующего. Итак, метод Фуджисаки распространяется и на эти два варианта.

Следовательно, схема RSA-OAEP и ее варианты являются доказуемо стойкими к атаке IND-CCA2.

В заключение отметим, что тот же результат применим и к схеме Рабин–OAEP. Следовательно, инверсия функции Рабина, т.е. извлечение квадратного корня по модулю N в произвольной точке, приводит к разложению модуля N на простые множители. Саймон может это сделать с помощью теоремы 6.17.3 (раздел 6.6.2), извлекая случайное число x и считая число c^* результатом шифрования числа x с помощью функции шифрования Рабина. Схема Рабин–OAEP является более стойкой, чем схема RSA-OAEP, поскольку факторизация является более слабым условием, чем предположение, лежащее в основе схемы RSA.

15.2.5 Неэффективность метода сведения к противоречию для схемы RSA-OAEP

Схема RSA-OAEP весьма эффективна. Однако этого нельзя сказать о процессе "сведения к противоречию".

Выражение (15.2.22) означает время, необходимое Саймону-имитатору для двойного применения атакующего алгоритма $\mathcal{A}$, чтобы инвертировать функцию RSA в произвольной точке. В это выражение входит квадратный член q^2, где число q представляет собой количество запросов, которые атакующий алгоритм $\mathcal{A}$ должен послать случайному оракулу H.

Обратите внимание на то, что случайный оракул представляет собой идеализацию функции хэширования, допускающей очень эффективное вычисление. Таким образом, атакующий должен выполнить, скажем, 2^{50} вычислений функции хэширования. Итак, можно считать, что $q \approx 2^{50}$. Следовательно, квадратичный член q^2 в формуле (15.2.22) означает, что время, которое Саймон должен затратить для решения задачи RSA, имеет порядок

$$2^{100} \cdot O_B((\log_2 N)^3).$$

Как показано в разделе 4.6, время, необходимое для факторизации целого числа с помощью решетки числового поля (Number Field Sieve — NFS), выражается формулой (4.6.1). При обычном размере числа $|N| = 1024$ значение, вычисляемое по формуле (4.6.1), равно приблизительно 2^{86}. Таким образом, если для факторизации применяется решетка числового поля, противоречие, выражаемое оценкой $2^{100} \cdot O_B((\log_2 N)^3)$, теряет смысл. Саймон может инвертировать функцию RSA, основанную на 1024-битовом модуле, намного быстрее, не прибегая к помощи атакующего алгоритма $\mathcal{A}$. Таким образом, доказательство, основанное на "сведении к противоречию" для 1024-битового модуля RSA, становится некорректным.

Поскольку указанный модуль считается безопасным для применения во многих прикладных системах, некорректность доказательства стойкости схемы RSA-OAEP свидетельствует о недостаточно высокой степени полинома.

Доказательство, основанное на "сведении к противоречию", становится корректным, если модуль состоит из 2048 бит. В этом случае формула (4.6.1) имеет порядок 2^{116}.

15.2.6 Критика модели случайного оракула

Канетти (Canetti), Голдрайх (Goldreich) и Халеви (Halevi) высказали негативное отношение к доказательствам, основанным на применении модели случайного оракула [64, 65]. Они продемонстрировали, что существуют схемы цифровой подписи и шифрования, доказуемо стойкие в рамках модели случайного оракула, но

уязвимые при реализации в реальных приложениях. Их основная идея заключалась в изобретении плохих схем цифровой подписи или шифрования. В обычных условиях эти схемы работают хорошо, но при некоторых условиях (в основном, при нарушении случайности) становятся плохими.

Очевидно, что доказательство стойкости плохих схем в рамках модели случайного оракула остается корректным, поскольку исходный текст, подлежащий подписанию или шифрованию, является равномерно распределенным. Однако в реальных приложениях равномерное распределение остается недостижимой целью, и, следовательно, реальные приложения являются уязвимыми.

Способ, которым авторы строили свои схемы, довольно сложен. Заинтересованные читатели найдут его описание в работе [65].

Однако, рассмотрев элегантные и убедительные научные аргументы, интересно обнаружить, что эти три автора пришли к разным выводам относительно полезности модели случайного оракула. Они решили не скрывать своих разногласий и сформулировали свои выводы по отдельности.

Канетти (раздел 6.1 в работе [65]) придерживается наиболее скептичной точки зрения. Он считает, что модель случайного оракула представляет собой неудачную абстракцию и снижает ценность метода “сведения к противоречию”. Канетти предложил в качестве альтернативы направить научные исследования на поиск специфических полезных свойств модели случайного оракула.

Вывод Голдрайха (раздел 6.2 в работе [65]) является умеренно пессимистичным. Он считает, что проблема заключается в неполноте модели случайного оракула и рекомендует не включать в работы доказательства, использующие этот метод. Иначе говоря, такие доказательства нельзя считать корректными. Однако окончательный вывод Голдрайха довольно оптимистичен: он полагает, что модель случайного оракула имеет определенную ценность для проверки криптосистем на прочность. Голдрайх выразил надежду, что в будущем эта модель будет усовершенствована.

Вывод Халеви (раздел 6.3 в работе [65]) основан на использовании события, имеющего значимую вероятность. Он полагает, что нынешние успехи метода сведения к противоречию являются случайными: “нет никаких оснований утверждать, что все существующие схемы, стойкость которых доказана с помощью модели случайного оракула, в действительности являются стойкими”. Халеви считает, что лучше иметь доказательства стойкости современных стандартных схем, полученные с помощью модели случайного оракула, чем не иметь никакого доказательства.

15.2.7 Авторская точка зрения на ценность модели случайного оракула

Автор придерживается собственного мнения на этот счет. Для того чтобы сохранить объективность при обсуждении результатов, изложенных в главе, рассмотрим аргументы, касающиеся схемы RSA-OAEP.

Доказательство стойкости модели RSA-OAEP в рамках модели случайного оракула основано на следующем факте.

> Если схема заполнения является истинно случайной функцией, то результатом заполнения с помощью схемы OAEP является "идеальный исходный текст": он имеет равномерное распределение в пространстве исходных текстов функции RSA. Таким образом, исследование стойкости функции RSA, используемой в идеальном мире (см. раздел 9.2), показывает, что легче всего преодолеть защиту от атаки IND-CCA2, решив задачу RSA и применив алгоритм расшифровки.

Итак, доказательство, основанное на модели случайного оракула, предполагает, что схема шифрования путем заполнения использует идеальную функцию хэширования, а не случайного оракула, причем наиболее уязвимым местом для проведения атаки является именно функция хэширования. Для того чтобы достичь высокой стойкости схемы шифрования путем заполнения, необходимо уделить большое внимание разработке функции хэширования и обеспечению случайности входных данных.

С этой точки зрения модель случайного оракула имеет большое значение, поскольку она фокусирует внимание именно на этих проблемах.

15.3 Криптосистема с открытым ключом Крамера–Шоупа

Другой широко известной эффективной криптосистемой с открытым ключом, стойкой в атаке IND-CCA2, является **криптосистема Крамера–Шоупа** (Cramer–Shoup) [84].

15.3.1 Доказуемая стойкость при стандартных предположениях о неразрешимости задачи

Выше рассмотрена общая методология формального доказательства стойкости, заключающегося в "сведении" атаки на криптографическую систему к решению общепризнанно трудноразрешимой задачи (т.е. атакующий алгоритм изображается черным ящиком, решающим трудную задачу). Это доказательство, основанное на "сведении к противоречию" обладает двумя полезными свойствами.

Свойство 15.1. Свойства доказательства, основанного на "сведении к противоречию".

1. Редукция должна быть эффективной. В идеале удачливый взломщик, успешно организующий атаку, должен решить трудноразрешимую задачу, лежащую в основе криптографической схемы.
2. Условия неразрешимости, обеспечивающие стойкость криптографической схемы, должны быть как можно более слабыми. В идеале для стойкости схемы шифрования с открытым ключом, основанной на применении однонаправленной функции с секретом (OWTF), единственным условием должна быть невозможность инвертировать эту функцию (помните, что функции OWTF являются частным случаем схемы OWTP).

Свойство 15.1.1 имеет практическое значение: неэффективная редукция, даже за полиномиальное время, может уничтожить связь между атакой и решением трудноразрешимой задачи. Например, если редукция связана с полиномом степени 8, а параметр безопасности равен 1024, то временна́я сложность редукции имеет порядок $1024^8 = 2^{80}$. Если предположить, что существует успешная атака, взламывающая систему за 10^{-6} с, то на применение атакующего алгоритма для решения трудноразрешимой задачи понадобится 38 миллиардов лет! Такое доказательство стойкости не только бессмысленно, но и *противоречит* определению математического доказательства: известные методы решения трудноразрешимых задач могут оказаться намного эффективнее, чем редукция! Фактически, как показано в разделе 15.2.5, даже если сложность редукции ограничена полиномом второй степени, при стандартном размере параметра безопасности доказательство становится некорректным.

Может показаться, что желательное свойство 15.1.2 имеет меньшее практическое значение, поскольку оно выглядит слишком общим: если слабые предположения не ограничивают ход доказательства, то доказательство следует строить исключительно на них. Поскольку доказательство стойкости является довольно важной частью исследования, свойство 15.1.2 имеет большое практическое значение. Его важность особенно ярко проявляется при разработке криптографических систем: более слабые предположения легче удовлетворить, используя практичные и доступные криптографические конструкции. Следовательно, криптографические системы, основанные на слабых предположениях, имеют более высокий уровень стойкости, чем криптосистемы, основанные на более сильных предположениях.

Как показано выше, доказательство стойкости схемы RSA-OAEP, основанное на модели случайного оракула, не полностью удовлетворяет свойству 15.1.1, поскольку редукция становится некорректной уже при стандартном размере модуля RSA. Более того, это доказательство недостаточно удовлетворяет и свойству 15.1.2, поскольку кроме условия, касающегося функции RSA (предположе-

ние 8.3), для него необходимо выполнение намного более строгого ограничения: функция хэширования, применяемая в схеме OAEP, должна обладать свойствами случайного оракула. Это чрезмерно сильное предположение: как показано в разделе 10.3.1.2, в природе не существует случайного оракула. Следовательно, это предположение неразумно. Действительно, с точки зрения практики из доказательства стойкости схемы RSA-OAEP следует, что при конструировании этой схемы мы должны применять функцию хэширования очень высокого качества. К сожалению, это условие не гарантирует нам уверенности, которую дает математическое доказательство стойкости.

Формальное доказательство стойкости криптосистемы с открытым ключом, основанное исключительно на неразрешимости задачи OWTP, называется доказательством, основанным на **стандартных предположениях о неразрешимости** (standard intractability assumption(s)). Такое доказательство устанавливает реальную стойкость криптосистемы: оно гарантирует, что криптосистему невозможно взломать, не нарушив предположений о неразрешимости задачи, лежащей в ее основе.

Существует большое количество доказуемо стойких к атаке IND-CCA2 криптосистем, основанных на стандартных предположениях о неразрешимости. Примером такой системы является схема Долева [100], стойкая к атаке NM-CCA2 (эквивалент атаки IND-CCA2). Однако, как показано в разделе 14.5.3, необходимость применять в схеме Долева доказательство NIZK делает эту криптосистему непригодной для практического применения.

Криптосистема с открытым ключом Крамера–Шоупа [84] является первой криптосистемой с открытым ключом, обладающей практической эффективностью и доказанной стойкостью к атаке IND-CCA2 при стандартных предположениях о неразрешимости. В дальнейшем будет показано, что эта схема имеет строгое доказательство стойкости, основанное на “сведении к противоречию”, которое называется **линейной редукцией** (linear reduction). Итак, схема шифрования с открытым ключом Шоупа–Крамера идеально удовлетворяет условиям 15.1.

15.3.2 Схема Крамера–Шоупа

Схема шифрования с открытым ключом Крамера–Шоупа представляет собой вариант семантически стойкой схемы шифрования Эль-Гамаля (см. раздел 14.3.5), устойчивый к атаке CCA2. Как и в семантически стойкой схеме шифрования Эль-Гамаля, стандартное предположение о неразрешимости, гарантирующее стойкость схемы Крамера–Шоупа, относится к задаче принятия решений Диффи–Хеллмана (задаче DDH).

Криптосистема Крамера–Шоупа описывается алгоритмом 15.3.2.1.

Легко видеть, что часть зашифрованного текста (u_1, e) представляет собой зашифрованную пару из семантически стойкой криптосистемы Эль-Гамаля. Из

Алгоритм 15.1. Криптосистема с открытым ключом Крамера–Шоупа

Параметры ключа

Пусть G — абелева группа, имеющая большой простой порядок q. Группа G является пространством исходных текстов.

(* Предполагается, что существует схема шифрования, позволяющая зашифровать любой исходный текст в виде битовой строки и расшифровать его вновь. При заданном описании группы $\mathrm{desc}(G)$ такое шифрование легко реализовать, используя методы, описанные в разделе 14.3.5. *)

Чтобы установить компоненты ключа, Алиса должна выполнить следующие инструкции.

1. Выбрать два случайных простых числа $g_1, g_2 \in_U G$.
2. Выбрать пять случайных целых чисел $x_1, x_2, y_1, y_2, z \in_U [0, q)$.
3. Вычислить значения $c \leftarrow g_1^{x_1} g_2^{x_2}, d \leftarrow g_1^{y_1} g_2^{y_2}, h \leftarrow g_1^z$.
4. Выбрать криптографическую функцию хэширования $H : G^3 \mapsto [0, q)$.
5. Огласить кортеж (g_1, g_2, c, d, h, H) в качестве параметров открытого ключа, а кортеж (x_1, x_2, y_1, y_2, z) сохранить как закрытый ключ.

Шифрование

Чтобы послать Алисе строку $m \in G$, Боб извлекает случайное число $r \in_U [0, q)$ и выполняет следующие вычисления.

$$u_1 \leftarrow g_1^r, \quad u_2 \leftarrow g_2^r, \quad e \leftarrow h^r m, \quad \alpha \leftarrow H(u_1, u_2, e), \quad v \leftarrow c^r d^{r\alpha}.$$

Зашифрованным текстом является кортеж (u_1, u_2, e, v).

Расшифровка

Чтобы расшифровать кортеж (u_1, u_2, e, v), Алиса должна выполнить следующие операции.

1. $\alpha \leftarrow H(u_1, u_2, e)$.
2. Вывести $\begin{cases} m \leftarrow \dfrac{e}{u_1^z}, & \text{если } u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v, \\ \text{ОТКАЗ} & \text{в противном случае.} \end{cases}$

теоремы 14.2 (раздел 14.3.5) следует, что схема Эль-Гамаля является стойкой к атаке IND-CPA при условии неразрешимости задачи DDH.

Как и любая другая схема шифрования, стойкая к атаке CCA2, процедура расшифровки предусматривает проверку целостности данных. Допустим, что на пути к Алисе зашифрованный текст не подвергался изменениям. Тогда

$$u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = (u_1^{x_1} u_2^{x_2})(u_1^{y_1\alpha} u_2^{y_2\alpha}) = (g_1^{rx_1} g_2^{rx_2})(g_1^{ry_1\alpha} g_2^{ry_2\alpha}) = c^r d^{r\alpha} = v.$$

После проверки целостности данных процедура расшифровки полностью совпадает с семантически стойкой криптосистемой Эль-Гамаля. В дальнейшем будет показано, что процедура проверки целостности данных является очень эффективной: она предотвращает возможность создания корректного зашифрованного текста без применения соответствующей процедуры шифрования.

Может возникнуть следующий вопрос.

> "Почему стойкость этой схемы базируется исключительно на предположении о неразрешимости задачи DDH? Поскольку процедура проверки целостности данных использует функцию хэширования H, почему стойкость криптосистемы не зависит от ее свойств, например, от свойств случайного оракула?"

Разумеется, функция хэширования, применяемая в этой схеме, не должна быть слабой. Однако следует отметить, что защита, предусмотренная на этапе проверки целостности данных, основана исключительно на свойстве однонаправленности функции хэширования. Таким образом, необходимость использовать свойства случайного оракула отпадает. Например, функцию хэширования $H(x)$ можно реализовать с помощью операции g^x, выполненной в той же самой группе G. В этом случае будет использовано только свойство однонаправленности дискретного логарифма (задача DL, см. определение 8.2 в разделе 8.4). Связанное с этим предположение о неразрешимости задачи DL, является стандартным. Кроме того, оно слабее предположения о неразрешимости задачи DDH в этой же группе. Иначе говоря, если бы мы использовали предположение о неразрешимости задачи DDH в группе G, то предположение о неразрешимости задачи DL в группе G выполнялось бы автоматически. Именно по этой причине мы утверждаем, что стойкость схемы основана исключительно на предположении о неразрешимости задачи DDH. В противоположность этому анализ атаки Шоупа на схему f-OAEP (раздел 15.2.3.3) показывает, что доказательство стойкости этой схемы не может быть основано только на свойстве однонаправленности функции хэширования или предположении о неразрешимости соответствующей задачи.

15.3.2.1 Производительность

На первый взгляд, криптосистема Крамера–Шоупа использует более длинные ключи и больше операций возведения в степень, чем криптосистема Эль-Гамаля. Однако более глубокий анализ показывает, что разница между этими криптосистемами не так велика, как кажется.

Открытый ключ в схеме Крамера–Шоупа состоит из пяти элементов, принадлежащих группе G, полученных путем добавления дополнительных компонентов к открытому ключу схемы Эль-Гамаля. Зашифрованный текст представляет собой четверку элементов из группы G, т.е. вдвое больше, чем в схеме Эль-Гамаля. Для шифрования необходимо выполнить "пять" (на самом деле четыре) операций

возведения в степень, в то время как в схеме Эль-Гамаля — только две. Для расшифровки необходимо выполнить "три" (на самом деле две) операции возведения в степень, в то время как в схеме Эль-Гамаля — только одну.

Поясним, почему для шифрования (расшифровки) достаточно выполнить четыре (две) операции возведения в степень, а не пять (три), как ясно указано в схеме. Это возможно потому, что произведение двух степеней в формуле $g^x h^y$ можно вычислить с помощью одного возведения в степень. Этот метод описывается алгоритмом 15.2. Легко видеть, что этот алгоритм завершает работу, $\max(|x|, |y|)$ раз выполнив операцию "возведения в квадрат с последующим умножением". Обратите внимание на то, что ни этот алгоритм, ни описание схемы Крамера–Шоупа, не содержат групповых операций. Действительно, группой G может быть любая абелева группа, в которой выполняется предположение о неразрешимости задачи DDH.

Оценка производительности криптосистемы Крамера–Шоупа демонстрирует, что как по пропускной способности каналов, так и по эффективности вычислений, криптосистема Крамера–Шоупа вдвое превышает показатели криптосистемы Эль-Гамаля.

15.3.3 Доказательство стойкости

Читатели, которых интересует лишь описание криптосистемы Крамера–Шоупа, могут ограничиться изучением алгоритма 15.3.2.1 и перейти к разделу 15.4. Те же, кого интересует интуитивное доказательство стойкости этой криптосистемы, могут продолжать чтение раздела.

Для того чтобы разобраться в доказательстве стойкости криптосистемы Крамера–Шоупа, не требуется никаких специальных математических знаний. Достаточно знать основные сведения из теории групп, содержащиеся в разделе 5.2, и элементарные факты из линейной алгебры.

Доказательство стойкости криптосистемы Крамера–Шоупа основано на "сведении к противоречию": т.е. на редукции атаки IND-CCA2 к решению трудноразрешимой задачи. В частности, стойкость криптосистемы Крамера–Шоупа основана на неразрешимости следующей задачи.

> Пусть G — группа, имеющая простой порядок q, а $(g_1, g_2, u_1, u_2) \in \in G^4$ — произвольная четверка, в которой $g_1 \neq 1$ и $g_2 \neq 1$. Вопрос состоит в следующем: является ли кортеж (g_1, g_2, u_1, u_2) четверкой Диффи–Хеллмана? Иначе говоря, существуют ли числа $a, b \in [0, q)$, удовлетворяющие условию
>
> $$g_2 = g_1^a, u_1 = g_1^b, u_2 = g_1^{ab}? \qquad (15.3.1)$$

Поскольку группа G имеет простой порядок, то число $g_1 \neq 1$ является ее порождающим элементом (следствие 5.3). Следовательно, всегда существуют целые

Алгоритм 15.2. Произведение степеней

ИСХОДНЫЕ ДАННЫЕ:

$g, h \in A$, где A — алгебраическая структура;
x, y: целые числа из интервала $(0, \#A)$;
$\mathrm{Exp}(u, z)$: операция возведения в степень, возвращающая число u^z
(* см. алгоритм 4.3*)

РЕЗУЛЬТАТ:

$g^x h^y$

1. if $(|x| > |y|)$
   {
   $u \leftarrow \mathrm{Exp}(g, x(\mathrm{mod}\, 2^{|x|-|y|}))$;
   (* Операция возведения в степень использует $|x| - |y|$ младших битов числа x. *)
   $x \leftarrow x \div 2^{|x|-|y|}$; (* Символ $\div$ обозначает целочисленное деление. Эта операция отбрасывает $|x| - |y|$ младших битов числа x. Следовательно, теперь $|x| = |y|$. *)
   }
2. if $(|y| > |x|)$
   {
   $u \leftarrow \mathrm{Exp}(h, x(\mathrm{mod}\, 2^{|y|-|x|}))$;
   $y \leftarrow y \div 2^{|y|-|x|}$;
   }
3. $v \leftarrow gh$; (* Теперь $|x| = |y|$. *)
4. while $(x \neq 0)$ do
   {
   а) $u \leftarrow u^2$;
   б) if $(x(\mathrm{mod}\, 2) == 1 \wedge y(\mathrm{mod}\, 2) == 1) u \leftarrow uv$;
   в) if $(x(\mathrm{mod}\, 2) == 1 \wedge y(\mathrm{mod}\, 2) == 0) u \leftarrow ug$;
   г) if $(x(\mathrm{mod}\, 2) == 0 \wedge y(\mathrm{mod}\, 2) == 1) u \leftarrow uh$;
   д) $x \leftarrow x \div 2; y \leftarrow y \div 2$; (* Младший бит отбрасывается. *)
   }
5. return (u);
   (* Общее количество операций "возведения в квадрат с последующим умножением" равно $\max(|x|, |y|)$. *)

числа $a, b \in [0, q)$, удовлетворяющее двум первым уравнениям в системе (15.3.1). Вот почему знак вопроса относится только к последнему уравнению. Достаточно просто проверить, что система (15.3.1) эквивалентна уравнению

$$\log_{g_1} u_1 = \log_{g_2} u_2 (\mathrm{mod}\, q).$$

Из предположения Диффи–Хеллмана (предположение 14.2) следует, что в общем случае для абелевой группы эта задача является трудноразрешимой.

15.3.3.1 Схематичное описание доказательства стойкости

Предположим, что существует атакующий алгоритм $\mathcal{A}$, способный взломать криптосистему Крамера–Шоупа с помощью атаки IND-CCA2 со значимым преимуществом. Мы построим эффективный алгоритм редукции, прибегнув к помощи специального агента Саймона-имитатора, способного решить трудноразрешимую задачу Диффи–Хеллмана.

Саймон получает *произвольную* четверку $(g_1, g_2, u_1, u_2) \in G^4$, где $g_1 \neq 1$ и $g_2 \neq 1$. Этот кортеж может быть четверкой Диффи–Хеллмана (обозначим этот случай как $(g_1, g_2, u_1, u_2) \in \mathbf{D}$, а может и не быть ею (обозначим этот случай как $(g_1, g_2, u_1, u_2) \notin \mathbf{D}$).

Используя входную информацию, Саймон может создать для атакующего алгоритма $\mathcal{A}$ открытый ключ $PK = (g_1, g_2, c, d, h, H)$. Кроме того, в ходе атаки IND-CCA2 Саймон может по запросу атакующего алгоритма $\mathcal{A}$ конструировать зашифрованный оклик $C^* = (u_1, u_2, e, v)$, шифрующий подобранный исходный текст $m_b \in_U \{m_0, m_1\}$. (Здесь m_0 и m_1 — исходные тексты, подобранные атакующим алгоритмом $\mathcal{A}$, однако бит b атакующему не известен.)

Зашифрованный оклик C^* обладает следующими свойствами.

1. Если $(g_1, g_2, u_1, u_2) \in \mathbf{D}$, то оклик C^* является корректным зашифрованным текстом Крамера–Шоупа, шифрующим сообщение m_b с помощью открытого ключа PK. Доказательство корректности отклика C^* содержится в разделах 15.2.3.3 и 15.3.3.4. Независимо от того, применяется ли открытый ключ или нет, атакующий $\mathcal{A}$ проходит курс обучения криптоанализу, точно имитируемый Саймоном. Доказательство этого факта приведено в разделе 15.3.3.5. Итак, Саймон просит атакующего $\mathcal{A}$ напрячь все свои силы.
2. Если $(g_1, g_2, u_1, u_2) \notin \mathbf{D}$, то оклик C^* шифрует текст m_b в теоретико-информационном смысле по Шеннону (т.е. с помощью идеального шифрования (см. раздел 7.5)). Иначе говоря, зашифрованный текст равномерно распределяется по всему пространству зашифрованных текстов. Идеальное шифрование по Шеннону рассматривается в разделе 15.3.3.5. Более того, в разделе 15.3.3.5 будет показано, что качество идеального шифрования по Шеннону невозможно скомпрометировать с помощью обучения атакующего алгоритма. Итак, в этом случае атакующий алгоритм $\mathcal{A}$ не имеет никакого преимущества!

Именно разница между преимуществом атакующего алгоритма $\mathcal{A}$ в указанных двух случаях позволяет ему помочь Саймону решить задачу принятия решений Диффи–Хеллмана.

Перейдем к алгоритму редукции.

15.3.3.2 Редукция

Алгоритм редукции состоит из следующих этапов.

1. Получив четверку $(g_1, g_2, u_1, u_2) \in G^4$, Саймон создает открытый ключ криптосистемы Крамера–Шоупа и посылает его атакующему $\mathcal{A}$. Метод создания открытого ключа описан в разделе 15.3.3.3.
2. Саймон проводит для атакующего $\mathcal{A}$ курс обучения криптоанализу, который необходимо пройти до получения оклика. Метод, с помощью которого Саймон имитирует процедуру расшифровки, выполняемую оракулом $\mathcal{O}$, описан в разделе 15.3.3.5.
3. Саймон получает от атакующего алгоритма $\mathcal{A}$ пару подобранных открытых текстов m_0 и m_1, подбрасывает идеальную монету $b \in_U \{0, 1\}$, шифрует текст m_b, необходимый для создания зашифрованного оклика C^*, и посылает этот оклик атакующему $\mathcal{A}$. Метод, с помощью которого Саймон имитирует процедуру шифрования, выполняемую оракулом $\mathcal{O}$, описан в разделе 15.3.3.5.
4. Имитируя процедуру расшифровки, выполняемую оракулом $\mathcal{O}$, Саймон продолжает курс обучения атакующего $\mathcal{A}$, который необходимо пройти после получения оклика. Метод, с помощью которого Саймон имитирует процедуру расшифровки, выполняемую оракулом $\mathcal{O}$, описан в разделе 15.3.3.5.
5. В заключение Саймон получает от атакующего алгоритма $\mathcal{A}$ результат осмысленного угадывания бита b. Теперь Саймон способен ответить на вопрос: $(g_1, g_2, u_1, u_2) \in \mathbf{D}$ или $(g_1, g_2, u_1, u_2) \notin \mathbf{D}$?

Процесс редукции продемонстрирован на рис. 15.3.3.3. Он представляет собой игровую атаку IND-CCA2, разыгранную между Саймоном и атакующим алгоритмом $\mathcal{A}$. Саймон контролирует все линии связи алгоритма $\mathcal{A}$ с внешним миром, поэтому атакующий может общаться только с Саймоном. Со стороны Саймона игровая атака является имитацией. Однако, как будет показано в дальнейшем, эта имитация имеет идеальное качество, и, следовательно, атакующий алгоритм $\mathcal{A}$ не может отличить ее от реальной атаки.

15.3.3.3 Создание открытого ключа

Используя четверку $(g_1, g_2, u_1, u_2) \in G^4$, Саймон создает открытый ключ. Для этого он генерирует случайные числа

$$x_1, x_2, y_1, y_2, z_1, z_2 \in_U [0, q)$$

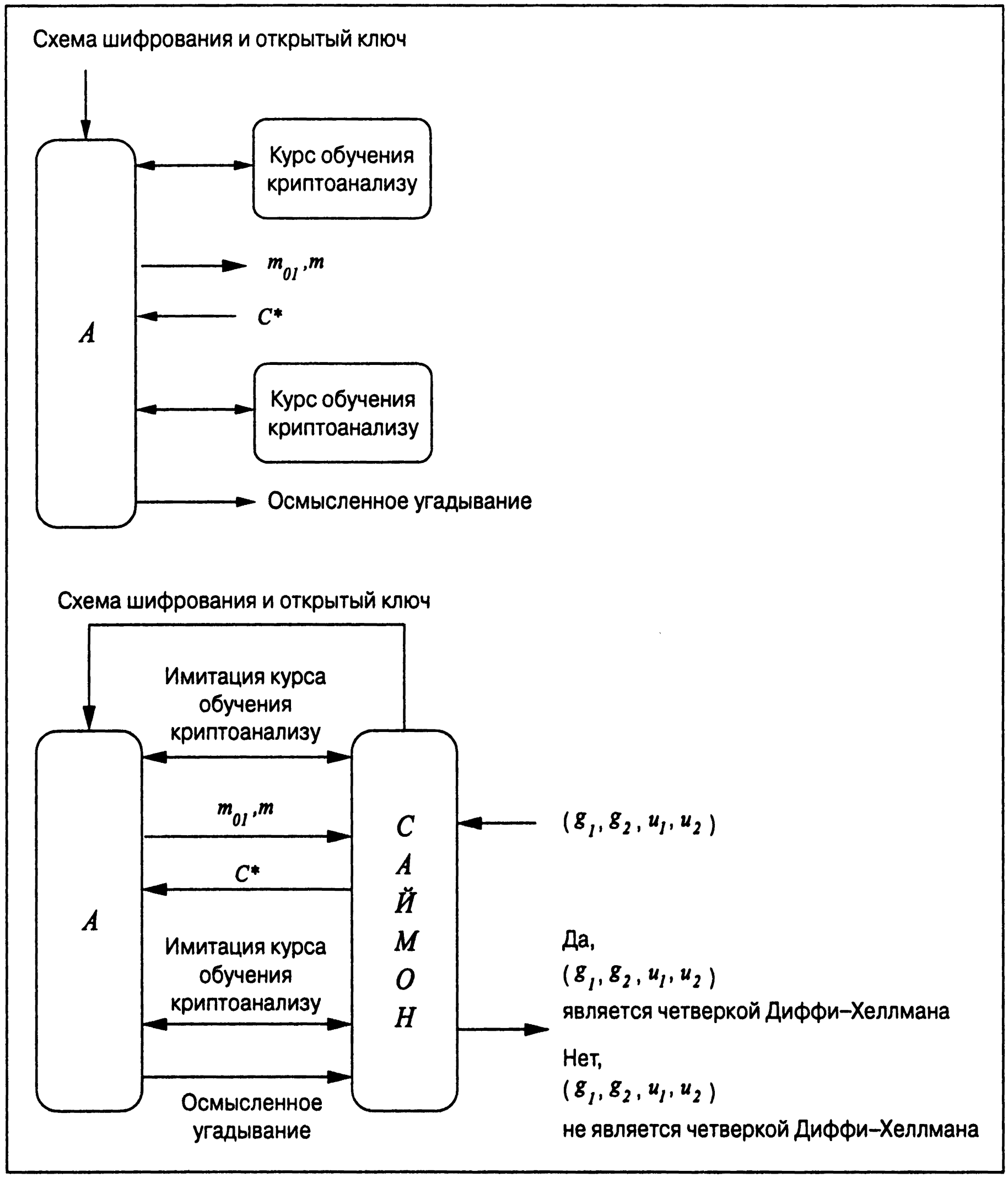


Рис. 15.4. Редукция задачи DDH к атаке на криптосистему Крамера–Шоупа

и вычисляет значения

$$c \leftarrow g_1^{x_1} g_2^{x_2}, \quad d \leftarrow g_1^{y_1} g_2^{y_2}, \quad h \leftarrow g_1^{z_1} g_2^{z_2}. \tag{15.3.2}$$

Кроме того, Саймон выбирает криптографическую функцию хэширования H. Открытый ключ, предназначенный для атакующего алгоритма $\mathcal{A}$, имеет вид

$$(g_1, g_2, c, d, h, H).$$

Закрытый ключ, который использует Саймон, имеет вид

$$(x_1, x_2, y_1, y_2, z_1, z_2).$$

Читатель мог уже заметить, что часть открытого ключа, а именно, компонент h, отличается от компонента, указанного в алгоритме 15.3.2.1. Однако это не проблема. Сначала убедимся, что число h является абсолютно корректным компонентом открытого ключа.

Заметим, что в выражении $h = g_1^{z_1} g_2^{z_2}$, где $g_1 \neq 1$, элемент g_1 является порождающим элементом группы G (следствие 5.3). Таким образом, существует число $w \in [0, q)$, такое что $g_2 = g_1^w$. Следовательно, для $z \equiv z_1 + wz_2 (\mathrm{mod}\, q)$

$$h = g_1^{z_1 + wz_2} = g_1^z. \tag{15.3.3}$$

Итак, действительно, число h полностью соответствует процедуре создания ключа в алгоритме 15.3.2.1.

Читатель может задать вопрос:

> "Как Саймон может использовать число $z \equiv z_1 + wz_2 (\mathrm{mod}\, q)$ в процедуре расшифровки, если ему неизвестно число $w = \log_{g_1} g_2 (\mathrm{mod}\, q)$?".

В главе 15.3.3.5 будет показано, что для любого корректно зашифрованного текста, соответствующего указанному открытому ключу, Саймон может применять число $z \equiv z_1 + wz_2 (\mathrm{mod}\, q)$ для расшифровки текста как "правильный" показатель степени, даже не зная его.

15.3.3.4 Имитация процедуры шифрования

Получив от атакующего алгоритма $\mathcal{A}$ два подобранных открытых сообщения m_0 и m_1, Саймон подбрасывает идеальную монету $b \in_U \{0, 1\}$ и шифрует текст m_b:

$$e = u_1^{z_1} u_2^{z_2} m_b, \quad \alpha = H(u_1, u_2, e), \quad v = u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}.$$

Зашифрованным окликом C^* является четверка (u_1, u_2, e, v).

Отметим, что эта процедура шифрования также отличается от обычной, в которой значение e вычисляется по формуле

$$e = h^r m_b,$$

где $r \in [0, q)$.

Однако этот факт не вызывает никаких трудностей. В то же время это чрезвычайно важно для доказательства стойкости. Попробуем разобраться в этой загадке.

1. $(g_1, g_2, u_1, u_2) \in \mathbf{D}$. Тогда, поскольку существует число $r \in [0, q)$, удовлетворяющее условию $u_1 = g_1^r, u_2 = g_2^r$, выполняется равенство

$$u_1^{z_1} u_2^{z_2} = (g_1^r)^{z_1} (g_2^r)^{z_2} = (g_1^{z_1} g_2^{z_2})^r = h^r.$$

Эта имитация шифрования точно соответствует схеме шифрования Крамера–Шоупа с заданным открытым ключом. Это именно то, к чему мы стремились, заставляя атакующий алгоритм $\mathcal{A}$ проявить всю свою силу.

2. $(g_1, g_2, u_1, u_2) \notin \mathbf{D}$. В этом случае существуют целые числа $r_1, r_2 \in [0, q)$, удовлетворяющие условию $r_1 \neq r_2 (\mathrm{mod}\, q)$, $u_1 = g_1^{r_1}$ и $u_2 = g_2^{r_2}$. Поскольку число $g_1 \neq 1$ является порождающим элементом группы G, существуют числа $\log_{g_1} g_2$, $\log_{g_1} h$, $\log_{g_1}\left(\frac{e}{m_0}\right)$ и $\log_{g_1}\left(\frac{e}{m_1}\right)$.

   Чтобы упростить изложение, предположим, что атакующий алгоритм $\mathcal{A}$ имеет неограниченную вычислительную мощность. Получив число e, содержащееся в зашифрованном оклике C^*, атакующий алгоритм $\mathcal{A}$, имеющий неограниченную вычислительную мощность, должен решить систему, состоящую из двух линейных уравнений относительно неизвестных целых чисел (z_1, z_2).

$$\begin{pmatrix} 1 & \log_{g_1} g_2 \\ r_1 & r_2 \log_{g_1} g_2 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} h \\ \log_{g_1}\left(\frac{e}{m_0}\right) \end{pmatrix} (\mathrm{mod}\, q), \tag{15.3.4}$$

$$\begin{pmatrix} 1 & \log_{g_1} g_2 \\ r_1 & r_2 \log_{g_1} g_2 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} h \\ \log_{g_1}\left(\frac{e}{m_1}\right) \end{pmatrix} (\mathrm{mod}\, q). \tag{15.3.5}$$

   Поскольку $(g_1, g_2, u_1, u_2) \notin \mathbf{D}$, выполняются условия $r_1 \neq r_2 (\mathrm{mod}\, q)$. Кроме того, заметим, что $\log_{g_1} g_2 \neq 0 (\mathrm{mod}\, q)$, так как $g_2 \neq 1$. Итак, матрицы систем (15.3.4), (15.3.5) имеют полный ранг, равный двум, поэтому обе системы имеют единственное решение (z_1, z_2). Атакующий алгоритм $\mathcal{A}$ не может определить, какой из этих вариантов является правильным. Следовательно, даже обладая неограниченными вычислительными возможностями, атакующий алгоритм $\mathcal{A}$ не имеет ни малейшего представления, какое сообщение зашифровано в отклике $C^* - m_0$ или m_1. Итак, в данном случае текст C^* шифрует сообщение m_b в теоретико-информационном смысле по Шеннону. Следовательно, атакующий алгоритм $\mathcal{A}$ не имеет никакого преимущества!

Следует отметить, что сказанное выше относится лишь к атаке CPA. Иначе говоря, предполагается, что атакующий алгоритм $\mathcal{A}$ является пассивным. Реальные взломщики так себя не ведут! Напомним, что атакующий алгоритм $\mathcal{A}$ прошел курс обучения криптоанализа после получения зашифрованного оклика. Например, если во втором варианте атакующий алгоритм $\mathcal{A}$, кроме систем (15.3.4) и (15.3.5), должен решить третью систему, возникающую в результате курса обучения криптоанализу в рамках атаки CCA2, схема Крамера–Шоупа перестанет быть стойкой в теоретико-информационном смысле по Шеннону.

В следующем разделе будет показано, как достигается точность имитации шифрования в ходе курса обучения криптоанализу во время атаки IND-CCA2.

15.3.3.5 Имитация процедуры расшифровывания

Получив от атакующего алгоритма $\mathcal{A}$ зашифрованный текст $C = (U_1, U_2, E, V)$, Саймон сначала должен выполнить процедуру проверки целостности данных, описанную в алгоритме 15.3.2.1. Если проверка завершилась успешно, зашифрованный текст считается корректным. Тогда Саймон вычисляет значение

$$m = \frac{E}{U_1^{z_1} U_2^{z_2}}, \tag{15.3.6}$$

и возвращает его атакующему алгоритму $\mathcal{A}$ как результат расшифровки. Если проверка завершается неудачей, зашифрованный текст считается некорректным, и Саймон должен вернуть отказ.

В свое время мы докажем теорему 15.1, утверждающую, что если зашифрованный текст $C = (U_1, U_2, E, V)$ является корректным, то $(g_1, g_2, u_1, u_2) \in \mathbf{D}$ с вероятностью $1 - \frac{1}{q}$. Следовательно, существует число $R \in [0, q)$, удовлетворяющее условию $U_1 = g_1^R, U_2 = g_2^R$. Итак,

$$U_1^{z_1} U_2^{z_2} = g_1^{Rz_1} g_2^{Rz_2} = (g_1^{z_1} g_2^{z_2})^R = h^R. \tag{15.3.7}$$

Следовательно, имитация расшифровки, выполненная Саймоном при вычислении формулы (15.3.6), является корректной. Вероятность противоположного события равна $\frac{1}{q}$. Этот факт устраняет сомнения, может ли Саймон правильно выполнить расшифровку, не имея закрытого компонента $z \equiv z_1 + z_2 \log_{g_1} g_2 (\mathrm{mod}\, q)$, сформулированные в разделе 15.3.3.3.

Способность правильно выполнять расшифровку корректных зашифрованных текстов позволяет Саймону обучать атакующий алгоритм $\mathcal{A}$ в рамках атаки IND-CCA2.

Покажем теперь, что курсы обучения криптоанализу не компрометируют качество зашифрованного оклика, скрывающего число m_b.

Для любого корректно зашифрованного текста, присланного атакующим алгоритмом $\mathcal{A}$, возвращенный результат расшифровки только подтверждает, что пара целых чисел (z_1, z_2) в тексте (15.3.7) зависит от тройки (U_1, U_2, h^R) точно так же, как пара компонентов открытого ключа (g_1, g_2, h) в третьем уравнении системы (15.3.2). Таким образом, атакующий алгоритм $\mathcal{A}$ не может получить никакой дополнительной информации о числах z_1 и z_2. Следовательно, если атакующий алгоритм $\mathcal{A}$ присылает корректно зашифрованный текст, то курс криптоанализа ему ни к чему.

Чтобы не утерять возможности, полученные в результате предыдущего курса криптоанализа, атакующий алгоритм $\mathcal{A}$ должен представить зашифрованный текст $C = (U_1, U_2, E, V)$, для которого $(g_1, g_2, U_1, U_2) \notin \mathbf{D}$. Если этот текст пройдет проверку, выполняемую Саймоном, атакующему алгоритму $\mathcal{A}$ будет возвращен результат расшифровки, связанный с зашифрованным окликом отношением,

известным только алгоритму $\mathcal{A}$. Поскольку предполагается, что атакующий алгоритм $\mathcal{A}$ является очень хитроумным, то, если $(g_1, g_2, U_1, U_2) \notin \mathbf{D}$, никогда нельзя быть уверенным, что полученный результат расшифровки как-то связан с зашифрованным окликом. Так как атакующий $\mathcal{A}$ убежден в существовании скрытой связи, больше нельзя утверждать, что шифрование числа m_b точно соответствует схеме Крамера–Шоупа в первом варианте или является стойким в теоретико-информационном смысле во втором варианте.

К счастью, если атакующий алгоритм $\mathcal{A}$ снова пошлет зашифрованный текст $C = (U_1, U_2, E, V)$, когда $(g_1, g_2, U_1, U_2) \notin \mathbf{D}$, он тут же получит отказ. Это свойство гарантирует теорема 15.1, которую мы вскоре докажем. В частности, будет показано, что вероятность отказа не меньше $1 - \frac{1}{q}$. Обратите внимание на то, что при противоположном событии помощь Саймона атакующему не нужна: с вероятностью $\frac{1}{q}$ он способен самостоятельно угадывать, корректен ли зашифрованный текст, поскольку размер группы G равен q.

Итак, атакующий алгоритм $\mathcal{A}$ не может предотвратить отказ, посылая неправильно зашифрованный текст.

Вероятность успешно пройти проверку, если посланный зашифрованный текст был некорректным, оценивается следующей теоремой.

Теорема 15.1. *Пусть (g_1, g_2, c, d, h, H) — открытый ключ в схеме шифрования Крамера–Шоупа в группе G, имеющей простой порядок q, причем $g_1 \neq 1$ и $g_2 \neq 1$. Если $(g_1, g_2, U_1, U_2) \notin \mathbf{D}$, то вероятность правильно решить следующую задачу равна $\frac{1}{q}$ независимо от применяемого алгоритма.*

ИСХОДНЫЕ ДАННЫЕ:

Открытый ключ $(g_1, g_2, c, d, h, H), (U_1, U_2, E) \in G^3$.

РЕЗУЛЬТАТ:

$V \in G : (U_1, U_2, E, V)$ — *корректный зашифрованный текст, по мнению владельца ключа.*

Примечание 15.1. *Мы упростили задачу создания корректного зашифрованного текста, сведя ее к вычислению четвертого компонента по заданной тройке (U_1, U_2, E) и открытому ключу. Обработка компонента V и вычисление одного из трех компонентов зашифрованного текста, по существу, представляют собой одну и ту же задачу. Однако, поскольку число V не является аргументом функции хэширования H, найти число V проще всего.* □

Доказательство. Для того чтобы создать корректный зашифрованный текст на основе введенных величин, алгоритм должен вычислить значение $V \in G$, удовлетворяющее условию

$$U_1^{x_1 + y_1 \alpha} U_2^{x_2 + y_2 \alpha} = V, \tag{15.3.8}$$

где числа x_1, y_2, x_2, y_2 — закрытый ключ владельца открытого ключа, а $\alpha = H(U_1, U_2, E)$.

Поскольку группа G имеет простой порядок q, число $g_1 \neq 1$ является порождающим элементом группы G (следствие 5.3). Итак, введем обозначение $r_1 = \log_{g_1} U_1, r_2 = \log_{g_2} U_2, w = \log_{g_1} g_2$. Кроме того, для любого элемента $V \in G$ существуют величины $\log_{g_1} d$ и $\log_{g_1} V$. Сочетая формулу (15.3.8) с конструированием компонентов открытого ключа c и d (которые неявно подвергаются проверке при генерации ключа), получаем следующую линейную систему.

$$\begin{pmatrix} 1 & 0 & w & 0 \\ 0 & 1 & 0 & w \\ r_1 & r_1\alpha & wr_2 & wr_2\alpha \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} c \\ \log_{g_1} d \\ \log_{g_1} V \end{pmatrix} (\mathrm{mod}\, q). \qquad (15.3.9)$$

Применяя метод Гаусса, приведем матрицу (15.3.9) к следующему виду.

$$\begin{pmatrix} 1 & 0 & w & 0 \\ 0 & 1 & 0 & w \\ 0 & 0 & w(r_2 - r_1) & w\alpha(r_2 - r_1) \end{pmatrix} (\mathrm{mod}\, q). \qquad (15.3.10)$$

Если $(g_1, g_2, U_1, U_2) \notin \mathbf{D}$, выполняется неравенство $r_1 \neq r_2 (\mathrm{mod}\, q)$. Кроме того, отметим, что $w \neq 0 (\mathrm{mod}\, q)$ (поскольку число $g_2 \neq 1$ также является порождающим элементом группы G). Итак, матрица (15.3.10) имеет полный ранг, равный трем, т.е. три строки являются линейно независимыми. Как известно, это означает, что для любого элемента $V \in G$ существуют (неединственные) решения $(x_1, y_1, x_2, y_2)(\mathrm{mod}\, q)$.

Итак, мы доказали, что, если входные данные удовлетворяют условиям теоремы, для всех q элементов группы G существует корректное число V, т.е. при заданных входных данных любой элемент $V \in G$ образует корректный зашифрованный текст (U_1, U_2, E, V). Однако для владельца ключа среди q возможных вариантов существует только одно число $V \in G$, согласованное с закрытым ключом (x_1, y_1, x_2, y_2). (Несмотря на то что для любого фиксированного элемента $V \in G$ существует несколько целочисленных решений системы (15.3.9), совершенно очевидно, что любому элементу $V \in G$ может соответствовать только одна фиксированная четверка (x_1, y_1, x_2, y_2).) Таким образом, установлена вероятность успеха при решении задачи, сформулированной в теореме. □

Итак, доказательство стойкости криптосистемы Крамера–Шоупа завершено.

Очевидно, что “сведение к противоречию” в этом доказательстве является линейным: атакующие способности алгоритма $\mathcal{A}$ точно соответствуют способности определять, принадлежит ли данная четверка множеству $\mathbf{D}$. Следовательно,

можно утверждать, что доказательство стойкости криптосистемы Крамера–Шоупа является строгой редукцией.

15.4 Обзор доказуемо стойких гибридных систем

Гибридные криптосистемы были описаны в разделе 8.15. Они стойки к атаке IND-CCA2 и пригодны для применения схем шифрования с открытым ключом. В этом разделе содержится обзор ряда гибридных криптосистем. Поскольку их количество велико, доказательство их стойкости не приводится. Интересующиеся читатели могут найти их в соответствующих работах.

Зашифрованный текст, созданный с помощью гибридной криптосистемы, имеет два компонента: **механизм инкапсуляции ключа** (key encapsulation mechanism — KEM) и **механизм инкапсуляции данных** (data encapsulation mechanism — DEM). Зашифрованную пару KEM-DEM можно записать в следующем виде.

$$\text{KEM} \parallel \text{DEM} = \mathcal{E}_{pk}^{\text{asym}}(K) \parallel \mathcal{E}_{pk}^{\text{sym}}(K)\,(\text{Сообщение}).$$

Получив эту пару, адресат должен расшифровать блок KEM, используя свой закрытый ключ, и получить эфемерный симметричный ключ K, а затем применить его для расшифровки блока DEM и извлечь сообщение.

Если блок KEM является результатом применения асимметричной схемы шифрования, стойкой к атаке IND-CCA2, то стойкость блока DEM к атаке IND является естественным следствием случайности эфемерного ключа. Нетрудно понять, что гибридная криптосистема, использующая структуру KEM-DEM, может быть стойкой к атаке IND-CCA2. Действительно, гибридные схемы, имеющие структуру KEM-DEM, должны рассматриваться как естественное решение задачи эффективного шифрования с открытым ключом, стойкого к атаке IND-CCA2.

Гибридные системы считаются наиболее естественными, поскольку они могут шифровать сообщения любой длины при небольших накладных расходах. В приложениях длина данных может изменяться, и в большинстве случаев она превышает длину фиксированного параметра безопасности криптосистем с открытым ключом, например, число n в схеме RSA-OAEP или число $\log_2(\#G)$ в схеме Крамера–Шоупа. Поскольку криптосистемы с открытым ключом имеют более высокие накладные расходы, чем симметричные криптосистемы, вполне естественно, что доказуемо стойкие схемы шифрования с открытым ключом, такие как схемы RSA-OAEP и Крамера–Шоупа, используются в гибридных системах только для создания блоков KEM, а шифрование данных осуществляется в виде серии блоков DEM.

К гибридным схемам шифрования относятся схемы KEM-DEM, предложенные Шоупом [271], схема FO, изобретенная Фуджисаки и Окамото [113], схема

HD-RSA, разработанная Пойнтшевалем (Pointcheval) [232], схема DHAES, спроектированная Абдаллой (Abdalla), Белларе и Роджуэем [4], вариант схемы Крамера–Шоупа, описанный Шоупом [269], а также схема REACT, созданная Окамото и Пойнтшевалем [223].

Схема Фуджисаки–Окамото имеет следующий вид.

$$\mathcal{E}_{pk}^{\text{hybrid}}(m) = \text{KEM} \parallel \text{DEM} = \mathcal{E}_{pk}^{\text{asym}}(\sigma, H(\sigma, m)) \parallel \mathcal{E}_{G(\sigma)}^{\text{sym}}(m),$$

где G и H — функции хэширования. В этой схеме результатом расшифровки блока KEM является пара $\sigma, H(\sigma, m)$. Получатель использует число σ как начальное значение функции хэширования G для создания симметричного ключа $G(\sigma)$, а затем применяет его при расшифровке блока DEM. В заключение получатель может проверить правильность расшифровки, вычислив значение $H(\sigma, m)$ повторно. Итак, эта схема позволяет получателю определить, был ли зашифрованный текст модифицирован или поврежден в ходе сеанса связи. Распознавание изменений зашифрованного текста обеспечивает стойкость криптосистемы против активных атак.

Схема HD-RSA, разработанная Пойнтшевалем, основана на неразрешимости **зависимой задачи RSA** (dependent RSA) [233]: по заданному тексту $A = r^e (\text{mod}\, N)$, зашифрованному схемой RSA, найти число $B = (r+1)^y (\text{mod}\, N)$. Совершенно очевидно, что эта проблема является трудноразрешимой, если невозможно найти корень числа A e-й степени по составному модулю N (задача RSA). Следовательно, блоком KEM в схеме HD-RSA является просто число $A = r^e (\text{mod}\, N)$, где $r \in \mathbb{Z}_N^*$ — случайное число. Получатель, знающий модель N, может определить число r по значению A и найти число B. В качестве симметричного ключа при шифровании блока DEM эта схема, как и гибридные схемы Шоупа и Фуджисаки–Окамото, использует число $K = G(B)$.

В гибридной схеме DHAES, изобретенной Абдаллой, Белларе и Роджуэем [4], блок DEM сопровождается кодом аутентификации сообщения (MAC), предназначенным для проверки целостности данных. Симметричные ключи (один — для блока DEM, а второй — для блока MAC) создаются с помощью функции хэширования $H(g^u, g^{uv})$, где g^u — блок KEM, а g^v — открытый ключ получателя. Совершенно очевидно, что владелец открытого ключа g^v может применить закрытый ключ v к KEM-блоку g^u, чтобы получить число g^{uv} и восстановить число $H(g^u, g^{uv})$, используемое при создании двух симметричных ключей. Без закрытого ключа v задача расшифровки эквивалентна решению вычислительной задачи Диффи–Хеллмана (определение 8.1). Задача вычисления числа $H(g^u, g^{uv})$ по заданным аргументам g^u и g^{uv} называется **задачей хэширования Диффи–Хеллмана** (Hash Diffie–Hellman problem — HDH).

Интересно, что если в схеме DHAES число g^{uv} непосредственно используется в качестве криптографического множителя (как в схемах Эль-Гамаля и Крамера–Шоупа), то семантическая стойкость основывается на следующей задаче: определить, является ли кортеж $(g, g^u, g^v, g^{uv}(= e/m_b))$ четверкой Диффи–Хеллмана.

Применение в этой гибридной схеме функции хэширования затрудняет доступ к четвертому элементу кортежа, и задача принятия решений становится легче, чем вычислительная задача. Напомним, что предположение о неразрешимости задачи, обеспечивающей стойкость криптосистемы, должно быть как можно более слабым. Читатели могут самостоятельно доказать, что по сложности задача HDH лежит между задачей CDH (определение 8.1 в разделе 8.4) и задачей DDH (определение 13.1 в разделе 13.3.4.3). Следует отметить, что "ослабление предположения" задачи HDH по сравнению с задачей DDH не является безусловным: для этого необходимо дополнительно ограничить функцию хэширования (что не было сделано для краткости изложения). К сожалению, неявное ограничение на функцию хэширования очень напоминает предположение о свойствах случайного оракула.

Гибридная схема Шоупа [269] использует ослабленное предположение о неразрешимости задачи, применяемой в схеме Крамера–Шоупа. В оригинальной схеме Крамера–Шоупа (алгоритм 15.1) шифрование сообщения m имело вид схемы Эль-Гамаля: $h^r m$. В "ослабленной" версии [269] число h^r скрыто с помощью функции хэширования $H(\ldots; h^r)$. Это затрудняет решение задачи DDH. Хэшированное значение $H(\ldots; h^r)$ используется для создания симметричных ключей, шифрующих блок DEM и применяемых в механизме проверки целостности данных. Шоуп назвал ослабленное предположение о неразрешимости задачи DDH "хеджированием с помощью хэширования" ("hedging with hash").

