Східноєвропейський національний університет
Імені Лесі Українки

Л.Я. Глинчук

КРИПТОЛОГІЯ

Навчально-методичний посібник

Луцьк
Вежа-Друк
2014

УДК 004.415.3(072)
ББК 32.811.4я73-9+32.973.26-018.2я73-9
Г 54

Рекомендовано до друку вченою радою
Волинського національного університету імені Лесі Українки
(витяг з протоколу № 8 від 23.02.2012р.)

Рецензенти:
Волошина Т.В., кандидат фізико-математичних наук, доцент Волинського національного університету імені Лесі Українки;
Гопанчук С.О., старший викладач кафедри прикладної математики Волинського національного університету імені Лесі Українки.

Глинчук Л.Я.
Г 54 Криптологія: навч.-метод. посіб. / Людмила Ярославівна Глинчук – Луцьк: Вежа-Друк, 2014. – 164 с.

ISBN 978-617-7181-77-3

Матеріал посібника поділяється на дві частини – теоретичну та практичну. У теоретичній частині подано матеріал усіх лекцій курсу, у практичній – лабораторний практикум, що дає змогу під керівництвом викладача детально розглянути питання, які ввійшли в це видання.

Рекомендовано студентам ВНЗ для вивчення предмету “Криптологія”, а саме для опанування теоретичного матеріалу й лабораторного практикуму з цієї дисципліни. Матеріал може також бути корисним студентам, які вивчають захист інформації.

УДК 004.415.3(072)
ББК 32.811.4я73-9+32.973.26-018.2я73-9

ISBN 978-617-7181-77-3

ЗМІСТ

ВСТУП

Визначальною особливістю сучасності є потреба вирішення складних багатоаспектних завдань, що супроводжуються розширенням інформаційного обміну про найновітніші досягнення науки, сприянням впровадженню їх результатів у життя.

Інформаційна епоха привела до докорінних змін у способі виконання функціональних обов'язків для численних професій. Сьогодні нетехнічний фахівець середнього рівня може виконувати роботу, яку раніше здійснював високо-кваліфікований програміст. Службовець у своєму розпорядженні ще ніколи не мав стільки точної й оперативної інформації як тепер.

Водночас використання комп'ютерів і автоматизованих технологій спричиняє появу низки проблем для керівництва організацією. Доступ до величезної кількості найрізноманітніших даних надають комп'ютери, часто об'єднані в мережі. Тому, дбаючи про безпеку інформації, важливо усвідомлювати наявність ризику, зумовленого автоматизацією і наданням щораз більшого доступу до конфіденційних, персональних чи інших даних. Збільшується кількість комп'ютерних злочинів, що врешті решт може призвести до економічних втрат. Відтак очевидно, що інформація – це ресурс, який треба захищати. Те, що в 60 – х роках називалося комп'ютерною безпекою, а в 70 – х – безпекою даних, тепер точніше називають інформаційною безпекою. Інформаційна безпека зумовлена важливістю інформації в сучасному суспільстві, розумінням того, що інформація – це коштовний ресурс – щось більше, ніж окремі елементи даних.

Інформаційною безпекою називають заходи для захисту інформації від неавторизованого доступу, руйнування, модифікації, розкриття і затримок у доступі.

Інформаційна безпека гарантує досягнення таких цілей:

- ✓ конфіденційність критичної інформації;
- ✓ цілісність інформації і пов'язаних з нею процесів (створення, введення, обробки, виведення);

- ✓ доступність до інформації у разі потреби;
- ✓ облік усіх процесів, пов’язаних з інформацією.

Деякі технології із захисту системи і забезпечення обліку всіх подій вбудовані в самий комп’ютер, інші – в програми. Деякі виконуються людьми і є реалізацією вказівок, що містяться у відповідних керівних документах.

Використання технологій, що входять у поняття інформаційної безпеки – запорука успішної діяльності будь – якої організації і її управлінської ланки.

Інформаційні процеси і діяльність, зумовлена ними, регламентуються стандартизованими нормами. Для спрощення обміну інформацією, захисту комерційної таємниці й авторських прав, статистичного аналізу, планування та ефективного керування на всіх ієрархічних рівнях глобальної інформаційної системи країни необхідне законодавче регулювання інформаційної діяльності організацій.

Щодо захисту інформації, то хоча розглянуті нами засоби не завжди надійні, оскільки сьогодні швидкими темпами розвивається не тільки техніка (у нашому випадку – комп’ютерна), але й методи, що дають змогу цю інформацію здобувати, цими засобами не слід нехтувати. Нашу епоху часто називають інформаційною, і вона несе величезні можливості, пов’язані з економічним зростом, технологічними нововведеннями. Володіння електронними даними, що стають найбільшою цінністю інформаційної ери, покладає на своїх власників права й обов’язки контролю їхнього використання. Файли і повідомлення, збережені на дисках, і ті, що пересилаються по каналах зв’язку, мають іноді більшу цінність, ніж самі комп’ютери, диски. Відтак перспективи інформаційного століття можуть бути реалізовані тільки в тому випадку, якщо окремі особи, підприємства й інші підрозділи, які володіють інформацією, що дедалі частіше має конфіденційний характер чи є особливо важливою, зможуть належно захистити свою власність від будь – яких загроз, вибрати такий рівень захисту, що відповідатиме їхнім вимогам безпеки. [7]

Тема 1. ОСНОВНІ ПОНЯТТЯ КРИПТОЛОГІЇ

Несанкціонований доступ до інформації — доступ до інформації з порушенням посадових повноважень співробітника, доступ до закритої для публічного доступу інформації з боку осіб, котрі не мають дозволу на доступ до цієї інформації. Також іноді несанкціонованим доступом називають одержання доступу до інформації особою, що має право на доступ до цієї інформації в обсязі, що перевищує необхідний для виконання службових обов'язків.

1. Загроза інформації та можливості прихованої її передачі.

Основні ознаки інформації, що має бути захищеною:

- ✓ наявність певного кола законних власників, що мають право користуватись цією інформацією;
- ✓ наявність зловмисників, що прагнуть оволодіти цією інформацією з корисливою метою для себе та на шкоду законним власникам.

Мета, яку прагнуть досягти зловмисники, називається загрозою. Основні види загроз такі:

- ✓ загроза розголошення конфіденційної (секретної) інформації;
- ✓ загроза цілісності (автентичності, істинності) інформації;
- ✓ загроза достовірності адресних ознак інформації.

Конфіденційна інформація – це відомості, які знаходяться у володінні, користуванні або розпорядженні окремих фізичних чи юридичних осіб і поширюються за їх бажанням відповідно до передбачених ними умов. Інформація, що є державною власністю, теж вважається конфіденційною.

Цілісність інформації – це відсутність спотворених, замінених або знищених інформаційних елементів.

Достовірність адресних ознак інформації означає відсутність фальсифікації її відправника або отримувача. Це виключає можливість відмовитись від фактів передачі або отримання інформації, якщо вона дійсно передавалась або отримувалась, а також можливість підтвердити передачу або отримання інформації, якщо вона насправді не передавалась і не отримувалась.

Щоб захиститись від загроз, можна скористуватись однієї з трьох можливостей передачі прихованої інформації:

- ✓ використання абсолютно надійного, недоступного для інших каналу зв'язку між абонентами; ця можливість вважається практично нереальною, оскільки, на сучасному рівні науки і техніки неможливо створити такий канал між віддаленими абонентами для неодноразової передачі великих обсягів інформації;
- ✓ використання загальнодоступного каналу зв'язку, але при цьому приховується сам факт передачі інформації; розробкою відповідних методів та засобів займається стенографія;
- ✓ використання загальнодоступного каналу зв'язку, але при цьому до інформації, що передається, застосовується таке перетворення, що зрозуміти її може тільки адресат; розробкою відповідних методів та способів перетворення інформації займається криптології. [8]

2. Основні поняття стеганографії.

Перші застосування стеганографічних методів відомі з глибокої давнини. Наприклад, відомий такий спосіб приховування письмового повідомлення: голову раба голили, на шкірі голови писали повідомлення, чекали, поки волосся виросте, і лише тоді раба відправляли до адресата.

Відомі методи тайнопису між рядків звичайного тексту молоком або спеціальними реактивами з наступною хімічною обробкою.

Відомий метод "мікроточки", коли повідомлення за допомогою сучасної техніки записується на дуже маленький носій (мікроточку), який пересилається звичайним поштовим листом, наприклад, під маркою.

Сучасні комп'ютерні технології викликали появу комп'ютерної стеганографії. Методами комп'ютерної стеганографії здійснюється приховування повідомлень у звичайних файлах. Їх називають контейнерами.

Порожній файл-контейнер містить тільки звичайну інформацію і не містить прихованої конфіденційної інформації. Контейнер заповнюють у відповідності з ключем, який визначає порядок занесення повідомлення у

контейнер, а також його здобування із нього. Ключ може розміщуватись у контейнері разом із повідомленням або передаватись окремо.

Основні вимоги до стеганографічних методів:

- ✓ метод повинен забезпечувати збереження основних властивостей контейнера після внесення в нього конфіденційного повідомлення та ключа (розмір, дата і час створення, зовнішній вигляд даних);
- ✓ якщо про наявність прихованого повідомлення зловмиснику стало відомо, то здобування його повинно являти собою складну обчислювальну задачу.

Основна задача будь-якого стеганографічного методу – шляхом передачі ззовні звичайної інформації приховати сам факт передачі конфіденційної інформації. Засобами стеганографії можна розв’язати також додаткові спеціальні задачі:

- ✓ камуфлювання програмного забезпечення, коли його використання незареєстрованими користувачами є небажаним; наприклад, воно може бути закамуфльовано під стандартне програмне забезпечення (наприклад, текстовий редактор) або під файли мультимедіа (наприклад, під звукове супроводження комп’ютерних іграшок);
- ✓ захист авторських прав; при цьому у файл, який авторизується, вноситься спеціальна позначка, яка залишається невидимою для ока, але розпізнається спеціальним програмним забезпеченням.
- ✓ Методи комп’ютерної стеганографії розвиваються за двома основними напрямками:
- ✓ методи, що базуються на використанні властивостей комп’ютерних форматів;
- ✓ методи, що базуються на природній надмірності аудіо- та відеоінформації.

У свою чергу методи, що базуються на використанні властивостей комп’ютерних форматів є такі:

- ✓ комп’ютерні мультимедійні формати мають резервні поля. Звичайно вони заповнюються нульовими даними і не враховуються програмою;

- ✓ врахування особливостей форматування текстових файлів. Наприклад, у текст можуть бути внесені додаткові пропуски між словами, можуть бути змінені положення рядків, розміщення слів у реченнях, параметри абзаців, тощо;
- ✓ використання прихованих полів, не відображуваних на екрані. Наприклад, використання кольору шрифту, що співпадає з кольором фону;
- ✓ приховування даних у не використовуваних місцях дисків., наприклад, в нульовій доріжці;
- ✓ використання імітації, яка базується на побудові спеціального змістовного тексту, що приховує в собі повідомлення. При цьому, наприклад, для знаків повідомлення можуть вибиратися певні позиції. Окремий випадок цього методу – акростих, наприклад:

У червоному намисті
Коло річки в зелен-листі
Розпишалася калина,
А із нею й тополина.
Їде візник, тихо стане,
На калину ніжно гляне,
А в уяві щось постане.