15.5 Литературные заметки о прикладных доказуемо стойких криптосистемах с открытым ключом

Первой работой, посвященной прикладным криптосистемам с открытым ключом, устойчивым к активным атакам, является статья Дамгарда (Damgard) [88]. В ней описан способ предотвращения активных атак на криптосистему с открытым ключом с помощью процедуры проверки целостности данных. Впоследствии этот метод лег в основу общей стратегии проектирования криптосистем, стойких к активным атакам. Однако две схемы Дамгарда, описанные в его работе, оказались уязвимыми для атаки CCA2 [311].

Ченг (Zheng) и Себерри (Seberry) предложили практичные схемы шифрования и цифровой подписи, гарантирующие стойкость к атаке CCA2 [310, 311]. Общая идея этих схем заключалась в усовершенствовании однонаправленной функции, применяемой в учебных прототипах (в схеме Эль-Гамаля), с помощью функции хэширования. Это важная идея, которая позднее легла в основу модели случайного оракула, используемой при доказательстве стойкости (см. раздел 15.2). Доказа-

тельство стойкости схем к атаке IND-CCA2, изложенное в работе [310], основано на нестандартном предположении, получившем название "исключительности пространств, генерируемых функциями" ("sole-samplability of spaces induced by functions"), а также на предположении о неразрешимости задачи Диффи–Хеллмана. Впоследствии Солдера (Soldera) обнаружил, что одна из схем Ченга и Себерри на самом деле уязвима к атаке IND-CCA2 [280, 281].

Обнаружив неполноту доказательства стойкости схемы RSA-OAEP, Шоуп предложил для рандомизированной схемы заполнения на основе алгоритма RSA модификацию, получившую название OAEP+ [270]. Он доказал ее стойкость с помощью более строгой редукции, чем редукция, примененная при доказательстве стойкости схемы RSA-OAEP в работе Фуджисаки и его соавторов [114]. Повышенная строгость редукции обеспечивается благодаря тому, что преимущество Саймона при инвертировании функции RSA линейно зависит от преимущества, с которым Злоумышленник взламывает криптосистему. Однако, поскольку время, затрачиваемое Саймоном на инвертирование функции RSA, по-прежнему ограничено квадратичной функцией, зависящей от количества запросов, которые Злоумышленник вынужден посылать случайному оракулу, процесс редукции остается неэффективным (см. раздел 15.2.5). Бонэ (Boneh) также предложил модификации схемы OAEP, получившие названия Simple-OAEP (SAEP) и Simple-OAEP+ (SAEP+) [49]. Однако эти схемы не позволяли восстанавливать сообщения (эта проблема будет рассматриваться при описании некоторых схем рандомизированного заполнения в разделе 16.4.4.2). Недавно Корон (Coron) и его соавторы [83] показали, что для шифрования с помощью алгоритма RSA можно применять другую схему рандомизированного заполнения, получившую название "вероятностная схема цифровой подписи с восстановлением сообщения (Probabilistic Signature Scheme with Message Recovery — PSS-R). Ее исходный вариант был предложен Белларе и Роджуэем в работе [26] (см. следующую главу). По существу, эти авторы догадались, что применение функций хэширования в схеме заполнения отменяет требование избыточности при проверке целостности данных (например, наличия младших нулевых разрядов в схеме RSA-OAEP). Эта схема будет описана в следующей главе. Как и схемы SAEP и SAEP+, схема PSS-R обладает слабыми возможностями для восстановления сообщения, если в ее основе лежит шифрование с помощью алгоритма RSA (см. раздел 16.4.4.2).

Подобно схеме RSA-OAEP, схемы рандомизированного заполнения, упомянутые в предыдущем абзаце, обладают стойкостью к атаке IND-CCA2, доказанной в рамках модели случайного оракула. Однако, поскольку стойкость схемы RSA-OAEP к атаке IND-CCA2 была доказана вновь (раздел 15.2.4), причем схема RSA-OAEP долгое время была стандартом шифрования с помощью алгоритма RSA, и схема OAEP позволяет лучше восстанавливать сообщения, пока неясно, сможет ли новая схема стать такой же популярной.

Приемы заполнения, разработанные для функции OWTP, привели к созданию оптимальных схем. Однако функция OWTP на самом деле обладает редкими свойствами. Вероятно, среди распространенных функций, применяемых в криптографии с открытым ключом, единственными OWTP-функциями являются функции RSA и Рабина (над квадратичными вычетами). Более того, модель случайного оракула, на которой основано доказательство стойкости схем заполнения до сих пор не позволяла выполнить "сведение к противоречию". Некоторые исследователи предлагают изобрести доказуемо стойкие схемы для однонаправленной функции общего вида. Другие авторы изобретают схемы, усовершенствующие схемы заполнения, заменяя функции OWTP однонаправленными функциями с секретом общего вида. Многие функции, применяемые в криптографии с открытым ключом, не являются перестановками (например, функция Эль-Гамаля). Следовательно, такая модификация является полезной. Фуджисаки и Окамото [112], а такжде Пойнтшеваль [234] предложили две обобщенные схемы, обладающие этим свойством. Однако эти схемы оказались неоптимальными: во время расшифровки для распознавания ошибок требуется выполнять повторное шифрование. Поскольку расшифровка часто выполняется с помощью медленных вычислительных устройств, например, смарт-карт, таких схем следует избегать.

Разумеется, гибридные системы образовали целое семейство схем шифрования с открытым ключом, доказуемо стойких к атаке IND-CCA2. Их обзор содержится в разделе 15.4.

В заключение напомним, что для практичных схем шифрования с открытым ключом, доказуемо стойких к атаке IND-CCA2, механизм проверки целостности данных обеспечивает их защиту без идентификации источника (см. раздел 10.5). В большинстве реальных приложений этого недостаточно, поэтому в криптографии с открытым ключом для идентификации источника широко применяются схемы цифровой подписи.

Недавно был изобретен новый криптографический примитив, получивший название **зашифрованной подписи** (signcryption). Схема зашифрованной подписи объединяет в одно целое шифрование и цифровую подпись. Такое объединение позволяет повысить эффективность шифрования с открытым ключом и в то же время добиться дополнительных свойств, необходимых для электронной коммерции: идентификации источника сообщений и невозможности отказаться от авторства. Поскольку этот криптографический примитив появился после того, как понятие доказуемой стойкости криптосистем с открытым ключом, возникшее в 1997 году в работе Ченга [309], получило широкое признание, исследователи с готовностью стали применять его при разработке своих схем цифровой подписи. Это понятие будет исследовано в следующей главе.

15.6 Резюме

В главе подробно описаны две популярные криптосистемы с открытым ключом, которые не только обладают доказанной стойкостью и пригодны для практического применения, т.е. неуязвимы для атаки IND-CCA2, но и эффективны. Эффективность этих криптосистем сравнима с эффективностью их учебных аналогов. Схемы шифрования, рассмотренные в главе, представляют собой большой шаг вперед по сравнению с предшествующими схемами, основанными на побитовом шифровании сообщений.

В криптосистемах с открытым ключом, предусматривающих проверку целостности данных на этапе расшифровки, зашифрованный текст содержит сообщение, сопровождаемое "цифровой подписью", созданной с помощью открытого ключа отправителя. Схема "цифровой подписи" позволяет восстанавливать сообщение. Следовательно, получатель может восстановить исходный текст и проверить подлинность подписи, используя собственный открытый ключ. Эта возможность является технически корректной. Единственная причина, по которой мы берем слова "цифровая подпись" в кавычки, заключается в том, что лицом, поставившим подпись, может быть кто угодно. Следовательно, криптографическое преобразование не гарантирует подлинности подписи в полном смысле этого слова. Несмотря на это, трудности, возникающие при фальсификации подписи у злоумышленника, не знающего процедуры и не владеющего открытым ключом, эффективно предотвращают атаку на основе подобранного зашифрованного текста. Это основная причина, по которой схема шифрования (как и две криптосистемы, описанные в главе), должна быть неуязвимой для атаки CCA2.

При доказательстве стойкости этих криптосистем было введено и проанализировано несколько важных понятий: модель случайного оракула, формальное доказательство путем сведения к противоречию и строгость редукции.

В главе приведен обзор различных гибридных криптосистем, сочетающих методы симметричного и асимметричного шифрования, с одной стороны, и достоинства криптосистем с открытым ключом, с другой.

В заключение рассматривается литература, относящаяся к изучаемой теме.

Упражнения

15.1. Какую роль играет случайный аргумент в алгоритме RSA-OAEP? Какова роль фиксированной строки 0^{k_1} в этом алгоритме?

15.2. Ширина полосы пропускания алгоритма шифрования измеряется размером исходного текста, который он может зашифровать при заданном параметре безопасности. Предположим, что длина параметра безопасности в схеме RSA-OAEP равна 2048 бит, а размер случайного аргумента — 160 бит. Какова полоса пропускания такой схемы RSA-OAEP?

15.3. Опишите модель случайного оракула, используемую при доказательстве стойкости криптосистем?

15.4. Какие ограничения налагаются моделью случайного оракула на доказательство стойкости криптосистемы?

15.5. Почему имитация случайного оракула, описанная в разделе 15.2.1, должна достигаться с помощью упорядоченного списка, который в начальный момент пуст?

15.6. В чем заключается "противоречие" в методе "сведения к противоречию" при доказательстве стойкости криптосистем, основанном на вычислительной сложности задачи?

15.7. Почему зашифрованный оклик в редукционном доказательстве должен быть случайным?

15.8. Почему Саймон должен несколько раз выполнять атакующий алгоритм при доказательстве стойкости схемы RSA-OAEP?

15.9. Почему модуль схемы RSA-OAEP, состоящий из 1024 бит, слишком мал для того, чтобы предотвратить атаку, несмотря на то, что модуль невозможно разложить на простые множители современными методами?

15.10. В схеме Крамера–Шоупа также используется функция хэширования. Должна ли она обладать свойствами случайного оракула для того, чтобы схема была доказуемо стойкой?

15.11. Предположим, что криптосистема Крамера–Шоупа модифицирована следующим образом:

$$e \leftarrow h^r + m.$$

Таким образом, расшифровка выполняется путем вычитания, а остальные части остаются без изменения. Докажите, что модифицированная схема является стойкой к атаке CCA2, т.е. любую активную атаку можно распознать. Является ли эта схема стойкой к атаке IND-CCA2?

15.12. Почему стоимость вычисления величины $g^x h^y (\mathrm{mod}\, p)$ сравнима со стоимостью одной операции возведения в степень?

15.13. Распространите алгоритм 15.2 на случай $f^x g^y h^z (\mathrm{mod}\, p)$.

15.14. Опишите гибридную криптосистему.

15.15. В приложениях длина секретных данных намного больше размера параметра безопасности в криптосистемах с открытым ключом, а в симметричных схемах шифрования — намного меньше. Какие алгоритмы шифрования следует применить при передаче таких данных: 1) RSA; 2) AES; 3) RSA-OAEP; 4) Эль-Гамаля; 5) Крамера–Шоупа или 6) гибридную криптосистему?

Глава 16

Сильная и доказуемая стойкость схем цифровой подписи

16.1 Введение

В определение схемы цифровой подписи (определение 10.2 из раздела 10.4) входит условие $\mathsf{Verify}_{pk}(m, s) = \mathsf{False}$ с "огромной" вероятностью, если пара (m, s) является подделкой, созданной без применения предписанной процедуры. Однако мы не уточнили, насколько высокой должна быть эта вероятность. Кроме того, как указано в разделе 10.4.9, учебное понятие стойкости схем цифровой подписи (т.е. сложность создания подделки "с нуля") является слишком слабым, чтобы получить применение на практике. Следовательно, аргументы, приведенные в главе 10, носят слишком неформальный характер. Это объяснялось тем, что в тот момент мы еще не обладали достаточными знаниями, позволяющими провести формальное и строгое доказательство стойкости.

В двух предшествующих главах изложена методика формального доказательства стойкости схем шифрования с открытым ключом, в том числе доказательство сильной стойкости (например, IND-CCA2), описаны метод "сведения к противоречию", модель случайного оракула и стандартная модель доказательства стойкости. Эти понятия образуют фундамент для формального анализа стойкости схем цифровой подписи. Как и при исследовании стойкости схем шифрования с открытым ключом, при изучении схем цифровой подписи необходимо исследовать два аспекта.

- **Трудность подделки подписи в ходе наиболее распространенных атак на схемы цифровой подписи.** Наиболее распространенной атакой на схемы цифровой подписи является **атака на основе адаптивно подобранных сообщений** (adaptive chosen-message attack). Адаптивный алгоритм атаки обладает открытым ключом объекта атаки и заставляет пользователя играть роль **оракула подписи** (oracle signing service) для любого заранее подобранного сообщения. Такой алгоритм способен адаптировать свои запросы

в соответствии с собранными парами сообщение-подпись. Можно считать, что в ходе адаптивной атаки атакующий алгоритм проходит курс обучения подделке подписи. Задачей атакующего, пославшего достаточно большое количество адаптивно подобранных сообщений и получившего в ответ соответствующие подписи, является создание *новой* пары сообщение-подпись, согласованной с открытым ключом пользователя. Здесь слово "новое" означает сообщение, которое никогда ранее не подписывалось пользователем.

- **Формальное доказательство стойкости**. Это доказательство основано на методе "сведения к противоречию". Оно представляет собой эффективное преобразование, демонстрирующее, что любой успешный алгоритм подделки (например, в результате адаптивной атаки) сводится к решению трудноразрешимой задачи из теории вычислительной сложности. "Противоречие" основано на широко распространенном убеждении, что не существует эффективных алгоритмов решения трудноразрешимых задач.

Голдвассер, Микали и Ривест систематически исследовали эти аспекты в своей знаменитой работе [127]. Они реализовали схему цифровой подписи и доказали ее неуязвимость к атаке на основе адаптивно подобранных сообщений. В этой схеме используется понятие "расцепленных" перестановок ("claw-free permutation"): две перестановки f_0 и f_1, определенные на одной и той же области, называются расцепленными, если невозможно найти тройку (x, y, z), такую что $f_0(x) = f_1(y) = z$. Голдвассер и его соавторы реализовали такие перестановки с помощью задачи факторизации целого числа. Эта схема позволяет подписать любую случайную строку без добавления распознаваемой избыточности, т.е. без применения функций хэширования при форматировании сообщения. Однако эта схема носит побитовый характер и, следовательно, не является идеальной для приложений. И все же работа Голдвассера и его коллег [127] заложила основы теории сильной (т.е. пригодной для приложений) стойкости схем цифровой подписи.

16.1.1 Структурная схема главы

В разделе 16.2 вводится понятие сильной стойкости схемы цифровой подписи. Раздел 16.3 посвящен редукционному доказательству стойкости схемы цифровой подписи Эль-Гамаля. В разделе 16.4 описываются прикладные схемы цифровой подписи, основанные на методе рандомизированого заполнения и однонаправленных перестановках с секретом (в основном, функции RSA и Рабина). В разделе 16.5 изучаются схемы шифрования подписи и их сильная стойкость.

16.2 Сильная стойкость схемы цифровой подписи

Введем необходимые определения.

Во-первых, будем представлять схему цифровой подписи в виде тройки (Gen, Sign, Verify), элементы которой определены в разделе 10.4 (определение 10.2). Поскольку защита схем цифровой подписи от подделок обеспечивается с помощью криптографических функций хэширования, будем считать, что в алгоритмах Sign и Verify используются одна или несколько сильных функций хэширования. Под сильной функцией хэширования подразумевается функция, имитирующая поведение случайного оракула (раздел 10.3.1.2). Действительно, все доказательства стойкости, приведенные в главе, основаны на модели случайного оракула (см. раздел 15.2.1).

Приведем определение атаки на основе адаптивно подобранных сообщений.

Определение 16.1 (Атака на основе адаптивно подобранных сообщений). *Пусть k — положительное целое число. Адаптивный алгоритм подделки цифровой схемы* (Gen, Sign, Verify) *представляет собой вероятностный алгоритм, время работы которого полиномиально зависит от параметра k. Он получает на вход открытый ключ pk, где $(pk, sk) \leftarrow_U \mathsf{Gen}(1^k)$, и пытается подделать подписи в соответствии с ключом pk. Фальсификатору разрешается посылать запросы и получать подписи на сообщениях, выбранных заранее. В ходе моделирования этого процесса фальсификатор получает доступ к алгоритмам подписи и хэширования, время работы которых полиномиально зависит от параметра k.*

Фальсификатор осуществляет $(t(k), \mathrm{Adv}(k))$-взлом схемы, если за время $t(k)$ он с вероятностью $\mathrm{Adv}(k)$ создает корректную подпись, т.е. пару сообщение-подпись (m, s), удовлетворяющую условию $\mathsf{Verify}_{pk}(m, s) = \mathsf{True}$, где m — распознаваемое сообщение, созданное с помощью функции хэширования, используемой в схеме, но еще не поступавшее на вход алгоритма Sign. *Здесь $t(k)$ — полином, а $\mathrm{Adv}(k)$ — значимая величина, зависящая от параметра k.*

Понятие значимой величины приведено в разделе 4.6.

Мы упростили это определение, не включив в него два выражения, оценивающие количество подписей и хэшированных запросов, созданных фальсификатором. Обе оценки являются полиномиальными: поскольку фальсификатор представляет собой полиномиальный алгоритм, зависящий от параметра k, количество подписей и хэшированных запросов ограничено полиномом.

Определение 16.2 (Стойкая схема цифровой подписи). *Схема цифровой подписи* (Gen, Sign, Verify) *называется $(t(k), \mathrm{Adv}(k))$-стойкой, если не существует фальсификатора, способного осуществить $(t(k), \mathrm{Adv}(k))$-взлом схемы при всех достаточно больших параметрах k.*

Определение 16.2 используется для доказательства стойкости схем цифровой подписи путем "сведения к противоречию". Предположим, что заданная цифровая схема допускает $(t(k), \mathrm{Adv}(k))$-взлом, где $t(k)$ — полином, а $\mathrm{Adv}(k)$ — значимая величина. Мы построим редукцию, позволяющую перевести полином $t(k)$ в полином $t'(k)$, а величину $\mathrm{Adv}(k)$ — в величину $\mathrm{Adv}'(k)$ так, что трудноразрешимая задача, лежащая в основе схемы цифровой подписи, будет допускать $(t'(k), \mathrm{Adv}'(k))$-взлом. Если процесс редукции достаточно эффективен, то значение $t'(k)$ будет малым, а величина $\mathrm{Adv}'(k)$ близкой к величине $\mathrm{Adv}(k)$, т.е. значимой. Следовательно, взлом цифровой схемы становится возможным, если трудноразрешимая задача допускает $(t'(k), \mathrm{Adv}'(k))$-решение, что считается невозможным. Это противоречие доказывает стойкость схемы. Понятие эффективной редукции описано в разделе 15.2.5.

Как и в предыдущей главе, при доказательстве стойкости схем цифровой подписи с помощью редукции будет использоваться специальный агент Саймон-имитатор. Он будет играть роль объекта атаки, посылая атакующему алгоритму свои подписи, поставленные на полученные запросы. Для этого используется имитация оракула подписи. Для того чтобы фальсификатор проявил всю свою силу, имитируемый оракул подписи не должен отличаться от истинного автора подписи. Поскольку фальсификатор полиномиально ограничен, достаточно применить понятие полиномиальной неразличимости, введенное в определении 4.15 (раздел 4.7).

В оставшейся части главы фальсификатор называется Злоумышленником и является активным атакующим алгоритмом.

16.3 Сильная и доказуемая стойкости схем цифровой подписи Эль-Гамаля

Долгое время (1985–1996) после появления схемы цифровой подписи Эль-Гамаля (раздел 10.4.6) и его аналогов (например, схемы Шнорра, описанной в разделе 10.4.8.1, и схемы DSS, изученной в разделе 10.4.8.2) считалось, что сложность их подделки связана с вычислением дискретного логарифма в большой подгруппе конечного поля. Однако до 1996 года этот факт не был формально доказан.

Строгое доказательство этого факта было получено Пойнтшевалем и Штерном (Stern) [235]. Для этого они применили мощное средство: модель случайного оракула (ROM) [22]. Основные идеи, положенные в основу этой модели, представлены в разделе 15.2.1. Модель Пойнтшеваля и Штерна представляет собой остроумную конкретизацию общей модели случайного оракула.

16.3.1 Тройная схема цифровой подписи Эль-Гамаля

Рассмотрим типичный вариант схемы цифровой подписи Эль-Гамаля, неуязвимость которого доказана с помощью модели ROM. В этом варианте на вход схемы поступает ключ подписи sk, открытый ключ pk и сообщение M, представляющее собой строку бит, а результатом ее работы является подпись сообщения M, представляющая собой тройку (r, e, s).

- Компонент r называется **фиксатором** (commitment). Он фиксирует эфемерное целое число ℓ, называемое **передачей** (committal). Это число не зависит от величин, использованных при создании предыдущих подписей. Обычно фиксатор выглядит как $r = g^{\ell} (\mathrm{mod}\, p)$, где g и p — часть открытых параметров схемы цифровой подписи.
- Компонент e представляет собой криптографическую функцию хэширования $H(M, r)$.
- Компонент s называется **подписью**. Он представляет собой линейную функцию, зависящую от фиксатора r, передачи ℓ, сообщения M, функции хэширования $H()$ и закрытого ключа подписи sk.

Такая схема цифровой подписи называется **тройной** (triplet signature scheme).

Исходная схема цифровой подписи Эль-Гамаля, описанная алгоритмом 10.3, не является тройной, поскольку в ней не используется функция хэширования, и она уязвима для экзистенциальной подделки. Однако вариант схемы Эль-Гамаля, в котором используется функция хэширования (см. раздел 10.4.7.2), становится стойким к экзистенциальной подделке и может классифицироваться как тройная схема.

Схема цифровой подписи Шнорра (алгоритм 10.4) также является тройной. Подпись сообщения M, созданная этим алгоритмом, представляет собой тройку (r, e, s), где $e = H(M, r)$, причем H — некая функция хэширования. Следует отметить, что в схеме Шнорра нет необходимости посылать величину r верификатору, поскольку ее можно вычислить как $g^s y^e$.

Опишем метод редукции, примененный Пойнтшевалем и Штерном для доказательства неуязвимости тройной схемы цифровой подписи. Этот метод называется **ветвящейся редукцией** (forking reduction).

16.3.2 Ветвящаяся редукция

Как показано в разделе 10.4.7.1, многократное использование эфемерного ключа (передачи ℓ или фиксатора r) в схеме цифровой подписи Эль-Гамаля приводит к раскрытию секретного ключа подписи. Компрометация закрытого ключа подписи сводится к эффективному решению трудноразрешимой задачи: вычисления дискретного логарифма открытого ключа в группе по модулю, являющемуся большим простым числом.

Доказательство стойкости тройной схемы Эль-Гамаля основано на повторной передаче фиксатора, предназначенной для раскрытия секретного ключа. Поскольку фальсификация подписи эффективно сводится к неразрешимой задаче — вычислению дискретного логарифма (предположение 8.2 в разделе 8.4), корректность доказательства зависит от эффективности процесса редукции.

В редукционном доказательстве стойкости тройной схемы Эль-Гамаля функция хэширования описывается с помощью случайного оракула (random oracle — RO), поведение которого рассмотрено в разделе 10.3.1.2. В модели ROM все случайные оракулы имитируются Саймоном. Кроме того, Саймон способен имитировать процедуру подписи и таким образом отвечать на запросы Злоумышленника. Итак, Саймон-имитатор может организовать для Злоумышленника курс обучения, необходимый для подготовки к фальсификации подписи. Следовательно, чтобы достичь успеха, Злоумышленник должен легко обучаться и создавать фальшивые подписи со значимой вероятностью. Саймон может использовать фальшивую подпись для решения трудноразрешимой задачи. В случае тройной схемы Эль-Гамаля этой задачей является вычисление дискретного логарифма в конечном поле. Процесс редукции проиллюстрирован на рис. 16.3.2.1.

В следующих двух разделах при описании процесса редукции широко используются интуитивные рассуждения. Приведенные оценки вероятности можно считать верхними границами точных оценок Пойнтшеваля и Штерна. Несмотря на это, при достаточно крупном параметре безопасности их можно использовать для констатации противоречия. Читатели, которых не устраивает такой уровень точности, могут обратиться непосредственно к работе [236].

16.3.2.1 Стойкость к неадаптивной атаке

Для начала рассмотрим стойкость тройной схемы Эль-Гамаля к неадаптивной атаке.

Пусть $(\mathsf{Gen}(1^k), \mathsf{Sign}, \mathsf{Verify})$ — *тройная* схема цифровой подписи Эль-Гамаля (т.е. вариант алгоритма 10.3), в которой простое число p удовлетворяет следующему условию: существует k-битовое простое число q, являющееся делителем числа $p-1$, причем число $(p-1)/q$ не имеет больших простых множителей.

Допустим, что Злоумышленник — успешный фальсификатор подписей в схеме $(\mathsf{Gen}(1^k), \mathsf{Sign}, \mathsf{Verify})$, а Саймон-имитатор контролирует все каналы, связывающие Злоумышленника с внешним миром, как показано на рис. 16.1. Однако в рамках неадаптивной атаки Злоумышленник не проходит "курс обучения подделке подписи", поскольку он никогда не запрашивает образец подписи.

Саймон генерирует случайное число $y \in \mathbb{Z}_p^*$. Его цель — вычислить дискретный логарифм числа y с базой, представляющей собой порождающий элемент g по модулю p, т.е. найти число x, удовлетворяющее условию $y \equiv g^x \pmod p$. Саймон использует Злоумышленника как "черный ящик", так что удачная подделка новой подписи на заранее подобранном сообщении предоставляет Саймону достаточно

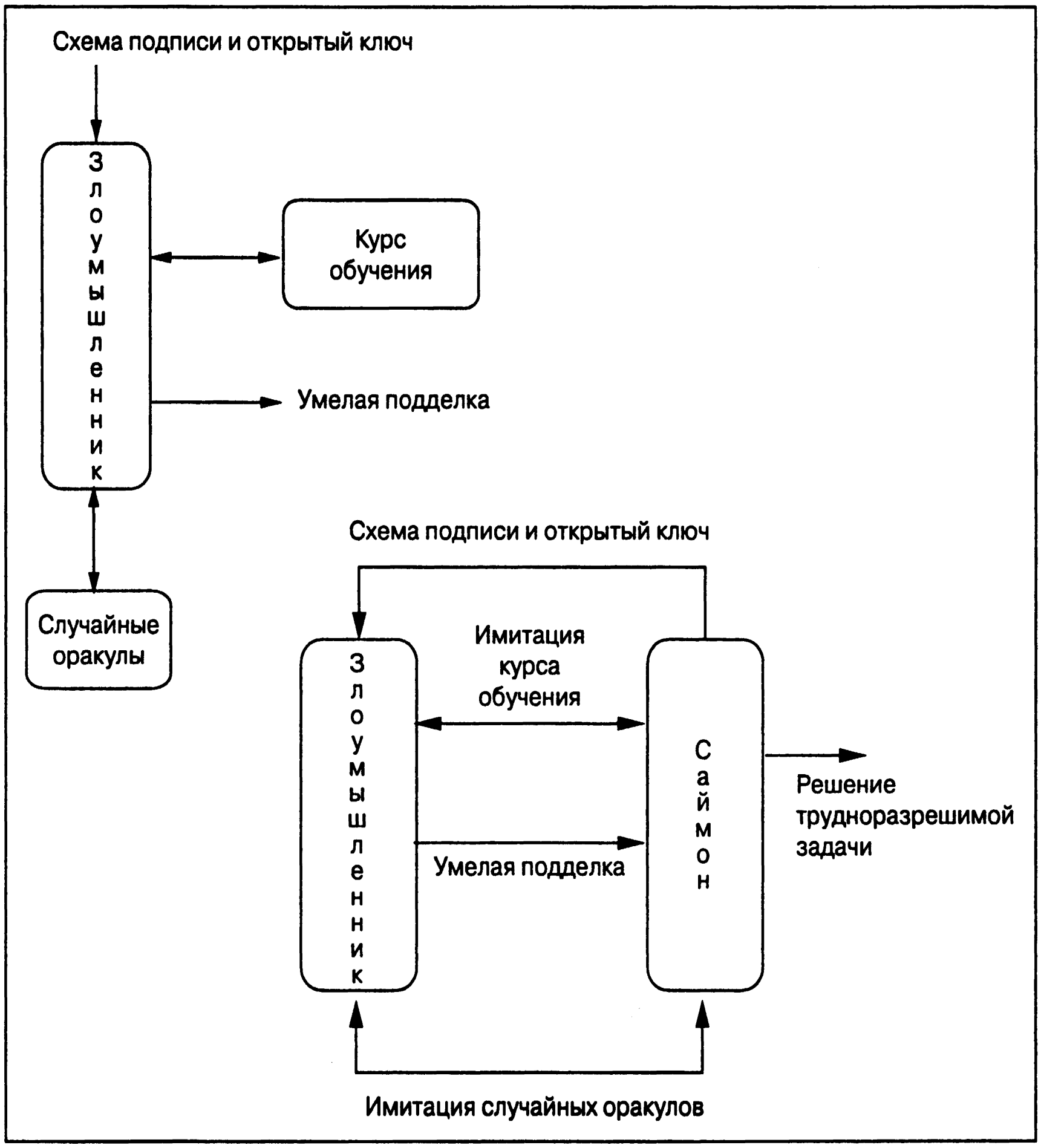


Рис. 16.1. Сведение подделки подписи к решению трудноразрешимой задачи

информации для вычисления дискретного логарифма. Интуиция подсказывает, что число y должно быть произвольным, в противном случае редукция становится бесполезной.

Допустим, что вероятность успешной подделки равна $\mathrm{Adv}(k)$, где число k является значимой величиной, а время, затраченное Злоумышленником на подделку, ограничено полиномом $t(k)$, зависящим от параметра k. Вычислим вероятность $\mathrm{Adv}'(k)$, с которой Саймон находит дискретный логарифм, и время его работы $t'(k)$. Разумеется, пары $(t(k), \mathrm{Adv}(k))$ и $(t'(k), \mathrm{Adv}'(k))$ связаны между собой.

Многократные вызовы Злоумышленника на первом этапе

Саймон вызывает Злоумышленника $1/\operatorname{Adv}(k)$ раз. Поскольку Злоумышленник считается успешным фальсификатором, при некоторых условиях (каких — уточним позднее) он с вероятностью, равной единице (поскольку его алгоритм выполняется $1/\operatorname{Adv}(k)$ раз), выдаст корректную подпись (r, e, s) сообщения M в схеме (Gen, Sign, Verify). Иначе говоря,

$$e = H(M, r),$$
$$y^r r^s \equiv g^e (\operatorname{mod} p),$$

где $|e| = k$.

При этом Злоумышленник должен несколько раз вычислить функцию хэширования H. В модели ROM, продемонстрированной на рис. 16.1, Злоумышленник должен послать Саймону запросы, предназначенные для случайного оракула. В ответ Саймон играет роль случайного оракула: имитирует функцию хэширования H, поддерживая H-список, содержащий элементы $((M_i, r_i), e_i)$, упорядоченные по сообщениям M_i. Здесь пары (M_i, r_i) представляют собой запросы, а число e_i — случайные ответы.

Поскольку Злоумышленник полиномиально ограничен, он может послать случайному оракулу лишь $n = q_H$ запросов, где величина q_H — значение полинома, зависящего от параметра k. Пусть пары

$$Q_1 = (M_1, r_1), Q_2 = (M_2, r_2), \ldots, Q_n = (M_n, r_n) \qquad (16.3.1)$$

представляют собой n запросов случайному оракулу, посланных Злоумышленником. Кроме того, пусть числа

$$R_1 = e_1, R_2 = e_2, \ldots, R_n = e_n$$

обозначают n ответов, полученных от Саймона. Поскольку $|H| = k$, ответы Саймона равномерно распределены по множеству $\{1, 2, 3, \ldots, 2^k\}$.

Благодаря этому свойству, если Злоумышленнику удалось подделать подпись (r, e, s) на сообщении M, он должен послать запрос (M, r) и получить ответ $e = (M, r)$. Иначе говоря, при некотором $i \in [1, n]$ должно выполняться условие $(M, r) = (M_i, r_i)$. Вероятность того, что пара (M, r) не будет послана в виде запроса, равна 2^{-k}. (Это число представляет собой вероятность того, что Злоумышленник правильно угадает случайный ответ Саймона $R_i = e_i$, не посылая ему запроса.) Считая величину 2^{-k} пренебрежимо малой, можно утверждать, что пара $((M, r), e)$ принадлежит H-списку Саймона.

Следует иметь в виду одно важное обстоятельство: без запроса, посланного Саймону, и его ответа, Злоумышленник не может подделать подпись. Вероятность противоположного события равна 2^{-k}, т.е. пренебрежимо мала. Учитывая это, можно считать, что Злоумышленник "вынужден" подделывать подпись одного из n сообщений в списке (16.3.1).

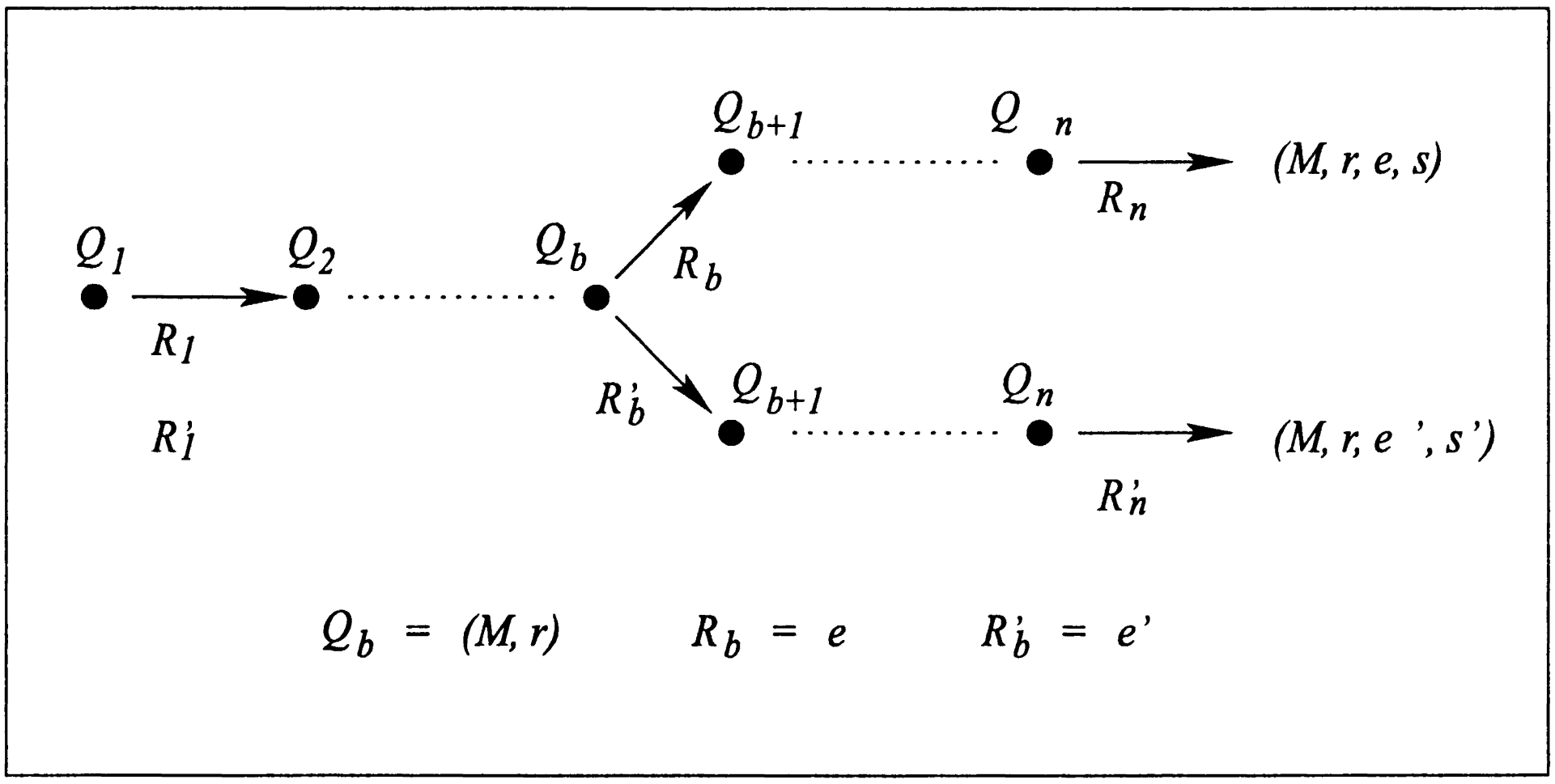


Рис. 16.2. Успешное разветвление ответов на запросы случайному оракулу

Многократные вызовы Злоумышленника на втором этапе

Саймон снова вызывает Злоумышленника $1/\operatorname{Adv}(k)$ раз при тех же условиях. Иначе говоря, повторяются n запросов, перечисленных в списке (16.3.1). Однако на этот раз Саймон создает n других случайных ответов, имеющих равномерное распределение.

Необходимо подчеркнуть, что, поскольку ответы по-прежнему равномерно распределены по множеству $\{1, 2, 3, \ldots, 2^k\}$, они остаются правильными.

Получив вторую порцию ответов, Злоумышленник должен снова проявить все свои способности и с вероятностью, равной единице, создать корректную подпись (r', e', s') сообщения M'. Как и в первый раз, пара (M', r') должна совпадать с одной из пар Q_j в списке (16.3.1) при некотором $j \in [1, n]$. Вероятность противоположного события равна 2^{-k}.

На рис. 16.2 проиллюстрировано "успешное разветвление запросов Злоумышленника". Это событие происходит, когда в обоих сеансах вызова Злоумышленника пары поддельных пар сообщение-подпись $(M, (r, e, s))$ и $(M', (r', e', s'))$ удовлетворяют условию $(M, r) = (M', r')$. Заметим, что в каждом из сеансов выполнения алгоритма Злоумышленник может подделать подпись для пары (M_i, r_i), где $i \in_U [1, n]$ — случайное число, имеющее равномерное распределение и не должно фиксироваться. Применяя парадокс дней рождения (см. раздел 3.6), получаем, что вероятность этого события (т.е. $i = j = b$) приблизительно равна $1/\sqrt{n}$. Следует иметь в виду, что, если число i во втором сеансе фиксировано, вероятность успешного разветвления равна $1/n$.

Напомним, что число n является полиномиально ограниченным, так что величина $1/\sqrt{n}$ не является пренебрежимо малой. Иначе говоря, Саймон получает

две поддельные корректные подписи (r, e, s) и (r', e', s') со значимой вероятностью $1/\sqrt{n}$. Кроме того, поскольку во втором сеансе Саймон порождает другие равномерно распределенные ответы, условие $e' \neq e(\mathrm{mod}\, q)$ должно выполняться с огромной вероятностью $1 - 2^{-k}$.

При успешном разветвлении Саймон способен вычислить дискретный логарифм. Посмотрим, как он это делает.

Вычисление дискретного логарифма

Получив две корректные подделки, Саймон может вычислить следующие величины.

$$y^r r^s = g^e (\mathrm{mod}\, p),$$
$$y^r r^{s'} = g^{e'} (\mathrm{mod}\, p).$$

Поскольку g — порождающий элемент по модулю p, для некоторого целого числа $\ell < p - 1$ выполняется условие $r = g^\ell (\mathrm{mod}\, p)$. Кроме того, $y = g^x (\mathrm{mod}\, p)$. Следовательно,

$$xr + \ell s = e(\mathrm{mod}\, q),$$
$$xr + \ell s' = e'(\mathrm{mod}\, q).$$

Поскольку из условия $e' \neq e(\mathrm{mod}\, q)$ следует, что $s' \neq s(\mathrm{mod}\, q)$, выполняется равенство

$$\ell = \frac{e - e'}{s - s'} (\mathrm{mod}\, q).$$

В заключение, если $q \mid r$, редукция оказывается невозможной. Это условие делает возможной атаку Блайхенбахера [41] на схему цифровой подписи Эль-Гамаля (см. раздел 10.4.7.1). Однако, поскольку атака Блайхенбахера возможна лишь при злонамеренном выборе параметров открытого ключа, при случайном выборе открытого ключа вероятность события $q \mid r$ равна $1/q$ и является пренебрежимо малой. Следовательно, не стоит опасаться, что Злоумышленник успешно подделает подписи $(M, \xi q, H(M, \xi q), s)$ при некотором целом числе ξ, поскольку это событие имеет пренебрежимо малую вероятность. Таким образом, с огромной вероятностью числа r и q являются взаимно простыми, и Саймон может вычислить значение $x(\mathrm{mod}\, q)$ по формуле

$$x = \frac{e - \ell s}{r} (\mathrm{mod}\, q).$$

Напомним, что число $(p - 1)/q$ не имеет больших простых множителей, т.е. величина $x(\mathrm{mod}\, p - 1)$ вычисляется достаточно легко.

Поскольку числа r, e и e' содержатся в двух списках Саймона, а числа s и s' являются результатом вычислений, выполненных Злоумышленником, Саймон действительно может применить описанный выше метод для вычисления дискретного логарифма числа y с базой g по модулю p. В этом методе Саймон использует Злоумышленника как "черный ящик": его не интересует технология, которую применяет Злоумышленник — ему нужен лишь результат.

Результат редукции

Итак, получены следующие результаты редукции.

1. Преимущество, с которым Саймон вычисляет дискретный логарифм, равно

$$\mathrm{Adv}'(k) \approx \frac{1}{\sqrt{q_H}},$$

поскольку число q_H полиномиально ограничено, а величина $\mathrm{Adv}'(k)$ является значимой.

2. Время работы Саймона оценивается как

$$t' \approx \frac{2\,(t + q_H)}{\mathrm{Adv}(k)},$$

где число t обозначает время, потраченное Злоумышленником на подделку подписи. Эффективность этого алгоритма редукции обсуждается в разделе 16.3.2.3.

Теоретической основой редукционного доказательства, использующего модель случайного оракула, является **лемма о ветвлении** (forking lemma) [235].

Примечание 16.1. *Метод ветвления работоспособен, поскольку Саймон-имитатор повторно генерирует ответы случайного оракула, так что одному и тому же множеству вопросов, заданных Злоумышленником, соответствуют два совершенно разных множества ответов. Это выглядит так, будто Злоумышленник слишком глуп и не способен понять, что на один и тот же вопрос он получает два разных ответа. В то же время Злоумышленник достаточно умен, чтобы подделать цифровую подпись. Следует иметь в виду, что Злоумышленник — это вероятностный алгоритм, единственным предназначением которого является создание корректных поддельных подписей, если его окружение является корректным, а ответы случайного оракула имеют* правильное распределение. *Мы не должны считать, что вероятностный алгоритм имеет дополнительные функциональные возможности, т.е. обладает человеческим сознанием и понимает, когда его дурачат. Фактически, возвращая Злоумышленнику правильно распределенные ответы, Саймон вовсе* не дурачит *его.* □

16.3.2.2 Стойкость к атаке на основе адаптивно подобранных сообщений

Рассмотрим атаку на основе адаптивно подобранных сообщений.

Метод редукции в этом случае остается практически неизменным. Однако теперь Злоумышленнику, кроме запросов случайному оракулу, разрешается создавать q_s подписанных сообщений. Следовательно, Саймон-имитатор должен, помимо ответов на запросы случайному оракулу, реагировать на подписанные запросы, причем эти ответы Злоумышленник может подвергать верификации с помощью алгоритма Verify_{pk}. Саймон должен делать это, даже несмотря на то, что у него нет ключа подписи. Именно эту часть информации он стремится получить с помощью Злоумышленника! Процедуру подписи Саймон имитирует.

Таким образом, нам достаточно показать, что в рамках модели ROM Саймон действительно удовлетворяет подписанные запросы Злоумышленника с высоким качеством.

Поскольку в алгоритме подписи используется функция хэширования, моделируемая случайным оракулом, для каждого запроса M, подписанного Злоумышленником, Саймон выбирает новый элемент $r < p$ и создает запрос случайному оракулу (M, r) от имени Злоумышленника, а затем возвращает как ответ случайного оракула, так и подписанный ответ Злоумышленнику. Создание нового числа r для каждого нового запроса точно соответствует процедуре подписи. Саймон никогда не должен повторно использовать ни одно число r.

Посмотрим, что именно должен сделать Саймон. Для подписанного запроса M Саймон генерирует случайные числа u и v, которые меньше $p - 1$, и вычисляет следующие величины.

$$r \leftarrow g^u y^v (\mathrm{mod}\, p);$$
$$s \leftarrow -rv^{-1} (\mathrm{mod}\, p - 1);$$
$$\ell \leftarrow -ruv^{-1} (\mathrm{mod}\, p - 1).$$

Саймон возвращает число ℓ в качестве ответа случайного оракула на запрос (M, r) и тройку (r, e, s) — в качестве подписи сообщения M (т.е. подписанный ответ на подписанный запрос M). Читатели могут убедиться, что возвращенная подпись действительно является корректной. Этот алгоритм имитации подписи точно совпадает с алгоритмом экзистенциальной подделки, описанным в разделе 10.4.7.2.

В рамках модели случайного оракула имитация подписи имеет точно такое же распределение, как и подпись, созданная алгоритмом, которая применяется случайным оракулом вместо функции хэширования H. Вот почему Злоумышленник ничего не подозревает. Итак, "имитация обучения", обеспечиваемая Саймоном (см. рис. 16.1), имеет высокое качество, и Злоумышленник должен быть удовле-

творен как подписанными ответами, так и ответами случайного оракула. Он должен проявить все свои способности, а метод редукции приведет нас к требуемому противоречию.

Подведем итоги.

Теорема 16.1. *Пусть* $(\mathsf{Gen}(1^k), \mathsf{Sign}, \mathsf{Verify})$ *— одна из тройных схем цифровой подписи Эль-Гамаля, в которой простое число* p *удовлетворяет следующему условию: существует* k*-битовое простое число* q*, являющееся делителем числа* $p-1$*, причем число* $(p-1)/q$ *не имеет больших простых множителей. Если алгоритм атаки на основе адаптивно подобранных сообщений может взломать схему за время* $t(k)$ *с преимуществом* $\mathrm{Adv}(k)$*, то задачу о вычислении дискретного логарифма по модулю* p *можно решить за время* $t'(k)$ *с преимуществом* $\mathrm{Adv}'(k)$*, где*

$$t'(k) \approx \frac{2(t(k) + q_H\tau) + O_B(q_s k^3)}{\mathrm{Adv}(k)},$$

$$Adv'(k) \approx \frac{1}{\sqrt{q_H}}.$$

Здесь q_s *и* q *— количество подписей и запросов оракулу* H *соответственно, а величина* τ *— время, необходимое оракулу* H *для ответа на запрос.* □

Величина k^3 представляет собой количество битовых операций, необходимых для возведения в степень по k-битовому модулю (эта оценка была выведена в разделе 4.3.2.6).

16.3.2.3 Выводы

- Вновь продемонстрирована мощь модели случайного оракула при доказательстве стойкости. Исследование стойкости тройной схемы цифровой подписи Эль-Гамаля показало следующее: если соответствующий алгоритм является действительно случайной функцией, то проще всего подделать подпись, вычислив дискретный логарифм, а затем повторяя действия автора сообщения. Этот результат аналогичен выводам, полученным при исследовании битовой стойкости в главе 9.
- Итак, доказательство стойкости, основанное на модели случайного оракула, демонстрирует, что наиболее уязвимым местом реальной схемы цифровой подписи является функция хэширования, если, коненчно, атакующий не придет к выводу, что атаковать функцию хэширования труднее, чем вычислить дискретный логарифм. Следовательно, основное внимание при разработке схемы цифровой подписи нужно уделить функции хэширования.
- Показано, что преимущество, с которым Саймон вычисляет дискретный логарифм, равно $\frac{1}{\sqrt{q_H}}$, где q_H — количество запросов случайному оракулу H,

которые должен послать Злоумышленник. Для того чтобы Саймон мог вычислять дискретный логарифм с постоянным преимуществом, процесс редукции необходимо выполнить $\sqrt{q_H}$ раз. Это увеличивает время работы Саймона до величины

$$\sqrt{q_H}\frac{2\left(t+q_H\tau\right)+O_B\left(q_s\left(\log p\right)^3\right)}{\mathrm{Adv}}.$$

Если функция хэширования вычисляется эффективно, разумно предложить фальсификатору вычислить ее 2^{50} раз (см. раздел 15.2.5). Следовательно, в редукционном доказательстве Злоумышленнику разрешается сделать 2^{50} запросов случайному оракулу. Иначе говоря, следует положить $q_H = 2^{50}$. В этом случае мы получим следующую оценку времени, которое понадобится Саймону для вычисления дискретного логарифма.

$$O\left(\frac{2^{75}}{\mathrm{Adv}}\right).$$

Эта оценка означает, что процесс редукции не очень эффективен. Следовательно, полученное противоречие теряет смысл, если простое число p состоит из 1024 бит, а преимущество Adv мало. Однако, если простое число p состоит из 2048 бит, противоречие становится существенным.

Несмотря на то что процесс редукции не имеет идеальной эффективности, метод ветвящейся редукции, предложенный Пойнтшевалем и Штерном, стал первым редукционным доказательством стойкости тройной схемы цифровой подписи Эль-Гамаля.

- Ирония заключается в том, что доказательство неуязвимости против атаки на основе адаптивно подобранных сообщений, представляющей собой наиболее сильное понятие стойкости схем цифровой подписи, основано на предположении, что схема уязвима для экзистенциальной подделки. Однако эта ситуация отличается от доказательства стойкости схемы RSA-OAEP, изложенного в разделе 15.2.4, где в качестве открытого показателя степени алгоритма шифрования RSA, мы предложили использовать число три. Экзистенциальная уязвимость схемы цифровой подписи, использующей однонаправленные функции с секретом, не является существенной слабостью.
- Несмотря на то что стандарт цифровой схемы DSS не является тройной схемой (функция хэширования получает на вход только битовую строку, а не сообщение и фиксатор), ее неуязвимость несложно доказать с помощью модели случайного оракула. Для этого достаточно предположить, что Саймон способен документировать все запросы случайному оракулу и запросы на подпись, связанные с определенным ключом. В этом случае на повторные

запросы можно возвращать старые ответы. Возможно, для доказательства стойкости схемы DSS ее следует представить в виде тройной схемы Эль-Гамаля, хэшировав фиксирующее значение.

- Пойнтшеваль и Штерн [235] предложили доказательство стойкости схемы цифровой подписи Фиата (Fiat) и Шамира (Shamir) [109]. В доказательстве использовался тот факт, что схема Фиата и Шамира, по существу, является тройной. Эта схема цифровой подписи представляет собой модификацию схемы идентификации с нулевым разглашением, описанной в следующей главе.

16.3.3 Метод "тяжелых строк"

Существует еще один метод доказательства неуязвимости схем цифровой подписи Эль-Гамаля. Он называется **методом "тяжелых строк"** (heavy row technique) и предложен Фейджем (Feige), Фиатом и Шамиром [106] для доказательства стойкости схемы идентификации с нулевым разглашением Фиата и Шамира [109]. (Свойства этого протокола с нулевым разглашением будут изучены в разделе 18.2.2.) Поскольку этот протокол идентификации легко преобразовать в тройную схему Фиата–Шамира (хотя и не Эль-Гамаля), метод "тяжелых строк" непосредственно распространяется на тройные схемы Эль-Гамаля. Этот факт доказан в работе [222]. Рассмотрим краткое описание метода "тяжелых строк" для доказательства стойкости тройных схем цифровой подписи из семейства Эль-Гамаля.

Предположим, что Злоумышленник фальсифицирует подпись с преимуществом Adv. Тогда количество вызовов Злоумышленника, которые должен сделать Саймон, прямо пропорционально величине $1/\,\mathrm{Adv}$ (оно равно $3/\,\mathrm{Adv}$).

Представим себе гигантскую бинарную матрицу H, состоящую из q строк и q столбцов. Строки соответствуют всем возможным случайным выборам первого элемента тройки в схеме цифровой подписи Эль-Гамаля, а столбцы — второго элемента. Элемент h_{ij} матрицы H равен 1, если тройка (i, j, s) является корректной подписью, и 0 — в противном случае. Строка называется "тяжелой", если она содержит по крайней мере две единицы.

Лемма 16.1 (Лемма о "тяжелых строках"). *Вероятность появления единицы в матрице* H *и в "тяжелых" строках равна по крайней мере* $1/2$. □

Эта лемма верна, поскольку "тяжелые" строки содержат больше единиц, чем остальные.

Поскольку Злоумышленник может фальсифицировать подпись в тройной схеме с преимуществом Adv, в матрице H есть $\mathrm{Adv} \cdot q^2$ единиц. Выполняя свой алгоритм $1/\,\mathrm{Adv}$ раз, Злоумышленник должен создать корректную подделку (i, j, s). По лемме о "тяжелых строках" вероятность того, что i-я строка окажется "тяжелой", равна по крайней мере $1/2$. Теперь, выполняя свой алгоритм еще $2/\,\mathrm{Adv}$ раз

при фиксированном числе i, Злоумышленник успешно создает еще одну корректную подпись (i, j', s'), где $j' \neq j$.

Как мы уже знаем, эти подписи позволяют вычислить дискретный логарифм, т.е. приводят к противоречию.

Представленное выше описание метода "тяжелых строк" является интуитивным. Мы не стали упоминать о парадоксе дней рождения, который может привести к увеличению вероятности. Точное описание метода содержится в работе [222].

16.4 Прикладные варианты схем цифровой подписи RSA и Рабина

Функции RSA и Рабина являются однонаправленными перестановками с секретом (OWTP). В результате учебные варианты этих схем цифровой подписи (см. разделы 10.4.2 и 10.4.4) являются детерминированными алгоритмами. Это означает, что подпись сообщения M однозначно определяется заданными ключом (sk, pk) и сообщением M.

В криптографии детерминированность нежелательна. В учебной схеме цифровой подписи Рабина детерминированность открывает возможность для разрушительной атаки, продемонстрированной в разделе 10.4.5. Атака на основе адаптивно подобранных сообщений позволяет Злоумышленнику вычислить два разных квадратных корня выбранного сообщения и разложить модуль на множители. Следовательно, прикладные варианты схем RSA и Рабина должны быть вероятностными.

16.4.1 Подпись с помощью рандомизированного заполнения

Вероятностные варианты схем RSA и Рабина были разработаны Белларе и Роджуэем [26]. Они назвали свой метод **вероятностной схемой цифровой подписи** (probabilistic signature scheme — PSS). Эта схема представляет собой модификацию функций RSA (и Рабина) с помощью рандомизированного заполнения. Для простоты рассмотрим лишь вариант схемы Рабина.