Методи, що базуються на природній надмірності аудіо- та відеоінформації, використовують надмірність цифрових фотографій, цифрового звуку та цифрового відео. Відомо, що молодші розряди цифрових відліків містять мало корисної інформації. Їх заміна іншою інформацією практично не впливає на якість відтворення.

В літературі наведено такі приклади з використанням указаної надмірності.

Приклад 1. Одна секунда цифрованого звучання з частотою дискретизації 44100 Гц та з рівнем відліку 8 біт у стерео режимі дозволяє приховати близько 10 Кбайт за рахунок заміни молодших розрядів звукових даних розрядами повідомлення. При цьому зміна значень відліків складає менше 1%. Це практично не виявляється при прослуховуванні файлу більшістю людей.

Приклад 2. Графічні кольорові файли за схемою RGB кодують кожну точку трьома байтами. Зміна кожного із трьох молодших бітів приводить до

зміни яскравості точки менш як на 1%. Це дозволяє приховувати у стандартному графічному малюнку обсягом 800 Кбайт близько 100 Кбайт повідомлення. При звичайному перегляді зображення це не помітно. [8]

3. Предмет криптологія

Криптологія – наука, що складається з двох напрямків: криптографії та криптоаналізу.

Криптографія – наука, що займається розробкою шифрів. При цьому, шифром називається такий метод або спосіб перетворення повідомлень, який забезпечує захист інформації в них від зловмисників.

Криптоаналіз – це наука про методи та способи розкриття зашифрованих повідомлень, а також про тактику та стратегію їх застосування. Внаслідок застосування криптоаналізу можливо також фальсифікувати або саме повідомлення, або його адресні ознаки (наприклад, видати підроблене повідомлення за істинне).

Шифруванням називається процес застосування шифру до повідомлення, що має бути захищеним. Внаслідок шифрування відкритий текст перетворюється у шифроване повідомлення (криптограму).

Дешифрування – процес, обернений шифруванню, який теж здійснюється за відомими правилами шифру.

Сучасна термінологія не вважає синонімами терміни “кодування” та “шифрування”. Це пояснюється тим, що термін “кодування” охоплює ті методи та способи перетворення повідомлень, які розглядаються в рамках наукового напрямку теорії інформації та кодування. Вказані методи та способи перетворення повідомлень вирішують інші задачі (наприклад, пов’язані із стисненням даних або захистом даних від випадкових спотворень у каналах зв’язку).

Особу, яка вирішує задачу перехвату повідомлень та розкриття зашифрованих повідомлень за допомогою методів криптоаналізу, називають *противником*. Розкриття зашифрованих повідомлень називають *зламуванням шифру*. Окрему спробу зламування шифру називають *атакою на шифр.*

Крім зламування шифру противник може робити спроби отримати інформацію, що захищається, іншими способами. Найбільш відомий з них – агентурний. При цьому противник певним способом схиляє до співпраці одного із законних власників, і за допомогою цього агента отримує доступ до потрібної інформації. В таких випадках криптографія безсила. [8]

4. Основні принципи криптології

Принцип рівної міцності захисту. На шляху від одного законного власника до іншого інформація може захищатись різними способами в залежності від загроз, що виникають. Так утворюється ланцюг захисту інформації з ланками різного типу. Противник прагне знайти найслабкішу ланку, щоб з найменшими витратами добратися до інформації. Законні власники повинні враховувати це в своїй стратегії захисту інформації криптографічними методами: безглуздо робити якусь ланку дуже міцною, якщо є слабкіші ланки.

Принцип доцільності захисту. На сучасному рівні технічного розвитку засоби зв'язку, засоби перехоплення повідомлень, а також засоби захисту інформації вимагають занадто багато витрат. Тому існує проблема співвідношення вартості інформації, витрат на її захист та витрат на її здобування. Перш ніж захищати інформацію криптографічними методами, треба вирішити два питання:

1) Чи отримає противник внаслідок атаки інформацію, що буде більш цінною, ніж вартість самої атаки?
2) Чи є інформація, яку захищає її власник, більш цінною, ніж вартість захисту?

Відповідь на ці два питання визначає доцільність захисту та вибір підходящих засобів криптографічного захисту.

Принцип використання ключа. Розробка хорошого шифру – справа надзвичайно трудомістка. Тому бажано збільшити термін цього шифру і використовувати його для шифрування якнайбільшої кількості повідомлень. Але при цьому виникає небезпека, що противник вже зламав шифр і вільно

читає шифровані повідомлення. Саме тому в сучасних шифрах використовують ключі.

Ключем в криптографії називають змінюваний елемент шифру, який застосовується до шифрування конкретного повідомлення. При цьому вважають, що сам шифр (крім ключа) є відомим противнику і доступним для вивчення. Оригінальність подання повідомлення забезпечується тільки періодично змінюваним ключем. Знання ключа дозволяє швидко та просто відновити початковий текст. Без знання ключа дешифрування тексту має бути практично недосяжним.

Принцип стійкості шифру. Здатність шифру протидіяти різноманітним атакам на нього називається *стійкістю шифру.* З математичної точки зору проблема отримання строго доведених оцінок стійкості для будь-якого шифру ще не вирішена. Ця проблема відноситься до проблем нижніх оцінок обчислювальної складності задачі, ще нерозв'язаних математично. Тому стійкість конкретного шифру оцінюється шляхом різноманітних спроб його зламування, а отримані результати оцінюють в залежності від кваліфікації криптоаналітиків, що атакують цей шифр. Таку процедуру називають *перевіркою стійкості.*

Принцип Керкхоффа. Стійкість сучасного шифру має визначатися, в першу чергу, ключем. Зміст цього принципу полягає в тому, що захищеність інформації не повинна залежати від таких факторів, які важко змінити при появі загрози. При використанні ключів законним власникам інформації легше перешкоджати противнику, оскільки міняти їх можна досить часто. Але, тепер законним власникам виникає інша задача – як таємно обмінятись ключами перед тим, як обмінюватися шифрованими повідомленнями.

Принцип використання різноманітних шифрів. Не існує єдиного шифру, що підходить до всіх випадків. Вибір шифру залежить від особливостей інформації (може мати різний характер, тобто бути документальною, телефонною, телевізійною, комп'ютерною тощо), від цінності інформації, від обсягів інформації, від потрібної швидкості її передачі, від тривалості захисту (державні та військові таємниці зберігаються десятками років, біржеві –

декілька годин), від можливостей противника (можна протидіяти окремій особі, можна протидіяти потужній державній структурі), а також від можливостей власників із захисту своєї інформації. [8]

5. Класифікація шифрів

До основних характеристик сучасних методів шифрування можна віднести:

- ✓ довжину ключа;
- ✓ складність алгоритму перетворення даних;
- ✓ розмір даних, що обробляються;
- ✓ спосіб роботи з ключами і т. д.

Існують різні підходи до класифікації шифрів:

- ✓ за методом шифрування – шифри заміни та шифри перестановки;
- ✓ за технологією шифрування – блокові шифри та потокові шифри;
- ✓ за особливостями ключів – симетричні шифри та асиметричні.

У своїй роботі “Теорія зв’язку у відкритих системах” (1949) Клод Шеннон розглядав класифікацію шифрів за методом шифрування.

Шифр *заміни* здійснює перетворення, при якому літери або якісь інші фрагменти відкритого тексту замінюються відповідними фрагментами шифрованого тексту.

Шифр *перестановки* здійснює перетворення, при якому літери або якісь інші фрагменти переставляються місцями безпосередньо у відкритому тексті.

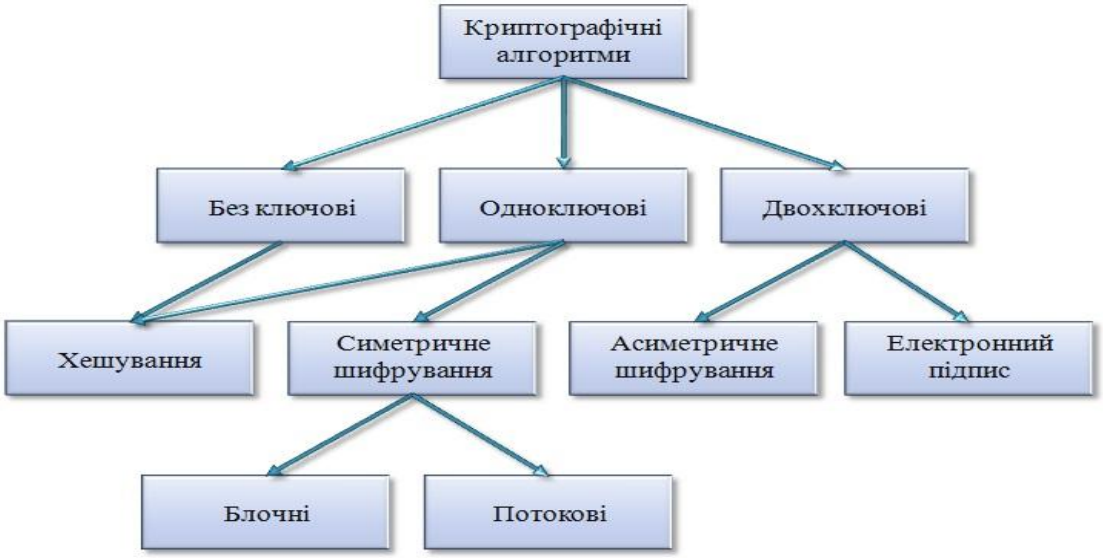
В *блокових* шифрах метод шифрування застосовують до блоку відкритого тексту, який має певні розміри (кількість знаків). В *потокових* шифрах метод шифрування застосовують до кожного знаку відкритого тексту окремо.

Якщо один і той же алгоритм, а також один і той же ключ використовується і для шифрування, і для дешифрування повідомлень, то такий метод шифрування називається *симетричним*. Зрозуміло, що цей єдиний ключ має бути секретним і відомим тільки відправнику та отримувачу повідомлення. Тому ці методи також називаються одно ключовими шифрами або *шифрами з секретним ключем*. Для симетричних шифрів є характерною нерозв’язувана

проблема, яка полягає в ускладненнях з передачею абоненту секретного ключа, а також в неможливості переконатись в аутентичності отриманого абонентом ключа.

Якщо алгоритми шифрування та дешифрування різні, і якщо використовуються два ключі – один для шифрування, а другий – для дешифрування повідомлення, то такий метод шифрування називається *асиметричним***.** Один з цих ключів є секретним, інший – відкритим. Тому асиметричні методи шифрування називаються двохключовими шифрами або шифрами з відкритим ключем.

В загальному всі криптографічні алгоритми поділяються так як на малюнку:



Ефективність використання симетричних шифрів визначається високою швидкодією при обробці великих обсягів інформації.

Ефективність використання асиметричних шифрів визначається відсутністю необхідності пересилання секретних ключів.

Вся сукупність засобів (шифри, ключі, програмно-апаратні засоби), яка забезпечує криптографічний захист інформації від загроз противника, називається *криптографічною системою.* Особливістю криптографічної системи є те, що два учасники секретного зв’язку повністю довіряють один одному.

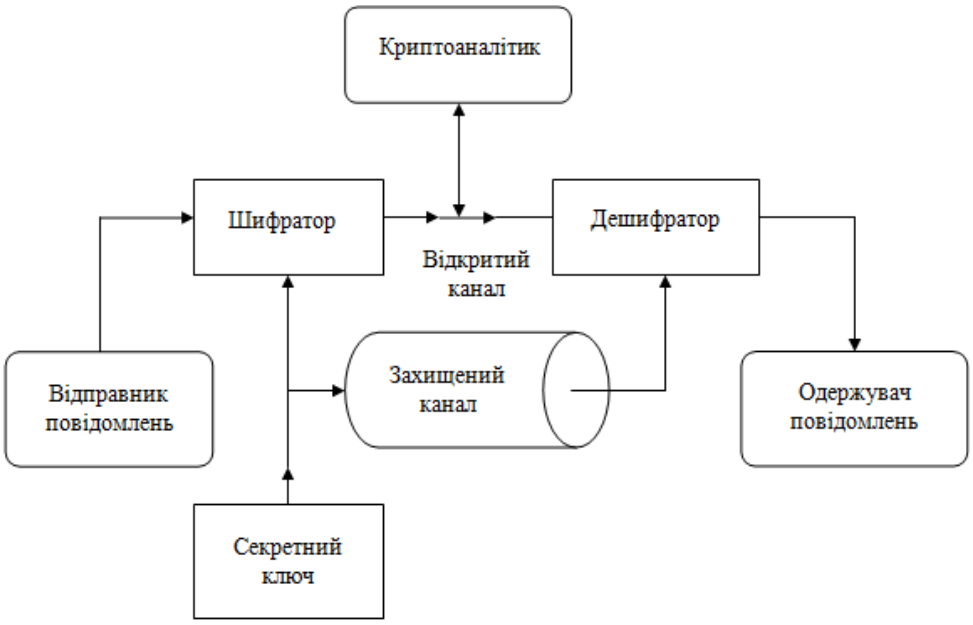


Рис. 1.1 Схема криптосистеми з таємним ключем (симетрична)

Криптосистема з рисунка може діяти у 2-х варіантах:

1) повна система, тобто в якій і відправник і одержувач можуть переходити з однієї ролі в іншу, при цьому система шифрування та дешифрування повинні бути і у відправника і у одержувача;
2) криптосистема односторонньої дії, у якій відправник може тільки відправляти, а одержувач тільки приймати з відповідними засобами шифрування і дешифрування у кожної з сторін.

6. Поняття абсолютно стійкого шифру

Одним з найважливіших результатів Клода Шеннона був висновок про існування та єдність абсолютно стійкого шифру. Абсолютно стійкий шифр має три ознаки:

- ✓ одноразовість використання;
- ✓ повна випадковість ключа;
- ✓ рівність довжин відкритого тексту та ключа або ключ більший від тексту.

У разі відсутності хоча б однієї з цих ознак шифр втрачає властивість абсолютної стійкості і з’являються принципові умови його зламування (хоча їх, можливо, буде важко реалізувати).

Теоретично противник з необмеженими ресурсами може зламати будь-який неабсолютно стійкий шифр.

Приклад. У 1999 році автор твору “Книга коду” Саймон Сінгх запропонував 25 тис. доларів тому, хто зможе зламати неабсолютно стійкий шифр, найскладніший за всю історію криптографії. Саймон Сінгх, доктор фізичних наук Кембріджського університету, разом з доктором Полом Лейландом, що працює в Кембріджі на компанію Microsoft, на протязі двох років в умовах повної секретності створили 10 криптограм зі зростаючою складністю. Але всього через рік, восени 2000 року, команда шведських комп’ютерщиків, очолюваних фахівцем із захисту інформації Фредриком Алмгреном, зламали всі 10 шифрів. Вони випередили конкурентів з усього світу і отримали винагороду. На це їм знадобилося 70 років комп’ютерного часу.

Указані ознаки роблять абсолютно стійкий шифр занадто дорогим. Тут виникають проблеми, пов’язані із значним збільшенням обсягу шифрованих даних (повідомлення плюс ключі), із забезпеченням всіх абонентів достатнім запасом секретних випадкових ключів, а також із виключенням їх повторного застосування (при кількості абонентів, більшій двох).

З цих причин абсолютно стійкі шифри застосовують тільки у випадках передачі невеликих обсягів особливо важливої державної інформації. Звичайні користувачі вимушені застосовувати неабсолютно стійкі шифри.

Прикладом реалізації абсолютно стійкого шифру є шифр Вернама. Цей шифр здійснює побітове додавання (по модулю 2) n-бітового відкритого тексту та n-бітового ключа: $y_i = x_i \oplus k_i$, $i=1, \ldots n$. Тут $x_1, x_2, \ldots, x_n$ – відкритий текст, $k_1, k_2, \ldots, k_n$ – ключ, $y_1, y_2, \ldots, y_n$ – криптограма. Різновид шифру Вернама для десяткових чисел в наш час знаходить застосування у системах військового зв’язку у вигляді, так званих, шифрувальних блокнотів.

Тема 2. КЛАСИЧНІ ШИФРИ ПЕРЕСТАНОВКИ

1. Загальна характеристика шифрів перестановки

Шифри перестановки відносяться до симетричних. Це означає, що один і той же алгоритм, а також один і той же ключ використовуються і для шифрування, і для дешифрування повідомлень. При цьому алгоритм та ключ шифрування з метою використання для дешифрування можуть бути шляхом певних перетворень представлені в іншій формі. Але це не означає, що вони різні.

Шифр перестановки полягає в тому, що окремі знаки або певні групи знаків за певними правилами переставляються місцями безпосередньо у відкритому тексті.

У найпростішому випадку шифр перестановки використовується як блоковий. Це означає, що в процесі шифрування знаки відкритого тексту переставляють в межах деяких блоків фіксованого розміру.

Стійкість шифру перестановки залежить від розміру блоку, а також від рівня складності порядку перестановки.

Найстародавніший приклад застосування шифру перестановки датується V-м століттям до н.е. Спартанці шифрували свої повідомлення за допомогою пристрою під назвою «скитала».

На циліндричну палицю (на скиталу) намотували спіраллю виток до витка стрічку пергамента. На ній писали вздовж осі палиці декілька рядків повідомлення. Після розмотування стрічки знаки на ній виявлялись розташованими хаотично.

	п	о	в	і	
	д	о	м	л	
	е	н	н	я	

Наприклад, відкритий текст «повідомлення», записаний в три рядки по одній літері на ширину стрічки, дає криптограму «пдеоонвмніля». Ключем такого шифру є діаметр палиці (або, що теж саме, кількість рядків тексту навколо палиці). [8]

2. Звичайна перестановка

Розглянемо блок відкритого тексту $T=(T_1,T_2,T_3,...,T_N)$ довжиною N і відповідний блок попарно різних індексів $\sigma=\sigma(K_1,K_2,K_3,...,K_N)$, де $1\leq K_i\leq N$ для всіх $1\leq i\leq N$. Тут блок індексів σ є ключем шифрування.

Звичайною перестановкою знаків даного тексту T називається його перевпорядкування таким чином, що знак з позиції $\sigma(i)=K_i$ у відкритому тексті переміщується у позицію i у криптограмі.

Виберемо ключем шифрування вектор індексів $\sigma=(2, 4, 3, 1)$ при N = 4. Він показує наступний взаємозв'язок індексів: $\sigma(1)=2$, $\sigma(2)=4$, $\sigma(3)=3$ і $\sigma(4)=1$ (знак відкритого тексту з порядковим номером 2 перетворюється у знак криптограми з порядковим номером 1, знак відкритого тексту з порядковим номером 4 перетворюється у знак криптограми з порядковим номером 2, знак відкритого тексту з порядковим номером 3 залишається знаком криптограми з тим же порядковим номером, знак відкритого тексту з порядковим номером 1 перетворюється у знак криптограми з порядковим номером 4. На підставі такого ключа текст «шифр» буде зашифровано у криптограму «ирфш». Інакше кажучи, значення i = 2, 4, 3 і 1 дають послідовність перенесення знаків із відкритого тексту у криптограму, тобто «и», «р», «ф» і «ш».

При дешифруванні можна скористуватись тим же ключем σ. При цьому треба врахувати, що індекси у складі ключа показують порядкові номери розташування послідовних знаків криптограми при перенесенні їх у відкритий текст. Наприклад, індекси у складі ключа $\sigma=(2, 4, 3, 1)$ при дешифруванні криптограми «ирфш» показують порядкові номери розташування її знаків при перенесенні їх у відкритий текст (перший знак "и" має у відкритому тексті порядковий номер 2, другий знак "р" має у відкритому тексті порядковий номер

4, третій знак "ф" має у відкритому тексті той же порядковий номер 3 і, нарешті, четвертий знак "ш" має у відкритому тексті порядковий номер 1).

Ще один приклад: відкритий текст "ШИФРУВАННЯ ПЕРЕСТАНОВКОЮ" при використанні ключа σ=(3, 8, 1, 5, 2, 7, 6, 4) для блоку довжиною N=8 перетворюється на криптограму "ФНШУИАВР_СНЕЯЕРПНЮТВАОКО".

Загальна можлива кількість перестановок заданого тексту T із N знаків рівна N!, значення якого швидко зростає зі збільшенням N. [8]

3. Звичайні рядково-стовпчикові табличні перестановки

Це перестановки, коли даний текст записується у прямокутну таблицю певного розміру по рядках, а зашифрований текст прочитується із таблиці по стовпчиках. Основний варіант шифрування полягає в тому, що спочатку отримують шифрувальну таблицю шляхом заповнення її послідовних рядків зліва направо. Після цього отримують криптограму шляхом прочитування її послідовних стовпчиків згори вниз. Наприклад, для повідомлення «перестановки» за допомогою шифрувальної таблиці розміром 3x4 (три рядки і чотири стовпчики) буде отримано криптограму «псоетвракени».

п	е	р	е
с	т	а	н
о	в	к	и

Зрозуміло, що ключем шифру є розмір таблиці (для наведеного прикладу 3x4). І для того, щоб отримати відкритий текст повідомлення, слід вписати криптограму в таблицю того ж самого розміру по стовпчиках, а прочитати по рядках.