Как и схема заполнения OAEP (см. рис. 15.1), схема заполнения PSS основана на функциях хэширования. В схеме RSA-OAEP процедура шифрования представляет собой преобразование, использующее однонаправленную функцию RSA. В схеме RSA-PSS процедура подписания — это преобразование, использующее секрет функции RSA, поскольку автор подписи владеет закрытым ключом.

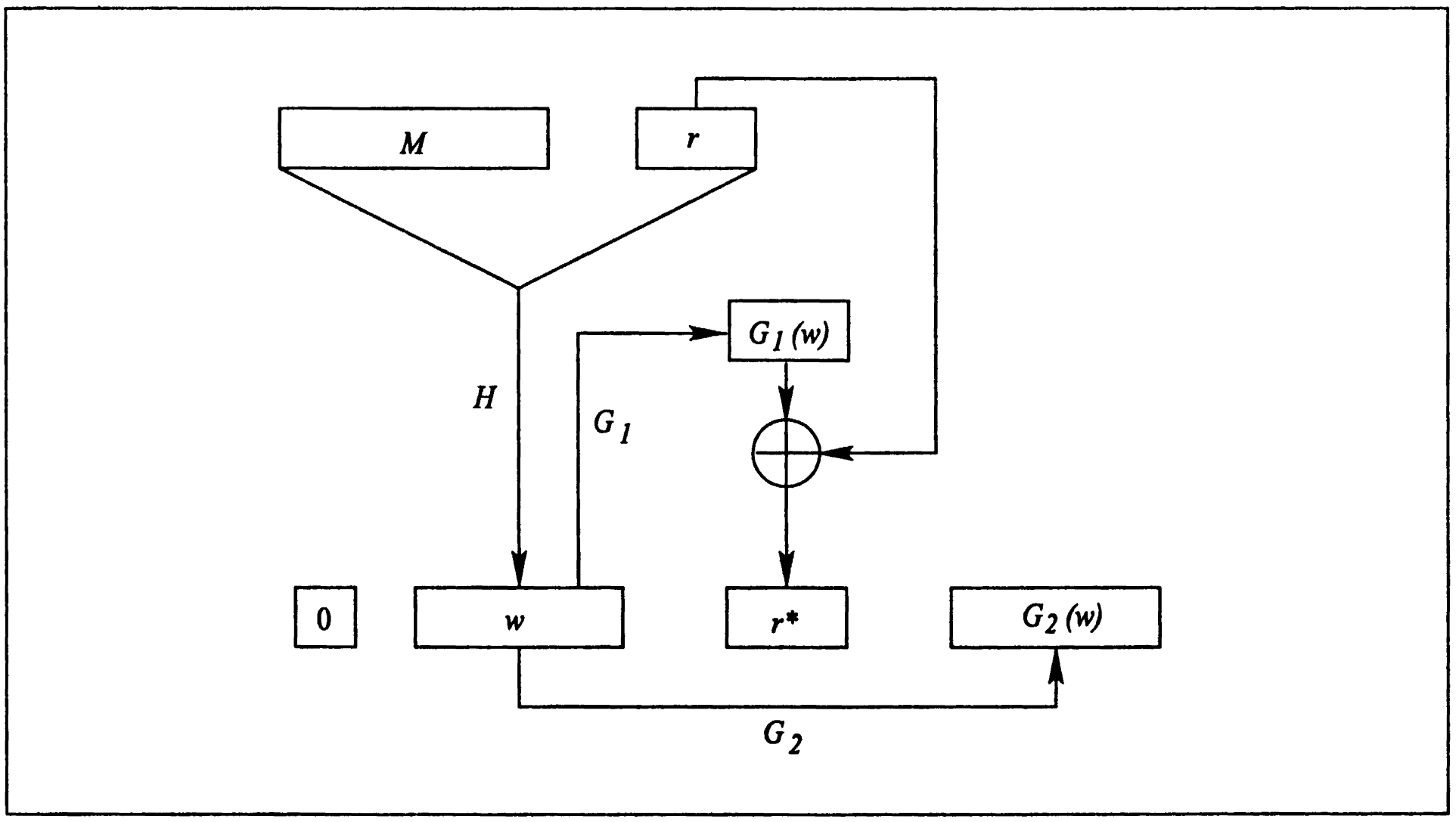


Рис. 16.3. Заполнение PSS

16.4.2 Вероятностная схема цифровой подписи — PSS

Рассмотрим алгоритм для функции RSA. Для схемы Рабина рассуждения проводятся аналогично.

Процедура заполнения по схеме PSS продемонстрирована на рис. 16.3, а схема цифровой подписи описывается алгоритмом 16.1.

Алгоритмы подписи и верификации используют две функции хэширования. Первая функция, $H : \{0,1\}^* \mapsto \{0,1\}^{k_1}$, называется компрессором (compressor), а вторая функция, $G : \{0,1\}^{k_1} \mapsto \{0,1\}^{k-k_1-1}$ — генератором (generator). При анализе стойкости эти функции хэширования имитируют случайных оракулов.

Какую роль играет ведущий нуль? Зная длины функций хэширования и случайного ввода, можно определить, что в результате заполнения возникают $k-1$ бит. Итак, добавление ведущего нуля приводит к созданию k-битовой строки, которую можно интерпретировать как целое число, которое меньше числа N. Это необходимо для правильного возведения в степень по модулю. Другим способом, гарантирующим, что в результате заполнения длина строки будет меньше числа N, является заполнение k-битовой строки с последующей проверкой. Этот метод включен в описание алгоритма заполнения RSA-OAEP (алгоритм 10.6), который незначительно отличается от исходного вариата, описанного в работе [24].

16.4.2.1 Доказательство стойкости

Формальное доказательство стойкости схемы RSA-PSS основано на методе редукции, использующем модель случайного оракула [26]. Оно снова сводится

Алгоритм 16.1. Вероятностная схема цифровой подписи (PSS)

Параметры ключа

Пусть $(N, e, d, G, H, k_0, k_1) \leftarrow_U Gen(1^k)$, где тройка (N, e, d) — параметры ключа RSA, причем пара (N, e) — открытый компонент, а число $d = e^{-1} (\mathrm{mod}\, \phi(N))$ — закрытый компонент; $k = |N| = k_0 + k_1$ такие, что числа 2^{-k_0} и 2^{-k_1} являются пренебрежимо малыми; G и H — функции хэширования, удовлетворяющие условиям

$$G : \{0,1\}^{k_1} \mapsto \{0,1\}^{k-k_1-1}, \quad H : \{0,1\}^* \mapsto \{0,1\}^{k_1}.$$

(* Битовая строка, являющаяся результатом функции G, разбивается на две подстроки — строку G_1, состоящую из k_0 старших битов, и строку G_2, состоящую из остальных $k - k_1 - k_0 - 1$ бит. *)

Генерация подписи

$\mathsf{SignPSS}(M, d, N) =$

$r \leftarrow_U \{0,1\}^{k_0}; w \leftarrow H(M \parallel r); r^* \leftarrow G_1(w) \oplus r;$

$y \leftarrow 0 \parallel w \parallel r^* \parallel G_2(w);$

$\mathsf{return}(y^d (\mathrm{mod}\, N))$.

Верификация подписи

$\mathsf{VerifyPSS}(M, U, e, N) = y \leftarrow U^e (\mathrm{mod}\, N);$

Представить строку y в виде $b \parallel w \parallel r^* \parallel \gamma$.

(* Иначе говоря, b — первый бит строки y, w — следующие k_1 бит, r^* — следующие k_0 бит, γ — остальные биты. *)

$r \leftarrow r^* \oplus G_1(w);$

if $(H(M \parallel r) = w \wedge G_2(w) = \gamma \wedge b = 0)$ return(True)

else return(False).

к противоречию: успешная подделка приводит к инвертированию функции RSA, т.е. к решению трудноразрешимой задачи. Процесс редукции очень похож на процесс редукции алгоритма заполнения RSA в схеме шифрования (например, см. раздел 15.2).

В частности, редукция в доказательстве стойкости схемы RSA-PSS также представляет собой преобразование процесса фальсификации в частичное инвертирование функции RSA (см. раздел 15.2.3.4, в котором атака IND-CCA2 приводит к раскрытию числа s^*, являющегося корнем e-й степени зашифрованного оклика c^*). Несмотря на это доказать стойкость схемы цифровой подписи легче, чем доказать неуязвимость схемы шифрования: частичное инвертирование функции RSA может непосредственно привести к ее полному инвертированию без повторного выполнения алгоритма Злоумышленника, как к схеме шифрования. Это возмож-

но благодаря вычислительной природе процесса фальсификации подписи: при успешной фальсификации Злоумышленник должен передать Саймону пару сообщение-подпись, которую можно верифицировать с помощью однонаправленной функции (в частности, с помощью функции RSA). В противоположность этому при успешной атаке IND-CCA2 Злоумышленник предоставляет Саймону намного меньше информации, просто угадывая один бит. По этой причине Саймон не имеет ни одной однонаправленной функции, чтобы установить связь между угаданным исходным текстом и зашифрованным окликом. В результате инверсия оказывается частичной. Итак, при шифровании редукция означает повторное выполнение алгоритма Злоумышленника путем сдвига позиции частичной инверсии, что приводит к полной инверсии функции.

Непосредственным итогом полной инверсии является эффективная редукция: преимущество Adv, с которым Злоумышленник фальсифицирует подпись, почти точно преобразуется в преимущество Саймона Adv', т.е. $\mathrm{Adv}' \approx \mathrm{Adv}$. Белларе и Роджуэй назвали результат такой редукции **точной стойкостью** (exact security) схемы цифровой подписи, основанной на алгоритме заполнения RSA.

Поскольку доказательство стойкости схемы цифровой подписи RSA-PSS довольно сложно и очень похоже на доказательство стойкости схемы шифрования RSA-OAEP, мы не приводим его. Заинтересованные читатели могут найти его полное описание в работе [26].

16.4.3 Схема PSS-R с восстановлением сообщения

Поскольку схема шифрования RSA-OAEP позволяет владельцу закрытого ключа восстановить зашифрованное сообщение, можно представить себе противоположную ситуацию: схема цифровой подписи, основанная на заполнении с возможностью восстановления сообщения, также может разрешить *любому* лицу, владеющему корректным открытым ключом, восстановить подписанное сообщение. Именно этим свойством обладает схема PSS-R (Probabilistic Signature Scheme *with Message Recovery*). Белларе и Роджуэй распространили эту схему заполнения на функции RSA и Рабина [26].

Мы опишем вариант исходной схемы заполнения PSS-R, разработанной Белларе и Роджуэем. Этот вариант предложен Короном (Coron) и его соавторами [83]. Мы выбрали его, поскольку авторы доказали не только стойкость для ситуации, когда подпись создана с помощью секрета функции RSA, но и стойкость шифрования, когда зашифрованный текст создан с помощью однонаправленной части функции RSA. Таким образом, описанная схема цифровой подписи является стойкой к атаке на основе адаптивно подобранных сообщений, а соответствующая схема шифрования — стойкой к атаке IND-CCA2.

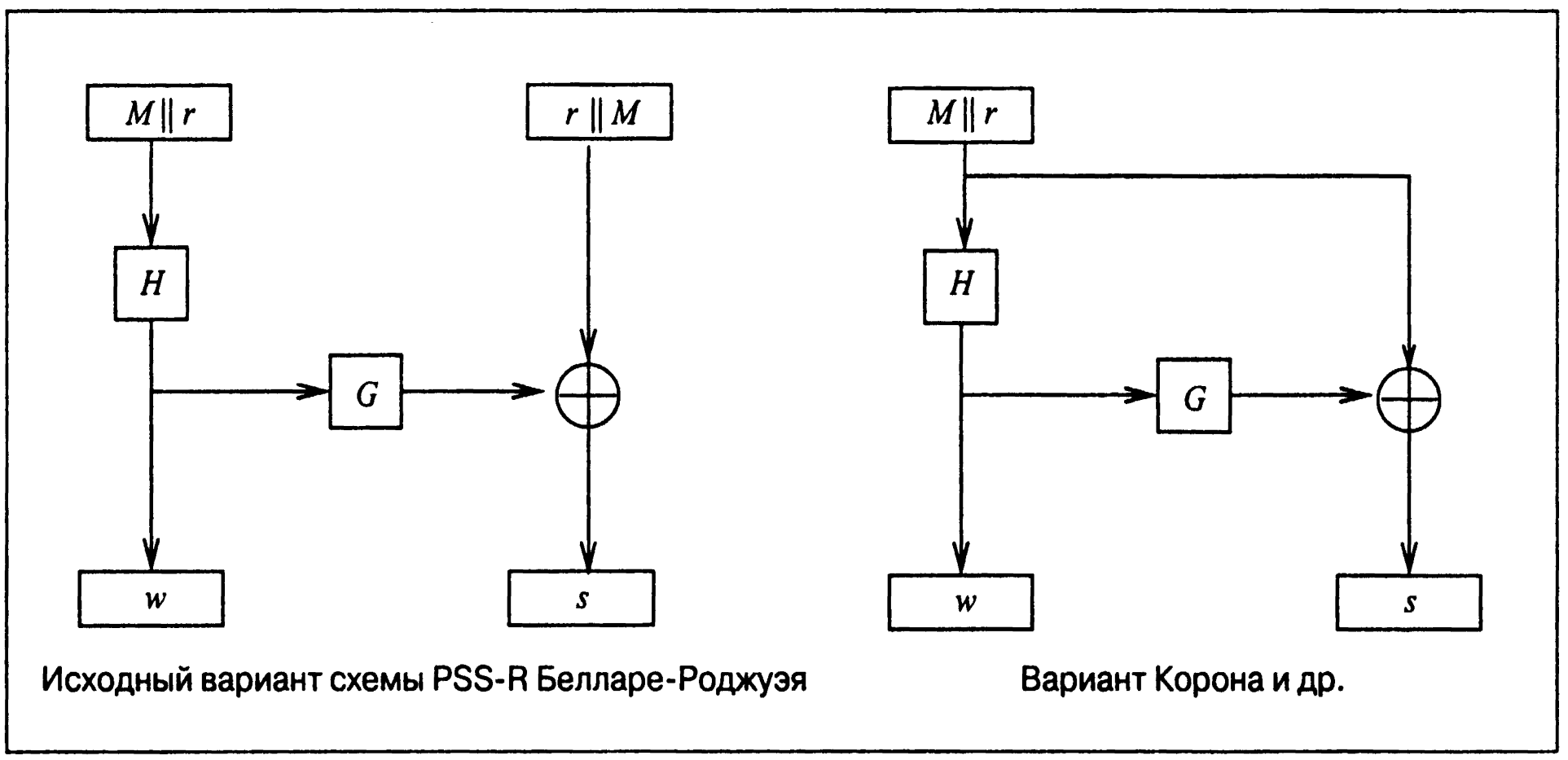


Рис. 16.4. Заполнение PSS-R

16.4.4 Универсальная схема заполнения PSS-R для подписи и шифрования

На рис. 16.4 показаны два вида заполнения PSS-R: слева изображена схема Белларе–Роджуэя [26], а справа — схема Корона и его соавторов [83]. Универсальная схема заполнения для подписи и шифрования описана в виде алгоритма 16.4.4.1.

В универсальной схеме заполнения RSA процедура подписания и шифрования называется PSS-R-Padding. Она получает на вход сообщение $M \in \{0,1\}^{k-k_1-k_0}$, а также показатель степени и модуль функции RSA. Показатель степени функции RSA играет роль параметра d при генерации подписи и параметра e — при шифровании. Отметим, что в отличие от схемы подписи PSS, в которой сообщение могло иметь неограниченную длину, теперь длина сообщения равна $k - k_1 - - k_0$. Процедура верификации подписи и расшифровки с проверкой целостности данных называется PSS-R-UnPadding. На ее вход поступает число $U < N$ и компоненты ключа RSA, а результатом ее работы является элемент множества $\{\mathsf{True}, \mathsf{False}\} \cup \{0,1\}^{k-k_1-k_0}$. Если первым компонентом вывода является значение True, остальная битовая строка представляет собой восстановленное сообщение, в противном случае остальная часть — это нулевая строка NULL.

16.4.4.1 Доказательство стойкости

Доказательство стойкости схемы шифрования и цифровой подписи RSA-PSS-R имеет много общего с доказательствами стойкости как схемы шифрования RSA-OAEP (первая часть), так и схемы цифровой подписи RSA-PSS (вторая часть). Следовательно, повторять его не имеет смысла. Детали можно найти в работе [83].

Алгоритм 16.2. Универсальная схема заполнения RSA для подписи и шифрования

Параметры ключа

Пусть $(N, e, d, G, H, k_0, k_1) \leftarrow_U \mathsf{Gen}(1^k)$, где тройка (N, e, d) — компоненты ключа RSA, причем пара (N, e) — открытый компонент, а число $d = e^{-1} (\mathrm{mod}\, \phi(N))$ — закрытый компонент; $k = |N| = k_0 + k_1$ такие, что числа 2^{-k_0} и 2^{-k_1} являются пренебрежимо малыми; G и H — функции хэширования, удовлетворяющие условиям

$$G : \{0,1\}^{k_1} \mapsto \{0,1\}^{k-k_1-1}, \quad H : \{0,1\}^{k-k_1-1} \mapsto \{0,1\}^{k_1}.$$

Генерация подписи

$\mathsf{PSS\text{-}R\text{-}Padding}(M, x, N) =$

1. $r \leftarrow_U \{0,1\}^{k_0}; w \leftarrow H(M \parallel r); s \leftarrow G(w) \oplus (M \parallel r); y \leftarrow (w \parallel s);$
2. if $(y \geqslant N)$ go to 1;
3. return$(y^x (\mathrm{mod}\, N))$.

Верификация подписи

$\mathsf{PSS\text{-}R\text{-}UnPadding}(U, x, N) = y \leftarrow U^x (\mathrm{mod}\, N);$

Представить строку y в виде $w \parallel s$.
(* Иначе говоря, w — первые k_1 бит, а s — остальные $k - k_1$ бит. *)
Представить строку $G(w) \oplus s$ в виде $M \parallel r$.
(* Иначе говоря, M — первые $k - k_1 - k_0$ бит, а r — остальные k_0 бит. *)
if $(H(M \parallel r) = w)$ return $(\mathsf{True} \parallel M)$
else return$(\mathsf{False} \parallel \mathsf{Null})$.

16.4.4.2 Выводы

- Для того чтобы длина результата заполнения в схеме PSS-R-Padding была меньше числа N, применяется метод проб и ошибок. Вероятность ошибки при повторении проверки i раз равна 2^{-i}. Как альтернативу в качестве заполнителя можно применить ведущий нуль.
- При шифровании по схеме PSS-R-Padding проверка целостности данных сводится к проверке значения функции хэширования. Этот метод отличается от схемы заполнения OAEP, в которой проверке подвергалась строка избыточных нулей.
- Анализ стойкости схемы шифрования PSS-R-Padding к атаке IND-CCA2, по существу, совпадает с доказательством стойкости схема RSA-OAEP. Он осуществляется с помощью редукции атаки к частичному инвертированию функции RSA, когда величина w известна. Иначе говоря, если Злоумышленник может взломать систему с преимуществом Adv, то в ходе атакующей игры с Саймоном-имитатором Злоумышленник должен посылать запросы

к оракулу G приблизительно с тем же преимуществом. Поскольку реализация атакующего алгоритма приводит к частичному инвертированию, чтобы выполнить полное инвертирование, его необходимо провести несколько раз. Как показано в разделе 15.2.4, для того, чтобы получить противоречие, алгоритм Злоумышленника нельзя выполнять больше двух раз (т.е. сложность редукции представляет собой полиномиальный алгоритм степени 2).

- Даже если алгоритм Злоумышленника выполняется всего два раза, редукция оказывается не очень точной. Последствия этого факта указаны в разделе 15.2.5. Для того чтобы получить необходимое противоречие, модуль функции RSA в схеме RSA-PSS-R должен состоять как минимум из 2048 бит.
- Для того чтобы выполнялось условие $|w| > \frac{|N|}{2}$, алгоритм Злоумышленника необходимо реализовать не меньше двух раз. Следовательно, $|M|\,|r| \leqslant \frac{|N|}{2}$. Итак, схема заполнения RSA-PSS-R имеет довольно узкую полосу пропускания для восстановления сообщений: размер восстанавливаемого сообщения не должен превышать половину размера модуля. Как правило, выбирая параметры $k = |N| = 2048$ и $k_0 = 160$, можно получить максимум $|M| = \frac{|N|}{2} - k_0 = 1024 - 160 = 862$, т.е. модуль $|M|$ составляет приблизительно 42% от величины $|N|$.
- По причинам, указанным в разделе 16.4.2.1, процесс редукции в доказательстве стойкости схемы цифровой подписи RSA-PSS-R к атаке на основе адаптивно подобранных сообщений является неполным. Это объясняется тем, что успешная фальсификация подписи может привести к полному инвертированию функции RSA. Таким образом, в отличие от доказательства, рассмотренного в предыдущем абзаце, для стойкости схемы условие $|w| > \frac{|N|}{2}$ не является обязательным. Достаточно установить параметры k_0 и k_1 так, чтобы величины 2^{-k_0} и 2^{-k_1} стали пренебрежимо малыми. На практике часто выбирают $k_0 = k_1 = 160$. Таким образом, величина $|M| = k - k_1 - k_0$ может быть довольно большой. В наиболее типичном случае, когда $k = |N| = 2048$ и $k_0 = k_1 = 160$, выполняется условие $|M| = 2048 - 320 = 1728$, т.е. величина $|M|$ составляет 84% от величины $|N|$.

16.5 Зашифрованная подпись

Для того чтобы предотвратить фальсификацию и обеспечить секретность письма, автор должен подписать его и запечатать в конверт и лишь затем отправить его адресату. Как правило, в практике обмена секретными сообщениями цифровая подпись и шифрование данных выполняются отдельно и непосредственно: отправитель подписывает и шифрует сообщение, а получатель расшифровывает его и верифицирует подпись.

На подпись и шифрование затрачиваются машинные циклы, а дополнительные биты удлиняют сообщение. Стоимость криптографических операций обычно выражается через относительное увеличение размера сообщения и затраты времени на вычисления, выполняемые отправителем и получателем. В прямой процедуре шифрования-подписания стоимость доставки аутентичного секретного сообщения зависит от стоимости цифровой подписи и шифрования. Как правило, такой способ оказывается неэкономным.

Зашифрованная подпись (signcryption) — это конструкция с открытым ключом, позволяющая эффективно реализовать процедуру подписи и шифрования сообщений. Она обеспечивает три необходимые функциональные возможности: секретность, аутентичность и невозможность отречения от авторства. Поскольку на практике эти возможности довольно часто необходимы одновременно, Жень (Zheng) предложил понятие зашифрованной подписи [309], которое эффективным образом объединяет в себе процедуру подписания и шифрования сообщений.

16.5.1 Схема зашифрованной подписи Женя

Жень предложил две очень похожие схемы зашифрованной подписи под названием SCS1 и SCS2 [309]. Они используют две весьма близкие схемы цифровой подписи Эль-Гамаля — SDSS1 и SDSS2 соответственно.

Напомним, что в тройной схеме Эль-Гамаля фиксатор r в цифровой подписи (r, s, e) обычно вычисляется по формуле $r = g^k (\mathrm{mod}\, p)$, где числа g и p представляют собой компоненты открытого ключа, а передача k является целым числом, независящим от величин, использованных в предыдущих подписях. Кроме того, в схеме цифровой подписи Шнорра (алгоритм 10.4), представляющей собой вариант тройной схемы Эль-Гамаля, отправитель не обязан посылать получателю фиксатор. Способ, которым создается цифровая подпись, позволяет получателю восстанавливать фиксатор с помощью формулы $r = g^s y^e (\mathrm{mod}\, p)$.

Итак, если отправитель сообщения (автор подписи) вычисляет фиксатор особым образом, так что его может восстановить только легальный получатель (с помощью своего открытого ключа), то значение фиксатора можно использовать как симметричный ключ, общий для отправителя и получателя, а для обеспечения секретности сообщения — применять симметричное шифрование.

Вот как выглядит приблизительное описание схемы зашифрованной подписи Женя: для обеспечения секретности в качестве симметричного ключа шифрования сообщений используется восстанавливаемое значение фиксатора в тройной схеме цифровой подписи Эль-Гамаля. Это краткое и абстрактное описание можно представить в виде тройки (c, e, s), где c — зашифрованный текст, созданный симметричным алгоритмом шифрования, а (e, s) — второй и третий элемент тройной подписи. Первый элемент тройки Эль-Гамаля (как правило, обозначаемый как r) может восстановить только предназначенный получатель.

Поскольку схемы SCS1 и SCS2 очень похожи, мы опишем только схему SCS1 (см. алгоритм 16.5.1.1). Для простоты изложения в спецификации используются обозначения, принятые в схеме Эль-Гамаля, за исключением того, что вместо символа r используется символ K (значение фиксатора в тройной схеме цифровой подписи Эль-Гамаля), обозначающий симметричный ключ.

Покажем, что система, описанная алгоритмом 16.3, одновременно является криптосистемой и схемой цифровой подписи, т.е. 1) процедура расшифровки, выполняемая Бобом, действительно восстанавливает исходный текст, снабженный зашифрованной подписью Алисы, и 2) Алиса подписывает сообщение.

Для того чтобы доказать первое утверждение, достаточно показать, что Боб может восстановить число $K = g_B^u (\mathrm{mod}\, p)$, зашифрованное Алисой. Процедура расшифровки выглядит следующим образом.

$$K = (g^e y_A)^{sx_B} (\mathrm{mod}\, p) \equiv (g^{x_B})^{se} (g^{x_A x_B})^s \equiv y_B^{s(e+x_A)} \equiv y_B^u (\mathrm{mod}\, p).$$

Итак, действительно, Боб восстанавливает число K, зашифрованное Алисой. Используя число K_1, выделенное из числа K, Боб, конечно, может расшифровать текст c и восстановить сообщение M.

Чтобы доказать второе утверждение, заметим, что после вычисления значения $K = g_B^u (\mathrm{mod}\, p)$, тройка (K_2, e, s) образует тройную подпись Эль-Гамаля на восстановленном сообщении M. Следовательно, алгоритм 16.3 действительно является схемой цифровой подписи.

16.5.1.1 Выводы

- **Эффективность.** Схема SCS1 весьма эффективна как с вычислительной, так и коммуникативной точек зрения. Для того чтобы создать зашифрованную подпись отправитель сообщения выполняет одно возведение в степень, одно хэширование и одно симметричное шифрование. Для того чтобы выполнить обратную операцию, получатель выполняет приблизительно такой же объем вычислений, если выражение $(g^e y_A)^{sx_B}$ можно переписать в виде $g^{esx_B} y_A^{sx_B}$ и применить алгоритм 15.2. С точки зрения ширины полосы пропускания, учитывая, что симметричное шифрование сообщения не приводит к увеличению размера сообщений, зашифрованную подпись можно уложить в $2|q|$ бит плюс длина сообщения. Эта длина совпадает с шириной пропускания при передаче подписи Эль-Гамаля. Более того, применение симметричного алгоритма шифрования делает схему пригодной для эффективной пересылки основного объема данных (например, с помощью блочного шифра в режиме CBC). По существу, схему SCS1 можно рассматривать как гибридную схему шифрования с открытым ключом (см. раздел 15.4).
- **Стойкость.** Для защиты подписи от подделки Жень предусмотрел вполне разумные средства. Поскольку схема SCS1, по существу, представляет собой

Алгоритм 16.3. Схема зашифрованной подписи Женя SCS1

Установка системных параметров

Доверенный орган выполняет следующие операции.

1. Установить параметры системы (p, q, g, H).
   (* Параметры из схемы цифровой подписи Шнорра (алгоритм 10.4). *)
2. Выбрать симметричный алгоритм шифрования $\mathcal{E}$.
   (* Например, в качестве алгоритма $\mathcal{E}$ можно выбрать AES. *)

Параметры $(p, q, g, H, \mathcal{E})$ распространяются между всеми системными пользователями.

Генерация открытого/закрытого ключа пользователя

Пользователь Алиса генерирует случайное число $x_A \leftarrow_U \mathbb{Z}_q$ и вычисляет величину

$$y \leftarrow g^{x_A} (\mathrm{mod}\, p).$$

Параметрами открытого ключа Алисы являются числа $(p, q, g, y_A, H, \mathcal{E})$. Ее закрытым ключом является число x_A.

Зашифрованная подпись

Для того чтобы послать Бобу сообщение M, снабженное зашифрованной подписью, Алиса должна выполнить следующие операции.

1. Сгенерировать случайное целое число u из отрезка $[1, q]$, вычислить значение $K \leftarrow y_B^u (\mathrm{mod}\, p)$ и разбить его на две части, K_1 и K_2, соответствующей длины.
2. Найти число $e \leftarrow H(K_2, M)$.
3. Вычислить значение $s \leftarrow u(e + x_A)^{-1} (\mathrm{mod}\, q)$.
4. Найти число $c \leftarrow \mathcal{E}_{K_1}(M)$.
5. Послать Бобу текст (c, e, s), снабженный зашифрованной подписью.

Расшифровка зашифрованной подписи

Получив от Алисы текст (c, e, s), снабженный зашифрованной подписью, Боб должен выполнить следующие операции.

1. Восстановить число K по параметрам e, s, g, p, y_A и x_B : $K \leftarrow (g^e y_A)^{s x_B} (\mathrm{mod}\, p)$.
2. Разбить число K на две части: K_1 и K_2.
3. Найти число $M \leftarrow \mathcal{D}_{K_1}(c)$.
4. Принять число M как корректное сообщение, подписанное Алисой, только если $e = H(K_2, M)$.

тройную схему Эль-Гамаля с восстанавливаемым фиксатором, стойкость схемы к атаке на основе адаптивно подобранных сообщений непосредственно следует из доказательства стойкости тройной схемы Эль-Гамаля, предложенного Пойнтшевалем и Штерном [235] (см. раздел 16.3). Однако Жень не смог провести редукцию атаки IND-CCA2 к решению трудноразрешимой задачи, поскольку в схеме используется симметричный алгоритм шифрования. Возможно, причиной неудачи является следующий факт: только легальный получатель может восстановить фиксатор K при атаке на основе адаптивно подобранных зашифрованных текстов.

- **Невозможность отречься от авторства.** Невозможность отрицать свое авторство представляет собой важное преимущество во многих приложениях, например, в электронной коммерции. Цифровые подписи позволяют обеспечить это свойство, поскольку подпись сообщения можно подвергнуть универсальной верификации. Если две стороны спорят о паре сообщение-подпись, в качестве арбитра можно пригласить третью сторону. Для зашифрованной подписи универсальная верификация невозможна. Поэтому для того, чтобы отправитель сообщения не мог отречься от авторства, необходимо приложить дополнительные усилия. Именно это происходит в схеме зашифрованной подписи Женя. Для верификации тройной подписи в этой схеме необходимо восстановить фиксатор K и применить его для вычисления закрытого ключа. Итак, непосредственно привлечь третью сторону в качестве арбитра невозможно. Жень предложил, чтобы в случае спора между получателем (Бобом) и отправителем (Алисой) Боб провел доказательство с нулевым разглашением, продемонстрировав арбитру, что ему известна подпись Алисы. Соответствующий протокол доказательства с нулевым разглашением Жень не привел. Несмотря на то что такой протокол нетрудно разработать самостоятельно, для этого простую процедуру верификации придется преобразовать в интерактивный протокол. Это является наиболее серьезным дефектом схемы зашифрованной подписи Женя.

16.5.2 Двух зайцев — одним выстрелом: создание зашифрованной подписи с помощью алгоритма RSA

Мэлоун (Malone) и Мао (Mao) предложили схему зашифрованной подписи под названием “двух зайцев — одним выстрелом” (“two birds one stone” — TBOS) [182]. Выбор такого названия будет обоснован позднее. Схема зашифрованной подписи TBOS основана на алгоритме RSA. Авторы схемы привели редукционные доказательства ее сильной стойкости к взлому и подделке подписи. Оба доказательства

основаны на модели случайного оракула и практической невозможности инвертировать функцию RSA.

Схема зашифрованной подписи TBOS очень проста. Она дважды "заворачивает" ("wrap") сообщение при подписании и шифровке с помощью функции RSA: отправитель (т.е. Алиса) сначала подписывает сообщение, "заворачивая" его в секретную часть своей собственной функции RSA, а затем шифрует подпись, "заворачивая" ее в однонаправленную часть функции RSA, принадлежащей получателю (Бобу). Таким образом, обозначив через (N_A, e_A) и (N_A, d_A) параметры открытого и закрытого ключей Алисы соответственно, а через (N_B, e_B) и (N_B, d_B) — параметры открытого и закрытого ключей Боба, сообщение M, подписанное по схеме TBOS, можно представить в следующем виде.

$$\left[M^{d_A} \pmod{N_A}\right]^{d_B} \pmod{N_B}.$$

Хотя идея весьма проста; в учебном варианте алгоритма RSA она совершенно бесполезна, поскольку модуль функции RSA, принадлежащей Алисе, может оказаться больше, чем модуль функции RSA, принадлежащей Бобу. В этом случае "внутренняя обертка" зашифрованного текста, представляющая собой целое число, всегда будет больше, чем модуль, использованный для ее "внешней обертки".

Несмотря на это следует отметить, что в прикладной схеме RSA, как шифрования, так и цифровой подписи, сообщение "заворачивается" только после обработки с помощью рандомизированной схемы заполнения. В таком случае системные пользователи должны применять модули одинакового размера, поскольку как отправители, так и получатели должны согласовывать схему заполнения (padding) и распаковки ("unpadding").

Если системные пользователи используют модули одинакового размера, "двойное заворачивание" работает прекрасно. Если "внутренняя обертка" превосходит модуль "внешней обертки", пользователь просто "отрезает" один бит (например, старший) от "внутренней обертки". Чтобы "заворачивание" стало возможным, оставшееся целое число должно быть меньше модуля "внешней обертки". Напомним, что получатель такого зашифрованного текста проверит целостность данных. На этапе верификации получатель может восстановить отрезанный бит с помощью метода проб и ошибок. Вот в чем заключается идея описанного метода.

Итак, пусть $|N_A| = |N_B| = k$. Обозначим через $\text{Padding}(M, r) \in \{0,1\}^k$ рандомизированное заполнение сообщения M с помощью случайного числа r. Тогда сообщение M, посланное Алисой Бобу в "двойной обертке" и снабженное зашифрованной подписью TBOS, выглядит следующим образом.

$$\left[\text{Padding}\,(M, r)^{d_A} \pmod{N_A}\, d\right]^{d_B} \pmod{N_B}.$$

Описав схему шифрования TBOS, укажем три преимущества, которыми она обладает.

- Схема позволяет создавать компактные зашифрованные тексты: размер текста зашифрованной подписи совпадает с размером текста RSA без подписи, а также с размером подписи RSA без шифрования. По этой причине схема получила название "двух зайцев — одним выстрелом". Это свойство весьма привлекательно во многих электронных коммерческих приложениях, в которых необходимо пересылать через Internet короткие сообщения (например, номер кредитной карточки), соблюдая секретность и гарантируя невозможность отрицания авторства. В этих приложениях схема TBOS позволяет создавать одну короткую шифрограмму. Это не только обеспечивает эффективность, но и упрощает разработку протокола электронной коммерции.
- Невозможность отрицать авторство гарантируется непосредственно: получатель Боб, "развернувший" зашифрованный текст, снабженный зашифрованной подписью, и, возможно, восстановивший "отрезанный" бит, обнаруживает подпись Алисы: $\text{Padding}\,(M,r)^{d_A} \pmod{N_A}$. Любой посредник может верифицировать эту подпись, применяя обычную процедуру.
- Доказательства стойкости схемы TBOS основаны на стойкости прикладных схем заполнения RSA и носят редукционный характер. Несмотря на то что они основаны на модели случайного оракула, редукционные доказательства используют только общеизвестные неразрешимые задачи (задача RSA, определение 8.4 и предположение 8.3 из раздела 8.7).

Докажем теперь, что Боб всегда может правильно расшифровать зашифрованную подпись. Это очевидно, если $N_A < N_B$. Для ситуации, когда $N_A > N_B$ с приблизительной вероятностью $1/2$, получаем следующее.

$$\sigma = \text{Padding}\,(M,r)^{d_A} \pmod{N_A} > N_B.$$

Однако, поскольку $|N_A| = |N_B| = k$,

$$\sigma < N_A < 2^k.$$

Следовательно, отбрасывая старший бит

$$\sigma' \leftarrow \sigma - 2^{k-1},$$

получаем,

$$\sigma' < 2^{k-1} < N_B.$$

Иначе говоря, Боб может правильно восстановить величину σ'. В дальнейшем будем считать, что в ходе верификации Боб сам определяет, следует ему восстанавливать "отрезанный" бит или нет.

Алгоритм 16.4. Двух зайцев — одним выстрелом: схема зашифрованной подписи RSA-TBOS

Параметры ключа

Пусть k — четное положительное целое число. Обозначим параметры открытого и закрытого ключей RSA, принадлежащих отправителю Алисе (соответственно получателю Бобу), как $(N_A, e_A), (N_A, d_A)$ (соответственно $(N_B, e_B), (N_B, d_B)$), причем $|N_A| = |N_B| = k$.

Пусть G и H — две функции хэширования, удовлетворяющие условиям

$$H : \{0,1\}^{n+k_0} \mapsto \{0,1\}^{k_1} \text{ и } G : \{0,1\}^{k_1} \mapsto \{0,1\}^{n+k_0},$$

где $k = n + k_0 + k_1$, а числа 2^{-k_0} и 2^{-k_1} — пренебрежимо малы.

Шифрование подписи

Когда Алиса снабжает сообщение $M \in \{0,1\}^n$ зашифрованной подписью, она выполняет следующие инструкции.

1. $r \leftarrow_U \{0,1\}^{k_0}$.
2. $\omega \leftarrow H(M \parallel r)$.
3. $s \leftarrow G(\omega) \oplus (M \parallel r)$.
4. If $s \parallel \omega > N_A$ goto 1.
5. $c' \leftarrow (s \parallel \omega)^{d_A} \pmod{N_A}$.
6. If $c' > N_B, c' \leftarrow c' - 2^{k-1}$.
7. $c \leftarrow c'^{e_B} \pmod{N_B}$.
8. Послать сообщение c Бобу.

Расшифровка подписи

Когда Боб расшифровывает криптограмму c, полученную от Алисы, он выполняет такие инструкции.

1. $c' \leftarrow c^{d_B} \pmod{N_B}$.
2. If $c' > N_A$ отказать.
3. $\mu \leftarrow c'^{e_A} \pmod{N_A}$.
4. Представить число μ в виде $s \parallel \omega$.
5. $M \parallel r \leftarrow G(\omega) \oplus s$.
6. If $H(M \parallel r) = \omega$, return M.
7. $c' \leftarrow c' + 2^{k-1}$.
8. If $c' > N_A$, отказ.
9. $\mu \leftarrow c'^{e_A} \pmod{N_A}$.
10. Представить число μ в виде $s \parallel \omega$.
11. $M \parallel r \leftarrow G(\omega) \oplus s$.
12. If $\omega \neq H(M \parallel r)$, отказ.
13. return M.

16.5.2.1 Схема RSA-TBOS

Схема RSA-TBOS, предложенная Мэлоун-Ли и Мао [182], использует схему заполнения PSS-R (раздел 16.4.4). Эта схема зашифрованной подписи представлена в алгоритме 16.4.

Выполнение шестого этапа при создании зашифрованной подписи гарантирует условие $c' < N_B$. Если число c' сразу не удовлетворяет этому условию, значит,

$N_A > c' > N_B$. Поскольку числа N_A и N_B состоят из k бит, число c' также состоит из k бит, а значит, присваивание $c' \leftarrow c' - 2^{k-1}$ эквивалентно удалению старшего бита из числа c'. Отсюда следует условие $c' < N_B$.

Обратите внимание на то, что этот шаг вынуждает включить в процедуру расшифровки дополнительный этап. В частности, возможно, придется дважды выполнить операцию $c'^{e_A} \pmod{N_A}$ (два числа c' могут отличаться на величину 2^{k-1}). В качестве альтернативы можно создать схему, в которой на этапе шифрования используется метод проб и ошибок. В этом случае необходимо повторить первые пять этапов шифрования с разными числами r, пока не выполнится условие $c' < N_B$.

Схема RSA-TBOS легко обеспечивает невозможность отрицания авторства. Получатель зашифрованной подписи выполняет процедуру расшифровки до второго этапа, на котором число c' может быть передано третьей стороне для верификации.

Несмотря на то что схема шифрования подписи RSA-TBOS обладает многими преимуществами, необходимо отметить один недостаток, вытекающий из применения схемы заполнения RSA-PSS-R: она имеет довольно узкую полосу пропускания для восстановления сообщений (см. раздел 16.4.4.2).

16.5.2.2 Доказательство стойкости

Мэлоун-Ли и Мао предложили редукционное доказательство сильной стойкости схемы зашифрованной подписи TBOS [182]. К свойствам сильной стойкости относятся стойкость шифрования к атаке IND-CCA2 и невозможность подделать подпись с помощью атаки на основе адаптивно подобранных сообщений.

Это доказательство аналогично доказательству стойкости схемы RSA-OAEP, приведенному в разделе 15.2. Заинтересованный читатель найдет его изложение в работе [182].

Вместо формального изложения доказательства рассмотрим интуитивные рассуждения, обосновывающие стойкость зашифрованной подписи. Стойкость к атаке IND-CCA2 вытекает из стойкости схемы шифрования, основанной на заполнении с помощью алгоритма RSA. Следовательно, достаточно обосновать невозможность подделать подпись в ходе атаки на основе адаптивно подобранных сообщений. Для этого необходимо вернуться к схеме заполнения OAEP.

Напомним редукционное доказательство стойкости схемы RSA-OAEP к атаке IND-CCA2 (раздел 15.2). Если Злоумышленнику неизвестна подлинная процедура шифрования, вероятность создать корректный *зашифрованный текст* пренебрежимо мала, несмотря на алгоритм, находящийся в распоряжении Злоумышленника, и адаптивный характер процедуры создания зашифрованных текстов.

Этот факт можно механически преобразовать в доказательство неуязвимости подписи в рандомизированной схеме заполнения: вероятность того, что Злоумышленник может подделать корректную пару сообщение-подпись, не зная процедуры

подписания (т.е. не зная правильного показателя степени), пренебрежимо мала, даже если атака основана на адаптивно подобранных сообщениях.

Разумеется, это доказательство носит неформальный характер, поскольку в нем не используется "сведение к противоречию". Формальное редукционное доказательство изложено в работе [182].

16.6 Резюме

В главе введено понятие сильной стойкости схем цифровой подписи: невозможность подделки в ходе атаки на основе адаптивно подобранных сообщений. Эта атака является аналогом атаки IND-CCA2 на схемы шифрования с открытым ключом. Основная идея этих атак заключается в том, что Злоумышленник проходит курс обучения криптоанализу. Криптографическая система обладает сильной стойкостью к атаке, если Злоумышленник неоднократно прошел курс обучения криптоанализу (при условии, что сложность его алгоритма и количество тренировок представляют собой полиномиально ограниченные величины).

Затем мы рассмотрели два важных семейства прикладных схем цифровой подписи — тройные схемы Эль-Гамаля и схемы рандомизированного заполнения с помощью однонаправленных перестановок с секретом, например, функций RSA и Рабина.

После этого приведено формальное доказательство сильной стойкости схем цифровой подписи из обоих семейств.

При доказательстве сильной стойкости схем из первого семейства мы изучили метод редукции, использующий модель случайного оракула, в основе которого лежит предположение, что вероятность правильных "ветвящихся ответов на вопросы фальсификатора" не является пренебрежимо малой величиной. Иначе говоря, множеству вопросов, поступающих от фальсификатора, могут соответствовать два множества совершенно разных ответов, имеющих корректное распределение (равномерное). Поскольку фальсификатором, задающим вопросы, является полиномиальный алгоритм, его интересует только корректность распределения, а не содержание ответов. Следовательно, несмотря на то, что на один и тот же вопрос поступают разные ответы, ветвящийся фальсификатор можно использовать для редукции атаки к решению трудноразрешимой задачи — вычислению дискретного логарифма. В качестве альтернативного метода доказательства приведена модель "тяжелых строк". Несмотря на то что оба метода являются строго формальными, редукционные алгоритмы не очень эффективны. Следовательно, доказательство имеет смысл только при больших значениях параметров безопасности.

Схемы цифровой подписи из второго семейства конструируются с помощью последовательной комбинации рандомизированных заполнений путем однонаправленных перестановок с секретом. Редукционное доказательство их стойкости

аналогично редукционному доказательству стойкости схем шифрования с открытым ключом, использующих рандомизированные заполнения с помощью однонаправленных перестановок с секретом, изученных в предыдущей главе. Несмотря на это, успешная атака (фальсификация подписи с помощью атаки на основе адаптивно подобранных сообщений) может привести к полному инвертированию однонаправленной функции. Таким образом, редукционное доказательство стойкости схем цифровой подписи с рандомизированным заполнением является строгим, т.е. способность атакующего алгоритма подделать подпись может быть полностью сведена к инвертированию одной из необратимых функций (т.е. соответствующей однонаправленной функции с секретом). Такое свойство называется точной стойкостью.

В заключительной части главы рассмотрены схемы зашифрованной подписи, представляющие собой эффективные и полезные криптографические примитивы. Аналогично другим схемам шифрования и цифровой подписи, рассмотренным в книге, схемы зашифрованной подписи используют две распространенные криптографические задачи: вычисление дискретного логарифма и факторизацию целого числа.

Упражнения

16.1. Что представляет собой “прикладная” стойкость цифровой подписи?

16.2. Если Злоумышленник — нехороший человек, почему мы должны позволять ему получать подписи на сообщениях по его выбору без всяких ограничений?

16.3. При доказательстве стойкости тройной схемы Эль-Гамаля в лемме о ветвящихся ответах Саймон дважды выполняет алгоритм Злоумышленника и на одно и то же множество запросов случайному оракулу возвращает два разных множества ответов. Следует ли считать, что Саймон таким образом обманывает Злоумышленника?

16.4. Обсудите полезность понятия экзистенциальной подделки тройной подписи Эль-Гамаля при доказательстве стойкости этой схемы.

16.5. Предположим, что алгоритм PSS подписывает одно и то же сообщение дважды. Какова вероятность, что он создаст одну и ту же подпись?

16.6. В упражнении 15.2 мы ввели понятие полосы пропускания схемы шифрования. Аналогичное понятие можно ввести для схемы цифровой подписи. Предположим, что мы выбрали те же самые параметры безопасности, что и в упражнении 15.2. Какова полоса пропускания в универсальной схеме заполнения с помощью алгоритма RSA (алгоритм 16.2) 1) при подписании; 2) при шифровании?

16.7. Почему две полосы пропускания, описанные в предыдущей задаче, отличаются друг от друга?

16.8. Обсудите различия между невозможностью отказа от авторства в схеме цифровой подписи Женя и в схеме зашифрованной подписи TBOS.

16.9. Аргументы, приведенные в пользу стойкости схемы зашифрованной подписи TBOS (в разделе 16.5.2.2), являются убедительными, но не строгими. Почему?

Подсказка: являются ли эти аргументы редукционными?

Глава 17

Формальные методы анализа протоколов аутентификации

17.1 Введение

В главе 11 показано, что протоколы аутентификации и протоколы согласования аутентифицированных ключей (в этой главе обе разновидности часто называются протоколами аутентификации) весьма уязвимы. Причем дефекты этих протоколов могут быть скрыты. Как создать действительно стойкие протоколы аутентификации? Эта тема находится в центре внимания многих исследователей. Некоторые из этих ученых являются криптографами и математиками, другие — специалистами по компьютерным наукам. В среде этих исследователей стало общепризнанным, что для анализа протоколов аутентификации необходимо развивать формальные подходы.

Формальные методы являются естественным продолжением неформальных. Формальными могут быть методологические, механические и логические понятия. Как правило, для моделирования поведения системы с помощью логических и математических средств формальный метод использует язык символов. Иногда формальный метод представляет собой экспертную систему, имитирующую поведение людей и даже пытающуюся моделировать человеческое мышление. Общим для всех этих методов является систематический, а иногда даже исчерпывающий подход к проблеме. Следовательно, формальные методы весьма полезны для анализа сложных систем.

В области формального анализа протоколов аутентификации различаются два подхода. Один из них использует формальные рассуждения о некоторых полезных свойствах, например, стойкости, а другой направлен на поиск нежелательных и опасных дефектов.

В рамках первого подхода анализируемый протокол должен выбираться или разрабатываться весьма тщательно. Анализ должен показать, что выбранный протокол действительно соответствует предъявляемым требованиям, которые также должны быть строго формализованы. Методы формального доказательства часто приспособлены для конкретного протокола и предполагают широкомасштабное

вмешательство человека, хотя в некоторых случаях они имеют более широкое применение. Этот подход развивается двумя школами исследователей: вычислителями и символистами. Вычислители стремятся выразить свойства стойкости с помощью измерения вероятности, а исследование правильности протокола сводится к доказательству определенной теоремы. Довольно часто в доказательствах стойкости используется метод редукции атаки к решению общепринятых трудноразрешимых задач из теории вычислительной сложности (см. главы 14 и 15). Символисты, к которым относятся специалисты по компьютерным наукам, выражают свойства стойкости в виде множества абстрактных символов, допускающих определенные манипуляции, иногда путем применения логических правил, иногда — с помощью механического средства для доказательства теорем. Результатом этих манипуляций должен быть ответ ДА или НЕТ.

В рамках второго подхода считается, что любой протокол аутентификации, выбранный наугад, тщательно разработанный или имеющий математическое доказательство стойкости, все равно может содержать ошибку. Ведь формально доказательство правильности демонстрирует лишь, что протокол соответствует множеству *установленных* желательных свойств. Вполне возможно, что такой протокол окажется скомпрометированным, если его дефект в процессе доказательства правильности не был учтен. Таким образом, этот подход направлен на исчерпывающий анализ всех возможных ошибок. Для формализации протокол представляется в виде системы с конечным множеством состояний, которую часто разбивают на частичные протоколы, выполняемые разными пользователями (включая Злоумышленника). Ошибку можно описать с помощью терминов общего характера. Например, при анализе секретности сообщений неправильное состояние протокола может возникнуть, если Злоумышленник получает некую часть информации, а при аутентификации сущностей неправильное состояние возникает, если посторонний пользователь получает права, которыми обладает законный пользователь. Этот подход тесно связан с формальным анализом сложных систем, используемым в компьютерных науках. По этой причине для исследования протоколов аутентификации часто применяются автоматические средства анализа.

17.1.1 Структурная схема главы

Техническая часть главы начинается с формализации описания протокола. В разделе 17.2 рассматривается метод последовательного уточнения спецификаций протокола аутентификации. Затем от средств формального описания протоколов аутентификации мы перейдем к методам их анализа. В разделе 17.3 описывается метод доказательства, основанный на вычислительной модели, в котором исследование сводится к доказательству определенной математической теоремы. Раздел 17.4 посвящен методам доказательства стойкости протоколов с помощью манипуляции логическими символами. Кроме того, в этом разделе излагается

логический подход и метод анализа стойкости протоколов, основанный на доказательствах теорем. В разделе 17.5 описываются методы формального анализа протоколов с помощью систем с конечным множеством состояний и поиска системных ошибок. Раздел 17.6 содержит обзор недавних работ, образующих мост между вычислительными и логическими методами доказательства правильности протоколов аутентификации.

17.2 Формальное описание протоколов аутентификации

Для начала покажем необходимость формального описания протоколов аутентификации. Спецификация протокола должна стать неотъемлемой частью любого формального метода анализа сложных систем. Если такой системой является протокол аутентификации, необходимо более точное описание применения криптографических преобразований.

Как показано в главах 2 и 11, многие протоколы аутентификации используют исключительно шифрование, и по этой причине стало общепринятым обозначать шифрование в рамках этих протоколов символами $\{M\}_K$. Это обозначение относится к части зашифрованного текста: чтобы создать его, отправитель должен выполнить шифрование, а, чтобы извлечь сообщение M, получатель должен выполнить расшифровку. Демонстрация этих криптографических возможностей доказывает, что пользователь владеет секретным ключом, т.е. идентифицирует его.

Таким образом, может показаться, что идея аутентификации с помощью шифрования довольно проста.

Однако на самом деле эта простота обманчива. Неверное применение этого механизма выявляет дефекты, присущие многим протоколам. В этом разделе мы продемонстрируем наиболее распространенные ошибки при использовании шифрования для аутентификации, а затем предложим метод разработки протоколов, основанный на последовательном уточнении криптографических преобразований.

17.2.1 Непригодность механизма шифрования-расшифровки при аутентификации

В разделе 11.4.1.5 были перечислены два “нестандартных” механизма создания протоколов аутентификации, основанных на шифровании. В этих механизмах отправитель порождает зашифрованнный текст $\{M\}_K$ и посылает его по назначению. Подлинный получатель имеет секретный ключ, позволяющий выполнить расшифровку и вернуть отправителю компонент, извлеченный из зашифрованного текста. Возвращенный компонент сообщения, часто содержащий идентификатор

"свежести", доказывает отправителю существование получателя и обеспечивает аутентификацию получателя.

17.2.1.1 Вредность

В механизме "оклик-отзыв", предназначенном для верификации "свежести" сообщения, процесс шифрования-расшифровки позволяет противнику воспользоваться помощью *оракула расшифровки* (см. разделы 7.8.2.1 и 8.9). В этом случае Злоумышленник может выполнить незаконную операцию, которую он не смог бы осуществить без посторонней помощи, поскольку ему не известен правильный криптографический ключ.

Оракул расшифровки открывает для Злоумышленника широкие возможности для манипуляций протокольными сообщениями и дискредитирует цели аутентификации. Атака Лоу на протокол аутентификации с открытым ключом Нидхема–Шредера (атака 2.3) ясно демонстрирует такой трюк: на этапах 1–7 Злоумышленник использует Алису в качестве оракула расшифровки для того, чтобы расшифровать случайное число Боба N_B и в дальнейшем общаться с Бобом, представляясь Алисой.

Оракул расшифровки предоставляет Злоумышленнику ценную информацию, необходимую для криптоанализа, например, для организации атак на основе подобранного исходного или зашифрованного текста. Большое количество таких приемов описано в главе 14.

Для того чтобы продемонстрировать свой криптографический мандат (т.е. обладание правильным ключом) в механизме "оклик-отзыв", получатель должен выполнить однонаправленное преобразование. Если для шифрования применяется симметричный криптографический метод, наиболее удачным является механизм 11.4.2. Если идентификатор "свежести" должен сохраняться в секрете, следует применять механизм 11.4.1. Однако в этом случае для защиты своего зашифрованного текста Боб должен выполнять проверку целостности данных с помощью однонаправленного преобразования (причины указаны в разделе 17.2.1.2). Иначе говоря, для защиты зашифрованного текста в механизме 11.4.1 необходимо применять механизм 11.4.2. Если для шифрования используются асимметричные методы, как правило, применяется механизм 11.4.3.