Шифр звичайних рядково-стовпчикових табличних перестановок є блоковим. Розмір блоку співпадає із загальною кількістю клітинок таблиці (для наведеного прикладу розмір блоку становить 3x4=12 знаків).

В літературі вказується, що можливе також заповнення таблиці по стовпчиках, а прочитування по рядках. І таку перестановку називають

звичайною стовпчиково-рядковою. Але елементарний аналіз показує, що це еквівалентно звичайному транспонуванню розмірів таблиці. Отже, говорити про такий варіант немає ніякого сенсу, а застосовувати – недоцільно. Дійсно, шифрувальна таблиця розміром 4x3 стовпчиково-рядкової перестановки дає для наведеного прикладу ту ж саму криптограму «псоетвракени».

п	с	о
е	т	в
р	а	к
е	н	и

Інші варіанти заповнення приводять до значного збільшення обсягу ключової інформації (треба указувати порядок заповнення) і тому теж не доцільні для застосування.

Суттєво кращий ефект підвищення криптографічної стійкості дає метод, коли даний текст шифрують, послідовно застосовуючи дві таблиці різного розміру. Таку перестановку називають *подвійною звичайною рядково-стовпчиковою табличною перестановкою.* Наприклад, шифрування повідомлення «стовпчик» за допомогою таблиць 2x4 і 3x3 дає криптограму «счвпоктиб» (останній знак – довільний). [8]

4. Рядково-стовпчикові табличні перестановки із застосуванням ключа стовпчиків

Такі перестановки мають більший рівень стійкості.

При такому способі шифрування обумовлюється не тільки розмір таблиці, але й ключове слово. Ключове слово має вигляд вектора індексів перестановок стовпчиків.

Процес шифрування полягає в тому, що над верхнім рядком таблиці записують ключ, довжина якого співпадає з кількістю її стовпчиків. Після цього здійснюють вписування відкритого тексту у таблицю по рядках звичайним способом. Наприклад, повідомлення

4	1	3	2
п	е	р	е
с	т	а	н
о	в	к	и

Криптограма утворюється шляхом прочитування по стовпчиках, але тепер стовпчики беруться не підряд, а у порядку, визначеному ключем. Таким чином отримуємо криптограму «етвениракпсо».

Для підвищення криптографічної стійкості методу даний текст шифрують, послідовно застосовуючи дві таблиці різного розміру та два ключа стовпчиків. Таку перестановку називають подвійною рядково-стовпчиковою табличною перестановкою із застосуванням ключів стовпчиків. [8]

5. Рядково-стовпчикові табличні перестановки із застосуванням ключа рядків

Шифр рядково-стовпчикових табличних перестановок із застосуванням ключа рядків принципово нічим не відрізняється від аналогічного шифру, який використовує ключ стовпчиків.

При такому способі шифрування ключове слово має вигляд вектора індексів перестановок рядків.

Процес шифрування полягає в тому, що зліва від першого стовпчика таблиці записують ключ, довжина якого співпадає з кількістю її рядків. Після цього здійснюють вписування відкритого тексту у таблицю по рядках, але не підряд, а у відповідності з ключем. Наприклад, повідомлення «перестановки» з ключем «3, 1, 2» дає наступну шифрувальну таблицю розміром 3х4.

3	о	в	к	и
1	п	е	р	е
2	с	т	а	н

Криптограма утворюється шляхом послідовного прочитування її стовпчиків згори вниз. Таким чином отримуємо криптограму «опсветкраиен».

Для підвищення криптографічної стійкості методу даний текст шифрують, послідовно застосовуючи дві таблиці різного розміру та два ключа рядків. Таку перестановку називають *подвійною рядково-стовпчиковою* табличною перестановкою із застосуванням ключів рядків. [8]

6. Рядково-стовпчикові табличні перестановки з двома ключами

При такому способі шифрування обумовлюються розмір таблиці, а також два ключових слова у вигляді векторів індексів, перше з яких діє на рядки, а друге – на стовпчики.

Процес шифрування полягає в тому, що зліва від першого стовпчика таблиці записують перший ключ, довжина якого співпадає з кількістю її рядків, а над верхнім рядком таблиці записують другий ключ, довжина якого співпадає з кількістю її стовпчиків. Після цього здійснюють вписування відкритого тексту у таблицю по рядках, але не підряд, а у відповідності з першим ключем, який діє на рядки. Наприклад, повідомлення «перестановки» з ключами «3, 1, 2» і «4, 1, 3, 2» дає наступну шифрувальну таблицю розміром 3x4.

	4	1	3	2
3	о	в	к	и

1	п	е	р	е
2	с	т	а	н

Криптограма утворюється шляхом прочитування по стовпчиках, причому стовпчики беруться не підряд, а у порядку, визначеному другим ключем, який діє на стовпчики. Таким чином отримуємо криптограму «ветиенкраопс». [8]

7. Табличні перестановки з використанням трафарету

Квадратним трафаретом називають нанесену на планшет квадратну матрицю з прорізаними віконцями. Планшет, як маску, накладають на папір того ж розміру, і у віконця вписують знаки повідомлення по порядку слідування рядків зліва направо. Після першого заповнення планшет

повертають на 90° за годинниковою стрілкою і процедуру вписування повторюють. Таким способом вписування знаків повідомлення у віконця може бути здійснене чотири рази.

Квадратний трафарет цікавий тим, що після кожного повороту віконця опиняються над незаповненими клітинками паперу. З цією метою розташування віконець на трафареті підбирають спеціально. Кількість віконець не повинна перевищувати четвертої частини від загальної кількості клітинок трафарету. Щоб описати трафарет, застосовують позначення: нуль – віконце відсутнє, одиниця – віконце є. Таким чином, весь трафарет можна представити у вигляді сукупності двійкових чисел, кожне з яких відповідає його рядку. Ці числа доцільно представити у десятковій системі числення. Зрозуміло, що сукупність цих чисел являє собою ключ шифру.

Якщо кількість віконець трафарету виявиться більшою кількості знаків повідомлення, то порожні віконця заповнюються випадковими знаками.

Наприклад, повідомлення «перестановки» при застосуванні квадратного трафарета розміром 4х4, що має чотири віконця, може бути перетворено у криптограму «сапобтвекрваеинг». Цікаво, що на останньому четвертому повороті трафарету знаки повідомлення вже вичерпались, і тому порожні віконця були заповнені додатковими випадковими знаками «абвг». Таким чином, в наведеному прикладі після останнього, отримаємо

		п		**с**							**о**		**а**		
			е		**т**					**в**		**б**			
	р						**а**	**к**						**в**	
Е						**н**			**и**						**г**

0° 90° 180° 270°

с	**а**	**п**	**о**
б	**т**	**в**	**е**
к	**р**	**в**	**а**
е	**и**	**н**	**г**

результат

Ключем цього шифру є розмір квадратного трафарету 4х4 та послідовність двійкових чисел "0010, 0001, 0100, 1000" або, що те ж саме, десяткових чисел "2, 1, 4, 8", які цей квадратний трафарет описують.

Для дешифрування криптограми потрібний такий же самий квадратний трафарет.

В наведеному вище прикладі трафарет має чотири віконця, що дорівнює четвертій частині від загальної кількості клітинок 4х4=16.

Слід відмітити, що для підвищення криптографічної стійкості кількість віконець трафарету можна зробити меншою. Розглянемо відповідній приклад, в якому квадратний трафарет розміром 4х4 має три віконця і описується послідовністю "0, 5, 0, 8".

				е							**а**		**в**		
	п		**е**			**с**									
								н		**о**			**к**		
Р						**т**									**и**

0° 90° 180° 270°

е	**в**		**а**
	п	**с**	**е**
н	**к**	**о**	
р		**т**	**и**

результат

За рахунок зменшення кількості віконець при шифруванні повідомлення «перестановки» після внесення знаків на останньому четвертому повороті залишаються вільними ще чотири клітинки "ев_а_псенко_р_ти". їх слід заповнити додатковими випадковими знаками «абвг» вже без трафарету. Таким чином, отримуємо криптограму "еваабпсенковргти".

Шифр табличних перестановок з використанням квадратного трафарету є блоковим. Розмір блоку можна визначити, якщо кількість віконець трафарету помножити на чотири (для першого прикладу розмір блоку становить 4х4=16 знаків, для другого прикладу – 3х4=12 знаків).

Кількість можливих квадратних трафаретів різко зростає зі збільшенням їх розміру: один для 2x2, 256 для 4x4, більше 100 тисяч – для 6x6. Ця кількість суттєво збільшується за рахунок використання зменшеної кількості віконець.

Крім квадратних трафаретів існують також прямокутні трафарети. Для них застосовують повороти на 180°, а також перекладання на зворотну сторону. Розглянемо відповідній приклад, в якому прямокутний трафарет розміром 4x3 має два віконця. Цей трафарет описується послідовністю “4, 1, 0, 0”.

т											**т**	**т**		**т**
		р							**е**			**е**		**р**
			а					**р**				**а**		**р**
					ф	**а**						**а**		**ф**

0° 90° звор. сторона 180° результат

За рахунок недостатньої кількості віконець при шифруванні повідомлення «трафарет» після внесення знаків на останньому четвертому положенні трафарету залишаються вільними ще чотири клітинки "т_те_ра_ра_ф". їх слід заповнити додатковими випадковими знаками «абвг» вже без трафарету. Таким чином, отримуємо криптограму "татебраврагф". [8]

Тема 3. КЛАСИЧНІ ШИФРИ ЗАМІНИ

1. Загальна характеристика шифрів заміни

У загальному випадку шифр заміни здійснює перетворення, при якому літери або якісь інші фрагменти відкритого тексту замінюються відповідними фрагментами шифрованого тексту.

Найпростіший випадок шифрування заміною полягає в тому, що знаки відкритого тексту, записані в одному (первинному) алфавіті, заміняють знаками, що взято із іншого (вторинного) алфавіту, у відповідності з наперед установленим правилом. При цьому один і той же знак на протязі тексту замінюється однаково.

Якщо використовується один і той же вторинний алфавіт, то шифр заміни називають моноалфавітним. Якщо вторинних алфавітів декілька, то шифр називають багатоалфавітним.

Одним із перших моноалфавітних шифрів заміни вважається полібіанський квадрат. У II столітті до н.е. грецький письменник та історик Полібій винайшов з метою шифрування квадратну таблицю розміром 5x5, заповнену літерами грецького алфавіту у випадковому порядку.

При шифруванні чергову літеру відкритого тексту знаходили у цьому квадраті, а у криптограму записували літеру, розташовану рядком нижче в тому ж стовпчику. Якщо літера знаходилась у нижньому рядку таблиці, то для криптограми брали саму верхню літеру з того ж стовпчика.

Таким чином, основним для будь-якого шифру заміни є поняття алфавіту, який являє собою фіксовану послідовність всіх використовуваних знаків. При цьому фіксується як порядок слідування знаків, так і їх загальна кількість. Знаки алфавіту нумеруються по порядку, починаючи з нуля, тобто $0 \le j < m$. Таким чином, m являє собою загальну кількість знаків в алфавіті і називається його обсягом.