Разумеется, механизмы 11.4.1 и 11.4.2 тоже позволяют пользователю, посылающему оклик, задействовать получателя в качестве оракула при создании пар "исходный текст-шифрованный текст"

$$N, \mathcal{E}_K(\ldots, N),$$
$$N, \mathsf{MDC}(K, \ldots, N),$$

где число N представляет собой идентификатор "свежести", выбираемый по усмотрению пользователя, посылающего оклик. Несмотря на это, если число N

является открытым, пересылка такой пары порождает меньше проблем, чем предоставление услуг оракула расшифровки.

Более того, во втором случае услуги оракула шифрования на самом деле остаются недоступными. Любое однонаправленное преобразование, необходимое для создания кода MDC, позволяет сжимать данные (см., например, разделы 10.3.1 и 10.3.3, в которых описано сжатие данных с помощью хэш-функции и блочного шифра). Сжатие данных означает потерю информации — в результате преобразование становится необратимым. Потеря информации делает пару "оклик-отзыв" бесполезной: они применяются только в механизме 11.4.2. Применение этих пар в сочетании с другими механизмами может породить ошибки.

17.2.1.2 Недостаточность

Шифрование сообщения должно сохранять целостность данных. Если такой защиты нет, активный противник может манипулировать зашифрованными сообщениями, что компрометирует протокол.

Для примера рассмотрим протокол аутентификации Нидхема–Шредера с симметричным ключом (его вариант, предложенный Деннингом и Сакко, описан в разделе 2.6.5.1). Будем предполагать, что алгоритм шифрования, используемый в протоколе, обеспечивает надежную защиту секретности любого компонента сообщения в зашифрованном тексте. Однако для простоты изложения предположим, что алгоритм шифрования *не* обеспечивает целостности данных. Заметим, что это предположение вполне обоснованно. На самом деле ни один алгоритм шифрования, который изначально не предназначен для защиты целостности данных, не обладает этим свойством, если исходный текст является достаточно случайным и не распознается после расшифровки.

В качестве примера выберем алгоритм шифрования AES (раздел 7.7), выполняемый в режиме CBC (раздел 7.8.2). Читатели могут провести аналогичные рассуждения для любого другого симметричного алгоритма шифрования, например, для шифрования с помощью одноразового блокнота. Необходимо отметить, что, независимо от алгоритма шифрования, наша атака не будет использовать его слабости.

Рассмотрим два первых шага протокола аутентификации с симметричным ключом Нидхема–Шредера.

1. Алиса $\rightarrow$ Тренту : $\mathit{Алиса}, \mathit{Боб}, N_A$.
2. Трент $\rightarrow$ Алисе : $\{N_A, K, \mathit{Боб}, Y\}_{K_{AT}}$,

где $Y = \{\mathit{Алиса}, K, T\}_{K_{BT}}$.

Пусть $P_1, P_2, \ldots, P_\ell$ — блоки исходного сообщения, представляющего собой строку

$$N_A, K, \mathit{Боб}, Y.$$

Для того чтобы протокол соответствовал требованиям практики, следует предположить, что размер сеансового ключа K не меньше, чем размер блока зашифрованного текста. Это вполне разумное предположение, поскольку сеансовый ключ должен содержать достаточно много информации. Случайное число N_A также должно быть достаточно большим и непредсказуемым. Поскольку первая часть числа N_A находится в блоке P_1, из предположения о размере сеансового ключа непосредственно следует, что блок P_2 хранит либо весь сеансовый ключ, либо его часть.

Отметим, что мы связали блок P_2 с ключом K лишь для простоты изложения. Если сеансовый ключ K слишком велик, он может храниться в нескольких блоках, начиная с блока P_2. Разумеется, Злоумышленник должен знать размер сеансового ключа K. В конце концов, эта информация не является секретной.

Пусть $IV, C_1, C_2, \ldots, C_\ell$ — блоки зашифрованного текста AES-CBC, соответствующие блокам исходного текста $P_1, P_2, \ldots, P_\ell$ (режим CBC описан в разделе 7.8.2). Кроме того, обозначим через

$$IV', C_1', C_2', \ldots, C_\ell'$$

блоки зашифрованного текста из предыдущего сеанса протокола связи между теми же самыми участниками протокола. Предполагается, что Злоумышленник записал все блоки старых зашифрованных текстов.

Чтобы атаковать протокол в ходе нового сеанса, Злоумышленник должен перехватить блоки нового зашифрованного текста, посланные Трентом Алисе.

2. Трент → Злоумышленнику ("Алисе") : $IV, C_1, C_2, \ldots, C_\ell$.

Затем Злоумышленник должен заменить эти блоки следующим образом.

2. Злоумышленник ("Трент") → Алисе : $IV', C_1, C_2', \ldots, C_\ell'$.

Иначе говоря, Злоумышленник должен заменить последние $\ell - 1$ блоков зашифрованного текста в ходе текущего сеанса соответствующими блоками старого зашифрованного текста, записанного в ходе предыдущих сеансов, и отправить это сообщение Алисе, маскируясь Трентом.

При расшифровке в режиме CBC число N_A остается на предписанном месте, поскольку результат расшифровки представляет собой функцию, зависящую от вектора инициализации IV и блока C_1 (см. раздел 7.8.2). Эта функция возвращает следующий "новый" сеансовый ключ (или первый блок "нового" сеансового ключа).

$$\hat{K} = D_{K_{AT}}\left(C_2'\right) \oplus C_1 = K' \oplus C_1' \oplus C_1.$$

Здесь символ K' обозначает старый сеансовый ключ (или его первый блок), распространявшийся в ходе предыдущего сеанса. В результате расшифровки Алиса обнаружит, что остальные $\ell - 1$ блоков исходного текста идентичны блокам, полученным ею ранее.

Поскольку K' — старый сеансовый ключ, не следует исключать вероятность того, что Злоумышленнику он каким-то образом уже известен (например, Алиса или Боб его случайно раскрыли). Следовательно, Злоумышленник может использовать ключ $\hat{K}$ (или конкатенацию ключа $\hat{K}$ с остальными блоками ключа K', если размер сеансового ключа превышает один блок) для того, чтобы общаться с Алисой, маскируясь Бобом.

Анализ этой атаки показывает, что, независимо от значения идентификатора "свежести" N_A, ни одно исходное сообщение, полученное от Алисы после расшифровки, нельзя считать "свежим"!

Существует много способов реализовать механизм шифрования-расшифровки в этом протоколе. Однако, поскольку его детали известны Злоумышленнику, каждый из них может предотвратить одну атаку, но оказаться уязвимым для другой.

Ошибочная идея, заключавшаяся в использовании режима шифрования CBC для защиты целостности данных, была положена в основу двух протоколов аутентификации, опубликованных в качестве предварительных стандартов [144, 145]. Общие принципы разработки этих протоколов изложены в работах [142, 146]. Разумеется, эти протоколы были обречены на провал [184, 185].

По нашему мнению, следовало бы защитить блоки зашифрованного текста с помощью соответствующего механизма, гарантирующего целостность данных, например, с помощью методов аутентификации сообщений, описанных в разделах 10.3.2 и 10.3.3 (коды для распознавания манипуляций). Эти методы основаны на применении однонаправленного преобразования, а не на механизме шифрования-расшифровки.

Итак, мы ясно продемонстрировали, что применения методов симметричного шифрования в протоколах аутентификации недостаточно для обеспечения безопасности.

В протоколах аутентификации, использующих асимметричные методы, механизм шифрования-расшифровки также совершенно недостаточен. В качестве примера укажем протокол аутентификации с открытым ключом Нидхема–Шредера (протокол 2.5). Атака Лоу на этот протокол (атака 2.3) четко демонстрирует его слабость. Позднее мы покажем (раздел 17.2.3.3), что подход, основанный на применении однонаправленного преобразования, позволяет исправить этот протокол и предотвратить атаку Лоу.

17.2.2 Уточненная спецификация протоколов аутентификации

Для того чтобы точнее описать протоколы аутентификации, Бойд и Мао предложили использовать для обозначения криптографических операций следующие символы [186].

- $\{M\}_K$ — результат шифрования сообщения M с помощью ключа K. Эта операция обеспечивает секретность сообщения M: оно может быть восстановлено только пользователем, обладающим ключом K^{-1}, т.е. ключом расшифровки, соответствующим ключу K. Отметим, что владелец ключа K^{-1} может *не* распознать результат расшифровки.
- $[M]_K$ — результат однонаправленного преобразования сообщения M с помощью ключа K. Эта операция обеспечивает целостность данных и *идентификацию источника сообщений* с помощью средств, описанных в разделе 10. Сообщение M в тексте $[M]_K$ не является секретным и может оказаться доступным даже без применения каких-либо криптографических операций. Пользователь, обладающий ключом K^{-1}, может проверять целостность данных в тексте $[M]_K$ и идентифицировать источник сообщения. Если целостность данных в сообщении M не нарушена, а источник поддается идентификации, процедура верификации возвращает значение ДА, в противном случае она возвращает значение НЕТ.
- На практике при реализации симметричного алгоритма шифрования или схемы цифровой подписи, использующей асимметричный алгоритм, текст $[M]_K$ можно представить в виде $(M, \mathrm{prf}_K(M))$, где prf_K — псевдослучайная функция с ключом (например, код аутентификации сообщений в режиме CBC-MAC, как в разделе 10.3.3, или криптографическая функция хэширования с ключом HMAC, как в разделе 10.3.2). Это представление является достаточно эффективным.

Уточненные обозначения относятся как к симметричным, так и к несимметричным методам шифрования. В первом случае ключи K и K^{-1} совпадают, а во втором — образуют пару согласованных ключей.

Следует подчеркнуть, что преобразование $[M]_K$ не только гарантирует целостность данных, но и позволяет *идентифицировать источник сообщения*. Если процедура верификации текста $[M]_K$ возвращает значение ДА, то, даже если сообщение M вообще не содержит никакой информации об источнике сообщений, верификатор может идентифицировать правильный источник с помощью использованного ключа верификации.

В последнее время идея о необходимости совмещения секретности и защиты целостности данных получила всеобщее признание. Ее применение в криптографии с открытым ключом подробно освещалось в главе 15. В работе [11] Айелло (Ailello) и его соавторы предложили для соответствующей объединенной процедуры обозначение $\{M\}_{K_1}^{K_2}$. Это значит, что сообщение M шифруется с помощью ключа K_1, а для защиты целостности данных используется ключ K_2.

17.2.3 Примеры уточненной спецификации протоколов аутентификации

Рассмотрим несколько примеров уточненных протоколов аутентификации.

17.2.3.1 Протокол аутентификации с симметричным ключом Нидхема–Шредера

В качестве первого примера рассмотрим протокол аутентификации с симметричным ключом Нидхема–Шредера.

Протокол 17.1. Уточненная спецификация протокола аутентификации с симметричным ключом Нидхема–Шредера

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ И ЦЕЛЬ:

те же, что и в протоколе 2.4.

1. Алиса $\rightarrow$ Тренту : *Алиса*, *Боб*, N_A.
2. Трент $\rightarrow$ Алисе : $\left[\{K\}_{K_{AT}}, N_A, \textit{Алиса}, \textit{Боб}\right]_{K_{AT}}$,
   $\left[\{K\}_{K_{BT}}, T, \textit{Алиса}, \textit{Боб}\right]_{K_{BT}}$.
3. Алиса $\rightarrow$ Бобу : $\left[\{K\}_{K_{BT}}, T, \textit{Алиса}, \textit{Боб}\right]_{K_{BT}}$.
4. Боб $\rightarrow$ Алисе : $[N_B]_K$.
5. Алиса $\rightarrow$ Бобу : $[N_B - 1]_K$.

В уточненной спецификации протокола аутентификации с симметричным ключом Нидхема–Шредера необходимость защиты целостности данных выделена явно. Если сообщения, которые Алиса и Боб получают от Трента и которыми они обмениваются друг с другом, в процессе передачи не искажались, то после выполнения процедуры верификации обе стороны могут быть уверены в том, что сеансовый ключ K согласован как с пользователями, так и с идентификатором "свежести". Это убеждает их в подлинности партнеров и "свежести" сеансового ключа. Очевидно, что любое несанкционированное изменение сообщения (см. раздел 17.2.1.2) легко распознать.

Анализ спецификации показывает, что секретность в этом протоколе обеспечивается на минимальном уровне: защищенным является только сеансовый ключ K. Это объясняется тем, что ключ является случайным, а значит, необходимо как можно меньше использовать средства шифрования, чтобы не давать Злоумышленнику информации для криптоанализа.

17.2.3.2 Протокол Ву–Лама

В качестве второго примера рассмотрим протокол Ву–Лама. Его уточненная спецификация приведена в протоколе 17.2 (ср. с протоколом 11.2). Этот пример ярко показывает, насколько эффективно уточненная спецификация предотвращает разнообразные атаки. Причины, позволяющие организовать такие атаки, станут совершенно очевидными: некорректные криптографические операции приводят к неправильному применению шифрования в протоколе аутентификации.

Протокол 17.2. Уточненная спецификация протокола Ву–Лама

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ И ЦЕЛЬ:

те же, что и в протоколе 11.2.

СОГЛАШЕНИЕ:

если процедура верификации возвращает значение НЕТ, выполнение протокола прекращается.

1. Алиса → Бобу: *Алиса*.
2. Боб → Алисе: N_B.
3. Алиса → Бобу: $[N_B]_{K_{AT}}$.
4. Боб → Тренту: $\left[Алиса, N_B, [N_B]_{K_{AT}}\right]_{K_{BT}}$.
   (* Примечание: Боб включает одноразовое случайное число N_B в свою часть однонаправленного преобразования, поскольку *сам* является источником идентификатора “свежести” .*)
5. Трент → Бобу: $[N_B]_{K_{BT}}$.
6. Боб принимает сообщение, если процедура верификации целостности данных в тексте $[N_B]_{K_{BT}}$ возвращает значение ДА, в противном случае он отказывается участвовать в протоколе.

Отметим, что ни одно сообщение в протоколе Ву–Лама (протокол 11.2) не является секретным, поэтому нет никакой необходимости обеспечивать секретность какого бы то ни было протокольного сообщения. Единственной целью криптографической защиты в этом протоколе является защита целостности данных и идентификация источника. Таким образом, уточненная спецификация протокола лишь описывает однонаправленное преобразование.

Покажем, что ни одна из атак на исходный протокол Ву–Лама и его “исправленные” варианты, описанные в разделе 11.7, не опасна для уточненной версии.

Во-первых, атака с помощью параллельных сеансов (атака 11.5) больше невозможна. Чтобы убедиться в этом, предположим, что Злоумышленник посылает Бобу сообщение $[N_B]_{K_{MT}}$ на параллельных этапах 3 и 3’.

3. Злоумышленник ("Алиса") → Бобу: $[N_B]_{K_{MT}}$.

3'. Злоумышленник →Бобу: $[N_B]_{K_{MT}}$.

Предположим также, что Боб не проверяет эти сообщения (поскольку для них не предусмотрено никаких проверок) и просто пересылает их на параллельных этапах 4 и 4'.

4. Боб → Тренту: $\left[Алиса, N_B, [N_B]_{K_{MT}}\right]_{K_{BT}}$.

4'. Боб → Тренту: $\left[Злоумышленник, N'_B, [N_B]_{K_{MT}}\right]_{K_{BT}}$.

Однако Трент обнаруживает на этих двух этапах две ошибки. Первая ошибка возникает при верификации на этапе 4: Трент использует для верификации сообщения $[N_B]_{K_{MT}}$ ключ K_{AT}, и на этом выполнение протокола прекращается. Вторая ошибка обнаруживается при верификации сообщения на этапе 4': Трент приходит к выводу, что два однонаправленных преобразования используют разные случайные числа, а значит, выполнение протокола также необходимо прекратить (в противном случае неясно, какое из этих двух чисел Трент должен вернуть Бобу).

Кроме того, очевидно, что атака с помощью отражения (атака 11.6) также больше невозможна, поскольку если Злоумышленник отражает сообщение, полученное на этапе 4, в виде сообщения на этапе 5, то процедура верификации, выполняемая Бобом на этапе 6, определенно вернет значение НЕТ.

Теперь ясно, что основным недостатком исходного протокола Ву–Лама было отсутствие криптографических операций.

17.2.3.3 Протокол аутентификации с открытым ключом Нидхема–Шредера

В заключение рассмотрим уточненную спецификацию протокола аутентификации с открытым ключом Нидхема–Шредера.

Как указано в разделе 2.6.6.3, протокол аутентификации с открытым ключом Нидхема–Шредера можно разбить на три этапа.

Протокол 17.3. Протокол аутентификации с открытым ключом Нидхема–Шредера

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ И ЦЕЛЬ:

те же, что и в протоколе 2.5.

1. Алиса → Бобу: $\{N_A, Алиса\}_{K_B}$.
2. Боб → Алисе: $\{N_A, N_B\}_{K_A}$.
3. Алиса → Бобу: $\{N_B\}_{K_B}$.

Здесь $\{\ldots\}_{K_A}$ и $\{\ldots\}_{K_B}$ — сообщения, зашифрованные с помощью открытых ключей Алиса и Боба. Следовательно, они могут быть расшифрованы только ими.

Таким образом, получив сообщение на этапе 2, Алиса должна верить, что только Боб мог расшифровать сообщение, полученное им на этапе 1, и вернуть ей случайное число N_A. Аналогично, получив сообщение на этапе 3, Боб должен верить, что только Алиса могла расшифровать послание, полученное ею на этапе 2, и вернуть ему случайное число N_B. Таким образом, вполне естественно ожидать, что после выполнения протокола обе стороны идентифицируют друг друга.

Однако атака Лоу (атака 2.3) опровергает эти ожидания. Учитывая аргументы против использования механизма шифрования-расшифровки в протоколах аутентификации, изложенные в разделе 17.2.1, можно рассмотреть атаку на исходный вариант протокола Нидхема–Шредера под новым углом зрения: она становится возможной из-за неправильного применения криптографических операций. Атака станет невозможной, если спецификацию протокола уточнить следующим образом.

Протокол 17.4. Уточненная спецификация протокола аутентификации с открытым ключом Нидхема–Шредера

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ И ЦЕЛЬ:

те же, что и в протоколе 2.5.

1. Алиса $\rightarrow$ Бобу: $\left\{[N_A, \mathit{Алиса}]_{K_A}\right\}_{K_B}$.
2. Боб $\rightarrow$ Алисе: $\left\{N_A, [N_B]_{K_B}\right\}_{K_A}$.
3. Алиса $\rightarrow$ Бобу: $\left\{[N_B]_{K_A}\right\}_{K_B}$.

В уточненной спецификации $[N_A, \mathit{Алиса}]_{K_A}$ — это сообщение, для верификации которого необходимо использовать открытый ключ Алисы K_A. Следовательно, преобразование $[N_A, \mathit{Алиса}]_{K_A}$ можно считать подписью Алисы. Итак, сообщение на этапе 1 представляет собой одноразовое случайное число, сгенерированное и подписанное Алисой, а затем зашифрованное с помощью открытого ключа Боба. Аналогично сообщение на этапе 2 сначала подписано Бобом, а затем зашифровано с помощью открытого ключа Алисы.

Поскольку второе сообщение подписано Бобом, атака Лоу на уточненную версию протокола становится невозможной. Злоумышленник может запустить выполнение протокола связи с Бобом, маскируясь Алисой и используя ее подпись (Алиса сама посылает Злоумышленнику свою подпись, а он может расшифровывать и зашифровывать ее с помощью открытого ключа Боба). Однако теперь Злоумышленник не может пересылать Алисе ответы Боба, как он это делал в атаке на сходный вариант протокола, поскольку Алиса заметит ошибку при верификации подписи Злоумышленника.

Несмотря на то что в разделе 2.6.6.4 мы предложили способы защиты протокола Нидхема–Шредера от атаки Лоу путем добавления в зашифрованные сообщения имен общающихся пользователей, этот прием был не совсем корректным. Действительно, если алгоритм шифрования является уязвимым (см. раздел 14.5.3), то нет гарантии, что пользователь, выполняющий расшифровку, будет уверен в правильности имени, указанного в сообщении. Очевидно, что для окончательного исправления протокола необходимо использовать корректные криптографические операции, а спецификацию протокола следует уточнить.

Спецификацию протокола аутентификации с открытым ключом Нидхема–Шредера можно уточнить иначе — с помощью метода "зашифруй и подпиши" (протокол 17.5).

Протокол 17.5. Второй вариант уточненной спецификации протокола аутентификации с открытым ключом Нидхема–Шредера

ПРЕДВАРИТЕЛЬНЫЕ УСЛОВИЯ И ЦЕЛЬ:

те же, что и в протоколе 2.5.

1. Алиса → Бобу: $\left[\{N_A\}_{K_B}, Алиса\right]_{K_A}$.
2. Боб → Алисе: $\left[\{N_A, N_B\}_{K_A}\right]_{K_B}$.
3. Алиса → Бобу: $\left[\{N_B\}_{K_B}\right]_{K_A}$.

Атака Лоу для этого варианта также не представляет опасности: теперь Злоумышленник даже не в состоянии запустить протокол связи с Бобом.

17.3 Вычислительные аспекты корректных протоколов — модель Белларе–Роджуэя

В главах 14 и 15 изложены идеи доказуемой стойкости, основанные на вычислительной модели, предложенной в знаменитой работе Голдвассера и Микали [125]. В этой модели секретность (один из нескольких аспектов конфиденциальности) подвергается атаке (одной из многочисленных игровых атак, моделирующих с определенной точностью поведение реальных атакующих алгоритмов, стремящихся взломать схемы шифрования с открытым ключом). Доказательство стойкости схем шифрования с открытым ключом к атакам связано с демонстрацией эффективного преобразования (полиномиальной редукции), сводящего атаку к решению одной из трудноразрешимых задач из теории вычислительной сложности. Невозможность решения такой задачи опровергает существование эффективной атаки, т.е. обнаруживает противоречие.

Следовательно, формальное доказательство стойкости в рамках вычислительной модели состоит из трех этапов.

1. Формальное поведение участников протокола и атакующего алгоритма: моделирование, как правило, осуществляется с помощью атакующей игры, в которой участвуют атакующий алгоритм и объект атаки.
2. Формальное определение стойкости: победа атакующего алгоритма в атакующей игре обычно выражается через (значимую) вероятность и оценку (разумной) временно́й сложности.
3. Формальная демонстрация существования полиномиальной редукции, сводящей атаку к решению трудноразрешимой задачи, которая, как правило, имеет вид математической теоремы.

Белларе и Роджуэй были первыми исследователями, предложившими вычислительный подход к доказательству стойкости протоколов аутентификации и протоколов согласования аутентифицированных ключей [23]. В своей знаменитой работе они смоделировали атаки на эти протоколы, разработали несколько простых протоколов аутентификации и согласования аутентифицированных ключей, а затем привели доказательство их стойкости. Эти доказательства основаны на редукции атаки к нарушению условия псевдослучайности, т.е. обнаружению отличий между значениями псевдослучайной функции и истинно случайной функции с помощью полиномиального алгоритма распознавания. Иначе говоря, доказательство отрицает существование псевдослучайных функций.

Читатели могут вернуться к разделу 4.7 и проанализировать предположения 4.1 и 4.2. Эти предположения являются основой современной криптографии. Отсюда следует, что результатом редукции должна быть либо ошибка, либо переворот всех основ современной криптографии. Поскольку ошибка более вероятна, редукция приводит к требуемому противоречию.

Мы рассмотрим наиболее простой результат работы Белларе и Роджуэя: доказательство стойкости двусторонней аутентификации сущностей, основанное на использовании общего симметричного ключа [23]. Несмотря на его простоту, этого достаточно для иллюстрации основных принципов, лежащих в основе вычислительной модели.

Описание работы Белларе и Роджуэя, посвященной протоколу аутентификации, состоит из трех этапов. В разделе 17.3.1 описывается формальная модель поведения участников протокола и Злоумышленника. В разделе 17.3.2 приводится формальное определение стойкости протокола взаимной аутентификации сущностей. В разделе 17.3.3 демонстрируется редукционное доказательство стойкости этого протокола.

17.3.1 Формальное моделирование поведения участников протокола

Протоколы, рассмотренные в работе [23], являются двусторонними. Каждый из двух участников протокола моделируется эффективным исполняемым кодом, имеющим входную и выходную информацию. Протокол состоит из сеансов связи между двумя участниками. Однако следует подчеркнуть, что в этих "сеансах связи" участвует Злоумышленник, который может манипулировать значениями, передаваемыми по каналу связи.

Итак, описание абстрактного протокола состоит из двух частей: во-первых, эффективно вычисляемой функции, принадлежащей участнику протокола, и, во-вторых, совокупности связей.

17.3.1.1 Формализация части протокола, выполняемой подлинным участником

С формальной точки зрения абстрактный протокол описывается функцией $\mathcal{P}$, имеющей полиномиальную сложность и следующие аргументы.

1^k Параметр безопасности $k \in \mathbb{N}$ (унарное представление параметра безопасности аргументируется в разделе 4.4.6 и определении 4.7).

i Идентификатор пользователя $i \in I$, выполняющего данную часть протокола. Будем называть этого пользователя "владельцем". Множество I состоит из пользователей, обладающих одним и тем же долговременным ключом.

j Идентификатор партнера, с которым общается владелец; $j \in I$.

K Долговременный симметричный ключ (т.е. секретная входная информация). В двустороннем протоколе симметричный ключ K принадлежит как владельцу, так и его партнеру.

conv Предыдущие сообщения — *conv* (от англ.: conversation — *прим. ред.*), представляющие собой строку битов. Эта строка увеличивается по мере выполнения протокола, причем новая строка приписывается к старой.

r Случайный аргумент владельца — читатель может считать число r одноразовым случайным числом, генерируемым владельцем.

Поскольку $\mathcal{P}(1^k, i, j, K, conv, r)$ имеет полиномиальную сложность, зависящую от размера аргументов (отметим, что размер параметра k равен 1^k), можно считать, что аргументы K и r имеют размер k, а размер аргументов i, j и $conv$ полиномиально зависит от параметра k.

Значениями функции $\mathcal{P}(1^k, i, j, K, conv, r)$ являются три числа.

m Следующее сообщение, подлежащее отправке, — $m \in \{0,1\}^* \cup$ {"сообщений нет"}. Это — открытое сообщение, подлежащее отправке адресату через открытую сеть.

δ Решение пользователя — $\delta \in$ {Принять, Отказать, Не принимать решения}. Пользователь решает, принять или отвергнуть идентификатор партнера по переговорам, либо не принимать решения вообще. Принятие идентификатора, как правило, откладывается до завершения протокола, а отклонить его можно в любой момент. Если пользователь принимает какое-либо определенное решение, значение δ больше не изменяется.

α Закрытый результат, вычисленный владельцем, — $\alpha \in \{0,1\}^* \cup$ {"результата нет"}. Читатели могут считать, что закрытым результатом, вычисленным владельцем при благоприятном исходе протокола, является согласованный сеансовый ключ.

Анализ показывает, что в формальную модель Белларе–Роджуэя входят важные компоненты протоколов аутентификации сущностей: криптографические операции, идентификаторы участников, идентификаторы "свежести" и сообщения. (Смысл этих компонентов описан в разделе 11.4.)

Иногда абстрактный протокол аутентификации сущностей, принадлежащих множеству I, обозначается как $\mathcal{P}(1^k, I)$.

Для любой пары $i, j \in I$ (т.е. для любых двух участников протокола, обладающих одним и тем же долговременным симметричным ключом) и для числа $s \in \mathbb{N}$ обозначим через $\prod_{i,j}^{s}$ игрока i, пытающегося аутентифицировать игрока j в ходе сеанса с меткой s. Эта попытка может инициироваться игроком i, а может быть и ответом на сообщение, полученное от пользователя j. Фактически мы всегда будем интерпретировать эту попытку как ответ на *запрос оракулу*, посланный Злоумышленником. Для этого необходимо формализовать понятие обмена информацией.

17.3.1.2 Формализация обмена информацией

Белларе и Роджуэй в своей работе следовали идее Долева и Яо [101] (см. раздел 2.3): Злоумышленник, т.е. атакующий алгоритм, контролирует всю сеть связи.

Злоумышленнику, контролирующему сеть, могут быть известны последовательности игроков $\prod_{i,j}^{s}$ и $\prod_{j,i}^{t}$ для любой пары $i, j \in I$ (т.е. для любых двух пользователей, обладающих общим долговременным симметричным ключом), даже если он не вычисляет их сам. Однако, будучи активным атакующим алгоритмом, Злоумышленник может делать намного больше, чем просто пассивно наблюдать за сообщениями. Он может организовывать сколько угодно *сеансов* связи и убеж-

дать любого пользователя (например, i) начать протокол, считая Злоумышленника другим подлинным пользователем (например, j).

Поскольку Злоумышленник является мощным и активным атакующим алгоритмом, он может использовать игроков $\prod_{i,j}^{s}$ и $\prod_{j,i}^{t}(i,j \in I, s,t \in \mathcal{N})$ в качестве *оракулов*, представляющих собой черные ящики. Это значит, что Злоумышленник может послать запрос оракулу $\prod_{i,j}^{s}$, передавая пользователю i аргументы $(i,j,s,conv)$. Аналогичным образом Злоумышленник может послать запрос оракулу $\prod_{j,i}^{t}$. Когда Злоумышленник посылает запрос оракулу $\prod_{i,j}^{s}$, используя аргументы $(i,j,s,conv)$, пользователь i добавляет к ним собственный секретный ключ K, случайное число r и вычисляет функцию $\prod^{s}(1^k,i,j,K,conv,r)$. После этого пользователь i отсылает в сеть результат m (если он существует) или строку "сообщений нет", а также решение δ, сохраняя у себя закрытый компонент результата α. Все результаты, посланные пользователем в сеть, оказываются в распоряжении Злоумышленника и могут быть использованы для организации атаки.

В модели Долева–Яо любой запрос считается пришедшим от Злоумышленника, пока оракул не примет положительного решения.

Не ограничивая общности, можно считать, что среди злоумышленников всегда есть относительно безопасный противник, называемый "безвредным". Он просто выбирает пары оракулов $\prod_{i,j}^{s}$ и $\prod_{j,i}^{t}$, а затем перехватывает и точно передает сообщения, которыми они обмениваются, начиная с оракула $\prod_{i,j}^{s}$. Другими словами, первый запрос безвредного противника выглядит так: $(i,j,s,\text{“”})$ (здесь символы “” обозначают пустую строку). В ответ Злоумышленник получает сообщение $m_1^{(i)}$. Второй запрос выглядит следующим образом: $(j,i,t,m_1^{(i)})$. В ответ противник получает сообщение $m_1^{(j)}$ и так далее, пока оба оракула не примут положительное решение и не прекратят свою работу. Таким образом, безвредный противник играет роль проводника, связывающего между собой пользователей i и j. Позднее мы убедимся, что в доказуемо стойком протоколе Злоумышленник, стремящийся пройти аутентификацию, вынужден играть роль безвредного противника.

Будем считать, что t-й запрос Злоумышленника посылается оракулу в момент времени $\tau = \tau_t \in \mathbb{R}$. Кроме того, потребуем, чтобы при $t < u$ выполнялось неравенство $\tau_t < \tau_u$.

17.3.2 Цель взаимной аутентификации: согласованные диалоги

Белларе и Роджуэй указали, что целью взаимной аутентификации сущностей являются **согласованные диалоги** (matching conversations). (Обратите внимание на множественное число.)

Диалог оракула $\prod_{i,j}^{s}$ представляет собой последовательность упорядоченных во времени сообщений, отсылаемых им в сеть, и получаемых на них ответов. Пусть $\tau_1 < \tau_2 < \cdots < \tau_R$ (где R — некоторое положительное целое число) — моменты времени, в которые оракул $\prod_{i,j}^{s}$ посылает сообщение. Диалог можно обозначить следующим образом.

$$conv = (\tau_1, m_1, m'_1), (\tau_2, m_2, m'_2), \ldots, (\tau_R, m_R, m'_R).$$

Эта запись означает, что в момент τ_1 оракул $\prod_{i,j}^{s}$ получает запрос m_1 и посылает в ответ сообщение m'_1. Затем в момент $\tau_2 > \tau_1$ оракул получает запрос m_2 и посылает в ответ сообщение m'_2 и так далее, пока в момент τ_R он не получит запрос m_R и пошлет в ответ сообщение m'_R.

Напомним, что в модели Долева–Яо оракул $\prod_{i,j}^{s}$ любой запрос должен считать сообщением от Злоумышленника, пока в некий момент m_R он не примет положительное решение. Удобно предположить, что диалог начинает Злоумышленник. Итак, если $m_1 =$ "", будем называть оракула $\prod_{i,j}^{s}$ инициатором диалога, в противном случае — ответчиком.

Пусть

$$conv = (\tau_0, \text{""}, m_1), (\tau_2, m'_1, m_2), (\tau_4, m'_2, m_3), \ldots, (\tau_{2t-2}, m'_{t-1}, m_t)$$

обозначает диалог оракула $\prod_{i,j}^{s}$. Будем говорить, что оракул $\prod_{j,i}^{t}$ ведет диалог $conv'$, согласованный с диалогом $conv$, если существует последовательность моментов времени $\tau_0 < \tau_1 < \tau_2 < \cdots < \tau_R$, такая что

$$conv = (\tau_1, m_1, m'_1), (\tau_3, m_2, m'_2), (\tau_5, m_3, m'_3), \ldots, (\tau_{2t-2}, m_t, m'_t),$$

где строка m'_t означает "нет сообщений".

Если оракулы $\prod_{i,j}^{s}$ и $\prod_{j,i}^{t}$ всегда ведут согласованные диалоги, то Злоумышленник не в состоянии организовать опасную атаку и вынужден играть роль безвредного противника.

Теперь мы готовы сформулировать понятие стойкости протокола взаимной аутентификации сущностей.

Определение 17.1. *Будем говорить, что протокол* $\prod(1^k, \{A, B\})$ *является стойким протоколом аутентификации, выполняемым пользователями* A *и* B*, если оба оракула* $\prod_{A,B}^{s}$ *и* $\prod_{B,A}^{t}$ *принимают положительное решение тогда и только тогда, когда они ведут согласованные диалоги, причем вероятность противоположного события пренебрежимо мала.*

Если протокол стоек, то из существования согласованных диалогов непосредственно следуют положительные решения оракулов. Доказать обратное утверждение, т.е. что из положительных решений следует существование согласованных

диалогов, намного сложнее. Следовательно, целью атаки на протокол является получение положительных решений, когда согласованных диалогов не существует. Таким образом, более корректным является следующее определение стойкости протокола.

Определение 17.2. *Будем говорить, что протокол* $\prod(1^k, \{A, B\})$ *является стойким протоколом аутентификации, выполняемым пользователями* A *и* B*, если вероятность выигрыша Злоумышленника пренебрежимо мала. Здесь выигрыш Злоумышленника означает, что оба оракула* $\Pi^s_{A,B}$ *и* $\Pi^t_{B,A}$ *принимают положительное решение, хотя между ними нет согласованных диалогов.*

17.3.3 Протокол MAP1 и доказательство его стойкости

Белларе и Роджуэй продемонстрировали формальное доказательство простого протокола взаимной аутентификации MAP1 ("mutual authentication protocol one").

Протокол 17.6. Протокол MAP1

ПРЕДУСЛОВИЕ:

Алиса (A) и Боб (B) обладают общим секретным симметричным ключом K, имеющим размер k;
R_A — одноразовое случайное число Алисы,
R_B — одноразовое случайное число Боба, причем оба числа имеют длину k.
$[x]_K$ — пара $(x, \mathrm{prf}_K(x))$, где $x \in \{0,1\}^*$,
$\mathrm{prf}_K : \{0,1\}^* \mapsto \{0,1\}^k$ — псевдослучайная функция, снабженная ключом K.

$$A \qquad\qquad B$$

$$A \parallel R_A \longrightarrow$$

$$\longleftarrow [B \parallel A \parallel R_A \parallel R_B]_K$$

$$[A \parallel R_B]_K \longrightarrow$$

Этот протокол начинается с того, что Алиса посылает Бобу сообщение $A \parallel R_A$, где R_A — случайное число Алисы, имеющее длину k. Боб отвечает, генерируя случайный оклик R_B длины k и отсылая обратно сообщение $[B \parallel A \parallel R_A \parallel R_B]_K$. Алиса проверяет корректность этого сообщения. Если сообщение имеет правильный формат и метку, Алиса отсылает Бобу сообщение $[A \parallel R_B]_K$ и аутентифицирует его. Затем Боб проверяет корректность сообщения Алисы. Если ее сообщение имеет правильный формат и метку, он, в свою очередь, аутентифицирует Алису.

Если между Алисой и Бобом вклинивается безвредный противник, то в моменты времени $\tau_0 < \tau_1 < \tau_2 < \tau_3$ Алиса принимает диалоги

$$conv_A = (\tau_0, \text{“”}, A \parallel R_A), (\tau_2, [B \parallel A \parallel R_A \parallel R_B]_K, [A \parallel R_B]_K),$$

а Боб — диалоги

$$conv_B = (\tau_1, A \parallel R_A, [B \parallel A \parallel R_A \parallel R_B]_K), (\tau_3, [A \parallel R_B]_K, \text{“сообщений нет”}).$$

Очевидно, что диалоги $conv_A$ и $conv_B$ являются согласованными.

Для доказательства стойкости протокола MAP1 к любой полиномиально ограниченной атаке Белларе и Роджуэй проанализировали два эксперимента. В первом эксперименте функция prf_K является абсолютно случайной. Иначе говоря, оракулы $MAP1^s_{A,B}$ и $MAP1^t_{B,A}$ одновременно распоряжаются функцией prf_K. Когда они применяют ее к аргументу x, результатом $\text{prf}_K(x)$ является строка, равномерно распределенная по множеству $\{0,1\}^k$. Разумеется, необходимо признать, что в реальном мире не существует способа реализации такой функции. Во втором эксперименте протокол MAP1 реализуется с помощью семейства псевдослучайных функций, которые применяются на практике.

Поскольку в первом эксперименте результат $\text{prf}_K(x)$ является k-битовой равномерно распределенной строкой, оракул $MAP1^s_{A,B}$ видит диалог $conv_A$ и убеждается, что равномерно распределенная строка $[B \parallel A \parallel R_A \parallel R_B]_K$ вычислена с помощью значения R_A, сгенерированного им самим. Значит, вероятность того, что эта строка была вычислена Злоумышленником, пренебрежимо мала и равна 2^{-k}. Эта величина не зависит от свойств Злоумышленника. Итак, оракул $MAP1^s_{A,B}$ считает, что требуемый партнер видит диалог $(\tau_1, A \parallel R_A, [B \parallel A \parallel R_A \parallel R_B]_K)$. По сути, это доказывает, что существует диалог, согласованный с диалогом $conv_A$ и вычисленный партнером оракула $MAP1^s_{A,B}$ с огромной вероятностью, зависящей от величины k.

Аналогично, когда оракул $MAP1^t_{B,A}$ видит диалог $conv_B$, он убеждается, что согласованный диалог создан его партнером с огромной вероятностью, зависящей от величины k.

Итак, если протокол MAP1 реализуется с помощью истинно случайной функции, общей для обоих оракулов, вероятность победы Злоумышленника является пренебрежимо малой относительно величины k. (Смысл победы Злоумышленника описан в определении 17.2.)

Оставшаяся часть доказательства сводится к обнаружению противоречия.

Предположим, что во втором эксперименте Злоумышленник побеждает со значимой вероятностью, зависящей от параметра k. Построим полиномиальный алгоритм распознавания T, отличающий случайные функции от псевдослучайных. Алгоритм T получает на вход функцию $g : \{0,1\}^* \mapsto \{0,1\}^k$, выбранную в ходе следующего эксперимента.

1. Подбрасываем идеальную монету C.
2. Если $C =$ "ОРЕЛ", функция g считается случайной.
3. Иначе генерируем случайное число K и полагаем $g = \mathrm{prf}_K$.

Цель алгоритма T — предсказать результат жеребьевки со значимой вероятностью. Его стратегия заключается в том, чтобы выполнить атаку на протокол MAP1, реализованный с помощью функции g.

Если Злоумышленник побеждает (заметим, что алгоритм T может определить, победил Злоумышленник или нет, поскольку контролирует обоих оракулов $MAP1^s_{A,B}$ и $MAP1^t_{B,A}$, а значит, видит их диалоги), то алгоритм T выдает результат $C =$ "ОРЕЛ" (т.е. функция g является псевдослучайной), в противном случае алгоритм T возвращает результат $C =$ "РЕШКА" (т.е. функция g является абсолютно случайной). Итак, преимущество, с которым алгоритм T отличает k-битовые случайные и псевдослучайные функции, равно преимуществу, с которым Злоумышленник может победить, т.е. является значимым. Это противоречит общепринятому убеждению, что не существует полиномиального алгоритма распознавания, позволяющего различать случайные и псевдослучайные функции (предположение 4.2).

На практике псевдослучайную функцию prf_K можно реализовать с помощью кода аутентификации сообщения в режиме CBC-MAC (см. раздел 10.3.3) или функции хэширования HMAC (см. раздел 10.3.2). Обе эти реализации являются эффективными.

17.3.4 Вычислительные модели доказательства корректности протоколов

В своей знаменитой работе [23] Белларе и Роджуэй рассмотрели правильность протоколов согласования аутентифицированного сеансового ключа (протоколов обмена ключами) между двумя партнерами. Злоумышленник выигрывает, если угадывает сеансовый ключ. Поскольку передаваемый сеансовый ключ шифруется с помощью долговременного общего ключа, угадать его так же трудно, как отличить псевдослучайную функцию от абсолютно случайной.

Позднее Белларе и Роджуэй распространили свой метод на трехсторонний случай, в котором третей стороной является сервер аутентификации [25].

Некоторые авторы пошли еще дальше: например, в работе [37] доказана стойкость протокола обмена ключами, в работах [36, 38] исследована стойкость протокола согласования ключей, в работах [21, 57] изучен протокол, основанный на паролях, в работах [18, 66] проанализирован протокол обмена ключами, а в работе [67] продемонстрирована стойкость протокола IKE.

17.3.5 Выводы

Введя понятие согласованных диалогов, Белларе и Роджуэй формализовали понятие стойкости протокола. Этот позволило выявить и устранить типичные недостатки протоколов, делавших их уязвимыми для атаки на основе отражения (раздел 11.7.4) или атаки на основе неправильной интерпретации (раздел 11.7.6). Кроме того, математический анализ стойкости протоколов вынуждает разработчиков применять более точные криптографические средства (раздел 11.7.8).

Очевидно, что доказательство стойкости протоколов тесно связано с их разработкой. Стойкими могут быть только хорошо разработанные протоколы.

Следует отметить недостатки, связанные с этим подходом. В определениях 17.1 и 17.2 не учитываются варианты, в которых аутентификация осуществляется без согласования диалогов, и вопрос о стойкости протокола остается открытым. Однако, поскольку оракул, принимающий положительное решение, на самом деле ошибается, в этих случаях протокол следует признать нестойким.

Возможно, ошибочное решение вообще не следует рассматривать. Для любого стойкого протокола Злоумышленник всегда может оборвать последнее сообщение и не дать оракулу принять положительное решение. Очевидно, непринятие решения не делает протокол нестойким. Однако следует помнить, что существуют нетривиальные ошибки аутентификации, которые не являются результатом атаки с помощью обрыва последнего сообщения и не описываются определениями 17.1 и 17.2. Один из таких примеров демонстрировался в разделе 12.1 при описании атаки 12.1 на протокол IKE.

Благодаря природе связи доказать стойкость протокола аутентификации намного сложнее, чем стойкость криптографического алгоритма. Подход Белларе и Роджуэя был шагом в правильном направлении. Их идеи в настоящее время развиваются многими последователями.

17.4 Доказательство корректности протоколов с помощью символических манипуляций

Доказательство корректности протоколов с помощью символических манипуляций основано на формальных методах исследований, принятых в компьютерных науках. В этих случаях стойкость демонстрируется с помощью операций над абстрактными символами. В одних работах для анализа стойкости используются формальные логические системы, а в других — автоматизированные средства для доказательства математических теорем, возвращающее результат ДА/НЕТ.

17.4.1 Доказательство теорем

Подход, основанный на доказательстве теорем, состоит в следующем.

- Определяется множество алгебраических или логических символов, использующихся для описания поведения системы или утверждений, которые могут быть как предпосылками (известными формулами), так и следствиями (результатами вывода).
- Постулируется набор аксиом, устанавливающих правила вывода формул.
- Поведение системы описывается в виде набора доказываемых теорем.
- В доказательстве теорем используются предпосылки и аксиомы, а также ранее доказанные теоремы.

Иногда процесс доказательства можно автоматизировать с помощью определенных правил применения аксиом и теорем. Как правило, для автоматизации доказательства применяются *правила подстановки термов* (term rewriting rules). Например, широко известно, что любое булевское выражение можно механически переписать в "конъюнктивной нормальной форме" (conjunctive normal form — CNF). Однако в большинстве случаев автоматизированное доказательство теорем является слишком долгим и утомительным. Хорошо известно, что длина автоматизированного доказательства может оказаться неполиномиально ограниченной величиной, зависящей от размера доказываемой формулы (понятие неполиномиально ограниченной величины введено в разделе 4.6).

Несмотря на то что автоматизированные доказательства теорем часто оказываются слишком длинными, этот метод нашел широкое применение при исследовании систем, поведение которых невозможно описать с помощью конечной структуры (например, системы с бесконечным пространством состояний). Примером такого доказательства является метод математической индукции. Однако для получения короткого доказательства необходимо вмешательство человека.

Необходимым свойством алгебраического доказательства теоремы является аксиоматическая система, сохраняющая так называемое свойство **конгруэнтности** (congruence property). Это свойство является обобщением понятия сравнимости между целыми числами, определенного в разделе 4.3.2.5, на произвольные алгебраические структуры. Бинарное отношение R, определенное в алгебраической структуре, называется конгруэнтностью, если для любой бинарной операции $\circ$ над элементами этой структуры из условий

$$R(x, u) \text{ и } R(y, v)$$

следует утверждение

$$R(x \circ y, u \circ v).$$

Конгруэнтность иногда называют также свойством подстановки или замены (substitutability). В таком случае один компонент системы можно заменить другим,

не нарушая согласованности между элементами системы. Без этого автоматизированное доказательство теорем становится необоснованным. По этой причине свойство подстановки часто называют свойством *семантической непротиворечивости* (soundness) системы для доказательства теорем. Семантически противоречивое доказательство теорем лишено смысла, поскольку может привести к ложному утверждению, например, к абсурдному выводу "1 = 2".

Полнота (completeness) системы для доказательства теорем зависит от того, обладает ли выбранная система аксиом свойством семантической истинности. Если система для доказательства теорем является полной, любое семантически истинное утверждение должно быть доказуемым, т.е. должна существовать последовательность применения аксиом, демонстрирующая *синтаксическую* истинность утверждения. Полнота системы желательна, но, как правило, автоматизированные системы доказательства теорем ею не обладают.

Целью доказательства теоремы является демонстрация требуемых свойств системы или обнаружение ошибок. Именно поэтому нежелательное свойство нельзя сформулировать в виде теоремы. Несмотря на это, невозможность доказать требуемое свойство при доказательстве теоремы часто приводит к выявлению скрытых ошибок.

Протоколы аутентификации представляют собой чрезвычайно уязвимые системы. Как правило, невозможно сначала разработать неуязвимый протокол, а потом формально доказать его стойкость с помощью доказательства теоремы. Следовательно, этот подход необходимо применять уже на этапе разработки протокола.

17.4.2 Логика аутентификации

Одной из первых попыток формализации понятия корректности протокола была "логика аутентификации", предложенная Барроузом (Burrous), Абади и Нидхемом и получившая сокращенное название логики BAN [61]. Логику BAN можно рассматривать как один из возможных способов доказательства теорем. Она представляет собой набор логических формул, моделирующих следующие действия участников протокола на основе интуитивных рассуждений. Формулы состоят из следующих выражений и компонентов.

- Пользователь **видит** (sees) сообщение.
- Пользователь **создает** (utters) сообщение.
- Пользователь **верит** (believes or provides jurisdiction over) в истинность утверждения.
- Два (или больше) пользователя **разделяют** (share) общий секрет (сообщение).
- Шифрование сообщения.
- "Свежесть" сообщения.

- Конъюнкция логических утверждений.
- Хороший общий ключ: его не может скомпрометировать Злоумышленник и он всегда остается "свежим".

Эта система аксиом основана на интуитивных представлениях о протоколе аутентификации. Например, требование "свежести" при аутентификации сообщения описывается следующим правилом ("верификация одноразового случайного числа").

$$\frac{P \text{ верит свежий}(X) \wedge P \text{ верит}(Q \text{ сказал } X)}{P \text{ верит}(Q \text{ верит } X)}. \qquad (17.4.1)$$

Здесь выражение Q верит X означает, что пользователь Q создал сообщение X *недавно*. Эта аксиома интерпретируется следующим образом: если пользователь P верит, что сообщение X является "свежим" и создано пользователем Q, то он должен верить и в то, что пользователь Q создал сообщение X недавно.

Анализ протокола с помощью логики BAN начинается с формулировки набора предположений о протоколе. Затем выполняется "идеализация" протокольных сообщений. Она представляет собой процесс преобразования протокольных сообщений в логические формулы. После этого к аксиомам применяются логические формулы, позволяющие установить требуемые свойства, например, качество ключа.

Поскольку логика BAN относится к доказательству теорем, она не содержит механизма, позволяющего выявлять недостатки протоколов. Однако следует отметить, что процесс доказательства можно выполнить в обратном направлении: от желаемой цели — к необходимому набору аксиом. Следовательно, логика BAN может выявить важные предположения, пропущенные при спецификации протокола. Игнорирование необходимых предположений часто ведет к дефектам. Большое количество недостатков множества протоколов аутентификации было раскрыто в работе [61]. Однако для обнаружения скрытых дефектов на основе логики BAN необходимо участие высококвалифицированного эксперта, проницательного и удачливого.

Процедура идеализации протокола может оказаться уязвимой. Протоколы, описанные в литературе, как правило, сводятся к последовательности сообщений, отправленных и полученных пользователями. Чтобы воспользоваться логикой BAN, аналитик должен выразить протокол в виде формул, относящихся к сообщениям, которыми обмениваются пользователи, и допускающих применения аксиом. Например, если Трент посылает сообщение, содержащее ключ K_{AB}, этап передачи сообщения выражается следующим образом.

$$T \text{ сказал } A \xleftrightarrow{K_{AB}} B.$$

Это означает, что ключ K_{AB} является качественным. Этот этап идеализации выглядит довольно простым. Однако на самом деле процедура идеализации явля-

ется довольно сложной. Мао обнаружил, что логика BAN допускает применение контекстно-свободных процедур идеализации [183]. Например, этап идеализации, описанный выше, выполнен без учета контекста протокола. Это довольно опасно. Мао утверждает, что идеализация протокола с помощью логики BAN должна быть контекстно-зависимым процессом.

Другим недостатком логики BAN является отсутствие формального определения семантических правил, на которых основана система аксиом. В результате систему для доказательства теорем с помощью логики BAN следует считать некорректной. Кроме того, некоторые аксиомы теряют смысл. Например, в правиле верификации одноразового случайного числа (17.4.1) выражение Q верит X часто содержит ошибку интерпретации (type error) для бoьшинства значений X (одноразового случайного числа): число X, как правило, не является логической формулой даже после идеализации. (В интерпретации правила верификации одноразового случайного числа необходимые исправления уже сделаны.)

Попытки усовершенствовать семантику и обеспечить корректность логики BAN предпринимались в работах [3, 286].

Интересной модификацией логики BAN является логика GNY, разработанная Гонгом (Gong), Нидхемом и Яхаломом (Yahalom) [131], расширенная логика ван Оршота (van Oorschot) [292], а также логика идентификации, предложенная Кайларом (Kailar) [156].

Логика GNY включает в себя понятие распознаваемости, основанное на информации о типе сообщения, предотвращающей ошибки, связанные с неправильной интерпретацией, а также понятие сообщения, принадлежащего пользователю, который либо создал, либо распознал его.

Расширенная логика ван Оршота облегчает проверку протокола согласования аутентифицированных ключей в криптографии с открытым ключом. Эта система была использована для анализа трех протоколов согласования ключей Диффи–Хеллмана, включая протокол STS (протокол 11.6, изученный в разделе 11.6.1).

Логика Кайлара (Kailar) предназначена для систем электронной коммерции. Она содержит синтаксические правила, позволяющие обеспечить идентификацию пользователей.

Все эти три модели, как и логика BAN, недостаточны для создания формальной семантической модели.

Несмотря на это, логика BAN, несомненно, является важной вехой. Она породила формальный подход к анализу протоколов аутентификации.

17.5 Методы формального доказательства: исследование состояния системы

Другой подход к формальному анализу сложных систем основан на моделировании поведения конечных систем. В этом случае состояния системы выражаются с помощью определенных отношений. Анализ поведения системы, как правило, сводится к исследованию пространства состояний и проверке заданных отношений. Эта методология называется **проверкой моделей** (model checking).

Проверка моделей должна гарантировать, что либо нежелательные состояния никогда не возникнут, либо в конце концов система будет находиться в требуемом состоянии. В первом случае говорят, что проверка обеспечивает *безопасность* (safety) системы, а во втором — *живучесть* (liveness).

По-видимому, для анализа протоколов аутентификации более предпочтительным является первый подход.

17.5.1 Проверка моделей

Проверка модели сводится к следующим действиям.

- Функционирование конечной системы моделируется с помощью *конечной системы переходов*, переводящей систему из одного состояния в другое в зависимости от происходящих событий. Такие системы называются "системами помеченных переходов" (labeled transition system — LTS).
  - Например, машина Тьюринга Div3, описанная в примере 4.1, представляет собой систему LTS, изменяющую свое состояние в зависимости от бита, считанного со входной ленты.
- Каждое состояние системы LTS выражается с помощью логической формулы.
  - Например, каждое состояние машины Div3 можно выразить с помощью пропозиционального логического высказывания в множестве $\{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$. Эти утверждения имеют следующий смысл: "считанная строка битов представляет собой целое число, сравнимое с нулем, единицей или двойкой по модулю три соответственно".
- Требуемое свойство системы также явно выражается логической формулой.
  - Например, целевое состояние машины Div3 можно выразить утверждением "считанная строка битов представляет собой целое число, кратное трем".
- Осуществляется символическое выполнение системы LTS, которое описывается "следом" ("trace")

$$\pi = f_0 e_1 f_1 e_2 \ldots e_n f_n,$$

где $f_0, \ldots, f_n$ — логические формулы, а $e_1, \ldots, e_n$ — события.

- Механическая процедура может проверить, является ли целевая формула логическим следствием формулы в "следе".
  - Например, при анализе функционирования машины Div3 можно убедиться, что формула "считанная строка битов представляет собой целое число, кратное трем" является логическим следствием любого конечного "следа".

Следует отметить, что в отличие от доказательства теорем, описывающих только *желательные* свойства, при проверке модели результирующая формула может выражать как желательное, так и *нежелательное* состояние системы. Например, утверждение

"Злоумышленнику известен новый сеансовый ключ K"

представляет собой формулу, выражающую нежелательное свойство протокола распределения сеансовых ключей. Если результирующая формула выражает нежелательное свойство, то проверка модели создает "след", приводящий к явному описанию системной ошибки. Таким образом, проверку моделей можно применять для выявления ошибок, скрытых в системе. Как правило, проверку моделей при анализе протоколов аутентификации применяют именно для этой цели.

17.5.1.1 Композиция систем при проверке моделей

При разработке сложных систем их, как правило, конструируют из более простых компонентов.

Например, для создания машины Тьюринга, распознающей строки битов, кратные шести (назовем эту машину Div6), можно просто сконструировать машину, распознающую четные числа (под названием Div2), и скомбинировать ее с машиной Div3, выполнив операцию "конъюнктивной композиции" (conjunctive composition). Заметим, что разработать машины Div2 и Div3 намного проще, чем создать машину Div6 с нуля.

В рамках конъюнктивной композиции машины Div2 и Div3 должны работать синхронно, считывая свои входные ленты, на которых записана одна и та же информация, и делая очередной ход. Такая машина распознает число, кратное шести, тогда и только тогда, когда обе составляющие ее машины распознают свои входные числа. Очевидно, что метод композиции позволяет правильно сконструировать машину Div6 (более простой вариант "конъюнктивной композиции" описывается в разделе 17.5.3).

Рассмотрим "дизъюнктивную композицию", состоящую из машин Div2 и Div3. Эта машина распознает строки, кратные двум *или* трем, т.е. числа из последовательности

$$0, 2, 3, 4, 6, 8, 9, 10, 12, 14, 15, \ldots$$

Термин "дизъюнктивная композиция" означает, что сложная машина должна прекратить работу, если одна из машин Div2 или Div3 примет положительное решение. Очевидно, что, не прибегая к методу дизъюнктивной композиции, разработать машину, функционирующую столь неуклюже, довольно сложно.

Выявление дефектов в протоколах аутентификации с помощью проверки моделей также можно упростить с помощью метода системной композиции.

Действительно, поиск ошибок в протоколах аутентификации представляет собой процесс проверки системы, которая всегда крупнее, чем системы, имитирующие отдельные части протокола. В лучшем случае, спецификация протокола только описывает законные действия пользователей. Однако удачная атака всегда описывает поведение более крупной системы, в которой Злоумышленник согласовывает свое поведение с законными пользователями (т.е. обманывает их или незаметно раскрывает секреты).

Следовательно, при анализе протоколов аутентификации с помощью проверки модели необходимо моделировать не только действия законных пользователей, но и типичное поведение Злоумышленника. Каждый из этих компонентов протокола является системой LTS. Эти компоненты объединяются в более сложные системы, а затем подвергаются проверке. Операция композиции в ходе проверки моделей часто моделирует асинхронную связь между компонентами системы. Термин "асинхронный" означает, что составная система может выполнять операции в зависимости от действий подсистем. Он описывает ситуацию, в которой поведение Злоумышленника может не зависеть от действий законных пользователей.

В начале раздела мы подчеркнули, что проверка моделей предназначена для анализа конечных систем. При анализе протоколов аутентификации это значит, что Злоумышленник должен быть полиномиально ограниченным: действия, связанные с неограниченной вычислительной мощностью, не рассматриваются.

При проверке моделей часто возникает проблема "комбинаторного взрыва": система отображается в крупную систему LTS, имеющую слишком большое количество состояний, для обработки которых имеющихся вычислительных ресурсов недостаточно. Эта проблема имеет особенно острый характер при анализе крупного программного или аппаратного обеспечения. Такие системы имеют огромное пространство состояний. К счастью, при разумном предположении, что Злоумышленник является полиномиально ограниченным, большинство протоколов аутентификации можно смоделировать с помощью относительно небольших LTS-систем. Следовательно, проверка моделей особенно эффективна при анализе протоколов аутентификации.

Рассмотрим два метода проверки моделей, применяемых при анализе протоколов аутентификации.

17.5.2 NLR-анализатор протоколов

Мидоуз разработал программу, написанную на языке ПРОЛОГ, под названием *NRL-анализатор протоколов* [194]. Здесь аббревиатура NRL означает Naval Research Laboratory of the United States of America (Исследовательская лаборатория военно-морского флота США).

Как и другие средства для анализа протоколов аутентификации, NRL-анализатор протоколов основан на модели Долева–Яо [101] (см. раздел 2.3). Итак, Злоумышленник может контролировать обмен сообщениями в сети, перехватывать их, читать, модифицировать или уничтожать, выполнять преобразование перехваченных сообщений (например, владея правильными ключами, он может выполнять шифрование или расшифровку) и пересылать их другим участникам протокола, маскируясь законным пользователем. Однако вычислительные возможности Злоумышленника полиномиально ограничены. Следовательно, существует определенное множество "слов", которые остаются неизвестными как в начале выполнения протокола, так и после его завершения, если протокол оказался стойким. Эти слова могут быть секретными сообщениями или криптографическими ключами, которые защищаются протоколом. Назовем эти слова "запретными" (forbidden words).

NRL-анализатор протоколов использует метод проверки моделей и является системой перезаписи термов (term-rewriting system). Он основан на модифицированной модели Долева–Яо, имеющей название "модель перезаписи термов Долева–Яо". Будем считать, что Злоумышленник может манипулировать этой системой. Если целью Злоумышленника является распознавание запретных слов, задача доказательства стойкости протокола формулируется следующим образом: запретные слова должны остаться запретными. И наоборот, доказательство уязвимости протокола означает, что в итоге запретные слова становятся известными Злоумышленнику.

В NRL-анализаторе протоколы моделируются с помощью глобальной конечной системы. Эта система состоит из нескольких локальных подсистем и содержит определенную информацию о состояниях, доступную для Злоумышленника. Каждая локальная подсистема описывает функционирование законного участника протокола. Такой способ конструирования системы следует стандартной методологии создания сложных систем из более простых компонентов (раздел 17.5.1.1).

Включение Злоумышленника в глобальную систему моделирует способ, которым он получает информацию в результате выполнения протокола. Целью Злоумышленника является генерация "запретных" слов путем перевода законных пользователей в определенное нежелательное состояние, противоречащее протоколу. Такое состояние называется "опасным". Если протокол содержит ошибку, пользователь может оказаться в опасном состоянии. В модели перезаписи термов опасное состояние эквивалентно возникновению последовательности термов, де-

монстрирующей, что некоторое слово, которое должно быть недоступным Злоумышленнику ("запретным"), становится ему известным.

В NRL-анализаторе протокола определяется набор правил перехода из одного состояния в другое. При определенных условиях правило перехода может "сработать" ("fired").

Для того чтобы правило сработало, должны выполняться два условия.

1. Злоумышленнику приписываются определенные слова.
2. Соответствующие локальные состояния оказываются связанными с определенными значениями.

После применения правила происходит следующее.

1. Законный пользователь выводит определенные слова, которые, таким образом, становятся известными Злоумышленнику.
2. Соответствующие локальные состояния оказываются связанными с новыми значениями.

Слова, связанные с правилом, подчиняются правилам перезаписи термов. Обычно в системе существуют три правила, выражающие понятие равенства, а также тот факт, что шифрование и расшифровка являются обратными функциями. Эти правила выглядят следующим образом.

$$\text{encrypt}(X, \text{decrypt}(X, Y)) \rightarrow Y$$

$$\text{decrypt}(X, \text{encrypt}(X, Y)) \rightarrow Y$$

$$\text{id\_check}(X, X) \rightarrow \text{ДА}$$

Чтобы выполнить анализ, пользователь посылает NRL-анализатору запрос о состоянии с помощью слов, известных Злоумышленнику (т.е. описание опасного состояния). Затем NRL-анализатор протокола выполняет обратный поиск, пытаясь обнаружить исходное состояние глобальной системы. Для этого программа, написанная на ПРОЛОГЕ, пытается применить к текущему состоянию правую часть правила перезаписи термов и свести ее к левой части, которая описывает предыдущее состояние. Если исходное состояние обнаружено, система действительно является нестойкой, в противном случае выполняется попытка доказать, что исходное состояние недостижимо. Для этого программа пытается показать, что любое состояние, из которого можно перейти в указанное состояние, также является недостижимым. Эта разновидность поиска часто приводит к возникновению бесконечного следа, в котором для того, чтобы Злоумышленник распознал слово A, необходимо, чтобы он узнал слово B, а для этого он должен узнать слово C, и т.д. Анализатор обладает определенными возможностями, позволяющими пользователю доказать лемму о недостижимости определенного класса состояний. Цель этой процедуры — уменьшить пространство состояний так, чтобы стал возможным исчерпывающий поиск, позволяющий определить стойкость протокола.

Следует отметить, что основной алгоритм, использованный в NRL-анализаторе, решает задачу о достижимости определенного состояния. Хорошо известно, что такой алгоритм может оказаться бесконечным. Следовательно, необходимо ограничить количество рекурсивных вызовов некоторых проверяющих процедур. Работа с NRL-анализатором требует от пользователя большой аккуратности при кодировании правил перехода и описании опасного состояния. Кроме того, NRL-анализатор особенно эффективен при проверке протоколов согласования ключей.

NRL-анализатор использовался для проверки большого количества протоколов аутентификации и успешно обнаружил их заранее известные дефекты. В частности, с его помощью были проанализированы протокол аутентификации с открытым ключом Нидхема–Шредера (протокол 2.5) [193] (в этой работе Мидоуз сравнил результаты анализа протокола с помощью NRL-анализатора и метода Лоу, основанного на применении модели FDR [181]), "протокол избирательного вещания" ("selective broadcast protocol"), предложенный Симмонсом (Simmons) [192, 274], протокол Татебаяши–Мацузаки–Ньюмана (Tatebayashi–Matsuzaki–Newman) [160], протокол IKE (Internet Key Exchange, см. раздел 12.2.3) [135, 195] и протоколы безопасных электронных транзакций (Secure Electronic Transaction — SET) [196, 259].

17.5.3 Метод CSP

Аббревиатура CSP означает *Communicating Sequential Processes* (многостадийные процессы передачи информации) и предложена в работе Хоара (Hoare) [137]. Позднее, уточнив семантику, это название заменили на TCSP (Theoretical CSP — теоретические многостадийные процессы передачи информации) [60]. В конце концов, в работе [138] исследователи вернулись к исходному названию — CSP.

Аббревиатура CSP означает целое семейство систем, называемое **алгеброй процессов** (process algebra). Это семейство основано на алгебраическом подходе к построению абстрактных вычислительных структур (см. главу 5). Поскольку CSP — это алгебра, она представляет собой язык *термов* (terms), для которых определены *операции* (operations). Эти операции подчиняются аксиоме замыкания, т.е. термы CSP образуют множество, замкнутое относительно указанных операций (см. определения 5.1, 5.12 и 5.13 в главе 5). Каждая операция имеет определенный смысл, согласованный со смыслом термов. Множество операций в алгебре CSP будет описано позднее.

17.5.3.1 Действия и события

В алгебре CSP система описывается в терминах действий, которые она может выполнять. **Действием** (action) называется конечная последовательность событий, возникающих последовательно. В частности, действие может описываться последовательностью нулевой длины, которая означает отсутствие каких-либо действий. Множество всех возможных событий (зафиксированных перед началом

анализа) называется **алфавитом процесса** (alphabet of a process) и обозначается символом Σ. Итак, для любого действия выполняется условие $a \in \Sigma^*$. Например, алфавитом процесса может служить множество $\Sigma = \{0, 1\}$, а действием, выполняемым таким процессом, может являться битовая строка, обладающая определенным свойством.

При моделировании протоколов или систем связи с помощью алгебры CSP действие представляет собой отдельное сообщение или последовательность сообщений. Если M и N — последовательности сообщений, то $M.N$ — также последовательность сообщений. Иногда при записи последовательности сообщений символ "." пропускается.

17.5.3.2 Процессы

Процессы являются компонентами системы. Они представляют собой сущности, описываемые алгеброй CSP в терминах возможных действий, которые они могут выполнять. Основные процессы алгебры CSP и их смысл указаны на рис. 17.1.

- *STOP* ("отсутствие действий" — бездействие).
- $a \rightarrow P$("префикс:" сущность выполняет действие a, а затем функционирует, как пользователь P).
- $P \,\square\, Q$ ("детерминированный выбор": сущность ведет себя как пользователь P или Q в зависимости от внешнего события).
- $P \sqcap Q$ ("недетерминированный выбор": сущность ведет себя как пользователь P или Q по непонятным причинам).
- $/a$ ("сокрытие:" не обращать внимание на действие a).
- $\mu X.P(x)$ ("рекурсия": повторить действия пользователя P над переменной X, т.е. $\mu X.P(x) \stackrel{\text{def}}{=} P(\mu X.P(x))$).
- $P \parallel Q$ ("параллельность:" пользователи P и Q *одновременно* выполняют *одно и то же* действие).
- $P \,|||\, Q$ ("чередование:" пользователи P и Q не обязаны выполнять одно и то же действие).

Рис. 17.1. Язык CSP

Основные операции, перечисленные на рис. 17.1, представляют собой строительные конструкции процессов в алгебре CSP, предназначенные для моделирования и описания функционирования конечной системы. Эти конструкции и семан-

тика операций делают язык CSP достаточно мощным для того, чтобы описывать сложные конечные системы.

Например, машину Тьюринга Div3 из примера 4.1, представляющую собой конечную систему, можно точно описать в виде процесса CSP. В этом описании используются только "префикс", "детерминированный выбор" и "рекурсия".

Пример 17.1. CSP-спецификация машины Тьюринга Div3

$$\begin{aligned}
&\text{Div3} \stackrel{\text{def}}{=} S_0;\\
&\quad S_0 \stackrel{\text{def}}{=} (0 \to S_0) \,\square\, (1 \to S_1) \,\square\, (\{\} \to \mathit{STOP});\\
&\quad S_1 \stackrel{\text{def}}{=} (0 \to S_2) \,\square\, (1 \to S_0);\\
&\quad S_2 \stackrel{\text{def}}{=} (0 \to S_1) \,\square\, (1 \to S_2);
\end{aligned}$$

□

Процесс Div3 рекурсивно определен с помощью многих подпроцессов. Все подпроцессы, за исключением операции *STOP*, реагируют на события из множества $\Sigma = \{0, 1\}$[1]. Операция *STOP* реагирует на пустое действие, прекращая работу машины. Легко убедиться, что машина Div3 функционирует следующим образом:

$$\text{Div3} = a \to \mathit{STOP} \text{ для всех } a \in \text{DIV3} \cup \{\}.$$

17.5.3.3 "Следы"

Процесс P представляет собой множество последовательностей возможных событий $traces(P)$. К этому множеству относится также пустой "след" $\{\}$ и последовательность 1001 (возможный "след" машины Div3).

Операция "." над двумя множествами "следов" T и T' определена следуюшим образом.

$$T.T' = \left\{tr.tr' \mid tr \in T \wedge tr' \in T'\right\},$$

где операция конкатенации "." определена в разделе 17.5.3.1.

17.5.3.4 Анализ процессов

Процесс P соответствует языку L (например, его спецификации), если все его "следы" являются частью языка L.

$$P \,\mathbf{sat}\, L \Leftrightarrow traces(P) \subseteq L.$$

Процесс P уточняет процесс Q, если $traces(P) \subseteq traces(Q)$. Значит, если процесс Q соответствует языку L (т.е. $Q \,\mathbf{sat}\, L$), то процесс P также соответствует ему.

---

[1] Напомним, что в разделе 17.5.3.1 при записи атомарного сообщения $\{e\}$ скобки пропущены. Все подпроцессы, за исключением операции *STOP*, реагируют на атомарные события $\{0\}$ и $\{1\}$ из алфавита Σ^*. Процесс S_0 реагирует также на пустое действие $\{\}$, которое прекращает работу машины Div3.

Например, справедливо утверждение Div3 **sat** DIV3 $\cup \{\}$, поскольку выполняется условие $traces\{\text{Div3}\} = \text{DIV3} \cup \{\}$. Более того, процесс Div3 уточняет процесс, обрабатывающий все битовые строки. Естественно, процесс *STOP* уточняет процесс Div3 (вообще говоря, процесс *STOP* уточняет любой процесс). Немного позднее мы опишем нетривиальный процесс, уточняющий процесс Div3.

Подход, основанный на проверке моделей, позволяет механически проверять отношения между конечными процессами, используя метод уточнения расхождений (Failures Divergences Refinement — FDR), разработанный компанией Formal Systems (Europe) Ltd. Детали этого метода можно найти на Web-странице `http://www.fsel.com/index.html`.

17.5.3.5 Композиция систем в алгебре CSP

В разделе 17.5.1.1 указывалось, что композиция систем играет чрезвычайно важную роль в проверке моделей. В алгебре CSP композиция систем достигается с помощью операций "параллельность" и "чередование".

Рассмотрим пример, демонстрирующий эффективность композиции систем в алгебре CSP. В разделе 17.5.1.1 был предложен метод реализации машины Div6, распознающей целые числа, кратные шести. Этот метод рассматривает машину Div6 как совокупность машин Div2 и Div3, обрабатывающих одну и ту же строку. Хотя этот метод достаточно прост, он представляет собой лишь абстрактную идею и не позволяет конкретизировать конструкцию машину Div6.

Создав машины Div2 и Div3 с помощью операций алгебры CSP, машину Div6 можно создать *механически*.

Пример 17.2. CSP-спецификация машины Div2

$$\text{Div2} \stackrel{\text{def}}{=} R_0;$$

$$R_0 \stackrel{\text{def}}{=} (0 \rightarrow R_0) \,\square\, (1 \rightarrow R_1) \,\square\, (\{\} \rightarrow STOP);$$

$$R_1 \stackrel{\text{def}}{=} (0 \rightarrow R_0) \,\square\, (1 \rightarrow R_1);$$

□

Теперь машину Div6 легко создать, механически применив операцию "параллельность".

Пример 17.3. CSP-спецификация машины Div6

$$\text{Div6} \stackrel{\text{def}}{=} \text{Div2} \parallel \text{Div3} = R_0 \parallel S_0;$$

$$R_0 \parallel S_0 \stackrel{\text{def}}{=} (0 \to R_0 \parallel S_0) \,\square\, (1 \to R_1 \parallel S_1) \,\square\, (\{\} \to \mathit{STOP} \parallel \mathit{STOP});$$

$$R_1 \parallel S_1 \stackrel{\text{def}}{=} (0 \to R_0 \parallel S_2) \,\square\, (1 \to R_1 \parallel S_0);$$

$$R_0 \parallel S_2 \stackrel{\text{def}}{=} (0 \to R_0 \parallel S_1) \,\square\, (1 \to R_1 \parallel S_2);$$

$$R_1 \parallel S_0 \stackrel{\text{def}}{=} (0 \to R_0 \parallel S_0) \,\square\, (1 \to R_1 \parallel S_1);$$

$$R_0 \parallel S_1 \stackrel{\text{def}}{=} (0 \to R_0 \parallel S_2) \,\square\, (1 \to R_1 \parallel S_0);$$

$$R_1 \parallel S_2 \stackrel{\text{def}}{=} (0 \to R_0 \parallel S_1) \,\square\, (1 \to R_1 \parallel S_2);$$

$$\mathit{STOP} \parallel \mathit{STOP} = \mathit{STOP}.$$

□

В примере спецификация

$$\mathit{STOP} \parallel \mathit{STOP} = \mathit{STOP}$$

представляет собой “аксиому поглощения” (поглощается операция $\parallel$).

Механически созданная машина Div6 действительно правильно реализует алгоритм (читатели могут убедиться в этом самостоятельно). Механическая модель, использующая метод FDR, подтверждает, что процесс Div6 уточняет процесс Div3 и процессы Div2. Отсюда следует, что процесс Div6 действительно реализует указанный алгоритм.

Аналогично на основе машин Div2 и Div3 можно создать машину, распознающую целые числа, кратные двум или трем. Эту механическую композицию можно создать, заменяя операцию “чередование” операцией “параллельность” каждый раз, когда объединяемые термы обозначают одно и то же действие, и применяя “аксиому блокировки”.

$$\mathit{STOP} \lVert\rVert P = \mathit{STOP} \parallel P = P \lVert\rVert \mathit{STOP} = P \parallel \mathit{STOP} = \mathit{STOP}.$$

Машина, полученная в результате такой композиции, довольно велика (имеет много состояний), поэтому мы не приводим ее спецификацию.

17.5.3.6 Анализ стойкости протоколов

Операция композиции в алгебре CSP делает ее особенно полезной для моделирования и описания поведения параллельных систем и средств связи. Именно это обстоятельство побудило некоторых исследователей утверждать, что алгебра CSP позволяет анализировать протоколы аутентификации [249, 250, 253]. Кроме того, существует модель FDR [248], которая используется в сочетании с алгеброй CSP для уточнения процессов. С помощью этой модели Лоу обнаружил скрытую ошибку в протоколе аутентификации Нидхема–Шредера [181].

При анализе конфиденциальности сообщений отношение "следования" $I \vdash m$ позволяет указать, как извлечь информацию из доступных данных. Аксиомы следования, описывающие извлечение информации, перечислены на рис. 17.2.

Пусть I — исходная информация.

- Если $m \in I$, то $I \vdash m$.
- Если $I \vdash m$ и $I \subseteq I'$, то $I' \vdash m$.
- Если $I \vdash m_1$ и $I \vdash m_2$, то $I \vdash (m_1, m_2)$ (спаривание).
- Если $I \vdash (m_1, m_2)$, то $I \vdash m_1$ и $I \vdash m_2$ (проекция).
- Если $I \vdash m$ и $I \vdash K \in \mathcal{K}$, то $I \vdash \{m\}_K$ (шифрование).
- Если $I \vdash \{m\}_K$ и $I \vdash K^{-1} \in \mathcal{K}$, то $I \vdash m$ (расшифровка).

Рис. 17.2. Аксиомы следствия CSP

Например,

$$\left(\{\{K_1\}_{K_2}\}_{K_3}, K_3\right) \vdash K_3$$

и

$$\left(\{\{K_1\}_{K_2}\}_{K_3}, K_3\right) \vdash \{K_1\}_{K_2},$$

но

$$\left(\{\{K_1\}_{K_2}\}_{K_3}, K_3\right) \vdash K_1 (\text{ложь}).$$

Аксиомы следования интуитивно описывают, как Злоумышленник извлекает информацию. Пытаясь скрытно взломать протокол, Злоумышленник может использовать исходную информацию, которой он владеет, и некоторые протокольные сообщения, перехваченные в сети. Кстати, множество I, в принципе, является бесконечным. Однако в реальности для данного протокола множество I можно сузить до конечного множества "интересной" информации. Более того, исследователи пользуются тем, что на самом деле им не обязательно конструировать множество I. Достаточно проверить условие $m \in I$ для некоторого конечного набора сообщений.

Подводя итоги, укажем, что в рамках алгебры CSP применение механических средств является важным элементом анализа функционирования систем. Эти механические средства применяют набор интуитивно определенных правил. Например, при конструировании системы из небольших компонентов с помощью операции композиции можно создать более крупные системы, применяя семантические правила, перечисленные на рис. 17.1. В процессе уточнения процессов механические средства позволяют применить процедуру уточнения (раздел 17.5.3.4) и аксиомы следования (рис. 17.2).

Синтаксис CSP, учитывающий линии связи между компонентами системы, не очень удобен для человека, зато вполне пригоден для реализации с помощью механических средств. (Следует отметить, что механическое конструирование машины Div6 с помощью операции композиции, описанное в примере 17.3, является *лаконичным* (succinct), поскольку в нем отсутствует какой бы то ни было обмен информацией *между* машинами Div2 и Div3.)

Поскольку протоколы аутентификации включают в себя обмен информацией между несколькими пользователями, большинству читателей, не являющимся специалистами в области формальных методов, может показаться, что модель CSP не пригодна для их анализа. Доказательство обратного утверждения приведено в работе Райана (Rayan) и Шнайдера (Schneider) [250].

17.6 Согласование двух точек зрения на формальное доказательство стойкости

В главе 14 приведены две разные точки зрения на доказательство стойкости.

Одна из них, вычислительная, описана в главе 14 и в разделе 17.3. Она основана на подробной вычислительной модели. Рассмотрим в качестве примера стойкость шифрования. Вычислительная модель рассматривает секретность как сложность распознавания исходного текста. Иначе говоря, атакующий алгоритм, имеющий в своем распоряжении два исходных текста и зашифрованный текст одного из них, не должен догадываться, какой именно исходный текст был зашифрован. Доказательство секретности обычно проводится с помощью сведения к противоречию, причем противоречием является решение одной из трудноразрешимых задач из теории вычислительной сложности.

Другая точка зрения, символическая, была изложена в разделах 17.4 и 17.5. Она основана на простом, но эффективном использовании формальных языков. Вернемся еще раз к доказательству стойкости шифрования. Символисты считают, что секретность — это сложность восстановления исходного текста с помощью аксиом следования, перечисленных на рис. 17.2. Механическое применение этих аксиом основывается либо на методах автоматического доказательства теорем, либо на приемах исследования состояния систем.

Эти две точки зрения возникли в разных научных сообществах, между которыми долгое время существовала пропасть. Символическая точка зрения привлекательна своей простотой, однако иногда эта простота оказывается обманчивой и приводит к неверным результатам. Например, некоторые символисты считают, что цифровая подпись и шифрование с помощью одной и то же пары секретных и открытых ключей аннулируют друг друга, приводя в пример некоторые учебные криптографические алгоритмы. Намного чаще символисты рассматривают

шифрование как детерминированную функцию, побуждая инженеров применять учебные алгоритмы шифрования.

Абади и Роджуэй взяли на себя труд уничтожить пропасть между вычислительной и символической точками зрения [2]. Они считают, что эти точки зрения должны не соревноваться, а дополнять друг друга. Абади и Роджуэй разработали следующие принципы.

- Взаимодействие между вычислительной и символической теориями должно укрепить основы формальной криптологии и выявить неявные предположения, скрытые в формальных методах. Оно должно сделать доказательства стойкости протоколов более убедительными, явно сформулировав требования к реализации криптографических операций.
- Для описания постоянно усложняющихся систем необходимо применять формальные методы вычислительной криптологии. Символический подход также основан на формальных принципах и даже допускает автоматическое доказательство. Кроме того, некоторые символические подходы точно описывают определенные наивные, но важные интуитивные предположения, принятые в криптографии. Эти интуитивные подходы следует укрепить методами вычислительной криптологии.

Работа Абади и Роджуэя посвящена вычислительному обоснованию символического подхода к шифрованию. Они стремились показать, что символическая и вычислительная точки зрения являются “почти гомоморфными”. Во-первых, с вычислительной точки зрения два неразличимых зашифрованных текста считаются эквивалентными. Во-вторых, с символической точки зрения эквивалентными считаются два зашифрованных текста, смысл которых невозможно выяснить с помощью аксиом следования. Таким образом, из эквивалентности зашифрованных текстов с символической точки зрения следует их неразличимость с вычислительной точки зрения. Итак, вычислительную точку зрения на стойкость можно считать формальным базисом символической теории.

В работе Абади и Роджуэя были заложены основы для дальнейшего формального доказательства стойкости других криптографических примитивов, таких как цифровые подписи, функции хэширования, а также протоколы аутентификации или распределения аутентифицированных ключей.

17.7 Резюме

В этой главе рассмотрены протоколы аутентификации, имеющие большое прикладное значение. На этот раз изучались формальные доказательства их стойкости.

Сначала была обоснована необходимость уточнения метода, позволяющего создавать спецификации протокола. Было указано, что неточности общепринятого

метода спецификации протоколов приводят к ошибкам и неправильному использованию криптографических средств в протоколах аутентификации. Затем был предложен уточненный метод спецификации протоколов и приведено несколько примеров, демонстрирующих его эффективность.

В главе описана методология формального анализа стойкости протоколов, как с вычислительной, так и символической точки зрения.

Обе теории имеют свои преимущества и недостатки. В главе указаны попытки согласовать их между собой, предпринятые в последнее время.

Формальный анализ протоколов аутентификации все еще остается на ранней стадии развития. Таким образом, материал, изложенный в этой главе, нельзя считать окончательным решением проблемы.

Упражнения

17.1. Для создания рандомизированных зашифрованных текстов широко используется режим CBC. Обеспечивает ли он секретность с прикладной точки зрения?

Подсказка: позволяет ли режим CBC предотвратить активную атаку?

17.2. В главе 2 описан процесс уточнения протоколов аутентификации "атака-исправление". Позволяет ли этот процесс создавать стойкие протоколы? Если нет, то почему?

17.3. Примените метод уточнения спецификаций обменов сообщениями в рамках протокола Kerberos (раздел 12.4.2). Укажите правильные спецификации, необходимые для аутентификации.

Подсказка: еще раз прочтите раздел 12.4.3.

17.4. Неправильное применение криптографических средств — распространенная ошибка, возникающая при разработке протоколов аутентификации и протоколов обмена аутентифицированными ключами. В чем именно проявляется эта ошибка?

17.5. Модель Белларе–Роджуэя, предназначенная для доказательства корректности протоколов аутентификации, основана на методе сведения к противоречию. В чем именно состоит противоречие?

17.6. Можно ли применить модель Белларе–Роджуэя для проверки корректности произвольного протокола аутентификации?

17.7. Примените операцию чередования для создания процесса CSP, распознающего целые числа, кратные двум или трем.

Подсказка: прочтите еще раз раздел 17.5.3.5. Поскольку процесс уточнения является достаточно долгим, необходимо применить алгебру CSP.

Часть VI

Криптографические протоколы

В настоящее время все больше коммерческих сделок, деловых операций и государственных функций осуществляется через Internet, в частности, через World Wide Web. Многие из этих операций должны быть секретными. К ним относятся покупки, выписывание счетов, банковские услуги, правительственные распоряжения и т.п. Как правило, для этих приложений необходимы аутентификация, секретность и невозможность отрицания авторства. Эти возможности могут обеспечить простые криптографические протоколы, такие как TLS (или SSL, описанный в разделе 12).

Однако существует множество операций, которые также можно выполнить с помощью Internet, но невозможно защитить простыми протоколами вроде TLS. К таким операциям относятся микроплатежи (необходимо максимально удешевить транзакции), электронные платежи (необходимо сохранить анонимность и предотвратить вымогательство), аукционы (необходимо отделить выигравший лот от проигравших), голосование (необходимо сохранить анонимность голосующих и предотвратить принуждение), обмен (необходимо соблюдать справедливость, учитывая, что участники обладают разными ресурсами), временны́е метки и подтверждение подлинности (необходимо заверить подлинность сообщения, даже если соответствующий механизм, например, цифровая подпись, в будущем может быть взломан), восстановление синхронизированных ключей (необходимо предусмотреть возможность раскрытия секрета после t умножений).

Более тонкие операции требуют более сильных криптографических протоколов. Последняя часть книги состоит из двух глав. В главе 18 описывается класс криптографических протоколов с нулевым разглашением (zero-knowledge protocols), образующий ядро "тонких" сетевых операций: они позволяют доказать требуемое свойство без раскрытия секрета. В главе 19 излагается конкретная реализация протокола "Подбрасывание монеты по телефону" (протокол 1.1). Эта реализация представляет собой очень стойкое решение, позволяющее проводить удаленную жеребьевку, в ходе которой необходимо использовать надежную и вызывающую взаимное доверие строку битов. При этом полученное решение является практичным и применяет широко известные криптографические средства, обеспечивая эффективность, сравнимую с эффективностью криптографии с открытым ключом.

Глава 18

Протоколы с нулевым разглашением

18.1 Введение

Одна из основных задач криптографии представляет собой двустороннюю интерактивную игру, в которой один участник (доказывающая сторона) *доказывает* другому участнику (проверяющей стороне) истинность утверждения, не раскрывая сущности доказательства. В этом случае проверяющий не может самостоятельно оценить истинность утверждения, поскольку ему неизвестна информация, доступная доказывающему. Эта игра называется протоколом (системой) **интерактивного доказательства** или **IP-протоколом** (interactive proof — IP). Доказательство, осуществляемое IP-протоколом (не следует путать IP-протокол с протоколом IP сети Internet. — *Прим. ред.*), можно назвать "*секретным доказательством*" ("proof in the dark"). Секретность этого доказательства состоит в том, что, во-первых, проверяющая сторона, убедившись в истинности доказываемого утверждения, не способна самостоятельно повторить доказательство, и, во-вторых, после завершения протокола никто из посторонних не способен понять ни одного сообщения, которыми обменивались доказывающая и проверяющая стороны.

Представим себе, что утверждение, которое необходимо доказать, не раскрывая сущности доказательства, является решением какой-либо знаменитой нерешенной математической задачи (например, доказательством гипотезы Гольдбаха[1]). В этом случае доказывающая сторона, опасающаяся плагиата, может пожелать скрыть технические детали доказательства от потенциально нечестного рецензента. Для этого она должна провести "секретное" доказательство, убедив рецензента (играющего роль проверяющей стороны в IP-протоколе) в корректности выводов, не давая никакой дополнительной информации.

Во многих реальных приложениях существуют намного более серьезные основания для проведения "секретных" доказательств. Как правило, IP-протоколы применяются для аутентификации сущностей. В отличие от обычных протоколов аутентификации, в которых пользователи ставят цифровые подписи, в IP-про-

[1] Каждое четное целое число больше двух можно представить в виде суммы двух простых чисел.

токоле доказывающая сторона, подлежащая аутентификации, не желает, чтобы сообщения стали доступными кому-либо, кроме проверяющей стороны, и выполняет "секретную" аутентификацию. Кроме того, IP-протоколы часто применяются для того, чтобы доказать, что часть скрытой информации имеет определенную структуру. Это необходимо в некоторых секретных приложениях (например, при проведении электронных аукционов), в которых скрытый номер (лот) должен находиться в допустимом диапазоне (например, необходимо доказать, что $x > y$ без раскрытия значений $\mathcal{E}_k(x)$ и $\mathcal{E}_k(y)$, представляющих собой предложения цены).

Рассматривая IP-протоколы, необходимо изучить два вопроса.

Вопрос 1. Сколько информации получит проверяющая сторона в ходе интерактивного доказательства?

Вопрос 2. Сколько раундов должна выполнить доказывающая сторона, чтобы убедить проверяющего?

Идеальным ответом на первый вопрос был бы "*нисколько*", или "*нуль*". IP-протокол, обладающий таким свойством, называется **протоколом с нулевым разглашением** или **ZK-протоколом** (zero-knowledge — ZK). Второй вопрос важен не только для практических приложений, но и для теории вычислительной сложности, поскольку решение этой проблемы связано с получением более низкой оценки сложности.

Глава содержит систематическое введение в теорию протоколов с нулевым разглашением (включая ответы на поставленные выше вопросы) и описание связанных с этим понятий. Эти понятия весьма важны, однако многие из них являются результатом многолетних исследований и не опубликованы в учебниках. Чтобы упростить изложение, для их иллюстрации используются конкретные протоколы.

18.1.1 Структурная схема главы

В разделе 18.2 изложены основные понятия, связанные с IP-протоколами. В разделе 18.3 описываются разнообразные ZK-свойства IP-протоколов. В разделе 18.4 проводится дифференциация между ZK-доказательством и ZK-аргументацией. Оценки вероятностей ошибок в двусторонних протоколах приводятся в разделе 18.5. Раздел 18.6 посвящен проблеме эффективности ZK-протоколов, а раздел 18.7 — неинтерактивным ZK-протоколам.

18.2 Основные определения

Протоколы с нулевым разглашением имеют не только огромное прикладное значение в криптографии, но и лежат в основе IP-протоколов, играющих важную роль в теории вычислительной сложности. В результате эти протоколы имеют большое количество определений. Приведем определения, имеющие отношение к прикладной криптографии.

18.2.1 Вычислительная модель

Отвлечемся пока от вопросов, связанных с эффективностью и возможностью раскрытия информации.

Рассмотрим вычислительную модель **интерактивной системы доказательства** (interactive proof system), которую предложили Голдвассер, Микали и Раков [126]. Обозначим основную модель протокола интерактивных доказательств через (P, V), где P — **доказывающая** (prover), а V — **проверяющая сторона** (verificator). Как правило, протокол (P, V) предназначен для проверки принадлежности определенного предложения языку, заданному над алфавитом $\{0, 1\}^*$. Общее определение языка приведено в разделе 18.2.2, а более узкое определение, связанное с криптографическими приложениями, формулируется в разделе 18.2.3.

Пусть L — язык над алфавитом $\{0, 1\}^*$. Стороны P и V получают образец $x \in L$, представляющий собой **общие входные данные** (common input). Доказательство принадлежности образца обозначается как $(P, V)(x)$. Обе стороны протокола связаны каналом связи, через который они обмениваются информацией.

$$a_1, b_1, a_2, b_2, \ldots, a_\ell, b_\ell. \tag{18.2.1}$$

Эта последовательность сообщений называется **стенограммой доказательства** (proof transcript). В стенограмме доказательства данные, передаваемые пользователем P, называются **стенограммой доказывающей стороны** (prover's transcript). Они чередуются с данными, передаваемыми пользователем V, которые называются **стенограммой проверяющей стороны** (verifier's transcript). Как общая длина стенограммы доказательства ℓ, так и длина каждой стенограммы $|a_i|, |b_i| (i = 1, 2, \ldots, \ell)$ ограничена полиномом, зависящим от параметра $|x|$. Доказательство принадлежности образца $(P, V)(x)$ языку L также должно быть ограничено полиномом, зависящим от объема входных данных $|x|$.

Результат работы протокола записывается в виде

$$(P, V)(x) \in \{\textsf{Принять}, \textsf{Отклонить}\}.$$

Эти два значения означают, что проверяющая сторона либо подтверждает, либо опровергает утверждение $x \in L$, высказанное доказывающей стороной P. Поскольку система (P, V) является вероятностной, при каждом x результат $(P, V)(x)$ является случайной величиной, зависящей от общих входных данных x, **закрытых входных данных** (private input) пользователя P и некоторых **случайных входных данных** (random input), общих для пользователей P и Q. Более того, элементы стенограммы доказательства (18.2.1) также являются случайными величинами.

Поскольку протокол (P, V) является двусторонней игрой, естественно предположить, что каждая сторона стремится получить дополнительное преимущество. С одной стороны, доказывающая сторона P должна быть заинтересована

в результате $(P, V)(x) =$ **Принять**, даже если $x \notin L$. Доказывающая сторона, руководствующаяся такой **жульнической** стратегией (cheating prover), обозначается как $\tilde{P}$. С другой стороны, проверяющая сторона V должна быть заинтересована в раскрытии информации о закрытых входных данных игрока P. Проверяющая сторона, следующая такой **нечестной** стратегии (dishonest verifier), обозначается как $\tilde{V}$.

18.2.2 Формальное определение протоколов интерактивного доказательства

Дадим определение протокола интерактивного доказательства.

Определение 18.1. *Пусть L — язык, заданный над алфавитом $\{0,1\}^*$. IP-протокол (P, V) называется системой интерактивного доказательства для языка L, если*

$$\text{Prob}[(P, V)(x) = \textbf{Принять} \mid x \in L] \geqslant \varepsilon \tag{18.2.2}$$

и

$$\text{Prob}[(\tilde{P}, V)(x) = \textbf{Принять} \mid x \notin L] \leqslant \delta, \tag{18.2.3}$$

где числа ε и δ являются константами, удовлетворяющими условиям

$$\varepsilon \in \left(\frac{1}{2}, 1\right], \delta \in \left[0, \frac{1}{2}\right). \tag{18.2.4}$$

Вероятностное пространство состоит из всех входных значений протокола (P, V) и всех случайных входных данных пользователей P и V.

Оценка (18.2.2) характеризует **полноту** (completeness) протокола (P, V). Величина ε называется **вероятностью полноты** (completeness probability) протокола (P, V). Это значит, что если $x \in L$, то проверяющая сторона V принимает предложение x с вероятностью, которая *не меньше величины ε*.

Оценка (18.2.3) характеризует **непротиворечивость** (soundness) протокола (P, V). Величина δ называется **вероятностью противоречивости** протокола $(\tilde{P}, V)$. Это значит, что если $x \notin L$, то проверяющая сторона V принимает предложение x с вероятностью, *не превышающей величины δ*.

Сравнивая определение 18.1 с определением 4.5 (из раздела 4.4), в котором описан класс алгоритмов $\mathcal{PP}$, получаем следующий результат.

Теорема 18.1. *$\mathcal{IP} = \mathcal{PP}$, где $\mathcal{IP}$ — класс всех языков, допускающих распознавание своих предложений с помощью IP-протоколов.* □

Более того, из раздела 4.4.1 следует, что оценку вероятности полноты (соответственно противоречивости) можно увеличить (соответственно уменьшить),

приблизив ее сколь угодно близко к единице (соответственно к нулю), последовательно и независимо повторяя протокол (P, V) полиномиально ограниченное количество раз (зависящее от размера входного предложения). В этом случае проверяющая сторона V принимает решение путем голосования.

Рассмотрим введенные понятия на примере протокола 18.1.

Протокол 18.1. Протокол интерактивного доказательства принадлежности подгруппе (* см. примечание 18.1 *)

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

1. f: однонаправленная функция, определенная в группе $\mathbb{Z}_n$ и удовлетворяющая гомоморфному условию:

$$\forall x, y \in \mathbb{Z}_n : f(x+y) = f(x) \cdot f(y).$$

2. $X = f(z)$ для некоторого $z \in \mathbb{Z}_n$.

ЗАКРЫТЫЕ ВХОДНЫЕ ДАННЫЕ АЛИСЫ:

$z < n$.

ЗАКЛЮЧЕНИЕ БОБА:

$X \in \langle f(1) \rangle$, т.е. элемент X порождается элементом $f(1)$.

Следующие шаги выполняются m раз.

1. Алиса генерирует число $k \in_U \mathbb{Z}_n$, находит число $\mathsf{Commit} \leftarrow f(k)$ и посылает его Бобу.
2. Боб генерирует число $\mathsf{Challenge} \in_U \{0, 1\}$ и посылает его Алисе.
3. Алиса вычисляет число $\mathsf{Response} \leftarrow \begin{cases} k, & \text{если } \mathsf{Challenge} = 0, \\ k + z \pmod n, & \text{если } \mathsf{Challenge} = 1, \end{cases}$ и посылает его Бобу.
4. Боб проверяет значение $f(\mathsf{Response}) \stackrel{?}{=} \begin{cases} \mathsf{Commit}, & \text{если } \mathsf{Challenge} = 0, \\ \mathsf{Commit}X, & \text{если } \mathsf{Challenge} = 1. \end{cases}$ Если проверка завершается неудачно, Боб посылает отказ и прекращает работу протокола.

Боб принимает доказательство.

Пример 18.1. В протоколе 18.1 Алиса является *доказывающей*, а Боб — *проверяющей стороной*. Общими входными данными Алисы и Боба является число $X = f(z)$, где функция f является однонаправленной и гомоморфной функцией, заданной над группой $\mathbb{Z}_n$. *Утверждение о принадлежности* формулируется Алисой и выглядит следующим образом: $X \in \{f(x) \mid x \in \mathbb{Z}_n\}$. Оно означает, что элемент X принадлежит подгруппе $\langle f(1) \rangle$, поскольку $X = f(1)^z$ (см. замеча-

ние 18.1, в котором показано, почему Бобу трудно решить эту задачу). *Закрытыми входными данными* Алисы является элемент $z \in \mathbb{Z}_n$ — прообраз элемента X при однонаправленном и гомоморфном отображении f.

В протоколе 18.1 обе стороны вступают в контакт m раз и создают следующую *стенограмму доказательства.*

$$\mathsf{Commit}_1, \mathsf{Challenge}_1, \mathsf{Response}_1, \ldots, \mathsf{Commit}_m, \mathsf{Challenge}_m, \mathsf{Response}_m.$$

Протокол выводит результат Принять, если каждая проверка, выполняемая Бобом, завершается успешно. В противном случае результатом является слово Отклонить.

Описанный протокол является *полным.* Иначе говоря, если Алиса знает прообраз z и следует инструкциям, то Боб всегда будет отвечать: Принять.

Полнота

Действительно, оценка *вероятности полноты* протокола (18.2.2) выполняется, причем $\varepsilon = 1$, поскольку ответы Алисы всегда успешно проходят проверку у Боба, т.е.

$$f(\mathsf{Response}) = \begin{cases} \mathsf{Commit}, & \text{если } \mathsf{Challenge} = 0, \\ \mathsf{Commit}\ X, & \text{если } \mathsf{Challenge} = 1, \end{cases}$$

при любом выборе случайного числа $\mathsf{Challenge} \in_U \{0, 1\}$.

Протокол является *непротиворечивым.*

Непротиворечивость

Оценим вероятность противоречивости δ.

Результат проверки, выполняемой Бобом на этапе 4, зависит от случайного выбора числа Challenge после получения числа Commit от Алисы. Проверка завершается успешно в двух случаях.

Вариант $\mathsf{Challenge} = 0$: Боб видит, что Алисе известен прообраз числа Commit.

Вариант $\mathsf{Challenge} = 1$: Боб видит число

$$\text{прообраз}(X) = \mathsf{Response} - \text{прообраз}(\mathsf{Commit}) (\mathrm{mod}\, n).$$

Поскольку Алиса не может предугадать случайный выбор Боба после получения передачи, если передаваемое число не равно единице, она должна знать прообраз(Commit), а значит, и прообраз(X).

Если Алиса не знает число прообраз(X), она может сжульничать, попытавшись угадать случайный бит оклика *перед* отправкой своей передачи. В "нечестном" доказательстве Алиса вычисляет передаваемое значение следующим образом.

- Выбирает случайное число $\mathsf{Response} \in \mathbb{Z}_n$.
- Угадывает число Challenge.
- Вычисляет число $\mathsf{Commit} \leftarrow \begin{cases} f(\mathsf{Response}), & \text{если } \mathsf{Challenge} = 0, \\ f(\mathsf{Response})/X, & \text{если } \mathsf{Challenge} = 1. \end{cases}$

Очевидно, что на каждом шаге Боб может отвергнуть ложное доказательство с вероятностью $1/2$. Следовательно, вероятность *противоречивости* (т.е. вероятность успешного обмана) равна $\delta = 1/2$. Если в течение m итераций Боб ни разу не отверг доказательство, вероятность успешного обмана не превосходит 2^{-m}. Успешный обман станет практически невозможным, если число m достаточно велико, т.е. число 2^{-m} является пренебрежимо малым. Например, если $m = 100$, обман становится крайне маловероятным. Следовательно, если Боб все же принял доказательство Алисы, значит, оно является правильным.

Позднее (в разделе 18.3.1 и в примере 18.2) мы проведем дальнейшее исследование понятия *идеального нулевого разглашения* (perfect zero-knowledge-ness): если функция f действительно является однонаправленной, то Боб, вынужденный применять полиномиально ограниченный алгоритм, не может выявить никакой информации о закрытых входных данных Алисы. □

Примечание 18.1. *Если функция f является гомоморфизмом, то $f(x) = f(1)^x$ для всех $x \in \mathbb{Z}_n$. Следовательно, в протоколе 18.1 Алиса доказывает свое знание дискретного логарифма числа X по основанию $f(1)$. Этот протокол назван "доказательством принадлежности подгруппе", поскольку это название имеет более общий характер. Следует подчеркнуть, что* $\mathrm{ord}[f(1)]$ *является секретным делителем числа n, т.е. в общем случае элемент $f(1)$ не порождает группу, состоящую из n элементов. Как правило, Боб не может непосредственно проверить принадлежность элемента подгруппе, не прибегая к помощи Алисы.* □

Замечание 18.1 утверждает, что решение задачи о принадлежности элемента подгруппе является трудной задачей. Приведем еще несколько свидетельств, подтверждающих сказанное. Отметим, что, хотя множество

$$L_n = \{f(x) = f(1)^x \mid x \in \mathbb{Z}_n\}$$

является циклической группой (поскольку она порождается элементом $f(1)$, см. раздел 5.2.3), Бобу нелегко проверить условие $\#L_n \stackrel{?}{=} n$. Для этого он должен разложить число n на простые множители (т.е. убедиться, что число $f(1)$ является первообразным корнем или корнем единицы n-й степени, см. теорему 5.11 из раздела 5.4.4). Боб может ответить ДА, не вступая в контакт с Алисой, только если $\#L_n = n$ (поскольку число $f(1)$ должно порождать все n элементов множества L_n). Таким образом, задача о принадлежности элемента подгруппе сводится к факторизации большого целого числа. Следовательно, чтобы затруднить решение этой задачи, целое число n в протоколе 18.1 должно быть достаточно большим. Именно по этой причине параметр безопасности в протоколе 18.1 должен иметь длину $\log n$.

В разделе 18.3.1.1 будет показан специальный выбор общих входных данных, при котором протокол 18.1 сводится к специальному варианту задачи о вычислении дискретного логарифма.

18.2.3 Результаты из теории вычислительной сложности

Этот материал можно пропустить без ущерба для понимания дальнейшего текста.

Докажем результат из теории вычислительной сложности, сформулированный выражением (4.5.1). В главе 4 мы еще не могли доказать этот факт, а теперь можем.

В *прикладной криптографии* нас интересуют только IP-протоколы, решающие задачи о принадлежности предложений подклассу языков $\mathcal{IP}$. Для любого языка L из этого подкласса вопрос $x \stackrel{?}{\in} L$ имеет два аспекта.

1. Неизвестно, существует ли полиномиальный (по $|x|$) алгоритм, детерминированный или вероятностный, позволяющий решить эту задачу. В противном случае игрок V может обойтись без помощи игрока P.
2. Задачу можно решить с помощью полиномиального (по $|x|$) алгоритма, если ему доступно свидетельство, предоставляющее дополнительную информацию.

Очевидно, что свойства 1 и 2 характеризуют класс $\mathcal{NP}$ (раздел 4.5). Точнее говоря, они описывают NP-задачу с немногочисленными свидетельствами. Поскольку $\mathcal{IP} = \mathcal{PP}$ (теорема 18.1), выполняется условие

$$\mathcal{NP} \subseteq \mathcal{PP}.$$

Следовательно, для любого языка $L \in \mathcal{NP}$ существует IP-протокол (P, V), т.е. для любого предложения $x \in L$ результат $(P, V)(x) =$ **Принять** достигается за время, полиномиально зависящее от параметра $|x|$.

Это свойство было *конструктивно* доказано несколькими авторами. Они сконструировали ZK- (IP-) протоколы для некоторых NP-полных языков (см. определение 4.11 в разделе 4.5.1) и решили **задачу о раскраске графа тремя красками** (Голдрайх, Микали и Уигдерсон (Wigderson) [124]), а также **задачу об истинности булевского выражения** (Чаум (Chaum) [71]). Сконструировав ZK-протокол (P, V) для NP-полного языка L, задачу о принадлежности $y \in L'$ для произвольного NP-языка можно решить следующим образом.

1. Игрок P сводит задачу $y \in L'$ к задаче $x \in L$, где L — NP-полный язык (например, предложение x может представлять собой вариант задачи о раскраске графа тремя красками или задачи об истинности булевского выражения). Поскольку игрок P знает предложение $y \in L'$, это преобразование можно

выполнить за время, ограниченное полиномом, зависящим от длины предложения y. Игрок P шифрует это преобразование и посылает зашифрованный текст игроку V.

2. Игрок P проводит ZK-доказательство, предоставляя игроку V возможность проверить корректность шифрования. В разделе 18.4.2 будет убедительно показано, что ZK-доказательство корректности шифрования строки нетрудно провести, если шифрование проводилось по вероятностной схеме Голдвассера–Микали.

Очевидно, что эти два этапа образуют ZK-протокол для доказательства принадлежности $x \in L$, т.е. корректное доказательство принадлежности $y \in L'$. Заметим, что в этом доказательстве нет никаких ограничений на язык L', кроме его принадлежности классу $\mathcal{NP}$.

Кроме того, легко видеть, что на практике такое общее доказательство является неэффективным. В разделе 18.6 будет показано, что в эффективном ZK- (и IP-) протоколе количество раундов должно быть ограничено линейной функцией, зависящей от параметра безопасности. Маловероятно, чтобы это ограничение было справедливым для общего протокола, поскольку в настоящее время не существует ни одного линейного метода редукции NP-задачи к NP-полной задаче. Все известные редукции являются полиномиальными, причем имеют высокую степень. Вот почему мы утверждаем, что ZK-решение задачи о принадлежности предложения произвольному NP-языку носит чисто теоретический характер, хотя и является довольно важным. Оно представляет собой *конструктивное* свидетельство включения $\mathcal{NP} \subseteq \mathcal{PP}$.

Равенство $\mathcal{NP} = \mathcal{PP}$ остается нерешенной задачей из теории вычислительной сложности.

18.3 Нулевое разглашение

Предположим, что на вопрос 1 (раздел 18.1) существует идеальный ответ (P, V) — протокол с нулевым разглашением, т.е. пользователь $\tilde{V}$ (или V) убеждается в корректности утверждения пользователя P, не узнав *ничего* нового о его закрытых входных данных.

Для того чтобы протокол (P, V) обладал этим свойством, необходимо ограничить вычислительную мощь пользователя V (или $\tilde{V}$) полиномом, зависящим от размера его входной информации. Очевидно, что без этого ограничения нельзя гарантировать нулевое разглашение, поскольку пользователь V, обладающий неограниченными вычислительными ресурсами может самостоятельно раскрыть секретные входные данные пользователя P.

В дальнейших разделах мы рассмотрим несколько аспектов нулевого разглашения.

- Идеальное нулевое разглашение (perfect ZK) (раздел 18.3.1).
- Честная проверка (раздел 18.3.2).
- Вычислительные ZK-протоколы (раздел 18.3.3).
- Статистические ZK-протоколы (раздел 18.3.4).

18.3.1 Идеальное нулевое разглашение

Пусть (P, V) — IP-протокол для языка L. Для любого предложения $x \in L$ выполнение протокола $(P, V)(x)$ приводит не только к выводу результата Принять, но и порождает стенограмму доказательства, в котором чередуются элементы стенограмм доказывающей и проверяющей стороны. Элементы стенограммы доказательства являются случайными величинами, зависящими от всех входных данных, включая случайные входные данные протокола (P, V).

Очевидно, что доказательство $(P, V)(x)$ раскрывает любую информацию о закрытых входных данных пользователя P только в том случае, если стенограмма доказательства допускает утечку информации.

Однако, если случайные величины в стенограмме доказательства являются *равномерно распределенными* (uniformly random) по соответствующему вероятностному пространству и не зависят от общих входных данных, то бессмысленно утверждать, будто они допускают утечку информации. Будем считать, что в этой ситуации (т.е. когда стенограмма доказательства состоит из равномерно распределенных случайных величин, не зависящих от общих входных данных) доказывающая сторона общается с проверяющей стороной на языке, не обладающем свойством *избыточности* (redundancy), т.е. имеющем *наибольшую энтропию* (highest possible entropy) (см. раздел 3.7.1). Следовательно, *не имеет значения, насколько умным* (или мощным) является проверяющий пользователь. Даже изучая этот язык очень долгое время, он не сможет извлечь никакой дополнительной информации!