Знаки відкритого тексту теж доцільно нумерувати, починаючи з нуля, тобто $0 \le i < n$. Тут n являє собою загальну кількість знаків у повідомленні. [8]

2. Моноалфавітна звичайна заміна Цезаря (шифр Цезаря)

Як повідомляє історик Гай Светоній, римський імператор Гай Юлій Цезар користувався у своєму військовому та особистому листуванні шифром, суть якого полягала у заміні кожної літери повідомлення на одну із інших літер 26-значного латинського алфавіту.

Щоб зрозуміти зашифроване повідомлення Цезаря, треба було кожну літеру в ньому замінити третьою, що йде після неї в алфавіті. При досягненні кінця алфавіту виконувався циклічний перехід до його початку.

Такий метод шифрування можна відобразити шифрувальною таблицею, в якій указано замінюючи знаки для кожного знаку криптограми. Використання таблиці очевидне: при шифруванні для кожного знаку відкритого тексту шукаємо відповідний знак криптограми, при дешифруванні – навпаки.

Номер (код)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Знак відкритого тексту	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Знак кріптограми	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

Отже, шифр Цезаря пов’язаний із використанням первинного алфавіту (для відкритого тексту) і вторинного алфавіту (для криптограми), циклічно зміщеного відносно первинного на три знаки вперед.

Зокрема, Цезар використовував свій шифр у листуванні з Цицероном (близько 50 р. до н.е.). А його відоме послання VENI VIDI VICI – «Пришёл, увидел, победил» своєму другові Амінтію у зашифрованому вигляді мало такий вигляд: SBKF SFAF SFZF.

Загальний випадок шифру Цезаря для первинного алфавіту деякого обсягу m ($0 \le j < m$) полягає в тому, що вторинний алфавіт циклічно зміщується відносно первинного на K знаків вперед ($0 \le K < m$). Значення K є ключем цього шифру.

Наприклад, для алфавіту “АБВГДЕЖЗИК” обсягом $m=10$ шифрувальна таблиця для $K=6$ має наступний вигляд:

Номер (код)	0	1	2	3	4	5	6	7	8	9
Знак відкритого	А	Б	В	Г	Д	Е	Ж	З	И	К

тексту										
Знак криптограми	Е	Ж	З	И	К	А	Б	В	Г	Д

Таким чином, відкритому тексту “ЖАБА” відповідає криптограма “БЕЖЕ” і навпаки.

Недоліки моноалфавітної звичайної заміни Цезаря:

- ✓ вона не маскує частот появи різних знаків відкритого тексту і тому її легко зламати на підставі аналізу частот появи знаків у криптограмі;
- ✓ у вторинному алфавіті зберігається той же самий алфавітний порядок знаків, що і в первинному;
- ✓ мала кількість можливих ключів (рівна обсягу алфавіту). [8]

3. Шифр Цезаря з ключовим словом

Шифр Цезаря з ключовим словом теж є моноалфавітним. Він має ту особливість, що порядок знаків у вторинному алфавіті у порівнянні з первинним є дещо іншим завдяки використанню ключового слова. Крім того, практично необмеженою стає кількість ключів.

Для шифрування як ключ вибирається деяке число $0 \le K < m$, а також ключове слово. Всі знаки ключового слова мають бути різні.

Ключове слово записують під знаками алфавіту, починаючи зі знаку, числовий код якого співпадає з числом *K*. Знаки алфавіту, що залишились, записують за ключовим словом в алфавітному порядку.

Припустимо, що для алфавіту “АБВГДЕЖЗИК” обсягом $m=10$ ключем вибрано число K = 3 і ключове слово БЕДА. Отримуємо шифрувальну таблицю:

Номер (код)	0	1	2	3	4	5	6	7	8	9
Знак відкритого тексту	А	Б	В	Г	Д	Е	Ж	З	И	К
Ключове слово				Б	Е	Д	А			
Знак криптограми	З	И	К					В	Г	Ж

В цій таблиці первинний алфавіт (для відкритого тексту) вказано в другому рядку, вторинний (для криптограми) – в третьому та четвертому. У відповідності з цією таблицею відкритий текст ЖАЖДА шифрується як АЗАЕЗ.

Вимога про відмінність всіх знаків ключового слова не є обов'язковою. В цьому випадку просто записують ключове слово без повторення однакових знаків. Наприклад, ключове слово ЖАЖДА записують як ЖАД.

В шифрі Цезаря з ключовим словом недолік моноалфавітної звичайної заміни Цезаря, пов'язаний з відсутністю маскування частот появи різних знаків відкритого тексту, зберігається. [8]

4. Шифр Гронсфельда

Шифр Гронсфельда являє собою модифікацію моноалфавітної звичайної заміни Цезаря більш складним ключем, який являє собою послідовність чисел. Кожне з чисел ключа має бути меншим обсягу алфавіту *m*. Цей ключ записують під відкритим текстом. Якщо ключ коротший відкритого тексту, то його повторюють циклічно. Кріптограму отримують як і в шифрі Цезаря, але здійснюють відлік такої кількості літер, яка указана відповідним числом ключа.

Таким чином, шифр Гронсфельда є багатоалфавітним, оскільки використовує декілька вторинних алфавітів (по кількості різних чисел у складі ключа).

Наприклад, для алфавіту "АБВГДЕЖЗИК" обсягом *m=10* шифрувальна таблиця для чисел ключа в межах від *K=1* до *K=6* має наступний вигляд:

Номер (код)	0	1	2	3	4	5	6	7	8	9
Знак відкритого тексту	А	Б	В	Г	Д	Е	Ж	З	И	К
Знак криптограми *K=1*	К	А	Б	В	Г	Д	Е	Ж	З	И
Знак криптограми *K=2*	И	К	А	Б	В	Г	Д	Е	Ж	З
Знак криптограми *K=3*	З	И	К	А	Б	В	Г	Д	Е	Ж
Знак криптограми *K=4*	Ж	З	И	К	А	Б	В	Г	Д	Е
Знак криптограми *K=5*	Е	Ж	З	И	К	А	Б	В	Г	Д
Знак криптограми *K=6*	Д	Е	Ж	З	И	К	А	Б	В	Г

Для прикладу зашифруємо відкритий текст "ЗАДВИЖКА", використовуючи ключ (5, 1, 3). Розмістимо ключ під відкритим текстом:

З	А	Д	В	И	Ж	К	А
5	1	3	5	1	3	5	1

Щоб зашифрувати першу літеру відкритого тексту З, треба використати перше число ключа 5. Це означає, що відповідний знак криптограми треба взяти із того рядка шифрувальної таблиці, для якого К=5. Отримуємо першу літеру криптограми В. Щоб зашифрувати другу літеру відкритого тексту А, треба використати друге число ключа 1. Це означає, що відповідний знак криптограми треба взяти із того рядка шифрувальної таблиці, для якого К=1. Отримуємо другу літеру криптограми К. Остаточно отримуємо криптограму ВКБЗЗГДК. [8]

5. Гомофонічна заміна

Гомофонічна заміна одному знакові відкритого тексту ставить у відповідність декілька різних символів криптограми. Цей метод застосовується для спотворення статистичних властивостей криптограм. Наприклад, для алфавіту “АБВГДЕЖЗИК” обсягом m=10 шифрувальна таблиця для кожного знаку алфавіту може містити набори по три різні символи у вигляді довільних двозначних чисел, вибраних випадковим способом:

Номер (код)	0	1	2	3	4	5	6	7	8	9
Знак відкритого тексту	А	Б	В	Г	Д	Е	Ж	З	И	К
Символи криптограми	17 31 48	23 44 63	14 89 42	55 52 11	37 88 25	97 51 15	47 67 33	76 19 59	27 64 73	77 38 45

Для прикладу зашифруємо відкритий текст “ЖАДАЖА”. Для шифрування першої літери відкритого тексту Ж застосовуємо відповідний перший символ криптограми 47. Аналогічно для літер відкритого тексту А і Д, які теж зустрічаються перший раз, використовуються відповідні перші символи криптограми 17 і 37. Четверта і п’ята літери криптограми А і Ж зустрічаються вдруге, тому для них відповідними символами криптограми будуть 31 і 67.

Остання шоста літера криптограми А зустрічається втретє, тому вона буде шифруватись символом 48. Остаточно криптограма має вигляд 471737316748. Бачимо, що при використанні шифру гомофонічної заміни кожний знак відкритого тексту замінюється відповідними символами криптограми по черзі. Після того, як весь набір символів для даного знаку вичерпався, здійснюється його повторне використання з початку. [8]

6. Шифруюча таблиця Трисемуса

Багато хто з істориків вважають Іоганна Трисемуса, аббата з Німеччини, одним із засновників сучасної криптології. У 1508 році Трисемус написав свій твір "Поліграфія" – перший друкований твір з криптології. В ньому він вперше описав свій шифр, відомий під назвою шифруюча таблиця Трисемуса. В цьому шифрі використовується прямокутна таблиця певного розміру (по можливості, якнайближча до квадратної) для запису знаків алфавіту і ключове слово, записане без повторення однакових знаків. Для заповнення шифрувальної таблиці використовується підхід, запозичений із шифру Цезаря з ключовим словом. Спочатку в таблицю по рядкам вписується ключове слово. Далі таблиця доповнюється рештою знаків в алфавітному порядку. Наприклад, для алфавіту "АБВГДЕЖЗИКЛМ" обсягом $m = 12$ шифрувальна таблиця може мати розмір 3х4. Виберемо ключове слово ЖАД. За таких умов шифрувальна таблиця має такий вигляд:

Ж	А	Д	Б
В	Г	Е	З
И	К	Л	М

Спосіб шифрування запозичений із полібіанського квадрата. При шифруванні чергову літеру відкритого тексту знаходили у шифрувальній таблиці, а у криптограму записували літеру, розташовану рядком нижче в тому ж стовпчику. Якщо літера знаходилась у нижньому рядку таблиці, то для криптограми брали саму верхню літеру з того ж стовпчика. Наприклад, ГЛАЗ шифрується як КДГМ. [8]

7. Біграмний шифр Плейфера

Даний шифр називається біграмним тому, що шифруються одночасно не один, а два сусідні знаки відкритого тексту. Шифрувальна таблиця Плейфера являє собою прямокутну матрицю (по можливості, якнайближчу до квадратної). Її розміри мають бути достатніми для розміщення всіх знаків алфавіту відкритого тексту. Матриця заповнюється знаками алфавіту випадковим способом. Наприклад, для алфавіту "АБВГДЕЖЗИКЛМ" обсягом $m=12$ шифрувальна таблиця може мати наступний вигляд:

Ж	А	Д	Б
В	Г	Е	З
И	К	Л	М

Процес шифрування складається з таких кроків.

1. Відкритий текст розбивається на пари знаків (біграми). В тексті повинна бути парна кількість знаків і не повинно бути біграм з однаковими знаками. Якщо ці умови не виконуються, то текст модифікують з утворенням незначних орфографічних помилок. *Наприклад*, можна замінити один із цих знаків іншим, вставити між ними дефіс або виключити один із них взагалі. Після цього кожна дана біграма відкритого тексту за допомогою шифрувальної таблиці перетворюється в результуючу біграму криптограми.

2. Якщо обидва знаки даної біграми відкритого тексту знаходяться в різних рядках та стовпчиках матриці, то вони вважаються протилежними кінцями діагоналі відповідного прямокутника. Результуючу біграму криптограми знаходять на кінцях другої діагоналі цього ж прямокутника. Знак, який знаходиться на лівому кінці першої діагоналі, замінюється знаком який знаходиться на лівому кінці другої діагоналі. Знак, який знаходиться на правому кінці першої діагоналі, замінюється знаком, який знаходиться на правому кінці другої діагоналі. Наприклад, даним біграмам відкритого тексту АЛ, МА, ДИ і КЕ відповідають результуючі криптограми КД, БК, ЛЖ і ГЛ.

3. Якщо знаки даної біграми знаходяться в одному й тому ж рядку, то кожний із знаків замінюється тим, що стоїть справа від нього (за останнім

знаком у рядку йде перший). Наприклад, даним біграмам ЖБ і ЛИ відповідають результуючі біграми АЖ і МК.

4. Якщо знаки даної біграми знаходяться в одному й тому ж стовпчику, то кожний із знаків замінюється тим, що стоїть нижче його (за останнім нижнім знаком йде самий верхній). Наприклад, даним біграмам ЖВ, ЛД і МЗ відповідають результуючі біграми ВИ, ДЕ і БМ. Наприклад, відкритий текст БАЗА перетворюється на криптограму ЖДБГ. [8]

8. Біграмний двотабличний шифр

Цей шифр винайдений у 1854 році англійцем Чарльзом Уитстоном. Такий метод використовує дві прямокутні таблиці однакового розміру *(по можливості, якнайближчі до квадрату)*, в кожній з яких випадковим способом розміщено один і той же алфавіт. Відкритий текст розбивають на пари знаків – біграми. Перший знак біграми відкритого тексту фіксується у першій таблиці, другий знак біграми – у другій. Між зафіксованими знаками вибудовується уявний прямокутник. Одна діагональ цього прямокутника з'єднує знаки біграми відкритого тексту, друга діагональ дає результуючу біграму до криптограми. Перший знак результуючої біграми теж прочитується із першої таблиці, другий знак біграми – із другої таблиці. Якщо знаки відкритого тексту потрапили в один і той же рядок, то і біграма криптограми береться з того ж рядка. Перший знак біграми криптограми береться із першої таблиці у стовпчику, номер якого такий же, як і номер стовпчика другого знаку біграми відкритого тексту. Другий знак біграми криптограми береться із другої таблиці у стовпчику, номер якого такий же, як і номер стовпчика першого знаку біграми відкритого тексту. При використанні алфавіту, який складається із десяти цифр, крапки та пропуску, шифрувальна таблиця може бути такою:

Таблиця 1			Таблиця 2		
2	7	.	0	2	7
6	0	3	_	.	4
1	4	9	6	8	5
_	5	8	3	1	9

У відповідності з цією таблицею біграму «78» буде зашифровано як «42», біграму «42» – як «78», біграму «59» – як «81» і т.д. А для повідомлення «2.718_3.14» одержимо криптограму «6252330465». Перевагою біграмного двотабличного шифру у порівнянні з біграмним шифром Плейфейра є можливість використання біграм з однаковими знаками. [8]

9. Координатні заміни

В координатних замінах знаки алфавіту використовуються для позначень координат шифрувальної таблиці. Якщо алфавіт має N знаків і мова йде про двохкоординатну заміну, то шифрувальна таблиця має форму квадрата розміром $N*N$. В окремих комірках таблиці випадковим способом розміщують всі N можливих пар знаків, а вертикалі та горизонаталі таблиці позначають знаками, розташованими в алфавітному порядку. Відкритий текст розбивають на пари знаків – біграми. Перший знак біграми відкритого тексту використовується як індекс рядка, другий знак біграми – як індекс стовпчика. На їх перетині знаходиться результуюча біграма до криптограми. *Наприклад,* при використанні алфавіту «0, 1, 2» шифрувальна таблиця може мати такий вигляд:

	0	1	2
0	10	21	01
1	00	20	11
2	12	02	22

Для такої шифрувальної таблиці повідомлення «1020110221» перетворюється на криптограму «0012200102».

Суть ускладненого координатного методу, полягає в тому, що таблиці з числами ставиться у відповідність таблиця з алфавітом, тоді кожній літері у відповідність ставляють числа, приклад:

	0	1	2		0	1	2
0	10	21	01	0	А	Б	В
1	00	20	11	1	Г	Д	Е
2	12	02	22	2	Ж	Й	К

Тоді, наприклад, слово ДАВАЙ перетвориться на «2010011002». [8]

Тема 4. АРИФМЕТИЧНІ ОСНОВИ КРИПТОГРАФІЇ (ч. 1)

1. Алгоритм ділення з остачею та основна властивість конгруентності

Множину всіх натуральних чисел 1, 2, 3, … будемо позначати через N, а через Z – множину цілих чисел 0, ±1, ±2, …

Нехай a, b – елементи Z, $a\neq 0$.

Означення. Кажуть, що $b|a$ (b ділить a), якщо існує таке ціле c, що $a=bc$.

Властивості:

1. $b|a$, $b|c \Rightarrow b|a\pm c$.
2. $b|a \Rightarrow b|ac$, $c \in Z$.
3. $a|b$, $b|a \Leftrightarrow a=\pm b$.

Теорема про ділення з остачею. Нехай a – ціле, b – натуральне. Тоді існують такі однозначно визначені q, $r \in Z$, $0\leq r<b$, що $a=bq+r$.

Іншими словами, будь яке ціле a подається через натуральне b єдиним способом, тобто $a=bq+r$, де, ціле $\boldsymbol{q}$ – називається ***неповною часткою***, $\boldsymbol{b}$ називається ***модулем***, $\boldsymbol{0 \leq r < b}$ – називається ***остачею від ділення або лишком*** числа a за модулем b.

Теорема залишається справедливою для будь-якого цілого $b\neq 0$ при умові, що обмеження $r< b$ замінюється на $r < |b|$.

Операція знаходження остачі r від ділення цілого числа a на натуральне b позначається як $r=a \bmod b$.

Операція знаходження неповної частки q від ділення цілого числа a на натуральне число b позначається як $q=a \operatorname{div} b$.

Неповну частку q та остачу r можна знайти за допомогою алгоритму:

1) Нехай $q=0$.
2) Обчислюємо $r=a-bq$.
3) Якщо умова $0 \leq r < b$ виявилась істинною, то виконання алгоритму закінчено; інакше – виконання алгоритму продовжимо.
4) Якщо $a>0$, то $q:=q+1$, якщо $a<0$ то $q:=q-1$

5) Повернемось до пункту 2.

Приклад 1.

Знайдемо q і r для чисел $a=43$ та $b=11$. Виконаємо звичайне ділення: $a/b = 43/11 \approx 3.9091$. Заокруглимо отриманий дріб до цілого в меншу сторону: отримаємо $q=3$. Тепер за формулою $r = a-bq = 43 – 11\cdot3 = 10$.

Приклад 2.

Знайдемо q і r для чисел $a=44$ та $b=11$. Виконаємо звичайне ділення: $a/b = 44/11 = 4$. Заокруглимо отриманий дріб до цілого в меншу сторону: отримаємо $q=4$. Тепер за формулою $r = a-bq = 44 – 11\cdot4 = 0$.

Приклад 3.

Знайдемо q і r для чисел $a=-64$ та $b=14$. Виконаємо звичайне ділення: $a/b =-64/14 \approx -4.5714$. Заокруглимо отриманий дріб до цілого в меншу сторону: отримаємо $q=-5$. Тепер за формулою $r = a-bq = -64 – 14\cdot(-5) =6$.

При остачі $r = 0$, кажуть, що:

- ✓ a ділиться на b *(націло або без остачі)*,
- ✓ b ділить a,
- ✓ b є дільник числа a,
- ✓ a є кратним числу b.

Означення. Якщо двом цілим a і b відповідає одна й та ж остача r, то вони називаються ***конгруентними за модулем*** c, тобто $a \equiv b \pmod c$. ***(або порівнюваними за модулем*** c*).*

Або по іншому

Означення. Цілі числа a і b називаються ***конгруентними за модулем*** $c \in N$, якщо вони дають одну і ту ж остачу при діленні на c. Пишуть $a \equiv b \pmod c$.

Основна властивість. $a \equiv b \pmod c$ тоді і тільки тоді, коли $(a-b)$ ділиться на c.

Це випливає з теореми про ділення з остачею для a і b:

$a = q_1\cdot c+r$, $b = q_2\cdot c+r$, де q_1, q_2 – цілі, $0 \le r < |c|$. Тоді $(a-b) = (q_1 - q_2)\cdot c$ ділиться на c.

Приклад 4.

Відомо, що $r = 17 \bmod 5 = 2$ ($a = 17, b = 5, a/b = 17/5 = 3.4, q=3$, тоді $r = a-bq = 17-5\cdot4=17-15=2$) і $r = 32 \bmod 5 = 2$. Отже, можна записати, що $32 \equiv 17(\text{mod } 5)$.

Приклад 5.

Відомо, що $r = 45 \bmod 7 = 3$ і $r = 24 \bmod 7 = 3$. Отже, можна записати, що $45 \equiv 24(\text{mod } 7)$.

Основна властивість конгруентності. Із конгруентності $a \equiv c(\text{mod } b)$ випливає рівність $a=c+bq$, де q – ціле. Це означає, що a і c відрізняються між собою на цілу кількість модулів b, тобто $a - c$ ділиться на b.

Приклад 6.

Оскільки $32 = 17 + 5\times3$, або $17=32+ 5\times (-3)$, або $r = (32 -17)\bmod 5 = 0$, або $r = (17 - 32)\bmod 5 = 0$. Відомо, що $32 \equiv 17(\text{mod } 5)$.

Приклад 7.

Оскільки $45 = 24 +7\times3$, або $24 = 45 +7 \times(-3)$, або $r = (45 - 24)\bmod 7 = 0$, або $r = (24 - 45)\bmod 7 = 0$. Відомо, що $45 \equiv 24(\text{mod } 7)$. [8]

2. Властивості конгруентностей по відношенню до арифметичних операцій

Припустимо, що $a_1 \equiv c_1 (\text{mod } b)$ і $a_2 \equiv c_2 (\text{mod } b)$. Тоді справедливі наступні конгруентності:

1) $(a_1 + a_2) \equiv (c_1 + c_2) (\text{mod } b)$ – суми конгруентних чисел конгруентні за тим же модулем;

2) $(a_1 \cdot a_2) \equiv (c_1 \cdot c_2) (\text{mod } b)$ – добутки конгруентних чисел конгруентні за тим же модулем;

3) $a_1^k = c_1^k (\text{mod } b)$ – степені конгруентних чисел конгруентні за тим же модулем.