Докажем теперь, что протокол 18.1 представляет собой идеальный ZK-протокол.

Пример 18.2. Проанализируем протокол 18.1. Стенограмма доказательства, созданная при выполнении протокола $(Алиса, Боб)(X)$, выглядит следующим образом.

$$\mathsf{Commit}_1, \mathsf{Challenge}_1, \mathsf{Response}_1, \ldots, \mathsf{Commit}_m, \mathsf{Challenge}_m, \mathsf{Response}_m,$$

где для $i = 1, 2, \ldots, m$ выполняются следующие условия.

- $\mathsf{Commit}_i = f(k_i)$, где $k_i \in_U \mathbb{Z}_n$.

  Очевидно, поскольку Алиса извлекает числа k_i из равномерно распределенной генеральной совокупности, величины Commit_i также равномерно

распределены по пространству значений функции f и не зависят от общих входных данных X.

- $\mathsf{Challenge}_i \in \{0,1\}$.

  Боб должен извлекать бит оклика из генеральной совокупности равномерно распределенных величин, но это требование можно снять.

- $\mathsf{Response}_i = k_i + z \cdot \mathsf{Challenge}_i (\mathrm{mod}\, n)$.

  Очевидно, что благодаря равномерному распределению чисел k_i величина $\mathsf{Response}_i$ равномерно распределена по группе $\mathbb{Z}_n$ при всех значениях $\mathsf{Challenge}_i \in \{0,1\}$ (даже если $\mathsf{Challenge}$ не является равномерно распределенной случайной величиной) и не зависит от общих входных данных X.

Следовательно, данные, посланные Алисой в ходе выполнения протокола 18.1, являются равномерно распределенными и не предоставляют Бобу никакой дополнительной информации о ее закрытых данных. Значит, протокол 18.1 является идеальным протоколом с нулевым разглашением. □

Из этого примера следует, что элементы стенограммы Алисы являются равномерно распределенными независимо от того, как Боб выбирает случайные биты оклика. Иначе говоря, Боб не может повлиять на распределение чисел в стенограмме Алисы. Следовательно, протокол 18.1 является идеальным протоколом с нулевым разглашением, даже если Боб ведет нечестную игру.

В идеальном протоколе с нулевым разглашением для получения стенограммы доказательства необязательно запускать сам протокол. Эту стенограмму, представляющую собой обычную строку, можно создать с помощью жеребьевки, выполняемой в течение времени, полиномиально зависящего от длины стенограммы. Это важное свойство идеальных ZK-протоколов описывается определением 18.2.

Определение 18.2. *IP-протокол для языка L называется идеальным протоколом с нулевым разглашением, если для любого предложения $x \in L$ стенограмму доказательства $(P,V)(x)$ можно создать с помощью алгоритма $\mathcal{EQ}(x)$, время работы которого полиномиально зависит от длины предложения x, с одним и тем же распределением вероятностей.*

Как правило, эффективный алгоритм $\mathcal{EQ}$ называется **имитатором** (simulator) ZK-протокола. Однако, если (P,V) — идеальный протокол с нулевым разглашением, алгоритм $\mathcal{EQ}$ называется не имитатором, а **уравнителем** (equator).

18.3.1.1 Протокол идентификации Шнорра

В протоколе 18.1 Боб использует биты оклика. Это приводит к большой вероятности противоречивости протокола: $\delta = 1/2$. Следовательно, для того, чтобы уменьшить ошибку до 2^{-m}, необходимо повторить протокол m раз. Обычно для

предотвращения жульничества со стороны Алисы достаточно положить $m = 100$. Необходимость большого количества раундов снижает эффективность протокола.

При некоторых параметрах безопасности вероятность противоречивости протокола можно снизить, что приведет к уменьшению количества необходимых раундов. Для этого Боб должен знать разложение числа n на простые множители. Обоснование этого условия будет приведено в разделе 18.6.1. Особая ситуация возникает, когда число n является простым. Рассмотрим **протокол идентификации Шнорра** (Schnorr's Identification Protocol), предложенный в работе [256] для идентификации смарт-карт.

Протокол идентификации Шнорра является разновидностью протокола 18.1, в котором функция $f(x)$ реализуется с помощью операции $g^{-x} (\mathrm{mod}\, p)$ над конечным полем $\mathbb{F}_p$, где подгруппа $\langle g \rangle$ имеет простой порядок $q \mid p-1$. Легко видеть, что функция $g^{-x} (\mathrm{mod}\, p)$ является гомоморфной. Более того, вследствие предположения 8.2 из раздела 8.4 для достаточно больших простых чисел p и q, например, $|p| = 1024, |q| = 160$, функция $g^{-x} (\mathrm{mod}\, p)$ является однонаправленной.

При таком выборе параметров Боб может использовать слегка увеличенные оклики, состоящие из $\log_2 \log_2 p$ бит.

Примечание 18.2. *Поскольку условие $q \mid p-1$ налагается явно, протокол идентификации Шнорра больше не должен решать задачу о принадлежности элемента определенной подгруппе. Теперь Боб может самостоятельно определить, принадлежит ли элемент y подгруппе $\langle g \rangle$, не прибегая к помощи Алисы: $y^q \equiv g^q \equiv \equiv 1 (\mathrm{mod}\, p)$. Следовательно, протокол идентификации Шнорра предназначен для решения более конкретной задачи: знает ли Алиса дискретный логарифм числа y по основанию g, представляющий ее криптографический мандат.* □

18.3.1.2 Стойкость протокола идентификации Шнорра

Полнота

Это свойство выполняется тривиальным образом, причем $\varepsilon = 1$. Доказательство этого факта читатели могут провести самостоятельно (упражнение 18.7).

Непротиворечивость

Предположим, что $\widetilde{\text{Алиса}}$ жульничает, т.е. не знает правильное значение дискретного логарифма. Получив число Commit от Алисы, Боб генерирует число $\mathsf{Challenge} \in_U \{0,1\}^{\log_2 \log_2 p}$ и ожидает отзыва.

$$\mathsf{Response} = \log_g \left[\mathsf{Commit} \cdot y^{\mathsf{Challenge}} (\mathrm{mod}\, p) \right] (\mathrm{mod}\, q).$$

Это уравнение демонстрирует, что при заданных числах Commit и y существуют $\log_2 p$ разных значений Response, соответствующих $\log_2 p$ разным значениям Challenge. При небольшой величине $\log_2 p$ наилучшей стратегией вычисления

Протокол 18.2. Протокол идентификации Шнорра

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

p, q: два простых числа, удовлетворяющих условию $q \mid p - 1$.
(* Типичный размер: $|p| = 1024, |q| = 160$. *)
$g : \mathrm{ord}_p(g) = q$;
$y : y = g^{-a} (\mathrm{mod}\, p)$.
(* Кортеж (p, q, g, y) состоит из параметров открытого ключа Алисы, сертифицированного органом авторизации. *)

ЗАКРЫТЫЕ ВХОДНЫЕ ДАННЫЕ АЛИСЫ:

$a < q$.

ЗАКЛЮЧЕНИЕ БОБА:

Алисе известен некоторый элемент $a \in \mathbb{Z}_q$, удовлетворяющий условию $y \equiv g^{-a} (\mathrm{mod}\, p)$.

Следующие шаги выполняются $\log_2 \log_2 p$ раз.

1. Алиса генерирует число $k \in_U \mathbb{Z}_n$, находит число $\mathsf{Commit} \leftarrow g^k (\mathrm{mod}\, p)$ и посылает его Бобу.
2. Боб генерирует число $\mathsf{Challenge} \in_U \{0,1\}^{\log_2 \log_2 p}$ и посылает его Алисе.
3. Алиса вычисляет значение $\mathsf{Response} \leftarrow k + a \cdot \mathsf{Challenge} (\mathrm{mod}\, p)$ и посылает его Бобу.
4. Боб проверяет число $\mathsf{Commit} = g^{\mathsf{Response}} y^{\mathsf{Challenge}} (\mathrm{mod}\, p)$.

   Если проверка завершается неудачно, Боб посылает отказ и прекращает работу протокола.

Боб идентифицирует Алису.
(* Для вычисления величины $g^{\mathsf{Response}} y^{\mathsf{Challenge}} (\mathrm{mod}\, p)$ Боб должен применить алгоритм 15.2, эффективность которого эквивалентна одному возведению в степень по модулю. *)

правильного ответа по величине $\mathsf{Commit} \cdot y^{\mathsf{Challenge}} (\mathrm{mod}\, p)$ является угадывание числа Challenge перед фиксацией числа Commit.

1. Генерируем число $\mathsf{Response} \in_U \mathbb{Z}_q$.
2. Угадываем значение $\mathsf{Challenge} \in_U \{0,1\}^{\log_2 \log_2 p}$.
3. Вычисляем величину $\mathsf{Commit} \leftarrow g^{\mathsf{Response}} y^{\mathsf{Challenge}} (\mathrm{mod}\, p)$.

Очевидно, что вероятность противоречивости при правильном угадывании на каждой итерации равна $1/\log_2 p$, т.е. вероятность противоречивости протокола, состоящего из одного раунда, равна $\delta = 1/\log_2 p$.

Поскольку вероятность противоречивости протокола идентификации Шнорра, состоящего из одного раунда, меньше, чем у протокола 18.1, его эффективность

выше. Действительно, в протоколе 18.1 для снижения вероятности ошибки до пренебрежимо малой величины $\delta = 2^{-m}$ необходимо выполнить m итераций, в то время как в протоколе идентификации Шнорра для этого достаточно $\ell = \frac{m}{\log_2 \log_2 p}$ раундов.

При $p \approx 2^{1024}$ и $m = 100$ получаем, что $\ell = 100/10 = 10$. Иначе говоря, увеличение длины оклика сокращает количество итераций по сравнению с протоколом 18.1 в 10 раз при той же самой вероятности противоречивости.

Идеальное нулевое разглашение

Для общих входных данных y можно построить уравнитель $\mathcal{EQ}(y)$, время работы которого ограничено полиномом, зависящим от величины $|p|$.

1. Алгоритм $\mathcal{EQ}$ инициализирует стенограмму Transcript в виде пустой строки.
2. For $i = 1, 2, \ldots, \log_2 \log_2 p$
   а) алгоритм $\mathcal{EQ}$ генерирует число $\mathsf{Response}_i \in_U \langle g \rangle$;
   б) алгоритм $\mathcal{EQ}$ генерирует число $\mathsf{Challenge}_i \in_U \{0,1\}^{\log_2 \log_2 p}$;
   в) алгоритм $\mathcal{EQ}$ находит число $\mathsf{Commit}_i \leftarrow g^{\mathsf{Response}_i} y^{\mathsf{Challenge}_i} (\mathrm{mod}\, p)$;
   г) $\mathsf{Transcript} \leftarrow \mathsf{Transcript} \parallel \mathsf{Commit}_i, \mathsf{Challenge}_i, \mathsf{Response}_i$.

Очевидно, что строку Transcript можно создать за полиномиальное время, причем ее элементы будут распределены точно так же, как элементы реальной стенограммы доказательства.

Анализ протокола идентификации Шнорра показывает, что увеличение размера оклика уменьшает количество итераций при той же самой вероятности противоречивости. Остается неясным, почему размер оклика необходимо увеличить на такую странную и небольшую величину, как $\log_2 \log_2 p$.

Увеличение размера оклика не только повышает эффективность протокола (позитивный результат), но и имеет негативные последствия (см. раздел 18.3.2). Будьте внимательны, размер имеет значение!

18.3.2 Нулевое разглашение при честной верификации

На первый взгляд, выбор величины $|\mathsf{Challenge}| = \log_2 \log_2 p$ в протоколе идентификации Шнорра ничем не обоснован. Например, кажется, что, выбрав число $|\mathsf{Challenge}| = \log_2 p$, мы получили бы еще более эффективный протокол: для достижения той же самой вероятности противоречивости ($\delta \approx 1/p$) понадобилась бы только одна итерация. Более того, уравнитель $\mathcal{EQ}$ можно создать аналогично протоколу идентификации Шнорра. Такому уравнителю для создания строки Transcript, состоящей из равномерно распределенных случайных величин, понадобилась бы только одна итерация.

Однако сделать это довольно трудно. Попробуем разобраться в причинах.

18.3.2.1 Как действует нечестный верификатор

Предположим, что $\widetilde{\text{Боб}}$ — **нечестный верификатор** (dishonest verifier), т.е. он не следует протокольным инструкциям и постоянно пытается заставить Алису раскрыть дополнительную информацию. Предположим, что $\widetilde{\text{Боб}}$ может сгенерировать такое большое число Challenge, что число $2^{\text{Challenge}}$ является неполиномиально ограниченной величиной. Тогда Боб может изобрести трюк, заставляющий Алису создать неправильную стенограмму, которую невозможно приравнять или подделать за полиномиальное время. Если $\widetilde{\text{Боб}}$ может сделать это, то по определению 18.3.1.1 протокол не может быть длиннее идеального протокола с нулевым разглашением.

Рассмотрим немного измененный протокол идентификации Шнорра, позволяющий $\widetilde{\text{Бобу}}$ выбирать число $\text{Challenge} \in \mathbb{Z}_q$, т.е. расширять пространство окликов с $\{0,1\}^{\log_2 \log_2 p}$ до всей группы $\mathbb{Z}_q$. Вот какие действия должен выполнить $\widetilde{\text{Боб}}$, реализуя модифицированный протокол идентификации Шнорра.

Получив число Commit, он применяет подходящую псевдослучайную функцию prf с достаточно большим пространством значений $\mathbb{Z}_q$ и вычисляет значение по следующей формуле.

$$\text{Challenge} \leftarrow \text{prf}(\text{“Осмысленное сообщение, подписанное Алисой} \parallel \text{Commit”}).$$

Величина Challenge, созданная таким образом, является псевдослучайной. Смысл предложения “Осмысленное сообщение, подписанное Алисой” будет разъяснен позднее.

Бедная Алиса, неспособная отличить случайные величины от псевдослучайных (предположение 4.2), не понимает, что число Challenge является псевдослучайным, и продолжает выполнять протокол, посылая обратно величину $\text{Response} \leftarrow k + a \cdot \text{Challenge} (\text{mod}\, p)$.

Напомним, что ответ Алисы удовлетворяет условию

$$\text{Commit} = g^{\text{Response}} y^{\text{Challenge}} (\text{mod}\, p), \tag{18.3.1}$$

поскольку именно эту операцию выполняет $\widetilde{\text{Боб}}$ на этапе верификации. Следовательно, Алиса помогает Бобу создать следующее уравнение.

$$\begin{aligned}\text{Challenge} = \text{prf}(&\text{“Осмысленное сообщение, подписанное Алисой”} \parallel \\ &g^{\text{Response}} y^{\text{Challenge}} (\text{mod}\, p)).\end{aligned} \tag{18.3.2}$$

Для любого постороннего лица уравнение (18.3.2) означает одно из двух.

1. Уравнение было составлено Алисой с помощью своих закрытых входных данных. В этом случае Алиса раскрывает свою связь с $\widetilde{\text{Бобом}}$.

2. $\widetilde{\text{Боб}}$ успешно взломал псевдослучайную функцию prf с достаточно большим пространством значений $\mathbb{Z}_q$, поскольку ему удалось создать уравнение вида

$$\mathsf{Challenge} = \mathrm{prf}(\ldots y^{\mathsf{Challenge}}).$$

Как известно, эта задача практически неразрешима, поскольку функция prf является однонаправленной.

Если $\widetilde{\text{Боб}}$ применяет полиномиально ограниченный алгоритм, посторонний наблюдатель, конечно, решит, что имеет место первый вариант. В стенограмме доказательства (Commit, Challenge, Response) пара (Commit, Challenge), удовлетворяющая условиям (18.3.1) и (18.3.2), представляет собой подпись сообщения "Осмысленное сообщение, подписанное Алисой", созданную по схеме Шнорра (см. алгоритм 10.4, в котором $\mathrm{prf} = H$). Поскольку эту подпись могла поставить только Алиса (напомним, что в разделе 16.3.2 было доказано, что схема цифровой подписи Шнорра является неуязвимой для подделки с помощью атаки на основе адаптивно подобранных сообщений), посторонний наблюдатель делает правильный вывод!

Алису должно немного утешить, что раскрытая ею информация не является чрезвычайно важной (хотя это зависит от конкретного приложения). Как показано в разделе 7.5.2, если Алиса генерирует числа $k \in_U \mathbb{Z}_q$ независимо от предыдущих значений, то величина

$$\mathsf{Response} = k + a \cdot \mathsf{Challenge} (\mathrm{mod}\, q)$$

обеспечивает шифрование секретного числа a с помощью одноразового блокнота, гарантируя стойкость в теоретико-информационном смысле. Это значит, что стенограмма доказательства остается $\widetilde{\text{Бобу}}$ неизвестной, причем посторонний наблюдатель не имеет никакой информации о секретном числе a, принадлежащем Алисе.

Однако, поскольку интерактивное доказательство сводится к подписи, которая не обязательно создается интерактивным способом, защита, обещанная интерактивным протоколом, оказывается взломанной: теперь любой посторонний наблюдатель может верифицировать результат доказательства. Это значит, что теперь доказательство больше не является секретным. По этой причине модифицированный вариант протокола идентификации Шнорра нельзя считать протоколом доказательства с нулевым разглашением!

Если протокол идентификации Шнорра использует большой оклик из группы $\mathbb{Z}_q$, он называется **протоколом с нулевым разглашением при честной верификации** (honest-verifier zero-knowledge protocol). В этом случае, если верификатор честно следует инструкциям, протокол гарантирует идеальное нулевое разглашение. Это объясняется тем, что верификатор генерирует абсолютно случайные оклики, позволяющие эффективно вычислять стенограмму доказательства.

Если в протоколе с нулевым разглашением при честной верификации (P, V) поведение участника V фиксируется, и он не может вынуждать участника P создавать некорректную стенограмму доказательства или стенограмму, не допускающую моделирования, то протокол $(P, \tilde{V})$ также остается идеальным. В разделе 18.3.2.3 будет показано, что этого можно добиться, ограничив размер оклика. Существует несколько способов ограничить поведение верификатора V.

- Вынудить участника V продемонстрировать свою честность при выборе случайного оклика. Идеальный протокол с нулевым разглашением, основанный на этой идее, будет описан в разделе 18.6.2.
- Заставить верификатора V сымитировать "доказательство". В этом случае нечестный верификатор разоблачит себя. В разделе 18.7.1 будет описан чрезвычайно эффективный протокол, использующий эту идею.

18.3.2.2 Эвристический метод Шамира–Фиата

Фиат и Шамир предложили общий метод преобразования протокола с нулевым разглашением при честной верификации в схему цифровой подписи [109]. Этот метод использует способ атаки, описанный в разделе 18.3.2.1. Пусть числа Commit, Challenge, Response образуют стенограмму в протоколе с нулевым разглашением при честной верификации. Для его преобразования в цифровую подпись сообщения $M \in \{0,1\}^*$ используется функция хэширования H:

$$\mathsf{Challenge} \leftarrow H(M \parallel \mathsf{Commit}).$$

Этот способ называется **эвристическим методом Фиата–Шамира** (Fiat–Shamir heuristic).

Легко проверить, что тройная схема цифровой подписи Эль-Гамаля (раздел 16.3.1) является частным случаем схем цифровой подписи, созданных с помощью эвристического метода Фиата–Шамира. Формальное доказательство стойкости тройной схемы цифровой подписи Эль-Гамаля (описанное в разделе 16.3.2) распространяется на любую схему цифровой подписи, которую можно получить из протокола с нулевым разглашением при честной верификации с помощью эвристического метода Фиата–Шамира.

Верификатор может проверить утверждение, скрытое с помощью однонаправленной функции (например, утверждение о принадлежности, или **утверждение о секретном свидетельстве** (witness hiding claim)), используя те же методы, что и при верификации цифровой подписи, поскольку эвристический метод Фиата–Шамира предусматривает открытое, а не секретное доказательство. Довольно часто такое утверждение называют **доказательством знания** (proof-of-knowledge). Благодаря стойкости к атаке на основе адаптивно подобранных сообщений (см. раздел 16.3.2), доказательство знания является эффективным способом демонстрации утверждений, скрытых с помощью однонаправленной функции.

В некоторых приложениях требуется доказать, что секрет имеет определенную структуру. В этих случаях доказательство не обязано быть секретным (т.е. доказывающая сторона не обязана отрицать свое участие в раунде). В таких приложениях доказательство знания является вполне адекватным и очень полезным методом.

18.3.2.3 И снова об идеальных протоколах с нулевым разглашением

Рассмотрим работу исходного варианта протокола идентификации Шнорра, в котором Боб руководствуется нечестной стратегией и пытается вынудить Алису поставить подпись по схеме Шнорра.

Однако теперь для любой псевдослучайной функции prf, генерирующей $\log_2 \log_2 p$ бит, уравнение (18.3.2) может *эффективно создать* любой пользователь, иначе говоря, любой пользователь может создать уравнение относительно стенограммы доказательства.

Предположим, что $\mathcal{EQ}$ — уравнитель. Все уравнители $\mathcal{EQ}$ должны генерировать случайное число Response $\in_U \mathbb{Z}_q$ и проверять, выполняется ли равенство (18.3.2) для каждого фиксированного числа Challenge из множества $\{0,1\}^{\log_2 \log_2 p}$. Если нет, алгоритм $\mathcal{EQ}$ просто пытается сгенерировать новое число $\mathsf{Response} \in_U \mathbb{Z}_q$. Метод проб и ошибок приводит к успеху до полного исчерпания всех $\log_2 \log_2 p$ бит в пространстве значений функций prf. Поскольку длина функции prf равна $\log_2 \log_2 p$ бит, пространство ее значений можно исчерпать за $\log_2 \log_2 p$ шагов, т.е за полиномиальное время, зависящее от числа p.

Создав уравнение, алгоритм $\mathcal{EQ}$ может задать число Commit, используя формулу (18.3.1). Итак, строка

$$\mathsf{Transcript} = \mathsf{Commit}, \mathsf{Challenge}, \mathsf{Response}$$

представляет собой "стенограмму доказательства", имитирующую отдельный раунд. Эту строку можно создать за $\log p$ единиц времени. Эта стенограмма удовлетворяет условиям

$$\mathsf{Challenge} = \mathrm{prf}(\text{"Осмысленное сообщение, подписанное Алисой"} \parallel \mathsf{Commit})$$

и

$$\mathsf{Commit} = g^{\mathsf{Response}} y^{\mathsf{Challenge}} (\mathrm{mod}\, p).$$

Однако эта строка вообще не является корректной стенограммой доказательства, и Алиса *не обязана* ее создавать!

Итак, длина оклика в протоколе с нулевым разглашением имеет значение!

18.3.3 Вычислительные ZK-протоколы

Как показано выше, для того, чтобы продемонстрировать, что IP-протокол $\left(P, \tilde{V}\right)$ является идеальным ZK-протоколом, необходимо создать уравнитель, эффективно генерирующий "стенограмму", имеющую то же распределение, что

и стенограмма, порождаемая протоколом $\left(P, \tilde{V}\right)$. Это требование можно ослабить, используя понятие **вычислительного протокола с нулевым разглашением** (computational zero-knowledge protocol).

Определение 18.3. *IP-протокол (P, V) для языка L называется вычислительным протоколом с нулевым разглашением, если для любого предложения $x \in L$ стенограмму доказательства $(P, V)(x)$ можно подделать с помощью полиномиального алгоритма $S(x)$, причем распределение вероятностей поддельной стенограммы полиномиально неотличимо от истинного.*

Понятие полиномиальной неразличимости введено в определении 4.15.

Для иллюстрации вычислительных ZK-протоколов изменим протокол 18.1. Будем считать, что однонаправленная и гомоморфная функция f определена в пространстве неизвестного объема, т.е. теперь число n в группе $\mathbb{Z}_n$ не знают оба участника протокола P и V. Рассмотрим способ, позволяющий сконструировать такую функцию f.

18.3.3.1 Создание однонаправленной и гомоморфной функции $f(x)$

Предположим, что участники P и V согласовали очень большое нечетное составное целое число N, причем ни один из них не знает его разложения на простые множители. Это довольно просто сделать, если каждый из участников предложит свое случайное число, которое используется для вычисления значения N. Однако мы не будем вдаваться в излишние подробности. Будем просто считать, что пользователи согласовали некий случайный элемент $a < N$, удовлетворяющий условию $\gcd(a, N) = 1$.

Поскольку случайное число N велико, то с большой вероятностью оно имеет простой множитель p, неизвестный участникам P и V, причем число $p - 1$ должно иметь большой простой множитель q, также неизвестный обоим сторонам протокола. Оставим в стороне вопрос, насколько большой должна быть эта вероятность, однако напомним, что для случайного и большого составного числа N существование таких больших простых чисел p и q объясняется теми же причинами, что и сложность его факторизации (см. раздел 8.8).

Поскольку числа a и N являются случайными и согласованными, то с большой вероятностью мультипликативный порядок $\mathrm{ord}_N(a)$ является большим и секретным целым числом. Вероятность того, что $q \mid \mathrm{ord}_N(a)$, не может быть меньше $1 - 1/q$, поскольку для любого простого числа $q \mid \phi(N)$ количество элементов группы $\mathbb{Z}_N^*$, порядок которых является взаимно простым с числом q, не больше $1/q$.

Предположим, что пользователи P и V "определили" функцию

$$f(x) \stackrel{\text{def}}{=} a^x \pmod N \tag{18.3.3}$$

для любого целого числа $x \in \mathbb{Z}_N$. Мы взяли слово "определили" в кавычки, поскольку областью определения этой функции является группа $\mathbb{Z}_{\mathrm{ord}_N(a)}$, а не $\mathbb{Z}_N$, т.е. для любого $x \in \mathbb{Z}_N$ всегда выполняется условие

$$f(x) = f(x(\mathrm{mod}\,\mathrm{ord}_N(a))).$$

Иначе говоря, аргументы функции f всегда принадлежат пространству $\mathbb{Z}_{ord_N(a)}$, размер которого меньше размера пространства $\mathbb{Z}_N$.

Легко видеть, что функция $f(x)$ является гомоморфной и однонаправленной. Ее гомоморфизм тривиально следует из равенства

$$f(x+y) = a^{x+y} = a^x \cdot a^y (\mathrm{mod}\, N).$$

Однонаправленность этой функции основана на невозможности вычислить дискретный логарифм по модулю p (напомним, что большое простое число $p \mid N$ нам неизвестно): найти число x из уравнения $f(x) = f(1)^x (\mathrm{mod}\, N)$ намного сложнее, чем определить число $x(\mathrm{mod}\, p-1)$ из уравнения $f(1)^x (\mathrm{mod}\, p)$, поскольку по предположению 8.2 функция $f(1)^x (\mathrm{mod}\, p)$ является однонаправленной.

18.3.3.2 Вычислительный протокол с нулевым разглашением

Используя функцию $f(x)$, построенную в разделе 18.3.3.1, можно сконструировать вычислительный протокол с нулевым разглашением.

Пример 18.3. Пусть (Алиса, $\widetilde{\text{Боб}}$) — вариант протокола 18.1, в котором используется гомоморфная однонаправленная функция $f(x)$, сконструированная в разделе 18.3.3.1, т.е. функция, определенная по формуле (18.3.3).

Теперь Алиса больше не знает число $n = \mathrm{ord}_N(a)$ и не может извлекать случайные числа из пространства $\mathbb{Z}_{\mathrm{ord}_N(a)}$, имеющие равномерное распределение. Для того чтобы Алиса могла провести доказательство (т.е. чтобы сохранялось свойство полноты), инструкции протокола для Алисы необходимо немного исправить (будем считать, что $z < N$).

1. Алиса генерирует число $k \in_U \mathbb{Z}_{N^2-z}$, вычисляет значение $\mathsf{Commit} \leftarrow f(k)$ и посылает его Бобу.
2. Боб . . . (* Без изменений. *)
3. Алиса вычисляет значение $\mathsf{Response} \leftarrow \begin{cases} k, & \text{если } \mathsf{Challenge} = 0, \\ k+z, & \text{если } \mathsf{Challenge} = 1, \end{cases}$ и посылает его Бобу.
4. Боб . . . (* Без изменений. *)

В этой модификации инструкции Боба остались неизменными. Однако в инструкции Алисы внесены два изменения. На первом шаге случайное число k

извлекается из группы $\mathbb{Z}_{N^2-z}$. Выбор такого пространства будет обоснован позднее. На шаге 3 (если Challenge $= 1$) Алиса вычисляет значение Response $\leftarrow k+z$, используя сложение в целочисленном пространстве $\mathbb{Z}$, т.е. без приведения по модулю. Алиса не может выполнить приведение по модулю, поскольку модуль $n = \mathrm{ord}_N(a)$ ей неизвестен.

Полнота и непротиворечивость этой модификации доказывается, как и прежде.

Однако теперь мы не можем больше утверждать, что этот вариант протокола является идеальным протоколом с нулевым разглашением, поскольку не можем сконструировать эффективный алгоритм создания уравнения, позволяющий создать стенограмму "доказательства", имеющую то же распределение, что и протокол $(\text{Алиса}, \widetilde{\text{Боб}})(X)$.

Действительно, обычные методы моделирования создают стенограмму с другим распределением. Как правило, алгоритм моделирования $\mathbb{S}$ сводится к следующим этапам.

1. Алгоритм моделирования $\mathcal{S}$ генерирует число Response $\in_U \mathbb{Z}_{N^2}$.
2. Алгоритм моделирования $\mathcal{S}$ генерирует число Challenge $\in_U \{0,1\}$.
3. Алгоритм моделирования $\mathcal{S}$ вычисляет значение Commit $\leftarrow f(\text{Response})/X^{\text{Challenge}} (\mathrm{mod}\, N)$.

Когда Challenge $= 1$, величина Response в стенограмме доказательства имеет равномерное распределение в интервале $[z, N^2)$, а в поддельной стенограмме величина Response равномерно распределена в интервале $[0, N^2)$. Это *разные* распределения. Если бы не число z, алгоритм $\mathcal{S}$ просто не смог бы моделировать поведение Алисы!

Несмотря на это, вариант $(\text{Алиса}, \widetilde{\text{Боб}})$ является вычислительным протоколом с нулевым разглашением. Это объясняется тем, что распределения величин $x \in {}_U[z, N^2)$ и $y \in_U [0, N^2)$ при $z < N$ неразличимы с вычислительной точки зрения. Из оценки

$$\mathrm{Prob}[y \leqslant z < N \mid y \in_U [0, N^2)] < \frac{N}{N^2} = \frac{1}{N} \tag{18.3.4}$$

следует, что

$$\left|\mathrm{Prob}\left[\text{Response} \in_U [z, N^2)\right] - \mathrm{Prob}\left[\text{Response} \in_U [0, N^2)\right]\right| < \frac{1}{N}.$$

По определению 4.15 (из раздела 4.7) значения Response в истиной и поддельной стенограмме доказательства невозможно отличить. Таким образом, мы построили полиномиальный имитатор $\mathbb{S}$, т.е. протокол $(\text{Алиса}, \widetilde{\text{Боб}})$ является вычислительным протоколом с нулевым разглашением. □

Теперь мы в состоянии объяснить, почему Алиса извлекает числа k из необычного пространства $\mathbb{Z}_{N^2-z}$.

Во-первых, поскольку алгоритм не предусматривает операции приведения по модулю, из числа N^2 необходимо вычесть z, чтобы число Response не стало больше N^2. Если это случится, протокол потеряет смысл!

Во-вторых, само число N^2 необходимо для того, чтобы оценка вероятности (18.3.4) и сам протокол гарантировали вычислительную эффективность. На самом деле число N^2 не обязательно должно быть очень большим. Вычислительный протокол с нулевым разглашением можно сконструировать при $N^{1+\alpha}$, где $\alpha > 0$ — произвольная константа. Читатели могут сами доказать это утверждение (подсказка: в правой части выражения (18.3.4) число $\frac{1}{N}$ можно заменить на $\frac{1}{N^\alpha}$).

В реальных приложениях протоколов с нулевым разглашением (например, в протоколе идентификации Шнорра) большинство однонаправленных функций реализуются с помощью доступных криптографических средств (например, так, как показано в разделе 18.3.3.1). Следовательно, вычислительные протоколы с нулевым разглашением являются наиболее важным прикладным понятием в теории ZK-протоколов.

18.3.4 Статистические ZK-протоколы

Голдвассер, Микали и Ракофф [126] ввели понятие **статистического протокола с нулевым разглашением** (statistical zero-knowledge protocol). IP-протокол называется статистическим протоколом с нулевым разглашением, если существует эффективный имитатор, позволяющий подделать стенограмму доказательства с точностью, которая не поддается статистическому распознаванию. Статистический алгоритм распознавания аналогичен полиномиальному алгоритму распознавания, введенному в определении 4.14, за исключением того, что время его работы не обязательно полиномиально ограничено. Отсюда следует, что статистический ZK-протокол подчиняется более строгим ограничениям, чем вычислительный.

По существу, вычислительный протокол (Алиса, Боб) в примере 18.3.3.1 является статистическим, поскольку выражение (18.3.4) означает, что событие "число Response в поддельной стенограмме меньше числа z" возникает с вероятностью, которая меньше пренебрежимо малой величины $1/N$.

Итак, с вероятностью больше $(N-1)/N$ число Response в обеих стенограммах превышает число z и является равномерно распределенным. Эти значения невозможно различить, даже если алгоритм будет работать бесконечно долго!

В принципе, статистические и вычислительные протоколы с нулевым разглашением несущественно отличаются друг от друга. Несмотря на это, поскольку понятие статистического протокола носит более строгий характер, в реальных приложениях оно более предпочтительно.

18.4 Доказательство или аргументация?

Ранее было явно указано, что для того, чтобы IP-протокол (P, V) обладал свойством нулевого разглашения (в любом из четырех указанных смыслов) вычислительная мощность верификатора V и $\tilde{V}$ должна быть ограничена полиномом, зависящим от размера общих входных данных. Однако до сих пор мы не делали никаких предположений о вычислительной мощи доказывающей стороны P или $\tilde{P}$.

18.4.1 Аргументация с нулевым разглашением

Внимательный читатель мог заметить, что, описывая все протоколы с нулевым разглашением, рассмотренные выше, мы считали, что участник P (или $\tilde{P}$) имеет полиномиально ограниченную вычислительную мощь. Однако доказательство непротиворечивости этих протоколов мы всегда начинали словами "если пользователь P (или $\tilde{P}$) не знает прообраз числа X ...".

Для языков из класса $\mathcal{IP} = \mathcal{PP}$ из этого утверждения следует, что пользователь P (или $\tilde{P}$) применяет полиномиально ограниченный алгоритм. Если предположить, что пользователь, применяющий неограниченный алгоритм, может вычислить прообраз однонаправленной функции f, то все предыдущие рассуждения о непротиворечивости протоколов становятся некорректными. Совершенно ясно, что, используя неограниченный алгоритм, пользователь P (или $\tilde{P}$) может по числу Challenge определить величину Response.

$$\mathsf{Response} \leftarrow \text{прообраз}\left(\mathsf{Commit} \cdot X^{\mathsf{Challenge}}\right).$$

В этом случае невозможно получить оценку вероятности противоречивости δ в выражении (18.2.3). При доказательстве непротиворечивости всех протоколов, рассмотренных ранее, величина δ была получена при неявном предположении, что пользователь P (или $\tilde{P}$) имеет ограниченные вычислительные ресурсы.

ZK-протокол (P, V) для языка L, в котором требуется, чтобы участник P (или $\tilde{P}$) имел полиномиально ограниченную вычислительную мощь, называется **протоколом аргументации с нулевым разглашением** (zero-knowledge argument protocol). Как правило, это требование необходимо для обоснования непротиворечивости протокола. Аргументация — это не строгое доказательство. В частности, она становится неубедительной, если пользователь P (или $\tilde{P}$) обладает неограниченными вычислительными ресурсами.

Итак, все протоколы, рассмотренные ранее, — идеальный, с честной верификацией, вычислительный и статистический — являются протоколами *аргументации* с нулевым разглашением. Протокол идентификации Шнорра также относится к этой категории. По существу, мы до сих пор не рассматривали ни одного **протокола доказательства с нулевым разглашением** (zero-knowledge proof protocol).

Перед тем как перейти к описанию протоколов доказательства с нулевым разглашением, необходимо четко разъяснить один вопрос. В большинстве реальных приложений, основанных на современных криптографических методах, пользователи секретной системы (включая доказывающую сторону в протоколе с нулевым разглашением), как правило, используют полиномиально ограниченные алгоритмы, а значит, не могут быстро решать NP-сложные задачи. Следовательно, протоколы аргументации с нулевым разглашением остаются весьма полезным понятием.

18.4.2 Доказательство с нулевым разглашением

В протоколе доказательства с нулевым разглашением непротиворечивость устанавливается без требования полиномиальной ограниченности пользователя P (или $\tilde{P}$).

Рассмотрим конкретный пример, в котором осуществляется доказательство того, что некое число является квадратичным вычетом. Этот протокол снова сводится к решению задачи о принадлежности: $x \in \mathrm{QR}_N$, где N — нечетное составное целое число.

18.4.2.1 Доказательство принадлежности числа множеству квадратичных вычетов

Пусть N — большое нечетное составное целое число, имеющее по крайней мере два простых нечетных множителя. В разделе 6.5 были рассмотрены квадратичные вычеты по модулю и установлены следующие факты из теории чисел.

Факт 1. Зная разложение числа N, с помощью алгоритма 6.5 для любого $x \in \in \mathrm{QR}_N$ можно эффективно вычислить квадратный корень y, удовлетворяющий условию $y^2 \equiv x (\mathrm{mod}\, N)$.

Факт 2. Для любого $x \in \mathrm{QNR}_N$ (квадратичного невычета) в группе $\mathbb{Z}_N^*$ не существует ни одного квадратного корня (шаг 1 в алгоритме 6.5 невыполним).

Факт 3. Если из условия $x \in \mathrm{QNR}_N$ следует, что $x \cdot y \in \mathrm{QR}_N$, то $y \in \mathrm{QNR}_N$ (читатели могут сами убедиться в этом, проверив все возможные варианты символов Якоби для чисел x, y и $x \cdot y$).

Используя эти факты, можно создать идеальный протокол *доказательства* с нулевым разглашением, позволяющий Алисе *доказать* Бобу, что заданное число является квадратичным вычетом по нечетному составному модулю. Этот протокол разработали Голдвассер, Микали и Раков [126].

Непротиворечивость

Предположим, что $x \in \mathrm{QNR}_N$ (т.е. Алиса жульничает). Оценим вероятность противоречивости δ. Разумеется, будем предполагать, что Алиса обладает неограниченными вычислительными ресурсами.

Протокол 18.3. Протокол доказательства с нулевым разглашением принадлежности числа множеству квадратичных вычетов

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

N: большое нечетное составное число, не являющееся степенью простого числа.

x: элемент множества QR_N.

ЗАКРЫТЫЕ ВХОДНЫЕ ДАННЫЕ АЛИСЫ:

$y \in \mathbb{Z}_N^* : y^2 \equiv x (\mathrm{mod}\, N)$.

ЗАКЛЮЧЕНИЕ БОБА:

$x \in \mathrm{QR}_N$.

Следующие шаги выполняются m раз.

1. Алиса генерирует число $u \in_U \mathrm{QR}_N$, вычисляет значение Commit $\leftarrow u^2 (\mathrm{mod}\, N)$ и посылает его Бобу.
2. Боб генерирует число $\mathsf{Challenge} \in_U \{0, 1\}$ и посылает его Алисе.
3. Алиса вычисляет $\mathsf{Response} \leftarrow \begin{cases} u, & \text{если } \mathsf{Challenge} = 0, \\ uy (\mathrm{mod}\, N), & \text{если } \mathsf{Challenge} = 1, \end{cases}$ и посылает его Бобу.
4. Боб проверяет значение

$$\mathsf{Response}^2 (\mathrm{mod}\, N) \equiv \begin{cases} \mathsf{Commit}, & \text{если } \mathsf{Оклик} = 0, \\ \mathsf{Commit}\ X (\mathrm{mod}\, N), & \text{если } \mathsf{Оклик} = 1. \end{cases}$$

Если проверка завершается неудачно, Боб посылает отказ и прекращает работу протокола.

Боб принимает доказательство.

Если $\mathsf{Challenge} = 0$, Боб видит, что число Response является квадратным корнем передаваемого числа, т.е. $\mathsf{Commit} \in \mathrm{QR}_N$.

Если $\mathsf{Challenge} = 1$, Боб видит, что число Response является квадратным корнем числа $\mathsf{Commit} \cdot x$, т.е. $\mathsf{Commit} \cdot x \in \mathrm{QR}_N$. Из факта 3 следует, что в этом случае $\mathsf{Commit} \in \mathrm{QNR}_N$.

Итак, если $x \in \mathrm{QNR}_N$, Боб видит, что $\mathsf{Commit} \in \mathrm{QR}_N$ или $\mathsf{Commit} \in \mathrm{QNR}_N$ в зависимости от того, какой случайный бит оклика он посылает Алисе: 0 или 1. Поскольку Алиса послала число Commit до того, как Боб извлек случайный бит оклика, она должна была правильно предугадать этот бит. Очевидно, что в этом случае вероятность ошибки равна $\delta = 1/2$. Поскольку Боб выполняет верификацию m раз, вероятность противоречивости протокола равна 2^{-m}.

Непротиворечивость протокола выполняется благодаря факту 2, вследствие которого даже Алиса, обладающая неограниченными вычислительными ресурсами, не может вычислить квадратный корень числа $x \in \mathrm{QNR}_N$ и угадать случайный бит оклика.

Полнота и идеальность нулевого разглашения

Полнота протокола непосредственно следует из факта 1.

Идеальность протокола можно продемонстрировать, сконструировав алгоритм уравнивания $\mathcal{EQ}$, генерирующий стенограмму доказательства.

1. Алгоритм $\mathcal{EQ}$ генерирует число $\mathsf{Response}_i \in_U \mathbb{Z}_N^*$.
2. Алгоритм $\mathcal{EQ}$ генерирует число $\mathsf{Challenge}_i \in_U \{0, 1\}$.
3. Алгоритм $\mathcal{EQ}$ вычисляет

$$\mathsf{Commit}_i \leftarrow \begin{cases} \mathsf{Response}_i^2 (\mathrm{mod}\, N), & \text{если } \mathsf{Challenge} = 0, \\ \mathsf{Response}_i^2 / x (\mathrm{mod}\, N), & \text{если } \mathsf{Challenge} = 1. \end{cases}$$

Легко проверить, что элементы поддельной и подлинной стенограмм имеют одинаковое распределение.

18.4.2.2 Доказательство принадлежности числа множеству квадратичных невычетов

Используя идеи, лежащие в основе протокола 18.3, можно создать протокол доказательства с нулевым разглашением принадлежности числа множеству квадратичных невычетов. Основная идея заключается в следующем.

При общих входных данных $x \in \mathrm{QNR}_N$ Боб может послать Алисе случайный оклик, используя формулы $\mathsf{Challenge} \leftarrow r^2 (\mathrm{mod}\, N)$ или $\mathsf{Challenge}' \leftarrow x r^2 (\mathrm{mod}\, N)$, где r — случайный элемент группы $\mathbb{Z}_N^*$. Очевидно, что $\mathsf{Challenge} \in \mathrm{QR}_N$, и Алиса, проверив это, может ответить ДА. С другой стороны, если число x действительно принадлежит множеству QNR_N, то по факту 3 $\mathsf{Challenge}' \in \mathrm{QNR}_N$. В этом случае Алиса также видит это и отвечает НЕТ.

Многократно посылая Алисе случайные оклики то из множества QR_N, то из множества QNR_N, Боб может проверить факт $x \in \mathrm{QNR}_N$, ориентируясь на правильные ответы Алисы. Подробное описание этого протокола приведено в работе [126].

Доказательства с нулевым разглашением принадлежности числа множеству квадратичных вычетов или невычетов позволяют проверить корректность шифрования произвольной битовой строки в схеме вероятностного шифрования Голдвассера–Микали (алгоритм 14.1). Кроме того, из них следует важный теоретический результат, изложенный в разделе 18.2.3.

18.5 Протоколы с двусторонней ошибкой

Во всех протоколах с нулевым разглашением (доказательства или аргументации), изученных нами до сих пор, вероятность полноты (18.2.2) всегда оценивалась величиной $\varepsilon = 1$, а вероятность противоречивости всегда была ненулевой — $\delta > 0$. Если $\varepsilon = 1$, то протокол является абсолютно полным, т.е., если доказывающая сторона не жульничает, верификатор всегда принимает доказательство. Используя терминологию, введенную для описания ошибок в рандомизированных алгоритмах (раздел 4.4), можно сказать, что все эти протоколы имеют **односторонюю ошибку** (one-sided error) и принадлежат подклассу Монте-Карло (т.е. "всегда высокочувствительные и, вероятно, точные", см. раздел 4.4.3). В таких протоколах односторонняя ошибка может совершаться Алисой, т.е. Алиса может жульничать и пытаться "доказать", что $x \in L$, хотя на самом деле $x \notin L$, а обманутый Боб может принять это "доказательство" (хотя вероятность противоречивости δ может сделать сколь угодно малой, независимо повторяя процесс доказательства).

Некоторые протоколы с нулевым разглашением допускают также возможность ошибки со стороны верификатора (Боба). Иначе говоря, в оценке вероятности полноты (18.2.2) $\varepsilon < 1$. Такие протоколы называются **протоколами с двусторонними ошибками** (two-sided errors) и принадлежат подклассу Атлантик-Сити (т.е. "вероятно, высокочувствительные и, вероятно, точные", см. раздел 4.4.5). Рассмотрим один из таких протоколов.

18.5.1 Доказательство с нулевым разглашением для разложения на два простых множителя

Протоколы доказательства с нулевым разглашением оказываются очень полезными при обосновании того, что нечетное составное целое число N имеет только два простых множителя, т.е. $N \in E_{2\_\text{Prime}}$ и является корректным RSA-модулем.

Как сказано в разделе 4.7, язык E_{2_Prime} называется **ансамблем** (ensemble). Любой элемент этого языка представляет собой нечетное составное целое число, имеющее два разных простых множителя. Такой язык невозможно отличить от другого ансамбля, языка E_{3_Prime}, состоящего из нечетных составных чисел, имеющих три простых множителя.

Предположим, что Алиса создает большое число $N \in E_{2\_\text{Prime}}$, зная его разложение на простые множители (т.е. просто перемножает два разных простых нечетных числа). С помощью идеального протокола с нулевым разглашением она может *доказать* Бобу, что $N \in E_{2\_\text{Prime}}$. Это доказательство использует все три факта из теории чисел, приведенных ранее, и два новых.

Факт 4. Если $N \in E_{2\_\text{Prime}}$, то ровно половина элементов из множества

$$J_N(1) = \left\{ x \mid x \in \mathbb{Z}_N^*, \left(\frac{x}{n}\right) = 1 \right\}$$

являются квадратичными вычетами, т.е. $\#\mathrm{QR}_N = \#J_N(1)/2$. Это объясняется тем, что только половина элементов этого множества имеет положительный символ Лежандра по обоим простым модулям. В то же время, чтобы символ Якоби остальных элементов был положительным, их символ Лежандра по обоим простым модулям должен быть отрицательным.

Факт 5. Если $N \notin E_{2\_\mathrm{Prime}}$ и число N не является ни простым, ни степенью простого числа, то не больше одной четверти элементов множества $J_N(1)$ являются квадратичными вычетами, т.е. $\#\mathrm{QR}_N \leqslant \#J_N(1)/4$. Этот факт обобщает факт 4 на ситуации, когда число N имеет три и больше простых множителя. Напомним, что для того, чтобы число x принадлежало множеству QR_N, необходимо, чтобы для каждого простого числа $p \mid N$ выполнялось условие $x(\mathrm{mod}\, p) \in \mathrm{QR}_p$.

При формулировке факта 5 мы потребовали, чтобы число N не было степенью простого числа. В противном случае, когда $N = p^i$, где i — целое число, все элементы множества $\mathrm{J}_N(1)$ являются квадратичными вычетами. К счастью, степень простого числа легко разложить на множители (см. подсказки к упражнениям 8.7 и 8.8).

Протокол 18.4 позволяет Алисе провести доказательство с нулевым разглашением принадлежности числа множеству E_{2_Prime}.

Исследуем стойкость протокола 18.4.

18.5.1.1 Стойкость

Во-первых, очевидно, что свойство идеального нулевого разглашения непосредственно наследуется от протокола 18.3. Итак, достаточно проанализировать полноту и непротиворечивость.

Полнота

Предположим, что Алиса честно сконструировала число $N \in E_{2\_\mathrm{Prime}}$. Однако после выполнения протокола Боб отказался принять ее доказательство. Он мотивирует это тем, что в случайном множестве Challenge оказалось меньше $3/8$ квадратов (Алисе не повезло!). Это может произойти, только если $\varepsilon < 1$.

В протоколах, рассмотренных ранее, верификатор не прощал никаких ошибок, даже если они происходили только один раз в ходе многократных раундов. Эти протоколы являются протоколами с односторонней ошибкой: если доказывающая сторона не жульничает, вероятность полноты равна $\varepsilon = 1$, а значит, верификатор не должен прощать ни одной ошибки. Поскольку в протоколе 18.4 $\varepsilon = \frac{1}{2}$ (если Алиса не жульничает, см. факт 4), Боб может выбрать больше половины невычетов и должен прощать некоторое количество ошибок. Однако, если количество ошибок превышает определенную величину, Боб должен понять, что Алиса жульничает, и отвергнуть ее доказательство.

Протокол 18.4. Протокол доказательства с нулевым разглашением того, что число N имеет два простых множителя

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

составное целое число N.

ЗАКРЫТАЯ ИНФОРМАЦИЯ АЛИСЫ:

разложение числа N.

ЗАКЛЮЧЕНИЕ БОБА:

$N \in E_{2\_\text{Prime}}$.

1. Боб удостоверяется, что N не является ни простым числом, ни степенью простого числа (например, применяя алгоритм Prime_Test и подсказку к упражнению 8.7).
2. Боб генерирует множество Challenge, состоящее из m случайных чисел, принадлежащих множеству $J_N(1)$, и отсылает его Алисе.
3. Обозначим через $x_1, x_2, \ldots, x_k$ все квадраты, принадлежащие множеству Challenge. Используя протокол 18.3, Алиса доказывает Бобу, что эти k элементов принадлежат множеству QR_N.
4. Если $k > \lfloor \frac{3}{8}m \rfloor$, Боб принимает доказательство, в противном случае он его отвергает.

   (* Здесь условие $k > \lfloor \frac{3}{8}m \rfloor$ представляет собой "практический критерий выбора меньшинством голосов". Сравните его с "критерием выборов большинством голосов", в котором $k > \frac{1}{2}m$. В рассматриваемом протоколе выбор большинством голосов применить просто невозможно, поскольку квадратичные вычеты числа N не образуют большинства в множестве $J_N(1)$. Выбор критерия голосования обосновывается в разделе 18.5.1.2. *)

Если Алиса не жульничала, и все же ее доказательство было отвергнуто, будем говорить, что произошло событие BadLuckAlice. Оценим его вероятность, учитывая критерий принятия решений, которым руководствуется Боб. Если $k > \lfloor \frac{3}{8}m \rfloor$, т.е. количество квадратичных вычетов в множестве окликов превышает $\frac{3}{8}$, Боб принимает доказательство, в противном случае он его отвергает. Выбор такого критерия голосования обосновывается в разделе 18.5.1.2.

Оценим вероятность полноты $\varepsilon(m)$ после m раундов, выразив ее через вероятность события BadLuckAlice.

$$\begin{aligned}\text{Prob}[\text{BadLuckAlice}] = \\ = \text{Prob}\left[\text{Боб отклонит доказательство} \mid N \in E_{2\_\text{Prime}}\right] < \\ < 1 - \varepsilon(m).\end{aligned}$$

Если $m = \#\mathsf{Challenge} < \#J_N(1)$, событие BadLuckAlice представляет собой сумму m испытаний Бернулли (см. раздел 3.5.2), состоящих из k успехов и $m-k$ испытаний, где $k \leqslant \lfloor \frac{3}{8}m \rfloor$. Поскольку Алиса сконструировала число $N \in E_{2\_\mathrm{Prime}}$, вероятность успеха и неудачи в множестве Challenge, состоящем из случайных элементов множества $J_N(1)$, равна $1/2$. Оценивая "левый хвост" биномиального распределения (раздел 3.5.2) и учитывая все возможные варианты числа k, получаем

$$1-\varepsilon(m) = \mathrm{Prob}[\mathsf{BadLuckAlice}] = \\ = \sum_{k=0}^{\lfloor \frac{3}{8}m \rfloor} \binom{m}{k} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{m-k} = \\ = \sum_{k=0}^{\lfloor \frac{3}{8}m \rfloor} \binom{m}{k} \left(\frac{1}{2}\right)^m.$$

Эта величина представляет собой площадь, ограниченную "левым хвостом" биномиального распределения, поскольку число $\frac{3}{8}m$ лежит слева от средней точки, равной $\frac{1}{2}m$.

Для того чтобы сделать вероятность события BadLuckAlice пренебрежимо малой, следует выбрать число $m = 2\,000$ (обоснование приведено в разделе 18.5.1.2). Тогда площадь фигуры, ограниченной "левым хвостом" биномиального распределения, равна следующей величине.

$$1-\varepsilon(2\,000) \approx 1{,}688 \cdot 10^{-29}.$$

Следовательно, $\varepsilon(2\,000)$ — огромная вероятность. Итак, если Алиса не жульничает, Боб принимает ее доказательство с огромной вероятностью.

По закону больших чисел (раздел 3.5.3) чем больше окликов сгенерирует Боб, тем выше вероятность полноты протокола. Кстати, если Боб сгенерирует $\#J_N(1)$ окликов (что практически невозможно), вероятность полноты станет равной единице, т.е. он никогда не сделает ошибки (событие BadLuckAlice никогда не произойдет).

Непротиворечивость

Для того чтобы оценить вероятность ошибки с другой стороны, предположим, что Алиса нечестно сконструировала число $N \notin E_{2\_\mathrm{Prime}}$, имеющее больше двух простых множителей. Не исключена возможность, что Боб примет "доказательство" Алисы, поскольку он может просто сгенерировать больше $\frac{3}{8}$ случайных окликов, являющихся квадратичными вычетами (Бобу не повезло!)

Обозначим через BadLuckBob условное событие, состоящее в том, что $N \notin E_{2\_\mathrm{Prime}}$, и, тем не менее, Боб принимает "доказательство". Для случайно

выбранного множества Challenge из факта 5 следует, что теперь вероятность успеха в серии испытаний Бернулли не превышает $\delta = \frac{1}{4}$, а вероятность неудачи не меньше $1 - \delta = \frac{3}{4}$. Применяя формулу биномиального распределения и суммируя все варианты $k > \lfloor \frac{3}{8}m \rfloor$, в которых Боб принимает "доказательство", получаем следующую оценку площади фигуры, ограниченной "правым хвостом".

$$\delta(m) = \text{Prob}[\textsf{BadLuckBob}] = \sum_{k=\lfloor \frac{3}{8}m \rfloor}^{m} \binom{m}{k} \left(\frac{1}{4}\right)^k \left(\frac{3}{4}\right)^{m-k}.$$

Для $m = 2\,000$ получаем

$$\delta(2\,000) \approx 1{,}847 \cdot 10^{-35}.$$

Таким образом, со стороны Алисы было бы чрезвычайно наивным жульничать и надеяться, что ее не поймают за руку!

Итак, мы закончили исследование нулевого разглашения, полноты и непротиворечивости протокола 18.4.

18.5.1.2 Обоснование "критерия голосования"

Если Алиса не жульничает, а вероятность полноты в одном раунде равна $\varepsilon = \frac{1}{2}$, то ровно половина элементов множества $J_N(1)$ является квадратичными вычетами. Для повышения вероятности полноты протокола 18.4 невозможно применить "критерий голосования большинством", описанный в разделе 4.4.1.1. В качестве критического значения выбрано число $\frac{3}{8}$, поскольку оно лежит посередине между величиной $\varepsilon = \frac{1}{2}$ (Алиса не жульничает) и $\delta = \frac{1}{4}$ (Алиса жульничает). Этот выбор делает оба события, BadLuckAlice и BadLuckBob, приблизительно равновероятными (точнее, одинаково невероятными).

Этот критерий называется "голосование меньшинством". По закону больших чисел (раздел 3.5.3), поскольку $\delta < \varepsilon$, в качестве критического значения можно выбрать их среднее и многократно повторять раунды (m раз), уменьшая вероятность $\delta(m)$ и увеличивая вероятность $\varepsilon(m)$. Это позволит распознавать жульничество с высокой степенью точности.

Для того чтобы вероятности обеих неудач были пренебрежимо малы (например, не превышали 2^{-100}), необходимо выполнить 2 000 повторений. Если выбрать меньшее количество повторений, вероятности ошибок могут резко возрасти. Например, при $m = 100$ (которое обычно считается "достаточным" количеством повторений) вероятность полноты равна $\varepsilon(100) \approx 0{,}993$ (т.е. событие BadLuckAlice происходит с вероятностью $1 - \varepsilon(100) \approx 0{,}007$), а вероятность противоречивости равна $\delta(100) \approx 0{,}0052$ (т.е. событие BadLuckBob происходит с вероятностью 0,0052). Эти оценки далеки от желаемых, поскольку вероятности неудач слишком высоки (ошибки будут возникать слишком часто).

Как правило, если величины ε и δ близки, протоколы с двусторонними ошибками становятся неэффективными.

Несколько исследователей, например, Грааф (Graaf) и Перальта (Peralta) [291], Камениш (Camenisch) и Мичелс (Michels) [63], а также Женнаро (Gennaro), Миччианикио (Miccianicio) и Рабин (Rabin) [120] предложили более эффективные ZK-протоколы с односторонней ошибкой ($\varepsilon = 1$) для доказательства того, что число N имеет два простых множителя.

Протокол, описанный нами выше, основан на идеях, изложенных Бергером (Berger), Каннаном (Kannan) и Перальта [32]. Его выбор объясняется двумя причинами. Во-первых, он является наиболее простым. Во-вторых, он имеет двусторонние ошибки, редко встречающиеся у протоколов с нулевым разглашением. Это позволило нам использовать его для демонстрации основных идей.

18.6 Эффективность раунда

Рассмотрим второй вопрос, сформулированный в разделе 18.1: сколько раундов необходимо для того, чтобы доказывающая сторона убедила в своей правоте проверяющую сторону? Это вопрос о так называемой **эффективности раунда** (round efficiency). Раундом называется полный цикл действий, связанных с отправкой и получением сообщений. Поскольку многие ZK- (и IP-) протоколы состоят из вычислений величин Commit (первый шаг участника P), Challenge (шаг участника V), Response (второй шаг участника P), мы будем часто называть раундом именно эти три действия.

Для того чтобы снизить вероятность ошибки в протоколах с нулевым разглашением, обычно используется большое количество раундов. В выражении (18.2.2) величина ε представляет собой нижнюю оценку вероятности полноты, а величина $1 - \varepsilon$ — ее оценку сверху. Как и оценку непротиворечивости, оценку полноты сверху необходимо минимизировать. Для того чтобы объективно оценивать эффективность раундов в протоколе с нулевым разглашением, необходимо оценивать вероятности ошибок в каждом отдельном раунде. Чем ниже оценка такой вероятности, тем выше эффективность сеанса.

В зависимости от вероятности ошибки в ходе отдельного раунда протоколы разделяются на три категории.

Логарифмические протоколы (logarithmic-round protocol). Эффективность отдельного раунда во всех протоколах с нулевым разглашением, рассмотренных нами ранее, за исключением протокола 18.4, была постоянной величиной, например, $1/2$ или $\log_2 \log_2 n$ (если в качестве параметра безопасности используется $\log_2 n$, как в протоколе 18.1 или протоколе идентификации Шнорра, величина $\log \log n$ считается постоянной). Для того чтобы сделать вероятность ошибки пренебрежимо малой, т.е. ограничить ее величи-

ной $1/(\log n)^c$ для всех констант c, протокол с постоянной вероятностью ошибки необходимо выполнить $\log n$ раз. По этой причине такие протоколы называются **логарифмическими**.

Полиномиальные протоколы. Эффективность раунда в логарифмическом протоколе фактически представляет собой линейный полином, зависящий от параметра безопасности. В некоторых протоколах с нулевым разглашением эффективность раунда оценивается полиномами более высокой степени. Протокол с нулевым разглашением для любого NP-языка, использующий полиномиальную редукцию к NP-полной задаче (см. раздел 18.2.3), называется **полиномиальным**.

Например, протокол 18.4 является полиномиальным. Во-первых, в нем используется повышенное количество раундов, поскольку он имеет двусторонние ошибки, а во-вторых, в ходе каждого раунда протокол 18.4 вызывает другой логарифмический протокол (протокол 18.3).

Константные (одноразундовые) протоколы . Если протокол с нулевым разглашением может снизить вероятность ошибки до пренебрежимо малой величины за небольшое количество раундов (или даже за один раунд), отпадает необходимость повторять логарифмическое количество раундов. Благодаря этому свойству такие протоколы называются **константными (одноразундовыми)**.

Повышению эффективности раундов в протоколах с нулевым разглашением посвящено много работ. В этой области получено много результатов. Рассмотрим два результата, касающихся решения задачи о принадлежности элемента подгруппе и вычисления дискретного логарифма.

- В разделе 18.6.1 получена нижняя оценка эффективности раунда в протоколе аргументации с нулевым разглашением для подгрупп $\mathbb{Z}_N^*$, где N — нечетное составное целое число. Этот результат носит негативный характер. Из него следует, что для снижения вероятности ошибки необходимо выполнить логарифмическое количество операций, т.е. константного протокола для доказательства принадлежности элемента подгруппе не существует.
- В разделе 18.6.2 изучен константный ZK-протокол доказательства равенства дискретного логарифма для элементов из конечного поля $\mathbb{F}_p$. Этот результат является положительным. Он свидетельствует о значительной эффективности протокола идентификации Шнорра (протокол 18.2).

18.6.1 Нижняя оценка эффективности раунда при решении задачи о принадлежности элемента подгруппе

Рассмотрим снова задачу о доказательстве (аргументации) принадлежности элемента подгруппе, поставленную перед протоколом 18.1. Теперь будем считать, что функция $f(x)$ реализуется так, как показано в разделе 18.3.3.1, т.е.

$$f(x) = y \equiv g^x (\mathrm{mod}\, N),$$

где N — большое нечетное составное число, а число g принадлежит группе $\mathbb{Z}_N^*$, имеющей большой простой порядок. Из этого следует, что

$$\{g^x (\mathrm{mod}\, N) \mid x \in \phi(N)\} \subset \mathbb{Z}_N^*,$$

т.е. количество элементов в этом подмножестве меньше, чем $\phi(N)$.

Предположим, что доказывающая сторона, Алиса, знает разложение числа N на простые множители. (Напомним, что в разделе 18.3.3 мы предполагали, что Алиса не знает факторизации числа N, а значит, протокол с нулевым разглашением является вычислительным.) Знание этого разложения позволяет Алисе провести *идеальное* доказательство с нулевым разглашением для $y \in \langle g \rangle$.

Сформулируем вопрос следующим образом.

> Можно ли улучшить эффективность протокола 18.1, увеличив размер оклика, генерируемого Бобом, как это сделано в протоколе идентификации Шнорра, если $f(x) = g^x (\mathrm{mod}\, N)$ и Алисе известно разложение составного целого числа N на простые множители?

Напомним, что, например, в протоколе идентификации Шнорра (протокол 18.2) мы немного увеличили длину оклика: $\mathsf{Challenge} \in \{0,1\}^{\log_2 \log_2 p}$. В результате эффективность протокола повысилась: вместо m раундов, предусмотренных в протоколе 18.1, для доказательства оказалось достаточно выполнить $\frac{m}{\log_2 \log_2 p}$ раундов, причем вероятность противоречивости не изменилась.

К сожалению, если Алиса знает разложение числа N на простые множители, такой прием становится невозможным. Проблема заключается не в нулевом разглашении. Она связана с вероятностью противоречивости. Вероятность противоречивости данного протокола равна $\delta = 1/2$ независимо от длины оклика. Если вероятность противоречивости является постоянной и значительной, протокол должен быть логарифмическим. Гэлбрайт (Galbraith), Мао и Патерсон (Paterson) обнаружили этот факт в работе [117].

Для простоты изложения рассмотрим вероятность противоречивости одноразундового трехшагового протокола, использующего длинный оклик (т.е., как показано в разделе 18.3.2, верификатор придерживается честной стратегии). Как будет

показано ниже, этот полученный результат оказывается справедливым для любой длины оклика, превышающей один бит.

Итак, опишем протокол с нулевым разглашением и честной верификацией под названием "Непригодный протокол" (протокол 18.5), предназначенный для доказательства принадлежности элемента одной из подгрупп $\mathbb{Z}_N^*$. Необходимо предупредить читателей, что этот протокол непригоден для использования ни в одном приложении. Мы описываем его лишь для демонстрации идеи.

Протокол 18.5. "Непригодный протокол"

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

большое составное целое число;

g, y: два элемента группы $\mathbb{Z}_N^*$, где g имеет большой порядок по модулю N, а $y \equiv g^z \pmod N$.

ЗАКРЫТАЯ ИНФОРМАЦИЯ АЛИСЫ:

целое число $z < \phi(N)$.

ЗАКЛЮЧЕНИЕ БОБА:

$y \in \langle g \rangle$, т.е. $y \equiv g^z \pmod N$ при некотором z.

1. Алиса генерирует число $k \in_U \mathbb{Z}_{\phi(N)}$, вычисляет значение $\mathsf{Commit} \leftarrow g^k \pmod N$ и отсылает его Бобу.
2. Боб генерирует равномерно распределенную случайную величину $\mathsf{Challenge} < N$ и отсылает ее Алисе.
3. Алиса вычисляет значение $\mathsf{Response} \leftarrow k + z \cdot \mathsf{Challenge} \pmod{\phi(N)}$ и посылает его Бобу.
4. Если $g^{\mathsf{Response}} \equiv \mathsf{Commit} \cdot y^{\mathsf{Challenge}} \pmod N$, Боб принимает доказательство, в противном случае он его отвергает.

На первый взгляд, поскольку размер числа Challenge велик, Алисе нелегко его угадать, и она вынуждена точно следовать инструкциям протокола, что снижает вероятность непротиворечивости до величины $\delta \approx 1/\phi(N)$. Если бы это было правдой, то протокол состоял бы только из одного раунда. К сожалению, эта оценка вероятности противоречивости некорректна. Прием, позволяющий Алисе обмануть Боба, продемонстрирован в примере 18.4.

Пример 18.4. С этого момента будем обозначать доказывающую сторону как Алиса, поскольку она придерживается нечестной стратегии.

Зная факторизацию числа N, Алиса может легко вычислить нетривиальный квадратный корень единицы, т.е. элемент $\xi \in \mathbb{Z}_N^*$, удовлетворяющий условиям $\xi \neq \pm 1$ и $\xi^2 \equiv 1 \pmod N$. Вычисление квадратного корня выполняется с помощью алгоритма 6.5. Она может выбрать этот элемент так, что $\xi \notin \langle g \rangle$.

Затем $\widetilde{\text{Алиса}}$ вычисляет общие входные данные:

$$Y \leftarrow \xi g^z (\mathrm{mod}\, N).$$

Очевидно, что $Y \in \xi\langle g\rangle$, т.е. Y — это класс смежности подгруппы $\langle g\rangle$. Подчеркнем, что $Y \notin \langle g\rangle$, поскольку $\xi \notin \langle g\rangle$ (свойства класса смежности описываются теоремой 5.1. в разделе 5.2.1).

Вместо вычисления значения Commit по инструкциям протокола, $\widetilde{\text{Алиса}}$ подбрасывает монету $b \in_U \{0,1\}$, пытаясь угадать четность оклика Боба. Затем она вычисляет величину Commit по следующей формуле.

$$\mathsf{Commit} \leftarrow \begin{cases} g^k (\mathrm{mod}\, N), & \text{если } b = 0, \\ \xi g^k (\mathrm{mod}\, N), & \text{если } b = 1. \end{cases}$$

В оставшейся части протокола $\widetilde{\text{Алиса}}$ должна следовать инструкциям.

Очевидно, что шансы правильно угадать число Challenge равны $1/2$. Если Алиса правильно угадала четный оклик $\mathsf{Challenge} = 2u$, Боб выполняет следующую верификацию.

$$g^{\mathsf{Response}} \equiv g^k g^{z2u} \equiv \mathsf{Commit} \cdot (\xi g)^{z2u} \equiv \mathsf{Commit} \cdot Y^{\mathsf{Challenge}} (\mathrm{mod}\, N).$$

Следовательно, Боб принимает доказательство. Если Алиса правильно угадала нечетный оклик $\mathsf{Challenge} = 2u + 1$, Боб выполняет верификацию следующим образом.

$$g^{\mathsf{Response}} \equiv g^k g^{z(2u+1)} \equiv \xi g^k (\xi g)^{z(2u+1)} \equiv \mathsf{Commit} \cdot Y^{\mathsf{Challenge}} (\mathrm{mod}\, N).$$

Следовательно, Боб снова должен принять доказательство.

Таким образом, независимо от длины оклика, как и в протоколе 18.5, вероятность противоречивости равна всего лишь $\delta = 1/2$. По этой причине мы назвали этот протокол “непригодным”. □

Поскольку Бобу неизвестна факторизация числа N, он не может самостоятельно решить задачу о принадлежности элемента подгруппе (см. замечание 18.1). Итак, в примере 18.4 Боб может предотвратить жульничество $\widetilde{\text{Алисы}}$ только с вероятностью $1/2$. Увеличение длины оклика ничего не дает!

Задача, описанная в примере 18.4, не возникала в вычислительном ZK-протоколе, приведенном в разделе 18.3.3.2, в котором функция $f(x)$ реализовывалась аналогично: $f(x) = a^x (\mathrm{mod}\, N)$, где N — нечетное составное целое число. Напомним, что в этом протоколе используются битовые оклики, а значит, вероятность его противоречивости равна $\delta = 1/2$. Протокол идентификации Шнорра также не страдает этим недостатком, поскольку группа $\langle g\rangle$ в этом протоколе имеет простой

порядок q и не содержит ни одного элемента, порядок которого был бы меньше q, за исключением единицы.

Используя нетривиальный квадратный корень единицы по модулю N, Алиса достигает максимальной вероятности угадывания, равной $\delta = 1/2$. В тривиальном случае $\xi = -1$ (другой тривиальный случай, $\xi = 1$, в атаке не используется) Боб может достичь большей убедительности: либо Y, либо $-Y$ принадлежит подгруппе $\langle g \rangle$. Однако, поскольку Алиса знает факторизацию числа N, а Боб — нет, используя китайскую теорему об остатках, она также может скрыть число g^k, используя другой множитель малого порядка, например, третьего. (С помощью теоремы 6.7 из раздела 6.2.3 Алиса может вычислять элементы любого порядка $d \mid \phi(N)$.) Таким образом, вероятность противоречивости не может быть пренебрежимо малой величиной. Протокол 18.1 остается всего лишь демонстрационной версией решения задачи о принадлежности элемента подгруппе при обычном выборе параметров безопасности, в который входят подгруппы $\mathbb{Z}_N^*$.

Итак, доказательство принадлежности подгруппе с нулевым разглашением состоит из логарифмического количества раундов.

В приложениях ZK-протоколов, описанных в следующей главе, мы рассмотрим доказательство принадлежности элемента группе $\mathbb{Z}_N^*$. Однако сократить затраты, связанные с выполнением логарифмических протоколов, нам не удастся, поскольку для этого необходимо особым образом выбирать число N.

18.6.2 Доказательство знания дискретного логарифма с постоянным количеством раундов

Протокол идентификации Шнорра (протокол 18.2) позволяет аргументировать с нулевым разглашением знание дискретного логарифма элемента из конечного поля $\mathbb{F}_p$. Выше было продемонстрировано, что этот протокол является логарифмическим.

Теперь покажем, что с помощью протокола идентификации Шнорра можно провести аналогичное доказательство с постоянным количеством раундов. Эта модификация называется протоколом Чаума (Chaum) [72]. Ее полное название — протокол ZK Dis-Log-EQ доказательства Чаума (Chaum's ZK Dis-Log-EQ Proof Protocol). Этот протокол позволяет провести доказательство с нулевым разглашением того, что два элемента имеют одно и то же значение дискретного логарифма.

Рассмотрим протокол ZK Dis-Log-EQ доказательства Чаума с параметрами безопасности, выбранными в протоколе идентификации Шнорра. Иначе говоря, пусть элемент $g \in F_p$, где p — нечетное простое число, и $\mathrm{ord}_p(g) = q$, где q — также нечетное простое число (следовательно, $q \mid p - 1$). Введем обозначение $G = \langle g \rangle$.

В протоколе ZK Dis-Log-EQ доказательства Чаума используется дополнительный элемент $h \in \langle g \rangle$, где $h \neq g$ и $h \neq 1$.

Протокол 18.6. Протокол ZK Dis-Log-EQ доказательства Чаума

ОБЩИЕ ВХОДНЫЕ ДАННЫЕ:

p, q: два простых числа, удовлетворяющих условию $q \mid p-1$.

(* Как правило, $|p| = 1024$, $|q| = 160$. *)

$g, h : \operatorname{ord}_p(g) = \operatorname{ord}_p(h) = q, g \neq h$.

(* Боб проверяет условия: $g \neq 1, h \neq 1, g \neq h, g^q \equiv h^q \equiv 1 (\operatorname{mod} p)$. *)

$X, Y : X = g^z (\operatorname{mod} p), X = h^z (\operatorname{mod} p)$.

ЗАКРЫТАЯ ИНФОРМАЦИЯ АЛИСЫ:

$z \in \mathbb{Z}_q$.

ЗАКЛЮЧЕНИЕ БОБА:

Алиса знает некий элемент $z \in \mathbb{Z}_q$, такой что $X \equiv g^z (\operatorname{mod} p)$ и $Y \equiv h^z (\operatorname{mod} p)$, т.е. $\log_g X \equiv \log_h X (\operatorname{mod} q)$.

1. Боб генерирует число $a, b \in_U \mathbb{Z}_q$, вычисляет значение $\mathsf{Commit}_B \leftarrow g^a h^b (\operatorname{mod} N)$ и отсылает его Бобу.

   (* Число Commit_B является окликом Боба. *)

2. Алиса генерирует число $c \in_U \mathbb{Z}_q$, вычисляет значения

$$\mathsf{Commit}_A^{(1)} \leftarrow \mathsf{Commit}_B g^c (\operatorname{mod} p),$$
$$\mathsf{Commit}_A^{(2)} \leftarrow \left(\mathsf{Commit}_A^{(1)}\right)^z (\operatorname{mod} p),$$

   и посылает их Бобу.

3. Боб сообщает Алисе числа a и b.

   (* Боб раскрывает содержание передачи, чтобы продемонстрировать правильность своего оклика. *)

4. Алиса проверяет условие $\mathsf{Commit}_B \equiv g^a h^b (\operatorname{mod} p)$.

   (* Если условие выполняется, она раскрывает Бобу число c, если нет — прекращает выполнение протокола. Если оклик Боба сконструирован правильно, значит, он заранее знал число $X^a Y^b (\operatorname{mod} p)$, предназначенное для Алисы. *)

5. Боб проверяет условия

$$\mathsf{Commit}_A^{(1)} \equiv \mathsf{Commit}_B g^c (\operatorname{mod} p),$$
$$\mathsf{Commit}_A^{(2)} \equiv X^c X^a Y^b (\operatorname{mod} p).$$

   Если оба равенства выполняются, доказательство принимается, в противном случае оно отклоняется.

Из этой спецификации следует, что протокол содержит четыре обмена сообщениями и выполняется только один раз. Ниже будет показано, что вероятность его непротиворечивости равна $\delta = 1/q$. Таким образом, протокол ZK Dis-Log-EQ доказательства Чаума весьма эффективен.

18.6.2.1 Стойкость протокола ZK Dis-Log-EQ доказательства Чаума

Полнота

Из описания протокола непосредственно следует, что вероятность его полноты равна $\varepsilon = 1$. Иначе говоря, если Алиса обладает элементом z и следует инструкциям протокола, Боб всегда принимает ее доказательство.

Непротиворечивость

Покажем, что протокол ZK Dis-Log-EQ действительно является протоколом доказательства, т.е. Алиса может обладать неограниченной вычислительной мощью. Для этого мы не будем налагать на объем ее вычислительных ресурсов никаких ограничений.

Предположим, что Алиса жульничает. Тогда ее общие входные данные (p, q, g, h, X, Y) удовлетворяют следующим условиям.

$$X \equiv g^z (\mathrm{mod}\, p) \text{ и } Y \equiv h^{z'} (\mathrm{mod}\, p) \text{ при некотором } z \neq z' (\mathrm{mod}\, q). \qquad (18.6.1)$$

Для того чтобы Боб принял доказательство Алисы, выполняя верификацию на шаге 5, Алиса должна послать ему на шаге 2 число $\mathsf{Commit}_A^{(2)}$, удовлетворяющее условию

$$\mathsf{Commit}_A^{(2)} \equiv X^c X^a Y^b (\mathrm{mod}\, p). \qquad (18.6.2)$$

Другими словами, получив от Боба числа a и b, Алиса должна раскрыть число $c \in \mathbb{Z}_q$, удовлетворяющее условию (18.6.2). Поскольку значения a и b зафиксированы Бобом на этапе 1, а числа $\mathsf{Commit}_A^{(1)}$ и $\mathsf{Commit}_A^{(2)}$ зафиксированы на этапе 2, формула (18.6.2) означает, что число $c \in \mathbb{Z}_q$ также фиксируется на этапе 2. Иначе говоря, Алиса не может изменить значение c после того, как она послала на этапе 2 свои передачи.

При фиксированном числе $c \in \mathbb{Z}_q$ получаем:

$$c \equiv \log_g \frac{\mathsf{Commit}_A^{(1)}}{g^a h^b} (\mathrm{mod}\, q). \qquad (18.6.3)$$

С учетом формулы (18.6.2) отсюда следует, что

$$c \log_g X \equiv \log_g \frac{\mathsf{Commit}_A^{(2)}}{X^a Y^b} (\mathrm{mod}\, q). \qquad (18.6.4)$$

Поскольку $h \in \langle g \rangle$ (равенство $\mathrm{ord}_p(h) = q$ позволяет Бобу подтвердить это, проверив условия $h \neq 1$ и $h^q \equiv 1 (\mathrm{mod}\, p)$), $h \equiv g^d (\mathrm{mod}\, p)$ при некотором $d \in \mathbb{Z}_q$, $d \neq 0 (\mathrm{mod}\, q)$. Следовательно, формулу (18.6.3) можно переписать так.

$$c - \log_g \mathsf{Commit}_A^{(1)} \equiv -a - bd (\mathrm{mod}\, q). \tag{18.6.5}$$

Аналогично, используя формулу (18.6.1), можно переписать формулу (18.6.4) в следующем виде.

$$c \log_g X - \log_g \mathsf{Commit}_A^{(2)} \equiv -az - bdz' (\mathrm{mod}\, q). \tag{18.6.6}$$

Для $z \neq z' (\mathrm{mod}\, q)$ формулы (18.6.5) и (18.6.6) образуют следующую систему линейных уравнений.

$$\begin{pmatrix} c - \log_g \mathsf{Commit}_A^{(1)} \\ c \log_g X - \log_h \mathsf{Commit}_A^{(2)} \end{pmatrix} = \begin{pmatrix} -1 & -d \\ -z & -dz' \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} (\mathrm{mod}\, q).$$

Матрица этой системы имеет полный ранг ($rank = 2$). Отсюда следует, что существует единственная пара чисел $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$. Эта пара удовлетворяет условиям, при которых вычисляется значение Commit_B на этапе 1, и выполняется верификация на этапе 5.

Однако на этапе 2, фиксируя число $c \in \mathbb{Z}_q$, Алиса получает только одно из уравнений (18.6.5), которое позволяет ей вычислить ровно q разных пар (a, b). Все они удовлетворяют системе (18.6.5), но только одна из них удовлетворяет условию (18.6.6), проверяемому на этапе 5. Таким образом, обладая неограниченными вычислительными ресурсами, Алиса может угадать правильную пару (a, b) с вероятностью $1/q$.

Итак, мы не только доказали, что вероятность противоречивости однораундового протокола Чаума равна $1/q$, но и показали, что он является протоколом доказательства (а не аргументации) равенства дискретного логарифма.

Идеальное нулевое разглашение

Протокол Чаума является идеальным протоколом доказательства с нулевым разглашением. Построим алгоритм уравнивания ("eguator") $\mathcal{EQ}$, позволяющий создать стенограмму, распределение которой идентично распределению подлинной стенограммы. Для общего входа (p, q, g, h, X, Y) алгоритм $\mathcal{EQ}$ выполняет следующие шаги.

1. Алгоритм $\mathcal{EQ}$ генерирует числа $a, b \in_U \mathbb{Z}_q$ и вычисляет значение $\mathsf{Commit}_B \leftarrow g^a h^b (\mathrm{mod}\, p)$.
2. Алгоритм $\mathcal{EQ}$ генерирует число $c \in_U \mathbb{Z}_q$ и вычисляет значения
   $\mathsf{Commit}_A^{(1)} \equiv \mathsf{Commit}_B g^c (\mathrm{mod}\, p)$,
   $\mathsf{Commit}_A^{(2)} \equiv X^c X^a Y^b (\mathrm{mod}\, p)$.

3. Алгоритм $\mathcal{EQ}$ выдает результат $\mathsf{Transcript} = \mathsf{Commit}_B, \mathsf{Commit}_A^{(1)}, \mathsf{Commit}_A^{(2)}, a, b, c$.

Очевидно, что полученная строка Transcript имеет распределение, идентичное распределению подлинной стенограммы.

Существует и другой, более убедительный способ доказательства идеального нулевого разглашения в протоколе Чаума. Во-первых, если Боб придерживается нечестной стратегии, создавая оклик, т.е. число Commit_B сконструировано неверно, он не получит никакого ответа. Во-вторых, если Боб правильно создает оклик, используя пару чисел $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$, значит, ему уже с первого этапа известно число $X^a Y^b (\mathrm{mod}\, p)$, "раскрываемое" Алисой. В обеих ситуациях Боб не получает никакой новой информации о закрытом числе Алисы!

18.6.2.2 Выводы

- Протокол ZK Dis-Log EQ можно использовать для идентификации. В этом случае пара (g, X) может считаться компонентом открытого ключа пользователя, сертифицированного соответствующим органом (см. раздел 13.2).
- Для вычисления значений $g^a h^b (\mathrm{mod}\, p)$ и $X^c X^a Y^b (\mathrm{mod}\, p)$ можно применять алгоритм 15.2, эффективность которого сравнима с однократным возведением в степень по модулю. Таким образом, Алиса и Боб должны будут выполнить примерно по три возведения в степень. В этом случае, если Алиса жульничает, вероятность принятия доказательства пренебрежимо мала. Для сравнения, чтобы достичь столь же малой вероятности ошибки в протоколе идентификации Шнорра, Алиса и Боб должны выполнить $\log_2 p \approx 10$ (если $p \approx 2^{1024}$) операций возведения в степень.
- Неограниченные вычислительные ресурсы доказывающей стороны повышают ценность протокола для приложений, в которых доказывающей стороной является мощный пользователь, например, правительственное агентство.
- Несмотря на непротиворечивость протокола, Алиса не обязана знать значение дискретного логарифма. Доказательство просто демонстрирует, что при возведении в степень она получает правильный ответ. Вполне возможно, что для этого она прибегает к услугам оракула. В протоколе идентификации Шнорра, даже если доказывающая сторона использует оракула, два правильных ответа позволяют извлечь значение дискретного логарифма и применить лемму о разветвлении для доказательства неуязвимости тройной схемы цифровой подписи Эль-Гамаля (см. раздел 16.3.2). В протоколе Чаума два правильных ответа не позволяют определить значение дискретного логарифма.
- На основе своего протокола Чаум предложил **неоспоримую схему цифровой подписи** (undeniable signature scheme) [72] (см. также работу Чаума и Антверпена (Antwerpen) [74]). "Неоспоримая схема цифровой подписи"

позволяет доказать авторство документа с помощью интерактивного протокола, не прибегая к процедуре верификации, предусмотренной в обычных схемах. Это дает возможность подписывающей стороне выбрать верификатор подписи и, таким образом, защитить свои права на тайну подписи. Это может оказаться полезным в некоторых приложениях, где публичная верификация подписи является нежелательной. Например, поставщик программного обеспечения ставит на своих продуктах цифровую подпись, чтобы подтвердить их аутентичность и безопасность, однако правом верификации этой подписи должны обладать только добросовестные покупатели. Используя неоспоримую подпись, поставщик может предотвратить пиратскую подделку.

18.7 Неинтерактивное нулевое разглашение

Интерактивные протоколы с нулевым разглашением предусматривают взаимодействие между пользователями. Несмотря на то что в одноразундовых и константных протоколах (например, в протоколе Чаума) количество взаимодействия мало, оба пользователя должны одновременно находиться на связи. Если бы протокол с нулевым разглашением был неинтерактивным, можно было бы использовать одностороннюю линию связи. Такой вид связи обладает несколькими преимуществами.

Рассмотрим умозрительную ситуацию, в которой пользователи P и V являются математиками (сценарий, описанный в работе [44]). Пользователь P хочет путешествовать по миру, доказывая математику V новые теоремы, не разглашая сути доказательства. В этом сценарии необходимо использовать неинтерактивное доказательство, поскольку пользователь P может не иметь фиксированного адреса и переезжать до получения очередного ответа. Эти два виртуальных пользователя оценят преимущества неинтерактивного доказательства с нулевым разглашением.

В начале главы 15 рассматривалось более реалистичное применение неинтерактивного доказательства с нулевым разглашением: создание доказуемо стойкой схемы шифрования с открытым ключом, неуязвимой для атаки CCA2 (хотя на самом деле мы стремились предостеречь от подобного подхода). Во всяком случае, возможность проведения неинтерактивного доказательства (или аргументации) с нулевым разглашением всегда полезна.

Блюм, Фельдман и Микали предложили **метод неинтерактивного доказательства с нулевым разглашением** (non-interactive zero-knowledge — NIZK), в котором пользователи P и V обладают общими битами оклика [44]. Эти биты могут генерироваться посредником, пользующимся доверием у пользователей P и V (такой доверенный источник случайных битов был назван Рабином **случайным маяком** (random beacon) [239]). Вполне возможна также ситуация, когда

оба пользователя сами заранее генерируют биты при личной встрече (например, в описанном выше сценарии математики могут согласовать биты до отъезда).

В разделе 18.3.2.2 описан эвристический метод Фиата–Шамира, позволяющий создавать неинтерактивное "доказательство знания"[2]. Однако неинтерактивность, обеспечиваемая методом Фиата–Шамира, достигается за счет потери нулевого разглашения: "закрытое доказательство" становится "открытым", т.е. доступным для открытой проверки.

Якобсон (Jakobsson), Сако (Sako) и Импальяццо (Impagliazzo) разработали новый способ, основанный на эвристическом методе Фиата–Шамира и позволяющий сохранить доказательство "закрытым" [153]. Они назвали этот метод **доказательством предназначенному верификатору** (designated verifier proof): если Алиса доказывает Бобу знание некоего факта, только Боб может проверить его правильность.

18.7.1 Неинтерактивное доказательство с нулевым разглашением и предназначенным верификатором

Метод неинтерактивного доказательства с нулевым разглашением и предназначенным верификатором основан на применении эвристического метода Фиата–Шамира к следующему логическому выражению.

"Утверждение Алисы истинно" $\vee$ "Боб сымитировал доказательство Алисы".

Алиса может создать "доказательство" этого логического выражения с помощью конструкции под названием **"передача с секретом"** (trapdoor commitment). Брассар (Brassard), Чаум и Крепо (Crepeau) [59] назвали ее **имитируемой передачей** (simulatable commitment).)

Передача с секретом — это специальное сообщение, которое Алиса создает с помощью открытого ключа Боба, т.е. предназначенного верификатора. Обозначим передачу с секретом, созданную с помощью открытого ключа Боба y_B, как

$$\mathsf{TC}(w, r, y_B).$$

Здесь w — передаваемое значение, а r — случайное число. Укажем два важных свойства передачи с секретом.

---

[2]Фраза "доказательство знания" всегда заключается в кавычки, поскольку, строго говоря, эвристический метод Фиата–Шамира позволяет проводить лишь аргументацию (см. раздел 18.4.1).

Свойство 18.1 (Свойства передачи с секретом).

1. Без секретного компонента ключа y_B передача является связанной, т.е. не существует эффективного алгоритма вычисления коллизии $w_1 \neq w_2$, удовлетворяющей условию $\mathsf{TC}(w_1, r, y_B) = \mathsf{TC}(w_1, r', y_B)$.
2. Используя закрытый компонент ключа y_B, легко вычислить любое количество коллизий.

Пример 18.5 (Схема на основе передачи с секретом.). Пусть (p, q, g) — числа из общих входных данных протокола идентификации Шнорра, а $y_B = g^{x_B} (\mathrm{mod}\, p)$ — открытый ключ Боба, где $x_b \in \mathbb{Z}_q$.

Если Алиса желает скрыть число $w \in \mathbb{Z}_q$, она генерирует число $r \in_U \mathbb{Z}_q$ и вычисляет передачу с секретом $\mathsf{TC}(w, r, y_B) \leftarrow g^w y_B^r (\mathrm{mod}\, p)$. Она может открыть значение $\mathsf{TC}(w, r, y_B)$, раскрыв пару (w, r). Проверим свойства передачи $\mathsf{TC}(w, r, y_B)$.

Во-первых, не зная закрытый ключ Боба x_B, Алиса может раскрыть передачу только с помощью пары (w, r). Предположим противное, т.е. что она знает другую пару (w', r'), позволяющую раскрыть передачу, где $w' \neq w (\mathrm{mod}\, q)$ (поскольку $r' \neq r (\mathrm{mod}\, q)$). Тогда, поскольку

$$1 = g^{w-w''} y_B^{r-r'} (\mathrm{mod}\, p),$$

получаем

$$y_B \equiv g^{x_B} \equiv g^{\frac{w'-w}{r-r'}} (\mathrm{mod}\, p),$$

т.е. Алиса знает число $x_B \equiv \frac{w'-w}{r'-r} (\mathrm{mod}\, q)$. Это противоречит предположению о том, что Алиса не знает числа x_B.

Во-вторых, используя ключ x_B, Боб может сгенерировать числа w_1, w_2, $r_1 \in_U \mathbb{Z}_q$, где $w' \neq w (\mathrm{mod}\, q)$. Затем он вычисляет параметр

$$r_2 \leftarrow \frac{w_1 - w_2 + r_1 x_B}{x_B}.$$

Легко убедиться, что $\mathsf{TC}(w_1, r_1, y_B) = \mathsf{TC}(w_2, r_2, y_B)$. □

В разделе 18.3.2.2 показано, что "доказательство знания", проведенное с помощью эвристического метода Фиата–Шамира, представляет собой тройку (Commit, Challenge, Response), построенную доказывающей стороной, т.е. Алисой. В этой тройке компонент Commit содержит число k, которое, зафиксировав, Алиса больше не может изменить.

В схеме NIZK, предложенной Якобсоном и его соавторами, доказательство представляет собой кортеж

$$(w, r, \mathsf{Commit}, \mathsf{Challenge}, \mathsf{Response}). \qquad (18.7.1)$$

Здесь пара (w, r) представляет собой открытую информацию, предназначенную для создания сообщения $\mathsf{TC}(w, r, y_B)$. Эта пара добавляется для того, чтобы предназначенный верификатор Боб мог использовать секретную информацию для поиска коллизий и моделирования доказательства Алисы.

Создание доказательства

Алиса создает доказательство (18.7.1) следующим образом.

P.1. Алиса генерирует числа $w, r \in_U \mathbb{Z}_q$ и вычисляет передачу $\mathsf{TC}(w, r, y_B) \leftarrow g^w y_B^r (\mathrm{mod}\, p)$.

P.2. Число Commit вычисляется так же, как и в эвристическом методе Фиата–Шамира: генерируя число $k \in_U \mathbb{Z}_q$ и вычисляя значение $\mathsf{Commit} \leftarrow g^k (\mathrm{mod}\, p)$.

P.3. Число Challenge генерируется, как обычно: с помощью функции хэширования (применяемой к необязательному сообщению M).

$$\mathsf{Challenge} \leftarrow h(\mathsf{TC}(w, r, y_B) \parallel \mathsf{Commit} \parallel [M]).$$

P.4. Для вычисления значения Response также используется число w.

$$\mathsf{Response} \leftarrow k + x_A(\mathsf{Challenge} + w) (\mathrm{mod}\, q).$$

Верификация доказательства

Получив кортеж доказательства (18.7.1) (и необязательное сообщение M), Боб выполняет его проверку, используя следующую процедуру.

V.1. Вычислить $\mathsf{Challenge} \leftarrow h(\mathsf{TC}(w, r, y_B) \parallel \mathsf{Commit} \parallel [M])$.

V.2. Проверить условие $g^{\mathsf{Response}} \stackrel{?}{\equiv} \mathsf{Commit}\, y_A^{\mathsf{Challenge}} y_A^w (\mathrm{mod}\, p)$. Если условие выполняется, доказательство принимается, в противном случае оно отклоняется.

Докажем стойкость этой схемы.

18.7.1.1 Стойкость

Полнота

Доказательство Алисы (18.7.1) очень похоже на кортеж (Commit, Challenge, Response), создаваемый в эвристическом методе Фиата–Шамира. Единственный элемент, который отличает это доказательство, — дополнительное число $y_A^w (\mathrm{mod}\, p)$. Это число умножается на правую часть выражения, проверяемого на этапе верификации (шаг V.2). Таким образом, эта схема является полной.

Непротиворечивость

Число $y_A^w (\mathrm{mod}\, p)$, известное предназначенному верификатору, является фиксированным, поскольку значение w в сообщении $\mathsf{TC}(w, r, y_B)$ фиксировано. Следовательно, учитывая первое свойство передачи с секретом, Алиса не может изменить число $y_A^w (\mathrm{mod}\, p)$, не зная ключа Боба x_B. Таким образом, если Боб уверен,

что его закрытый ключ Алисе не известен, то множитель $y_A^w (\mathrm{mod}\, p)$ является константой, а значит, тройка (Commit, Challenge, Response) — это подлинный кортеж, созданный Алисой с помощью эвристического метода Фиата–Шамира. Итак, непротиворечивость этой схемы зависит от непротиворечивости доказательства по методу Фиата–Шамира. Следует заметить, что, поскольку вычислительные ресурсы Алисы полиномиально ограничены (чтобы предотвратить инвертирование функции хэширования или открытого ключа Боба), эту схему следует считать аргументацией, а не доказательством.

Идеальное нулевое разглашение

Поскольку Бобу известна секретная информация x_B, с точки зрения любого постороннего наблюдателя, множитель $y_A^w (\mathrm{mod}\, p)$ больше не является константой. Наоборот, это число является объектом для манипуляций Боба. Действительно, поскольку Боб может свободно имитировать сообщение $\mathsf{TC}(w, r, y_B)$, кортеж (18.7.1) можно точно сымитировать.

Процедура имитации

Боб генерирует числа Response, $\alpha, \beta \in_U \mathbb{Z}_q$ и выполняет следующие операции.

S.1. $\mathsf{TC}(w, r, y_B) \leftarrow g^{\alpha} (\mathrm{mod}\, p)$.

S.2. $\mathsf{Commit} \leftarrow g^{\mathsf{Response}} y_A^{-\beta} (\mathrm{mod}\, p)$.

S.3. $\mathsf{Challenge} \leftarrow h(\mathsf{TC}(w, r, y_B) \parallel \mathsf{Commit} \parallel [M])$.

S.4. $w \leftarrow \beta - \mathsf{Challenge} (\mathrm{mod}\, q)$.

S.5. $r \leftarrow (\alpha - w)/x_B (\mathrm{mod}\, q)$.

S.6. Вывод кортежа $(w, r, \mathsf{Commit}, \mathsf{Challenge}, \mathsf{Response})$.

Покажем, что эта имитация доказательства является идеальной.

Во-первых, из выражения на шаге S.2, получаем, что

$$g^{\mathsf{Response}} \equiv \mathsf{Commit}\; y_A^{\beta} (\mathrm{mod}\, p).$$

Из шага V.3 следует, что

$$\mathsf{Commit}\; y_A^{\beta} \equiv \mathsf{Commit} y_A^{w+\mathsf{Challenge}} \equiv \mathsf{Commit}\; y_A^{\mathsf{Challenge}} y_A^w (\mathrm{mod}\, p).$$

Итак,

$$g^{\mathsf{Response}} \equiv \mathsf{Commit}\; y_A^{\mathsf{Challenge}} y_A^w (\mathrm{mod}\, p),$$

что полностью соответствует этапу верификации V.2.

Во-вторых, из шага S.5, следует, что

$$g^w y_B^r \equiv g^{w+rx_B} \equiv g^{\alpha} (\mathrm{mod}\, p).$$

Проверяя конструкцию $\mathsf{TC}(w, r, y_B)$, созданную на шаге S.1, легко убедиться, что секретная информация действительно имеет правильную структуру.

В заключение, нетрудно проверить, что, кроме правильной структуры, эти значения имеют требуемое распределение, генерируемое Алисой. Следовательно, алгоритм имитации, выполняемый Бобом, является алгоритмом уравнивания. Итак, мы доказали, что схема обладает идеальным нулевым разглашением.

18.7.1.2 Приложения

Якобсон и его соавторы предсказали интересные приложения своей схемы доказательства с предназначенным верификатором. Например, эту схему можно использовать в качестве альтернативы "неоспоримым подписям" (см. раздел 18.6.2.2): необязательное сообщение M можно рассматривать в качестве подписи Алисы, которую может верифицировать только предназначенный пользователь Боб. Рассмотрим ситуацию, в которой производитель программного обеспечения проверяет происхождение своей продукции. Если производитель Алиса использует "доказательство с предназначенным верификатором", то покупатель Боб не может убедить третью сторону в законном происхождении продукции, поскольку он может имитировать это доказательство.

Другим приложением является электронное голосование. Получив голос Кэрол, центр голосования должен послать ей подтверждение, сообщающее, что ее голос учтен. Очень важно, чтобы центр мог убедить Кэрол в правильности своего доказательства, предотвратив вмешательство Злоумышленника, стремящегося исказить результаты голосования. Если подтверждение создано с помощью "доказательства с предназначенным верификатором", то Злоумышленник не может проверить его корректность. Очевидно, что Кэрол может точно имитировать подтверждение своего голосования по отношению к любому кандидату по выбору Злоумышленника. Эта схема называется **электронным голосованием со свободным подтверждением** (receipt free electronic voting). Она изучена в работе Бенало (Benaloh) и Туинстра (Tuinstra) [30].

18.8 Резюме

В главе изучены протоколы с нулевым разглашением.

Сначала описаны интерактивные системы доказательства. Показано, что IP-протоколы тесно связаны с классом сложности $\mathcal{NP}$, изученным в главе 4. Это позволило лучше понять задачи из класса $\mathcal{NP}$. Как известно, для языка $L \in \mathcal{NP}$ задача $I \overset{?}{\in} L$ является легкоразрешимой (соответственно трудноразрешимой), если существует (соответственно не существует) дополнительная информация, облегчающая работу алгоритма. Как показывает изучение главы, интуитивно ясно, что задача является легкоразрешимой (трудноразрешимой), если верификатор может (не может) работать совместно с доказывающей стороной.

Сформулировано несколько понятий нулевого разглашения: идеальное, с честным верификатором, вычислительное и статистическое. Продемонстрированы различия между доказательством и аргументацией, рассмотрены протоколы с двусторонними ошибками, исследована задача об эффективности раундов и в заключение изучены неинтерактивные протоколы с нулевым разглашением. Каждое из понятий продемонстрировано на примере конкретного протокола. Надеемся, что такое изложение было доступным для читателей и дало им достаточно полное представление о протоколах с нулевым разглашением, которые предоставляют своим пользователям скорее умозрительные, чем практические преимущества.

Протоколы с нулевым разглашением представляют собой активно разрабатываемую область криптографии, связанную с компьютерными науками. Глава представляет собой всего лишь элементарное введение в этот предмет, поэтому читатели, заинтересовавшиеся данной темой, должны обратиться к специальной научной литературе.

Упражнения

18.1. Объясните следующие понятия, связанные с ZK-протоколами.

а) Общие входные данные.

б) Закрытые входные данные.

в) Случайные входные данные.

г) Полнота.

д) Непротиворечивость.

е) Стенограмма доказательства.

ж) Нечестная доказывающая сторона.

з) Нечестный верификатор.

и) Равенство.

к) Возможность имитации.

18.2. Объясните различия между следующими понятиями.

а) Идеальное нулевое разглашение.

б) Нечестный верификатор.

в) Вычислительный протокол с нулевым разглашением.

г) Статистический протокол с нулевым разглашением.

д) Доказательство с нулевым разглашением.

е) Аргументация с нулевым разглашением.

ж) Доказательство знания.

18.3. Невозможность отрицания цифровой подписи означает доказательство того, что автор подписи владеет эксклюзивным закрытым ключом (знанием), позволяющим ему создавать подпись. В чем заключается разница между этим доказательством знания и доказательством, обеспечиваемым протоколами с нулевым разглашением?

18.4. Может ли идеальный протокол доказательства с нулевым разглашением быть идеальным протоколом аргументации с нулевым разглашением?

18.5. Должна ли доказывающая сторона в ZK-протоколе быть полиномиально ограниченной? А верификатор?

18.6. Почему протокол идентификации Шнорра не может быть однораундовым?

18.7. Докажите полноту протокола идентификации Шнорра (протокол 18.2).

18.8. При описании вычислительного протокола с нулевым разглашением, приведенного в разделе 18.3.3.2, мы заметили, что Алиса может выбирать передаваемое значение из множества $Z_{N1+\alpha}$, где $\alpha > 0$ — малая и фиксированная величина. Почему?

18.9. Докажите факт 3 из раздела 18.4.2.1.

18.10. В некоторых протоколах с нулевым разглашением для понижения вероятности противоречивости применяется несколько раундов. Как правило, верификатор принимает доказательство, только если все раунды были безошибочными. Можно ли применить этот "критерий голосования" в протоколах с двусторонней ошибкой?

18.11. Почему в протоколе 18.4 используется "критерий голосования большинством"?

18.12. Почему протокол с двусторонней ошибкой не эффективен, в частности, когда поведение как честной, так и нечестной доказывающих сторон в отдельном раунде невозможно отличить?

18.13. Какой протокол называется константным (логарифмическим, полиномиальным)?

18.14. Можно ли упростить протокол 18.6, предусмотрев честное поведение доказывающей стороны и только три шага?

Подсказка: если Алиса непосредственно возводит оклик Боба в степень по модулю, то шаги 2, 3 и 4 можно сжать в отдельную передачу сообщения.

18.15. В чем заключается уязвимость описанной выше версии протокола 18.6 с честной доказывающей стороной?

Подсказка: вспомните четвертый пункт в разделе 18.6.2.2.

18.16. Что такое передача с секретом?

18.17. Где применяются неинтерактивные протоколы с нулевым разглашением?

Глава 19

Еще раз о “подбрасывании монеты по телефону”

В первом криптографическом протоколе, “Подбрасывание монеты по телефону” (протокол 1.1), использовалась “волшебная” функция f. Напомним ее свойства.

I. Для каждого целого числа x мы можем легко вычислить значение $f(x)$, однако по заданному значению $f(x)$ невозможно восстановить его прообраз x, например, определить, каким числом является x — четным или нечетным.

II. Невозможно найти пару чисел (x, y), таких что $x \neq y$ и $f(x) = f(y)$.

До сих пор эта функция остается для нас загадкой. Рассмотрим ее конкретную реализацию, не объясняя, почему при описании ее свойств дважды употребляется слово “*невозможно*”.

Практический способ реализации протокола 1.1 уже был предложен в разделе 1.2.1, в котором функция f представляла собой функцию хэширования, например, SHA-1. В этом случае для любого целого числа x значение $f(x)$ можно закодировать с помощью 40 шестнадцатеричных цифр и продиктовать Бобу по телефону.

Возможно, для детской игры этого и достаточно, но существует множество криптографических приложений, в которых участвуют партнеры, не доверяющие друг другу и стремящиеся согласовать между собой случайные числа. Нарушение случайности в этих приложениях может привести к тяжелым последствиям. Например, стандартные методы атаки, описанные в книге, как правило, сводятся к обману наивного пользователя, заставляя его выполнять криптографические операции над невинными, на первый взгляд, “случайными” числами. Если пользователь уверен, что случайное число, поступившее на обработку, является подлинным, атака становится невозможной. Следовательно, истинная случайность и знание, что полученное число действительно является случайным, очень сильно влияют на стойкость криптографической системы.

Рассмотрим еще одну причину, объясняющую необходимость достоверного источника случайных чисел. Вернемся к предыдущей главе, где рассмотрены протоколы с нулевым разглашением и честным верификатором. Эти протоколы основаны на применении вызывающих взаимное доверие случайных окликов, которые не должны вычисляться с помощью функции хэширования. Нечестный верификатор может атаковать протокол с нулевым разглашением и честным верификатором, генерируя псевдослучайные оклики с помощью функции хэширования (раздел 18.3.2.1). Следовательно, реализация протокола 1.1 (протокол подбрасывания монеты для генерации взаимно согласованных случайных чисел) с помощью функции SHA-1 непригодна для практических приложений.

Еще одна причина, по которой применение функции хэширования в протоколе подбрасывания монеты является нежелательным, — невозможность провести точный анализ стойкости, жизненно необходимый в серьезных приложениях.

Последний протокол, рассматриваемый в книге, представляет собой конкретную реализацию первого протокола. Пройдя курс современной криптографии, мы теперь в состоянии описать *хорошую* реализацию протокола 1.1 — знаменитый протокол “Подбрасывание монеты по телефону”, предложенный Блюмом [43].

19.1 Протокол Блюма “Подбрасывание монеты по телефону”

Рассмотрим спецификацию протокола Блюма “Подбрасывание монеты по телефону”. Этот протокол выполняется параллельно, позволяя сторонам согласовать случайное число, состоящее из m бит. Как и в протоколе 1.1, Алиса подбрасывает монету, а Боб угадывает результат.

В протоколе Блюма используется большое составное целое число $N = PQ$, где P и Q — два больших простых числа, удовлетворяющих условию

$$P \equiv Q \equiv 3 (\mathrm{mod}\, 4).$$

После опубликования протокола Блюма [43] эти целые числа стали называться **целыми числами Блюма**. Они обладают полезными криптографическими свойствами. В разделе 6.7 указаны некоторые факты из теории чисел, касающиеся чисел Блюма. Эти факты позволяют проанализировать стойкость протокола Блюма.

19.2 Анализ стойкости

В протоколе Блюма Алиса подбрасывает монету, а Боб пытается угадать результат. Следовательно, при анализе стойкости протокола необходимо оценить трудности, с которыми сталкиваются стороны, пытаясь организовать атаку. Существуют две возможности.

Протокол 19.1. Протокол Блюма "Подбрасывание монеты по телефону"

(* Этот протокол позволяет Алисе и Бобу согласовать строку, состоящую из m случайных битов. Как и в протоколе 1.1, Алиса подбрасывает монету, а Боб угадывает результат. *)

ПРАВИЛА:

- Стороны ставят цифровую подпись на каждое сообщение, посланное партнеру.
- Стороны прекращают выполнение протокола, если какая-либо проверка (включая верификацию цифровой подписи) обнаруживает нарушение правил.

1. Боб генерирует большое целое число Блюма $N = PQ$ и посылает его Алисе.
2. Алиса генерирует m случайных чисел: $x_1, x_2, \ldots, x_m \in_U \mathbb{Z}_N^*$. Результатами жеребьевки считаются числа $\left(\frac{x_i}{N}\right)$, $i = 1, 2, \ldots, m$. Затем она находит числа $y_1 \leftarrow x_1^2, y_2 \leftarrow x_2^2, \ldots, y_m \leftarrow x_m^2 (\mathrm{mod}\, N)$ и посылает их Бобу.
3. Боб генерирует случайные знаки $b_1, b_2, \ldots, b_m \in_U \{-1, 1\}$, пытаясь угадать знаки чисел $\left(\frac{x_i}{N}\right)$, где $i = 1, 2, \ldots, m$, и посылает их Алисе.

   (* Боб завершил угадывание результатов жеребьевки. *)
4. Алиса раскрывает Бобу числа $x_1, x_2, \ldots, x_m$.

   (* Алиса сообщает Бобу результаты его угадывания. *)
5. Боб проверяет условия $y_i \equiv x_i^2 (\mathrm{mod}\, N)$, где $i = 1, 2, \ldots, m$, и открывает Алисе числа P и Q.
6. Алиса проверяет условия $P \equiv Q \equiv 3 (\mathrm{mod}\, 4)$ и проверяет, являются ли числа P и Q простыми.
7. Алиса и Боб вычисляют согласованные случайные биты для $i = 1, 2, \ldots, m$.

$$r_i \leftarrow \begin{cases} 1, & \text{если Боб правильно угадал знак, т.е. } \left(\frac{x_i}{N}\right) = b_i, \\ 0 & \text{в противном случае.} \end{cases}$$

Алиса жульничает

Может ли Алиса подбросить монету и сообщить Бобу неверные результаты?

Боб обладает преимуществом при угадывании

Может ли Боб угадать результат с преимуществом больше $1/2$?

Оба эти вопроса допускают *количественный* ответ. Во-первых, жульничество Алисы связано с факторизацией целого числа Блюма N. Во-вторых, преимущество Боба при угадывании результата равно $1/2$.