Наведемо приклади виконання вказаних властивостей для наступних даних: відомо, що $27 \equiv 12 (\text{mod } 5)$ – їм відповідає остача 2; відомо також, що $34 \equiv 14 (\text{mod } 5)$ – їм відповідає остача 4.

Приклад 7.

Тоді $(27 + 34) \equiv (12 + 14)(mod\ 5) \rightarrow 61 \equiv 26(mod\ 5)$ – їм відповідає остача 1. Також і $(27 + 14) \equiv (12 + 34)(mod\ 5) \rightarrow 41 \equiv 46(mod\ 5)$ – їм теж відповідає остача 1.

Приклад 8.

Тоді $(27 \times 34) \equiv (12 \times 14)(mod\ 5) \rightarrow 918 \equiv 168(mod\ 5)$ – їм відповідає остача 3. Також і $(27 \times 14) \equiv (12 \times 34)(mod\ 5) \rightarrow 378 \equiv 408(mod\ 5)$ – їм теж відповідає остача 3.

Приклад 9.

Тоді $27^2 \equiv 12^2\ (mod\ 5) \rightarrow 729 \equiv 144(mod\ 5)$ – їм відповідає остача 4. Також і $27^3 \equiv 12^3\ (mod\ 5) \rightarrow 19683 \equiv 1728(mod\ 5)$ – їм відповідає остача 3.

На наведених властивостях конгруентностей по відношенню до арифметичних операцій базуються правила спрощеного обчислення ***остачі.***

Правило 1. Остачу для суми знайти простіше, якщо попередньо знайти остачі для кожного з доданків, тобто $(a+c)\ mod\ b = ((a\ mod\ b)+(c\ mod\ b))\ mod\ b$.

Приклад 10.

Пряме знаходження остачі для суми $(14+8)\ mod\ 5 = 22\ mod\ 5 = 2$. За правилом: $(14+8)\ mod\ 5 = (14\ mod\ 5 + 8\ mod\ 5)\ mod\ 5 = (4+3)\ mod\ 5 = 7\ mod\ 5 = 2$. Висновок: завдяки правилу проміжні результати не перевищують подвоєного модуля.

Правило 2. Остачу для добутку знайти простіше, якщо попередньо знайти остачі для кожного із співмножників, тобто $(a \cdot c)\ mod\ b = ((a\ mod\ b) \cdot (c\ mod\ b))\ mod\ b$.

Приклад 11.

Пряме знаходження остачі для добутку $(7 \cdot 8)\ mod\ 5 = 56\ mod\ 5 = 1$. За правилом: $(7 \cdot 8)\ mod\ 5 = ((7\ mod\ 5) \cdot (8\ mod\ 5))\ mod\ 5 = (2 \cdot 3)\ mod\ 5 = 6\ mod\ 5 = 1$. Висновок: завдяки правилу проміжні результати не перевищують квадрата модуля.

Правило 3. Остачу для степеня знайти простіше, якщо попередньо знайти остачу для основи, тобто $a^k\ mod\ b = (a\ mod\ b)^k\ mod\ b$.

Приклад 12.

Пряме знаходження остачі для степеня $5^4 \bmod 3 = 625 \bmod 3 = 1$. За правилом: $5^4 \bmod 3 = (5 \bmod 3)^4 \bmod 3 = 2^4 \bmod 3 = 16 \bmod 3 = 1$.

Висновок: завдяки правилу проміжний результат не перевищує k-го степеня модуля. При цьому використовуючи послідовне множення замість піднесення до степеня можна досягнути неперевищення квадрата модуля.

Таким чином, завдяки наведеним правилам виникає можливість обмеження величин проміжних результатів виконуваних операцій, що важливо з точки зору алгоритмізації та програмування. [8]

3. Найбільший спільний дільник (НСД)

Означення. Будь-яке ціле число, на яке діляться a і b називається їх *спільним дільником.*

Означення. Найбільший із спільних дільників натуральних чисел a і b називається *найбільшим спільним дільником* і позначається *НСД(a,b).*

Теорема. Для довільних двох цілих чисел, серед яких принаймні одне ненульове, існує *НСД* і він єдиний.

Лема 1. $НСД(a,0) = a$ для $a \neq 0$.

Лема 2. Для довільних цілих a і $b \neq 0$ із $a = bq + r$, (q – частка, r - остача) $\Rightarrow НСД(a,b) = НСД(b,r)$.

Ці дві леми лежать в основі алгоритму Евкліда знаходження НСД. Виконаємо наступне ділення з остачею:

$a = bq_1 + r_1,\ 0 \le r_1 < b,$

$b = r_1 q_2 + r_2,\ 0 \le r_2 < r_1,$

$r_1 = r_2 q_3 + r_3,\ 0 \le r_3 < r_1,$

…

$r_{n-2} = r_{n-1} q_n + r_n,\ 0 \le r_n < r_{n-1},$

$r_{n-1} = r_n q_{n+1}.$

$НСД(a,b) = НСД(b,r_1) = НСД(r_1,r_2) = \ldots = НСД(r_{n-1},r_n) = НСД(r_n,0) = r_n.$

Остання рівність виконується за лемою (1), а всі попередні за лемою (2). Таким чином, НСД двох чисел дорівнює останній відмінній від нуля остачі в алгоритмі Евкліда.

Алгоритм Евкліда скінченний, бо $|b| > r_1 > r_2 > \ldots > r_k > r_{k+1} > \ldots$ а числа r_i – невід'ємні цілі, тому на якомусь кроці обов'язково отримаємо $r_{n-1} = 0$.

Отже, остання не рівна нулю остача одночасно є найбільшим спільним дільником.

Приклад 13.

Застосуємо алгоритм Евкліда для знаходження НСД (175, 77). 175 = 77·2 + 21, 77 = 21·3 + 14, 21 = 14·1 + 7, 14 = 7·2. Остання додатна остача = 7, тому НСД (175, 77)=7.

Словесний опис алгоритму Евкліда для пошуку НСД заданих чисел a і b.

1. якщо *a>b,* то *a* замінити на *a mod b* інакше *b* замінити на *b mod a;*
2. якщо *a=0* або *b=0,* то обчислити *НСД=a+b* і закінчити алгоритм;
3. повернутись до пункту 1.

Приклад 14.

Таблиця виконання алгоритму при *a=140, b=12.*

a>b	A	B	a=0 або b=0
	140	12	
Так	8		Ні
Ні		4	Ні
Так	0		Так

НСД=0+4=4, тобто НСД(140,12)=4

Поняття найбільшого спільного дільника можна ввести і для декількох чисел $a_1, a_2, \ldots, a_n$. Його позначають НСД($a_1, a_2, \ldots, a_n$).

НСД декількох чисел можна обчислювати послідовно. Наприклад, НСД(a_1, a_2, a_3)=НСД(НСД(a_1, a_2), a_3). [8]

Тема 5. ПРОСТІ ЧИСЛА. ТЕСТУВАННЯ ЧИСЛА НА ПРОСТОТУ

1. Означення простого числа

Криптографія використовує дуже великі прості числа. На жаль, використання стандартних засобів мов програмування надає можливість працювати лише з такими цілими числами, значення яких не виходять за межі довгого цілого типу LongInt, для якого найбільше допустиме значення становить 2147483647. При цьому найбільше ціле число рівне 2147302891*(просте),* що далеко не задовольняє потреби криптографії. Саме тому дуже важливо розробляти швидкодіючі алгоритми обробки простих чисел.

Означення. Натуральне число $p>1$ називається *простим*, якщо воно не має інших натуральних дільників, крім 1 і p. Простим числом буде найменший, відмінний від 1 дільник цілого a, $a>1$.

Теорія чисел визначає прості числа наступним чином:

- ✓ всі натуральні числа, крім 1, мають, щонайменше, двох дільників – одиницю та самого себе;
- ✓ ті з них, що не мають ніяких інших дільників, називаються простими; наприклад, декілька перших простих чисел 2, 3, 5, 7, 11, 13, 17, 19;
- ✓ ті числа, які мають ще й інших дільників, називаються складеними; наприклад, 12 – складене число (його дільники 1, 2, 3, 4, 6, 12);
- ✓ число 1 розглядають окремо, не відносячи ні до простих, ні до складених, оскільки одиниця не має всіх властивостей, справедливих для всіх інших простих чисел.

Означення. Натуральні числа a і b, для яких $НСД(a,b)=1$, називаються *взаємно простими*.

Закономірності в розподілі простих чисел на даний час ще не знайдено.

Теорема. Існує нескінченно багато простих чисел.

Теорема. Будь-яке ціле число більше від одиниці розкладається на добуток простих множників. Цей розклад єдиний з точністю до порядку слідування множників. [8]

2. Утворення послідовності простих чисел

Будемо розглядати утворення простих чисел в межах від 2 до деякого заданого *X*.

Спосіб 1. Пов'язаний з послідовним перебором натуральних чисел від 2 до *X*. Для кожного з них реалізується тестування на простоту, і у разі позитивного результату чергове число переводиться у послідовність простих. У зв'язку з необхідністю тестування такий спосіб вважається не ефективним за витратами часу.

Спосіб 2. Спосіб відомий під назвою *решето Ератосфена*. Відповідний класичний алгоритм такий:

1) виписуємо всі натуральні числа від 2 до *X*;
2) обводимо перше число 2, а всі інші, кратні 2, викреслюємо;
3) відшукуємо найменше необведене і не викреслене, обводимо його, фіксуємо його значення L, а після цього викреслюємо кожне L-те число, тобто всі, кратні йому;
4) п.3 повторюємо, поки не залишиться необведених і незакреслених чисел.

Всі обведені числа є простими. Спосіб не вимагає виконання арифметичних операцій, що і визначає його швидкодію. У зв'язку з необхідністю попереднього виписування всіх натуральних чисел, такий спосіб вважається теж не ефективним за витратами пам'яті.

Вдвічі зменшити витрати пам'яті дозволяє *модифіковане решето Ератосфена*. Воно відрізняється від класичного лише тим, що виписується 2 і всі непарні натуральні числа, які не перевищують *X*. При цьому перше просте число 2 обводиться відразу.

Кількість отриманих простих чисел від 2 до *X* є значенням, так званої, функції $\pi(x)$, рівне кількості всіх простих чисел, що не перевищують *X*. [8]

3. Тестування числа на простоту, ймовірнісні тести

Тестуванням деякого числа *a* на простоту називається перевірка, є дане число *a* простим чи ні.