Стойкость к жульничеству Алисы

Для того чтобы сжульничать, Алиса должна найти коллизию, т.е. два элемента $z_1, z_2 \in \mathbb{Z}_N^*$, удовлетворяющие следующим условиям.

- $z_1^2 \equiv z_2^2 (\mathrm{mod}\, N)$.
- $\left(\frac{z_1}{N}\right) \neq \left(\frac{z_2}{N}\right)$.

Допустим, что Алисе удалось обнаружить коллизию. Из теоремы 6.18.1 следует, что $\left(\frac{-1}{N}\right) = 1$. Для этого необходимо, чтобы $z_1 \neq \pm z_2 (\mathrm{mod}\, N)$, т.е.$0 < z_1 \pm$ $\pm\, z_2 < N$. Предположим противное, т.е. $z_1 = -z_2 (\mathrm{mod}\, N)$. Тогда

$$\left(\frac{z_1}{N}\right) = \left(\frac{-1}{N}\right)\left(\frac{z_2}{N}\right) = \left(\frac{z_2}{N}\right),$$

что противоречит условию коллизии $\left(\frac{z_1}{N}\right) \neq \left(\frac{z_2}{N}\right)$.

Теперь из условий

$$0 < z_1 + z_2 < N, 0 < |z_1 - z_2| < N$$

и

$$z_1^2 - z_2^2 = (z_1 + z_2)(z_1 - z_2) \equiv 0 (\mathrm{mod}\, N)$$

получаем

$$0 < \gcd(z_1 + z_2, N) < N.$$

Итак, Алиса разложила число N на простые множители.

Таким образом, мы доказали, что жульничество Алисы по сложности эквивалентно факторизации числа N, т.е. решению общеизвестной трудноразрешимой задачи. Здесь мы воспользовались методом "сведения к противоречию", принятым при анализе стойкости. Следовательно, второе свойство "волшебной" функции означает невозможность решения трудноразрешимой задачи, да еще в реальном времени.

Итак, функция в протоколе Блюма действительно является однонаправленной.

Преимущество при угадывании

Покажем теперь, что преимущество Боба при угадывании равно $1/2$.

При i-м подбрасывании монеты Алиса посылает Бобу число $y_i \equiv x_i^2 (\mathrm{mod}\, N)$. Получив число y_i, Боб должен угадать знак числа $\left(\frac{x_i}{N}\right)$. По теореме 6.18.3 число y_i имеет точно два квадратных корня с положительным символом Якоби и точно два квадратных корня — с отрицательным. Используя алгоритм 6.5, Боб может вычислить каждый из этих четырех квадратных корней. Однако он не может знать, какой из этих корней выбрала Алиса, а значит, он не может угадать знак символа Якоби выбранного Алисой корня. С точки зрения Боба, эта функция представляет

собой неоднозначное отображение. Боб может лишь угадывать корень с вероятностью 1/2.

Эта оценка характеризует вероятность события, сформулированного в первом свойстве "волшебной" функции.

19.3 Эффективность

Анализируя протокол, мы можем измерить вычислительные затраты его участников.

Затраты Алисы

Основные затраты Алисы связаны с 1) выполнением m операций возведения в квадрат; 2) вычислением m символов Якоби и 3) проверкой числа на простоту. Для возведения в квадрат и вычисления символа Якоби необходимо выполнить $O_B((\log N)^3)$ операций Для проверки простоты числа нужно выполнить $O_B((\log N)^2)$ операций. Итак, если $m = \log N$, то общие затраты Алисы равны $C(\log N)^3$, где C — небольшая константа. В эту оценку входят затраты на создание и верификацию цифровых подписей. Проще говоря, полные вычислительные затраты Алисы сравнимы с затратами на выполнение нескольких шифровок по схеме RSA.

С точки зрения пропускной способности канала Алиса посылает $2(\log N)^2$ бит (при условии, что $m = \log N$).

Затраты Боба

Легко видеть, что вычислительные затраты Боба равны затратам Алисы плюс затраты на генерацию модулей RSA. Иначе говоря, вычислительные затраты Боба равны сумме затрат на создание ключа RSA и многократное шифрование по схеме RSA. Чтобы сформулировать этот результат точнее, заменим константу C в выражении для оценки затрат Алисы величиной $\log N$: $O_B((\log N)^4)$.

Затраты Боба на связь намного меньше, чем у Алисы, поскольку он должен пересылать только модули, m случайных битов и множитель модулей, т.е. $3 \log N$ бит.

Очевидно, что эти затраты вполне разумны.

19.4 Резюме

В главе получены количественные оценки эффективности и стойкости протокола Блюма "Подбрасывание монеты по телефону".

- Сильная измеримая стойкость
- Анализ стойкости, проведенный в разделе 19.2, показывает, что реализация однонаправленной функции в протоколе подбрасывания монеты является

очень строгой и допускает *количественную* оценку: одно из свойств "волшебной" функции связано с решением трудноразрешимой задачи (факторизации целого числа), а второе носит безусловный характер.

- Практическая эффективность
- Анализ эффективности, проведенный в разделе 19.3, показывает, что протокол позволяет участникам согласовать строку, состоящую из m случайных бит, затратив на это количество операций, сравнимое с обычным шифрованием в криптосистеме с открытым ключом, имеющим длину m. Легко видеть, что эта оценка вполне приемлема для практических приложений.
- Использование практичных конструкций
  - Протокол может использовать обычные схемы цифровой подписи, предусматривающее возведение в квадрат, вычисление символа Якоби большого числа и проверку простоты числа по методу Монте-Карло. Эти алгоритмы и операции являются стандартными и доступны в большинстве библиотек криптографических алгоритмов.

Таким образом, протокол Блюма полностью соответствует всем требованиям, предъявляемым к криптографическим алгоритмам, протоколам и системам, перечисленным в главе 1.

Глава 20

Послесловие

В середине 1970-х годов криптография вступила в новую эру. Ее начало ознаменовало два события: опубликование стандарта шифрования данных (US Data Encryption Standard) и изобретение криптографии с открытым ключом. Теоретическое и практическое значение криптографии стимулировало рост академических исследований и активизацию коммерческих приложений. В настоящее время криптография представляет собой обширное поле деятельности, в котором постоянно рождаются новые идеи и методы.

В книге рассмотрена относительно небольшая, но важная часть современной криптографии. Избранные темы включают в себя методы, схемы, протоколы и системы, либо ставшие строительными конструкциями современных систем защиты информации (например, криптографические примитивы, описанные в главах 7–10 и основные конструкции протоколов аутентификации, изложенные в главе 11), либо получившие широкое распространение (например, реальные системы аутентификации, описанные в главе 12, и прикладные схемы шифрования и цифровой подписи, которым посвящены главы 15–16), либо имеющие большое значение для будущих приложений в области электронной коммерции и услуг (например, схемы идентификации сущностей из главы 13 и протоколы с нулевым разглашением из главы 18).

Книга содержит систематическое и глубокое изложение избранных методов, имеющих не только прикладное, но и теоретическое значение. Особое внимание в книге уделено следующим темам.

- Преодоление слабости “учебных” криптографических схем и протоколов.
- Усиление понятия стойкости для применения не только в теории, но и на практике.
- Описание прикладных криптографических схем и протоколов.
- Формальная методология и методы анализа стойкости.
- Иллюстрация формального доказательства сильной стойкости некоторых схем и протоколов.

Кроме того, в книге изложены теоретические основы современной криптографии, позволяющие читателям самостоятельно продолжить освоение этой обширной области знаний.

Библиография

[1] Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. Technical Report DEC SRC Technical Report 125, Digital Equipment Corporation, November 1995.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, Spring 2002.

[3] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication (extended abstract). In *Proceedings of Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, August 1991.

[4] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: an encryption scheme based on the Diffie-Hellman problem. Submission to IEEE P1363: A symmetric Encryption, 1998. Available at `grouper.ieee.org/groups/1363/P1363a/Encryption.html`.

[5] C. Abrams and A. Drobik. E-business opportunity index — the EU surges ahead. Research Note, Strategic Planning, SPA-10-7786, GartnerGroup RAS Services, 21, July 2000.

[6] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). The Internet Engineering Task Force Request For Comments (IETF RFC) 3161, August 2001. Available at `www.ietf.org/rfc/rfc3161.txt`.

[7] C. Adams and S. Farrell. Internet X.509 Public Key Infrastructure Certificate Management Protocols. The Internet Engineering Task Force Request For Comments (IETF RFC) 2510, March 1999. Available at `www.ietf.org/rfc/rfc2510.txt`.

[8] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. Online News, August 2002. `www.cse.iitk.ac.in/users/manindra/primality.ps`.

[9] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, 1974.

[10] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. E.cient, DoS-resistant, secure key exchange for Internet Protocols. In B. Christianson et al., editor, *Proceedings of Security Protocols, Lecture Notes in Computer Science 2467*, pages 27–39. Springer-Verlag, 2002.

[11] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. Efficient, DoS-resistant, secure key exchange for Internet Protocols.

In *Proceedings of ACM Conference on Computer and Communications Security (ACM-CCS'02)*, pages 48–58. ACM Press, November 2002.

[12] Alctel. Understanding the IPSec protocol suite. White Papers Archive, March 2000. Available at `www.ind.alctel.com/library/whitepapers/wp_IPSec.pdf`.

[13] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM Journal of Computing*, 17(2):194–209, April 1988.

[14] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., 2001.

[15] R. Anderson, E. Biham, and L. Knudsen. Serpent: A proposal for the advanced encryption standard. AES proposal: National Institute of Standards and Technology (NIST), 1998. Also available at `www.cl.cam.ac.uk/~rja14/serpent.html`.

[16] L. Babai. Talk presented at the 21st Annual Symposium on Foundation of Computer Science. San Juan, Puerto Rico, October 1979.

[17] R. Baldwin and R. Rivest. The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS algorithms. The Internet Engineering Task Force Request For Comments (IETF RFC) 2040, October 1996. Available at `www.ietf.org/rfc/rfc2040.txt`.

[18] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key-exchange protocols. In *Proceedings of the 30th Annual Symposium on the Theory of Computing (STOC'98)*, pages 419–428. ACM Press, 1998.

[19] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among nations of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — Proceedings of CRYPTO'98, Lecture Notes in Computer Science 1462*, pages 26–45. Springer-Verlag, 1998.

[20] M. Bellare and S. Micali. Non-interactive oblivious transfer and applications. In G. Brassard, editor, *Advances in Cryptology — Proceedings of CRYPTO'89, Lecture Notes in Computer Science 435*, pages 547–557. Springer-Verlag, 1990.

[21] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'00, Lecture Notes in Computer Science 1807*, pages 139–155. Springer-Verlag, 2000.

[22] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, New York, 1993. ACM Press.

[23] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. Stinson, editor, *Advances in Cryptology — Proceedings of CRYPTO'93, Lecture Notes in Computer Science 773*, pages 232–249. Springer-Verlag, 1994.

[24] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. de Santis, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'94, Lecture Notes in Computer Science 950*, pages 92–111. Springer-Verlag, 1995.

[25] M. Bellare and P. Rogaway. Provably secure session key distribution –the three party case. In *Proceedings of 27th ACM Symposium on the Theory of Computing*, pages 57–66. ACM Press, 1995.

[26] M. Bellare and P. Rogaway. The exact security of digital signatures – How to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'96, Lecture Notes in Computer Science 1070*, pages 399–416. Springer-Verlag, 1996.

[27] S. M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix UNIX Security Symposium*, pages 1–16, July 1996.

[28] S. M. Bellovin and M. Merritt. Limitations of the Kerberos authentication system. *ACM Computer Communication Review*, 20(5):119–132, 1990.

[29] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, 1992.

[30] J. C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the 26th Annual Symposium on the Theory of Computing (STOC'94)*, pages 544–553, 1994.

[31] C. Bennett and G. Brassard. The dawn of a new era for quantum cryptography: the experimental prototype is working! *SIGACT News*, 20:78–82, Fall 1989.

[32] R. Berger, S. Kannan, and R. Peralta. A framework for the study of cryptographic protocols. In H. C. Williams, editor, *Advances in Cryptology — Proceedings of CRYPTO'85, Lecture Notes in Computer Science 218*, pages 87–103. Springer-Verlag, 1986.

[33] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4:3–72, 1991.

[34] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of two-party authentication protocols. In J. Feigenbaum, editor, *Advances in Cryptology — Proceedings of CRYPTO'91, Lecture Notes in Computer Science 576, Springer-Verlag*, pages 44–61, 1992.

[35] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999. London Mathematical Society Lecture Note Series 265.

[36] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Proceedings of the sixth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science, 1355*, pages 30–45. Springer Verlag, 1997.

[37] S. Blake-Wilson and A. Menezes. Security proofs for entity authentication and authenticated key transport protocols emplying asymmetric techniques. In *Proceedings of 1997 Security Protocols Workshop, Lecture Notes in Computer Science 1361*, pages 137–158. Springer Verlag, 1998.

[38] S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman key agreement protocols. In S. Tavares and H. Meijer, editors, *Proceedings of Selected Areas in Cryptography (SAC'98), Lecture Notes in Computer Science 1556*, pages 339–361. Springer Verlag, 1999.

[39] M. Blaze. E.cient, DoS-resistant, secure key exchange for Internet protocols (Transcript of Discussion). In B. Christianson et al., editor, *Proceedings of Security Protocols, Lecture Notes in Computer Science 2467*, pages 40–48. Springer-Verlag, 2002.

[40] M. Blaze, J. Feigenbaum, and J. Lacy. Distributed trust management. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[41] D. Bleichenbacher. Generating ElGamal signature without knowing the secret key. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT' 96, Lecture Notes in Computer Science 1070*, pages 10–18. Springer-Verlag, 1996.

[42] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal of Computing*, 15(2):364–383, May 1986.

[43] M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. In *Proceedings of the 24th IEEE Computer Conference*, pages 133–137, May 1981.

[44] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112, 1988.

[45] M. Blum and S. Goldwasser. An *efficient* probabilistic public-key encryption scheme which hides all partial information. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology — Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196*, pages 289–299. Springer-Verlag, 1985.

[46] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. In *Proceedings of 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 112–117, 1982.

[47] D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of 3rd Algorithmic Number Theory Symposium, Lecture Notes in Computer Science 1423*, pages 48–63. Springer-Verlag, 1997.

[48] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, February 1999.

[49] D. Boneh. Simplifed OAEP for the RSA and Rabin functions. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 275–291. Springer-Verlag, 2001.

[50] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0,292}$. In J. Stern, editor, *Advances in Cryptology — Proceedings of EUROCRYPT' 99, Lecture Notes in Computer Science 1592*, pages 1–11. Springer-Verlag, 1999.

[51] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 213–229. Springer-Verlag, 2001.

[52] D. Boneh, A. Joux, and P.Q. Nguyen. Why textbook ElGamal and RSA encryption are insecure (extended abstract). In T. Okamoto, editor, *Advances in Cryptology — Proceedings of ASIACRYPT'00, Lecture Notes in Computer Science 1976*, pages 30–43. Springer-Verlag, 2000.

[53] A. Bosselaers, H. Dobbertin, and B. Preneel. The new cryptographic hash function RIPEMD-160. *Dr. Dobbs*, 22(1):24–28, January 1997.

[54] C. Boyd. Hidden assumptions in cryptographic protocols. *IEE Proceedings, Part E*, 137(6):433–436, November 1990.

[55] C. Boyd and W. Mao. On a limitations of BAN logic. In T. Helleseth, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'93, Lecture Notes in Computer Science 765*, pages 240–247. Springer-Verlag, 1993.

[56] C. Boyd, W. Mao, and K. Paterson. Deniable authentication for Internet Protocols. In *International Workshop on Security Protocols, Lecture Notes in Computer Science (to appear)*, pages Pre–proceedings:137–150. Springer-Verlag, April 2003. Sidney Sussex College, Cambridge, England.

[57] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'00, Lecture Notes in Computer Science 1807*, pages 156–171. Springer-Verlag, 2000.

[58] S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI Technical Report, 1993.

[59] G. Brassard, D. Chaum, and C. Crepeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Schiences*, 37(2):156–189, 1988.

[60] S. C. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the Association of Computing Machinery*, 31(7):560–599, 1984.

[61] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report SRC Technical Report 39, Digital Equipment Corporation, February 1989.

[62] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S.M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford, and N. Zunic. MARS - a candidate cipher for AES. AES proposal: National Institute of Standards and Technology (NIST), 1998. Also available at `www.research.ibm.com/security/mars.html`.

[63] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In J. Stern, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'99, Lecture Notes in Computer Science 1592*, pages 106–121. Springer-Verlag, 1999.

[64] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th Annual Symposium on the Theory of Computing (STOC'98)*, pages 209–218. ACM Press, 1998.

[65] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. A new version of [64], October 2002. Available at `xxx.lanl.gov/ps/cs.CR/0010019`.

[66] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. P.tzmann, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'01, Lecture Notes in Computer Science 2045*, pages 453–474. Springer-Verlag, 2001.

[67] R. Canetti and H. Krawczyk. Security analysis of IKE's signature-based keyexchange protocol. In M. Yung, editor, *Advances in Cryptology — Proceedings of CRYPTO'02, Lecture Notes in Computer Science 2442*, pages 143–161. Springer-Verlag, 2002. Also available at eprint.iacr.org.

[68] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password interception in a SSL/TLS channel. To appear in CRYPTO'03, March 2003. Available at `lasecwww.epfl.ch/memo_ssl.shtml`.

[69] U. Carlsen. Cryptographic protocol flaws: know your enermy. In *Proceedings of The Computer Security Foundations Workshop VII*, pages 192–200. IEEE Computer Society Press, 1994.

[70] S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of a 512-bit RSA modulus. In B. Preneel, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'00, Lecture Notes in Computer Science 1807*, pages 1–18. Springer-Verlag, 2000.

[71] D. Chaum. Demonstrating that a public predicate can be satisfied without revealing any information about how. In A.M. Odlyzko, editor, *Advances in Cryptology —*

Proceedings of CRYPTO'86, Lecture Notes in Computer Science 263, pages 195–199. Springer-Verlag, 1987.

[72] D. Chaum. Zero-knowledge undeniable signatures (extended abstract). In I.B. Damgard, editor, *Advances in Cryptology — Proceedings of CRYPTO'90, Lecture Notes in Computer Science 473*, pages 458–464. Springer-Verlag, 1991.

[73] D. Chaum and T.P. Pedersen. Wallet databases with observers. In E.F. Brickell, editor, *Advances in Cryptology — Proceedings of CRYPTO'92, Lecture Notes in Computer Science 740*, pages 89–105. Springer-Verlag, 1993.

[74] D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology — Proceedings of CRYPTO'89, Lecture Notes in Computer Science 435*, pages 212–216. Springer-Verlag, 1990.

[75] B. Chor. *Two Issues in Public Key Cryptography, RSA Bit Security and a New Knapsack Type System*. MIT Press, 1985. An ACM Distinguished Dissertation.

[76] B. Chor and O. Goldreich. RSA/Rabin least significant bits are $\frac{1}{2}+\frac{1}{\text{poly}(\log N)}$ secure. In G.T. Blakley and D. Chaum, editors, *Advances in Cryptology — Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196*, pages 303–313. Springer-Verlag, 1985.

[77] J. Clark and J. Jacob. A survey of authentication protocol literature: version 1.0. Online document, November 1997. Available at `www.cs.york.ac.uk/jac/papers/drareview.ps.gz`.

[78] C. Cocks. An identity-based public-key cryptosystem. In *Cryptography and Coding: 8th IMA International Conference, Lecture Notes in Computer Science 2260*, pages 360–363. Springer, December 2001.

[79] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1996. Graduate Texts in Mathematics 138.

[80] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[81] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38:243–250, 1994.

[82] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'96, Lecture Notes in Computer Science 1070*, pages 178–189. Springer-Verlag, 1996.

[83] J. S. Coron, M. Joye, D. Naccache, and P. Paillier. Universal padding schemes for RSA. In M. Yung, editor, *Advances in Cryptology — Proceedings of CRYPTO'02, Lecture Notes in Computer Science 2442*, pages 226–241. Springer-Verlag, 2002.

[84] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology — Proceedings of CRYPTO'98, Lecture Notes in Computer Science 1462*, pages 13–25. Springer-Verlag, 1998.

[85] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proceedings of 6th ACM Conference on Computer and Communication Security*. ACM Press, November 1999.

[86] J. Daemen and V. Rijmen. AES Proposal: Rijndael. AES proposal: National Institute of Standards and Technology (NIST), October, 6, 1998. Available at `csrc.nist.gov/encryption/aes/`.

[87] J. Daemen and V. Rijmen. *The Design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002. ISBN:3540425802.

[88] I. Damgard. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology — Proceedings of CRYPTO'91, Lecture Notes in Computer Science 576*, pages 445–456. Springer-Verlag, 1992.

[89] D.W. Davies and W.L. Price. *Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer (second edition)*. John Wiley & Sons, 1989.

[90] D. Davis and R. Swick. Workstation services and Kerberos authentication at Project Athena. Technical Memorandum TM-424, MIT Laboratory for Computer Science, February 1990.

[91] R.A. DeMillo, G.L. Davida, D.P. Dobkin, M.A. Harrison, and R.J. Lipton. *Applied Cryptology, Cryptographic Protocols, and Computer Security Models*, volume 29. Providence: American Mathematical Society, 1983. Proceedings of Symposia in Applied Mathematics.

[92] R.A. DeMillo andM.J. Merritt. Protocols for data security. *Computer*, 16(2):39–50, Febrary 1983.

[93] D. Denning. *Cryptography and Data Security*. Addison-Wesley Publishing Company, Inc., 1982.

[94] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[95] T. Dierks and C. Allen. The TLS Protocol, Version 1.0. Request for Comments: 2246, January 1999.

[96] W. Diffie. The first ten years of public key cryptology. In G.J. Simmons, editor, *Contemporary Cryptology, the Science of Information Integrity*, pages 135–175. IEEE Press, 1992.

[97] W. Diffie and M. Hellman. Multiuser cryptographic techniques. In *Proceedings of AFIPS 1976 NCC*, pages 109–112. AFIPS Press, Montvale, N.J., 1976.

[98] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, IT-22(6):644–654, 1976.

[99] W. Diffie, P.C. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[100] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of 23rd Annunal ACM Symposium on Theory of Computing*, pages 542–552, 1991. Journal version in *SIAM Journal on Computing*, vol 30, no. 2, 391–437, 2000.

[101] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of IEEE 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, 1981.

[102] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.

[103] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. The Internet Engineering Task Force Request For Comments (IETF RFC) 2693, September 1999. Available at `www.ietf.org/rfc/rfc2693.txt`.

[104] eMarketer. Security online: Corporate & consumer protection, e-telligence for business. eMarketer Report, February 2003. Available at `www.emarketer.com`.

[105] A. Evans Jr., W. Kantrowitz, and E. Weiss. A user authentication scheme not requiring secrecy in the computer. *Communications of the ACM*, 17(8):437–442, 1974.

[106] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *ACM Special Interest Group on Algorithms and Computation Theory (SIGACT)*, pages 210–217, 1987.

[107] H. Feistel. Cryptography and computer privacy. *Sci. Am.*, 228(5):15–23, May 1974.

[108] N. Ferguson and B. Schneier. A cryptographic evaluation of IPsec. Counterpane Labs, 2000. Available at `www.counterpane.com/ipsec.pdf`.

[109] A. Fiat and A. Shamir. How to prove yourself: practical solutions of identification and signature problems. In A.M. Odlyzko, editor, *Advances in Cryptology — Proceedings of CRYPTO'86, Lecture Notes in Computer Science 263*, pages 186–194. Springer-Verlag, 1987.

[110] Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*. O'Reilly & Associates, May 1998. ISBN 1-56592-520-3.

[111] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL Protocol, Version 3.0. INTERNET-DRAFT, draft-freier-ssl-version3-02.txt, November 1996.

[112] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, editors, *Public Key Cryptography — Proceedings of PKC'99, Lecture Notes in Computer Science 1560*, pages 53–68. Springer-Verlag, 1999.

[113] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology — Proceedings of CRYPTO'99, Lecture Notes in Computer Science 1666*, pages 537–554. Springer-Verlag, 1999.

[114] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP Is secure under the RSA assumption. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 260–274. Springer-Verlag, 2001.

[115] K. Gaarder and E. Snekkenes. Applying a formal analysis technique to the CCITT X.509 strong two-way authentication protocol. *Journal of Cryptology*, 3(2):81–98, 1991.

[116] S. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Advances in Cryptology — Proceedings of ASIACRYPT'01, Lecture Notes in Computer Science 2248*, pages 495–513. Springer-Verlag, 2001.

[117] S. D. Galbraith, W. Mao, and K. G. Paterson. A cautionary note regarding cryptographic protocols based on composite integers. Technical Report HPL-2001-284, HP Laboratories, Bristol, November 2001.

[118] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[119] C. F. Gauss. *Disquisitiones Arithmeticae*. Translated by A. Arthur and S.J. Clark, 1996, Yale University Press, New Haven, 1801.

[120] R. Gennaro, D. Miccianicio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *5th ACM Conference on Computer and Communications Security, Fairfax, Virginia*, 1998.

[121] M. Girault. An identity-based identi.cation scheme based on discrete logarithms modulo a composite number. In I.B. Damgard, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'90, Lecture Notes in Computer Science 473*, pages 481–486. Springer-Verlag, 1991.

[122] M. Girault. Self-certified public keys. In D. W. Davies, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'91, Lecture Notes in Computer Science 547*, pages 490–497. Springer-Verlag, 1991.

[123] I. Goldberg and D.Wagner. Randomness and the Netscape browser, how secure is the World Wide Web? *Dr. Dobb's Journal*, pages 66–70, January 1996.

[124] O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In A.M. Odlyzko, editor, *Advances in Cryptology — Proceedings of CRYPTO'86, Lecture Notes in Computer Science 263*, pages 171–185. Springer-Verlag, 1987.

[125] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[126] S. Goldwasser, S. Micali, and C. Racko.. The knowledge complexity of interactive proof-systems. In *Proceedings of 17th Ann. ACM Symp. on Theory of Computing*, pages 291–304, 1985. A journal version under the same title appears in: *SIAM Journal of Computing* vol. 18, pp. 186–208, 1989.

[127] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[128] S. Goldwasser, S. Micali, and P. Tong. Why and how to establish a private code on a public network. In *Proceedings of 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 134–144, 1982.

[129] D. Gollmann. *Computer Security*. John Wiley & Sons, Inc., 1999. ISBN: 0-471-97884-2.

[130] D. Gollmann. Authentication — myths and misconceptions. *Progress in Computer Science and Applied Logic*, 20:203–225, 2001. Birkhauser Verlag Basel/Switzerland.

[131] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, 1990.

[132] F. T. Grampp and R. H. Morris. Unix operating system security. *AT&T Bell Laboratories Technical Journal*, 63(8):1649–1672, October 1984.

[133] C. G. Gunther. An identity-based key-exchange protocol. In J.-J. Quisquater and J. Vanderwalle, editors, *Advances in Cryptology — Proceedings of EUROCRYPT' 89, Lecture Notes in Computer Science 434*, pages 29–37. Springer-Verlag, 1990.

[134] N. M. Haller. The S/KEY one-time password system. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 151–157, 1994.

[135] D. Harkins and D. Carrel. The Internet key exchange protocol (IKE). The Internet Engineering Task Force Request For Comments (IETF RFC) 2409, November 1998. Available at `www.ietf.org/rfc/rfc2409.txt`.

[136] K. E. B. Hickman. The SSL Protocol. Online document, Feburary 1995. Available at `www.netscape.com/eng/security/SSL_2.html`.

[137] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8), 1978.

[138] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985. Series in Computer Science.

[139] P. Hofman. Features of proposed successors to IKE. INTERNET-DRAFT, draft-ietf-ipsec-soi-features-01.txt, May 2002.

[140] R. Housley and P. Hoffman. Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP. The Internet Engineering Task Force Request For Comments (IETF RFC) 2585, August 2001. Available at `www.ietf.org/rfc/rfc2585.txt`.

[141] D. Huhnlein, M. Jakobsson, and D. Weber. Towards practical non-interactive public key cryptosystems using non-maximal imaginary quadratic orders. In *Prodeedings of Selected Areas of Cryptography — SAC 2000, Lecture Notes in Computer Science 2012*, pages 275–287. Springer-Verlag, 2000.

[142] ISO/IEC. Information Processing — Modes of operation for an *n*-bit block cipher algorithm. International Organization for Standardization and International Electrotechnical Commission, 1991. 10116.

[143] ISO/IEC. Information Technology — Security Techniques — summary of voting on letter ballot No.6, Document SC27 N277, CD 9798-3.3 "Entity Authentication Mechanisms" — Part 3: Entity authentication mechanisms using a public key algorithm. International Organization for Standardization and International Electrotechnical Commission, October 1991. ISO/IEC JTC 1/SC27 N313.

[144] ISO/IEC. Information Technology — Security Techniques — Entity Authentication Mechanisms — Part 2: Entity authentication using symmetric techniques. International Organization for Standardization and International Electro-technical Commission, 1992. ISO/IEC JTC 1/SC 27 N489 CD 9798-2, 1992-06-09.

[145] ISO/IEC. Information Technology — Security Techniques — Entity Authentication Mechanisms — Part 2: Entity authentication using symmetric techniques. International Organization for Standardization and International Electro-technical Commission, 1993. ISO/IEC JTC 1/SC 27 N739 DIS 9798-2, 1993-08-13.

[146] ISO/IEC. Information Technology — Security Techniques — Entity Authentication — Part 1: General. International Organization for Standardization and International Electro-technical Commission, 1996. ISO/IEC JTC 1/SC 27 DIS 9798-1:1996 (E).

[147] ISO/IEC. Information Technology — Security Techniques — Entity Authentication — Part 2: Mechanisms using symmetric encipherment algorithms. International Organization for Standardization and International Electro-technical Commission, December 1998. ISO/IEC JTC 1/SC 27 N2145 FDIS 9798-2.

[148] ISO/IEC. Information Technology — Security Techniques — Entity Authentication — Part 3: Mechanisms using digital signature techniques. International Organization for Standardization and International Electro-technical Commission, October 1998. BS ISO/IEC 9798-3.

[149] ISO/IEC. Information Technology — Security Techniques — Entity Authentication — Part 4: Mechanisms using a cryptographic check function. International Organization for Standardization and International Electro-technical Commission, April 1999. ISO/IEC JTC 1/SC 27 N2289 FDIS 9798-4.

[150] ISO/IEC. Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms. International Organization for Standardization and International Electro-technical Commission, April 2000. ISO/IEC JTC 1/SC 27 9796-3.

[151] ISO/IEC. Information Technology — Security Techniques — Hash Functions — Part 3: Dedicated hash-functions. International Organization for Standardization and International Electro-technical Commission, November 2001. ISO/IEC JTC1, SC27, WG2, Document 1st CD 10118-3.

[152] ITU-T. Rec. X.509 (revised) the Directory — Authentication Framework, 1993. International Telecommunication Union, Geneva, Switzerland (equivalent to ISO/IEC 9594-8:1995.).

[153] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'96, Lecture Notes in Computer Science 1070*, pages 143–154. Springer-Verlag, 1996.

[154] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory, IV-th Symposium (ANTS IV), Lecture Notes in Computer Science 1838*, pages 385–394. Springer-Verlag, 2000.

[155] A. Joux and K. Nguyen. Separating decision Di.e-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, 2001/003, 2001. Available at http:/eprint.iacr.org/.

[156] R. Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on Sortware Engineering*, 22(5):313–328, May 1996.

[157] C. Kaufman. Comparison of IKEv2, JFK, and SOI requirements. The Internet Engineering Task Force: on line document, April 2002. Available at `www.ietf.org/proceedings/02mar/slides/ipsec-1/`.

[158] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. The Internet Engineering Task Force: INTERNET-DRAFT, draft-ietf-ipsec-ikev2-03.txt, October 2002. Available at `www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-03.txt`.

[159] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World, Second Edition*. Prentice-Hall PTR, 2002.

[160] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[161] S. Kent and R. Atkinson. IP Authentication Header. The Internet Engineering Task Force Request For Comments (IETF RFC) 2402, November 1998. Available at `www.ietf.org/rfc/rfc2402.txt`.

[162] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). The Internet Engineering Task Force Request For Comments (IETF RFC) 2406, November 1998. Available at `www.ietf.org/rfc/rfc2406.txt`.

[163] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. The Internet Engineering Task Force Request For Comments (IETF RFC) 2401, November 1998. Available at `www.ietf.org/rfc/rfc2401.txt`.

[164] J. Klensin. Simple mail transfer protocol. The Internet Engineering Task Force Request For Comments (IETF RFC) 2821, April 2001. Available at `www.ietf.org/rfc/rfc2821.txt`.

[165] L. R. Knudsen. *Block Ciphers — Analysis, Design and Applications*. Arhus University, 1994.

[166] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(5):203–209, 1987.

[167] P. C. Kocher. Timing attacks on implementations of Di.e-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology — Proceedings of CRYPTO'96, Lecture Notes in Computer Science 1109*, pages 104–113. Springer-Verlag, 1996.

[168] J. Kohl and C. Neuman. The Kerberos network authentication service (v5). The Internet Engineering Task Force Request For Comments (IETF RFC) 1510, September 1993. Available at `www.ietf.org/rfc/rfc1510.txt`.

[169] L. M. Kohnfelder. *Towards a Practical Public-key Cryptosystem*. MIT B.S. Thesis, MIT Department of Electrical Engineering, May 1978.

[170] E. Kranakis. *Primality and Cryptography*. John Wiley & Sons, 1986. Wiley-Teubner Series in Computer Science.

[171] H. Krawczyk. SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman protocols. Online document, 1996. Available at `www.ee.technion.ac.il/~hugo/sigma.html`.

[172] H. Krawczyk. SKEME: a versatile secure key exchange mechanism for Internet. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, pages 114–127. IEEE Computer Society Press, February 1996.

[173] L. Lamport. Constructing digital signatures from a one way function. SIR International, October 1979. Available at `www.csl.sri.com/papers/676/`.

[174] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[175] A. Lenstra and E. Verheul. The XTR public key system. In M. Bellare, editor, *Advances in Cryptology — Proceedings of CRYPTO'00, Lecture Notes in Computer Science 1880*, pages 1–19. Springer-Verlag, 2000.

[176] W. J. LeVeque. *Fundamentals of Number Theorey*. Dover Publications, Inc., 1977.

[177] R. Lidl and H. Niederpeiter. *Finite Fields*. Cambridge University Press, 1997. Encyclopedia of Mathematics and its Applications 20.

[178] R. J. Lipton. How to cheat at mental poker. Technical Report, Comp. Sci., Dept. Univ. of Calif., Berkeley, Calif., August 1979. (This is an internal technical report; a simple description of the attack is available in page 174 of [91]).

[179] G. Lowe. Some new attacks upon security protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society Press, June 1994.

[180] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[181] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In *Procedings of TACAS, Lecture Notes in Computer Science 1055*, pages 147–166. Springer-Verlag, 1996.

[182] J. Malone-Lee and W. Mao. Two birds one stone: Signcryption using RSA. In M. Joye, editor, *Topics in Cryptology — the Cryptographers' Track, Proceedings of the RSA Conference 2003 (CT-RSA 2003), Lecture Notes in Computer Science 2612*, pages 210–224. Springer-Verlag, April 2003.

[183] W. Mao. An augmentation of BAN-like logics. In *Proceedings of Computer Security Foundations Workshop VIII*, pages 44–56. IEEE Computer Society Press, June 1995.

[184] W. Mao and C. Boyd. On the use of encryption in cryptographic protocols. In P. G. Farrell, editor, *Codes and Cyphers — Proceedings of 4th IMA Conference on Cryptography and Coding*, pages 251–262, December 1993. The Institute of Mathematics and Its Applications, 1995.

[185] W. Mao and C. Boyd. On the use of encryption in cryptographic protocols, February 1994. Distributed by International Organization for Standardization (ISO) and International Electro-technical Commission (IEC) JTC1, SC27, WG2, Document N262: "Papers on authentication and key management protocols based on symmetric techniques." This ISO document distributes the paper published in [184].

[186] W. Mao and C. Boyd. Methodical use of cryptographic transformations in authentication protocols. *IEE Proceedings, Comput. Digit. Tech.*, 142(4):272–278, July 1995.

[187] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet security association and key management protocol (ISAKMP), version 10. INTERNETDRAFT: draft-ietf-ipsec-isakmp-10.txt, November 1998. Also available at `www.ietf.org/rfc/rfc2408.txt`.

[188] U. Maurer. Protocols for secret key agreement by public discussion based on common information. In E.F. Brickell, editor, *Advances in Cryptology — Proceedings of CRYPTO'92, Lecture Notes in Computer Science 740*, pages 461–470. Springer-Verlag, 1993.

[189] U. Maurer. Secret key agreement by public discussion from common information. *IEEE Transactions on Information Theory*, IT-39:733–742, 1993.

[190] U. Maurer and S. Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logairhms. *SIAM Journal of Computing*, 28(5):1689–1721, 1999.

[191] U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In D.W.Davies, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'91, Lecture Notes in Computer Science 547*, pages 498–507. Springer-Verlag, 1991.

[192] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1):5–53, 1992.

[193] C. Meadows. Analyzing the Needham-Schroeder public key protocol: a comparison of two approaches. In E. Bertino et al, editor, *Proceedings of Computer Security, ESORICS'96, Lecture Notes in Computer Science 1146*, pages 351–364. Springer-Verlag, February 1996.

[194] C. Meadows. The NRL Protocol Analyzer: an overview. *Journal of Logic Programming*, 26(2):113–131, February 1996.

[195] C. Meadows. Analysis of the internet key exchange protocol using the NRL Protocol Analyzer. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 216–231. IEEE Computer Society Press, May 1999.

[196] C. Meadows and P. Syverson. A formal specification of requirements for payment transactions in the SET protocol. In R. Hirschfeld, editor, *Proceedings of Financial Cryptography (FC'98), Lecture Notes in Computer Science 1465*, pages 122–140. Springer-Verlag, February 1998.

[197] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to a finite field. *IEEE Trans. Info. Theory*, 39:1636–1646, 1983.

[198] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[199] R. C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21:294–299, 1978.

[200] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. on Info. Theory*, 24:525–530, 1978.

[201] S. Micali and R. L. Rivest. Micropayments revisited. In B. Preneel, editor, *Topics in Cryptology — the Cryptographers' Track, Proceedings of the RSA Conference 2002 (CT-RSA 2002), Lecture Notes in Computer Science 2271*, pages 149–163. Springer-Verlag, 2002.

[202] S. P. Miller, C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. Project Athena Technical Plan Section E.2.1, 1987.

[203] V. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology — Proceedings of CRYPTO'85, Lecture Notes in Computer Science 218*, pages 417–426. Springer-Verlag, 1986.

[204] J. H. Moore. Protocol failures in cryptosystems. *Proceedings of the IEEE*, 76(5):594–601, 1988.

[205] J. H. Moore. Protocol failures in cryptosystems. In G. J. Simmons, editor, *Contemporary Cryptology, the Science of Information Integrity*, pages 541–558. IEEE Press, 1992.

[206] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 22(5):594–597, 1979.

[207] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. The Internet Engineering Task Force Request For Comments (IETF RFC) 2560, June 1999. Available at `www.ietf.org/rfc/rfc2560.txt`.

[208] M. Myers, X. Liu, J. Schaad, and J. Weinstein. Certi.cate Management Messages over CMS. The Internet Engineering Task Force Request For Comments (IETF RFC) 2797, April 2000. Available at `www.ietf.org/rfc/rfc2797.txt`.

[209] M. Naor and O. Reingold. Number theoretic constructions of efficient pseudoran-dom functions. In *Proceedings of FOCS'97*, pages 458–467, 1997.

[210] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of 22nd ACM Symposium of Theory of Computing*, pages 427–437, 1990.

[211] NBS. Data Encryption Standard. U.S. Department of Commerce, FIPS Publication 46, Washington, D.C., January 1977. National Bureau of Standards.

[212] R. Needham and M. Schroeder. Authentication revisited. *Operating Systems Review*, 21:7, 1987.

[213] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[214] B. C. Neuman and S. G. Stubblebine. A note on the use of timestamps as nonces. *ACM Operating Systems Review*, 27(2):10–14, April 1993.

[215] NIST. A Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). Federal Register Announcement August 30, 1991. National Institute of Standards and Technology.

[216] NIST. Digital Signature Standard. Federal Information Processing Standards Publication 186, 1994. U.S. Department of Commerce/N.I.S.T.

[217] NIST. Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB) 180-1, April 1995. U.S. Department of Commerce/N.I.S.T.

[218] NIST. Recommendation for block cipher modes of operation. NIST Special Publication 800-38A 2001 Edition, December 2001. U.S. Department of Commerce/N.I.S.T.

[219] NIST. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication (FIPS PUB) 197, November 2001. U.S. Department of Commerce/N.I.S.T.

[220] K. Nyberg and R. Rueppel. A new signature scheme based on the DSA giving message recovery. In *1st ACM Conference on Computer and Communications Security*, pages 58–61. ACM Press, 1993.

[221] A. M. Odlyzko. Discrete logarithms: the past and the future. *Designs, Codes and Cryptography*, 19:129–154, 2000.

[222] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identi.cation. In H. Krawczyk, editor, *Advances in Cryptology — Proceedings of CRYPTO'98, Lecture Notes in Computer Science 1462*, pages 345–370. Springer-Verlag, 1998.

[223] T. Okamoto and D. Pointcheval. REACT: rapid enhanced-security asymmetric cryptosystem transform. In D. Naccache, editor, *Topics in Cryptography, Cryptographers' Track, RSA Conference 2001 — Proceedings of CT-RSA'00, Lecture Notes in Computer Science 2020*, pages 159–175. Springer-Verlag, 2001.

[224] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Advances in Cryptology — Proceedings of EUROCRYPT' 98, Lecture Notes in Computer Science 1403*, pages 308–318. Springer-Verlag, 1998.

[225] H. Orman. The Oakley key determination protocol, version 2. draft-ietf-ipsecoakley-02.txt, 1996.

[226] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, 1987.

[227] Oxford. *Oxford Reference, Dictionary of Computing, Third Edition*. Oxford University Press, 1991.

[228] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — Proceedings of EUROCRYPT' 99, Lecture Notes in Computer Science 1592*, pages 223–238. Springer-Verlag, 1999.

[229] J. Patarin and L. Goubin. Trapdoor one-way permutations and multivariate polynomials. In Y. Han, T. Okamoto, and S. Qing, editors, *Information and Communications Security — Proceedings of ICICS'97, Lecture Notes in Computer Science 1334*, pages 356–368. Springer-Verlag, 1997.

[230] PKCS. Public Key Cryptography Standards, PKCS#1 v2.1. RSA Cryptography Standard, Draft 2, 2001. Available at `www.rsasecurity.com/rsalabs/pkcs/`.

[231] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[232] D. Pointcheval. HD-RSA: hybrid dependent RSA, a new public-key encryption scheme. Submission to IEEE P1363: A symmetric Encryption, 1999. Available at `grouper.ieee.org/groups/1363/P1363a/Encryption.html`.

[233] D. Pointcheval. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — Proceedings of EUROCRYPT' 99, Lecture Notes in Computer Science 1592*, pages 239–254. Springer-Verlag, 1999.

[234] D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In H. Imai and Y. Zheng, editors, *Public Key Cryptography — Proceedings of PKC'00, Lecture Notes in Computer Science 1751*, pages 129–146. Springer-Verlag, 2000.

[235] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'96, Lecture Notes in Computer Science 1070*, pages 387–398. Springer-Verlag, 1996.

[236] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[237] J. M. Pollard. Theorems on factorization and primality testing. *Proceedings of the Cambridge Philosophical Society*, 76:521–528, 1974.

[238] J. M. Pollard. Monte Carlo method for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[239] M. Rabin. Transaction protection by beacons. Technical Report Tech.Rep. 29-81, Aiken Computation Lab., Harvard University, Cambridge, MA, 1981.

[240] M. O. Rabin. Digitized signatures and public-key functions as intractible as actorization. Technical Report LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[241] C. Racko. and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology — Proceedings of CRYPTO'91, Lecture Notes in Computer Science 576*, pages 433–444. Springer-Verlag, 1992.

[242] R. Rivest and A. Shamir. PayWord and MicroMint: two simple micropayment schemes. *CryptoBytes, RSA Laboratories*, 2(1):7–11, Spring 1996.

[243] R. L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments 1321, April 1992.

[244] R. L. Rivest. S-expressions. INTERNET-DRAFT, May 1997. Available at `theory.lcs.mit.edu/~rivest/sexp.txt`.

[245] R. L. Rivest and B. Lampson. SDSI — A simple distributed security infrastructure. Invited Speech at CRYPTO'96, August 1996. Available at `theory.lcs.mit.edu/~cis/sdsi.html`.

[246] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[247] R. Sidney R. L. Rivest, M. J. B. Robshaw and Y. L. Yin. The RC6 Block Cipher, v1.1. AES proposal: National Institute of Standards and Technology (NIST), 1998. Available at `www.rsa.com/rsalabs/aes/`.

[248] A. W. Roscoe. Model checking CSP. In A.W. Roscoe, editor, *A Classical Mind: Essays in honour of C.A.R. Hoare*. Prentice-Hall, 1994.

[249] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *Proceedings of Computer Security Foundations Workshop VIII*, pages 98–107. IEEE Computer Society Press, June 1995.

[250] P. Ryan and S. Schneider. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2001.

[251] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the 2000 Symposium on Cryptography and Information Security, Okinawa, Japan*, January 2000.

[252] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comm. Math. Univ. Sancti. Pauli*, 47:81–92, Spring 1998.

[253] S. Schneider. Security properties and CSP. In *Proceedings of the 1996 IEEE Symposium in Security and Privacy*, pages 174–187. IEEE Computer Society Press, 1996.

[254] B. Schneier. *Secrets and Lies*. John Wiley & Sons, 2001.

[255] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: a 128-bit block cipher, AES proposal. AES proposal: National Institute of Standards and Technology (NIST), 1998. Available at `www.counterpane.com/twofish.html`.

[256] C. P. Schnorr. Efficient identification and signature for smart cards. In G. Brassard, editor, *Advances in Cryptology — Proceedings of CRYPTO'89, Lecture Notes in Computer Science 435*, pages 239–252. Springer-Verlag, 1990.

[257] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[258] L. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comp.*, 67(221):353–356, 1998.

[259] SET. Secure Electronic Transaction Speci.cation, Version 1.0. Online document, May 1997. Available at `www.setco.org/`.

[260] A. Shamir. Identity-based cryptosystems and signature schemes. In G.T. Blakley and D. Chaum, editors, *Advances in Cryptology — Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196*, pages 48–53. Springer-Verlag, 1985.

[261] A. Shamir, R. Rivest, and L. Adleman. Mental poker. In D. Klarner, editor, *The Mathematical Gardner*, pages 37–43, Boston, Mass, 1980. Prindle, Weber & Schmidt.

[262] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(3):379–423, July 1948.

[263] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:623–656, October 1948. Continued from July 1948 issue (i.e., [262]).

[264] C. E. Shannon. Communications theory of secrecy systems. *Bell Systems Technical Journal*, 28:656–715, October 1949.

[265] C. E. Shannon. Predilection and entropy of printed English. *Bell Systems Technical Journal*, 30:50–64, January 1951.

[266] R. Shirey. Internet Security Glossary. The Internet Engineering Task Force Request For Comments (IETF RFC) 2828, May 2000. Available at `www.ietf.org/rfc/rfc2828.txt`.

[267] P. W. Shor. Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal of Computing*, 26:1484–1509, 1997.

[268] P. W. Shor. Why haven't more quantum algorithms been found? *Journal of the ACM*, 50(1):87–90, January 2003.

[269] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'00, Lecture Notes in Computer Science 1807*, pages 275–288. Springer-Verlag, 2000.

[270] V. Shoup. OAEP reconsidered. In J. Killian, editor, *Advances in Cryptology — Proceedings of CRYPTO'01, Lecture Notes in Computer Science 2139*, pages 239–259. Springer-Verlag, 2001.

[271] V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). Distributed by International Organization for Standardization (ISO) and International Electro-technical Commission (IEC) JTC1, SC27, WG2, December 2001. An earlier version appeared in ISO/IEC JTC 1/SC 27 N2765 "Editor's contribution on public key encryption" (February 2001).

[272] J. H. Silverman. *The Arithmetic of Elliptic Curves*. Sprinber-Verlag, 1986. Graduate Texts in Mathematics.

[273] R. D. Silverman. Fast generation of random, strong RSA primes. *CryptoBytes*, 3(1):9–13, 1997.

[274] G. J. Simmons. How to (selectively) broadcast a secret. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 108–113. IEEE Computer Society Press, 1985.

[275] G. J. Simmons. A survey of information authentication. In G.J. Simmons, editor, *Contemporary Cryptology, the Science of Information Integrity*, pages 379–419. IEEE Press, 1992.

[276] D. Simon. On the power of quantum computation. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 116–123, 1994.

[277] S. Singh. *The Code Book*. Fourth Estate, 1999.

[278] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12:193–196, 1999.

[279] M. E. Smid and D. K. Branstad. The Data Encryption Standard, past and future. In G.J. Simmons, editor, *Contemporary Cryptology, the Science of Information Integrity*, pages 43–46. IEEE Press, 1992.

[280] D. Soldera. SEG — a provably secure variant of El-Gamal. Technical Report HPL-2001-149, Hewlett-Packard Laboratories, Bristol, June 2001.

[281] D. Soldera, J. Seberry, and C. Qu. The analysis of Zheng-Seberry scheme. In L. M. Batten and J. Seberry, editors, *7th Australian Conference in Information Security and Privacy — Proceedings of ACISP'02, Lecture Notes in Computer Science 2384*, pages 159–168. Springer-Verlag, 2002.

[282] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal of Computing*, 6(1):84–85, March 1977.

[283] M. Stadler. Publicly verifiable secret sharing. In U. Maurer, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'96, Lecture Notes in Computer Science 1070*, pages 190–199. Springer-Verlag, 1996.

[284] D. R. Stinson. *Cryptography: Theory and Practice*. CRC Press, Inc., 1995.

[285] P. Syverson. On key distribution protocols for repeated authentication. *ACM Operating Systems Review*, 27(4):24–30, October 1993.

[286] P. Syverson and P.C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of 1994 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1994.

[287] H. Tanaka. A realization scheme for the identity-based cryptosystem. In C. Pomerance, editor, *Advances in Cryptology — Proceedings of CRYPTO'87, Lecture Notes in Computer Science 293*, pages 340–349. Springer-Verlag, 1988.

[288] G. Trudik. Message authentication with one-way functions. *Computer Communication Review*, 22:29–38, 1992.

[289] S. Tsuji and T. Itoh. An ID-based cryptosystem based on the discrete logarithm problem. *IEEE Journal on Selected Areas in Communication*, 7(4):467–473, 1989.

[290] W. Tuchman. Hellman presents no shortcut solutions to the DES. *IEEE Spectrum*, 16(7):40–41, 1979.

[291] G. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In C. Pomerance, editor, *Advances in Cryptology — Proceedings of CRYPTO'87, Lecture Notes in Computer Science 293*, pages 128–134. Springer-Verlag, 1988.

[292] P.C. van Oorschot. Extending cryptographic logics of belief to key agreement protocols (extended abstract). In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 232–243, 1993.

[293] V. Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed systems. In *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 255–275, 1991.

[294] S. Vaudenay. Security flaws induced by CBC padding – Applications to SSL, IPSEC, WTLS. . . . In L. R. Knudsen, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'02, Lecture Notes in Computer Science 2332*, pages 534–545. Springer-Verlag, 2002.

[295] U. Vazirani and V. Vazirani. E.cient and secure pseudo-random number generation (extended abstract). In G.T. Blakley and D. Chaum, editors, *Advances in Cryptology — Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196*, pages 193–202. Springer-Verlag, 1985.

[296] E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In B. P.tzmann, editor, *Advances in Cryptology — Proceedings of EUROCRYPT'01, Lecture Notes in Computer Science 2045*, pages 195–210. Springer-Verlag, 2001.

[297] D. Wheeler. Transactions using bets. In M. Lomas, editor, *Security Protocols, Lecture Notes in Computer Science 1189*, pages 89–92. Springer-Verlag, 1996.

[298] M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

[299] M. Wiener. Efficient DES key search. Technical report, TR-244, School of Computer Science, Carleton University, Ottawa, May 1994.

[300] C. P. Williams and S. H. Clearwater. *Ultimate Zero and One*. Copernicus, Springer-Verlag New York, Inc., 2000.

[301] T. Y. C. Woo and S. S. Lam. Authentication for distributed systems. *Computer*, 25(1):39–52, January 1992.

[302] T. Y. C. Woo and S.S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, 28(3):24–37, July 1994.

[303] A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[304] T. Ylonen. The SSH (secure shell) remote login protocol. INTERNET-DRAFT,draft-ylonen-ssh-protocol-00.txt, September 1995.

[305] T. Ylonen. SSH authentication protocol. INTERNET-DRAFT, draft-ietfuserauth-16.txt, September 2002.

[306] T. Ylonen. SSH connection protocol. INTERNET-DRAFT, draft-ietf-connect-16.txt, September 2002.

[307] T. Ylonen. SSH protocol architecture. INTERNET-DRAFT, draft-ietfarchitecture-13.txt, September 2002.

[308] T. Ylonen. SSH transport layer protocol. INTERNET-DRAFT, draft-ietftransport-15.txt, September 2002.

[309] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) cost(signature) + cost(encryption). In B. Kaliski Jr., editor, *Advances in Cryptology — Proceedings of CRYPTO'97, Lecture Notes in Computer Science 1294*, pages 165–179. Springer-Verlag, 1997.

[310] Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *Special Issue on Secure Communications, IEEE Journal on Selected Areas on Communications*, 11(5):715–724, June 1993.

[311] Y. Zheng and J. Seberry. Practical approaches to attaining security against adaptively chosen ciphertext attacks (extended abstract). In E.F. Brickell, editor, *Advances in Cryptology — Proceedings of CRYPTO'92, Lecture Notes in Computer Science 740*, pages 291–304. Springer-Verlag, 1993.

[312] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, Massachusetts, 1995. Second printing.

Предметный указатель

P

C

Т

Научно-популярное издание

Венбо Мао

Современная криптография: теория и практика

Литературный редактор *О.Ю. Белозовская*
Верстка *А.Н. Полинчик*
Художественный редактор *В.Г. Павлютин*
Корректоры *З.В. Александрова,*
Л.А. Гордиенко,
О.В. Мишутина,
Л.В. Чернокозинская

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1.

Подписано в печать 20.06.2005. Формат 70×100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 61,9. Уч.-изд. л. 41,1.
Тираж 2000 экз. Заказ № 2440.

Отпечатано с диапозитивов
в ФГУП "Печатный двор" им. А. М. Горького
Федерального агентства по печати
и массовым коммуникациям.
197110, Санкт-Петербург, Чкаловский пр., 15.