Для цього достатньо перевірити всі можливі прості дільники цього числа, за винятком самого числа *a*. Така можливість пов'язана з тим фактом у теорії чисел, що будь-яке натуральне число *a* можна єдиним способом представити у вигляді добутку степенів простих чисел, тобто $a = p_1^{к1} * p_2^{к2} *_{...}* p_s^{кs}$ – *канонічний розклад числа на прості множники*, де $k_i \geq 1$ – кількість простих чисел, рівних *p*, присутніх у поданні числа *a*.

Якщо остача від ділення числа *a* на деякий можливий простий його дільник рівна нулю, то таке число *a* не є простим.

Всі можливі прості дільники числа *a*, які слід перевірити знаходяться в межах від 2 до $[\sqrt{a}]$ (ціла частина кореня з *a*). Для великого числа *a* така перевірка може тривати занадто довго. Тому важливе значення мають різні методи скорочення терміну перевірки.

Проблема визначення належності заданого натурального числа до класу простих чи складених чисел є дуже цікавою не тільки в математиці, а й у комп'ютерних науках. Відрізнити просте число від складеного, а також розкласти останнє на прості множники є однією з найважливіших задач арифметики. Пошук великих простих чисел необхідний, наприклад, для забезпечення надійності систем кодування інформації з відкритим ключем.

Для перевірки чисел на простоту користуються ймовірнісними тестами: Ферма, Соловай-Штрасена, Мілера-Рабіна.

Тест на простоту називається *ймовірнісним,* якщо в результаті його застосування не можна дати чіткої відповіді на запитання «чи є задане число простим, чи ні?», але можна виявити часткову інформацію стосовно простоти.

Наведені нижче тести дають таку інформацію про непарне ціле число *n:*

- ✓ якщо тест визначає, що *n* є складеним, то це дійсно так;
- ✓ якщо тест визначає, що *n* є простим, то зі ймовірністю, близькою до 1, можна вважати, що число є простим. [6]

4. Ймовірнісний тест на простоту Ферма

Тест базується на теоремі Ферма, яка стверджує, що якщо n − просте, то для довільного a, $1 \le a \le n - 1$, має місце рівність $a^{n-1} \equiv 1 (mod\ n)$. Якщо для заданого n знайдеться хоча б одне таке a, що $a^{n-1} \neq 1 (\text{mod n})$, то n не є простим.

Означення. Нехай n − непарне складене число. Число a, $1 < a < n - 1$, таке що $a^{n-1} \neq 1 (\text{mod n})$, називається *свідком Ферма* (свідком складеності) для n.

Означення. Нехай n − непарне складене число, a − ціле число, $1 < a < n - 1$. Число n називається *псевдопростим* за основою a, якщо $a^{n-1} \equiv 1\ (mod\ n)$. Число a називається *брехунцем Ферма* (брехунцем простоти) для n. Кількість брехунців Ферма для числа n будемо позначати через $fl(n)$ (Ferma liars).

Наприклад, для довільного складеного n число $a = 1$ завжди буде брехунцем Ферма, оскільки $1^{n-1} \equiv 1 (mod\ n)$.

АЛГОРИТМ

Вхід: непарне ціле число $n \ge 3$, параметр $t \ge 1$.

Вихід: визначення, чи є число n простим.

1. *for* $i \leftarrow 1$ *to* t *do*
   - 1.1. Обрати довільне ціле a, $2 \le a \le n - 2$.
   - 1.2. Обчислити $k \leftarrow a^{n-1} \ mod\ n$.
   - 1.3. *if* $k \neq 1$ *then return* («складене»).
2. *return* («просте»).

Якщо алгоритм дає відповідь «складене», то дійсно число є складеним. Якщо відповідь буде «просте», то або n є дійсно простим, або n є складеним, але має велику кількість брехунців. Чим більше значення параметра t, тим більше тестів буде зроблено і тим більша ймовірність того, що n є простим.

Приклад. Розглянемо складене число $n = 15$ та знайдемо його свідки та брехунці Ферма. Для цього складемо таку таблицю:

A	1	2	3	4	5	6	7
a^{14} mod 15	1	4	9	1	10	6	4

A	8	9	10	11	12	13	14
a^{14} mod 15	4	6	10	1	9	4	1

Свідками Ферма є числа 2, 3, 5, 6, 7, 8, 9, 10, 12, 13. Брехунцями Ферма є числа 1, 4, 11, 14.

Тест Ферма зручно використовувати для перевірки числа n на складеність, оскільки для більшості натуральних чисел кількість свідків більша за кількість брехунців.

Означення. Число n називається числом *Кармайкла,* якщо воно складене та для довільного a, $1 \le a \le n-1$, НСД $(a, n) = 1$, має місце рівність $a^{n-1} \equiv 1 (mod\ n)$. Найменше з них дорівнює $561 = 3 \cdot 11 \cdot 17$.

Критерій Корсельта. Для того, щоб складене число n було числом Кармайкла, необхідно і достатньо виконання двох умов:

- ✓ n не ділиться на квадрат простого числа;
- ✓ $n - 1$ ділиться на $p - 1$ для всякого простого дільника p числа n.

Приклад 1.

Простими дільниками числа 561 є 3, 11, 17. При цьому жоден з них не входить до розкладу навіть двічі, а число 560 ділиться на 2, 10 та 16:

$$560 : 2 = 280,\ 560 : 10 = 56,\ 560 : 16 = 35.$$

Твердження. Кожне число Кармайкла є добутком хоча б трьох простих чисел.

Приклад 2.

Числа Кармайкла в межах до 100 000:

561, 1105, 1729, 2465, 2821, 6601, 8911,

10 585, 15 841, 29 341, 41 041, 46 657,

52 633, 62 745, 63 973, 75 361.

Теорема (Чернік, 1939). Якщо $p = 6m + 1$, $q = 12m + 1$, $r = 18m + 1$ є простими числами, то число pqr є числом Кармайкла.

Приклад 3.

Якщо покласти $m = 1$, то отримаємо $p = 7$, $q = 13$, $r = 19$ – всі прості числа. Отже, $n = 7 * 13 * 19 = 1729$ − число Кармайкла.

Кількість чисел Кармайкла у натуральному ряді до 10^{12} дорівнює 8241, до 10^{13} – 19279, до 10^{14} – 44706, до 10^{15}– 105212. [6]

5. Ймовірнісній тест Соловай-Штрасена

Тест Соловай-Штрасена базується на критерії Ейлера: якщо n − просте, то $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) (\mathrm{mod}\, n)$ для всіх значень a, для яких НСД $(a, n)=1$.

Нехай n – непарне складене число, a − ціле число, $1 \le a \le n$ - 1.

1. Якщо НСД$(a, n) > 1$ або $a^{\frac{n-1}{2}} \neq \left(\frac{a}{n}\right) (\mathrm{mod}\, n)$, то число a називається *свідком Ейлера* (свідком складеності) для n.

2. Якщо НСД$(a,n) = 1$ та $a^{\frac{n-1}{2}} \neq \left(\frac{a}{n}\right) (\mathrm{mod}\, n)$, то число n називається *псевдопростим* за основою a. Число a називається *брехунцем Ейлера* (брехунцем простоти) для n. Кількість брехунців Ейлера для числа n будемо позначати через $el(n)$ (Euler liars).

АЛГОРИТМ

Вхід: непарне ціле число $n \ge 3$, параметр $t \ge 1$.

Вихід: визначення, чи є число n простим.

1. *for* $i \leftarrow 1$ to t *do*
   - 1.1. Обрати довільне ціле a, $2 \le a \le n$-2.
   - 1.2. Обчислити $k \leftarrow a^{(n-1)/2} \bmod n$.
   - 1.3. *if* $k \neq 1$ *and* $k \neq n$-1 *then return* («складене»).
   - 1.4. Обчислити символ Якобі $j \leftarrow \left(\frac{a}{n}\right)$.
   - 1.5. *if* $k \neq j(\mathrm{mod}\ n)$ *then return* («складене»).
2. *return* («просте»). [6]

6. Ймовірнісний тест Мілера-Рабіна

Тестом Мілера-Рабіна називається ймовірнісний тест перевірки на простоту, який було запропоновано Мілером з використанням ідей Рабіна.

Тест Мілера-Рабіна найбільш часто використовується на практиці і називається сильним тестом на простоту. Він базується на такому факті:

Нехай n – непарне просте число, причому $n - 1 = 2^s \cdot r$, де r – непарне. Нехай a – таке натуральне число, що НСД$(a,n) = 1$. Тоді має місце одна із рівностей: $a^r \equiv 1 (mod\ n)$ або $a^{2^j r} \equiv -1 (mod\ n)$ для деякого j, $0 \le j \le s - 1$.

Означення. Нехай n – непарне складене число, $n-1 = 2^s \cdot r$, де r – непарне, a – натуральне число, $1 \le a \le n - 1$.

1. Якщо $a^r \ne 1 (mod\ n)$ та $a^{2^j r} \ne -1 (mod\ n)$ для всіх j, $0 \le j \le s - 1$, то a називається *сильним свідком* (свідком складеності) для n.

2. Якщо $a^r \equiv 1 (mod\ n)$ або $a^{2^j r} \ne -1 (mod\ n)$ для деякого j, $0 \le j \le s - 1$, то a називається *сильним брехунцем* для n, а само число n - *сильним псевдо-простим* за основою a. Кількість сильних брехунців числа n будемо позначати через $sl(n)$ (strong liars).

АЛГОРИТМ

Вхід: непарне ціле число $n \ge 3$, параметр $t \ge 1$.

Вихід: визначення, чи є число n простим.

1. Записати $n - 1 = 2^s \cdot r$, де r – непарне.
2. *for i =1 to t do*
   - 2.1. Обрати довільне ціле a, $2 \le a \le n - 2$.
   - 2.2. Обчислити $y \leftarrow a^r \pmod n$.
   - 2.3. *if* $y \ne 1$ *and* $y \ne n - 1$ *then*

     $j \leftarrow 1$

     while $j \le s-1$ *and* $y \ne n-1$ *do*

     $y \leftarrow y^2 mod\ n$

     if $y = 1$ *then return*(«складене»).

     $j \leftarrow j+1$

     if $y \ne n-1$ *then return*(«складене»).
3. *return*(«просте»).

Приклад 4.

$n = 221 = 13 \cdot 17$ – складене число. $n - 1 = 220 = 2^2 \cdot 55$, $s = 2$, $r = 55$.

Нехай $a = 5$, $НСД(5, 221) = 1$. $a^r (mod\ n) \equiv 5^{55} (mod\ 221) \equiv 112 \ne -1$.

Вираз $a^{2^j r}$ будемо обчислювати для $j = 0, 1\ (0 \le j \le 1)$ поки не отримаємо – 1.

$j= 0: a^r(mod\ n) \equiv 5^{55}(mod\ 221) \equiv 112 \neq -1.$

$j= 0: a^{2r}(mod\ n) \equiv (5^{55})^2(mod\ 221) \equiv 168 \neq -1,$ що підтверджує складеність 221.

Нехай $a = 21, НСД(21,221) = 1.\ a^r(mod\ n) \equiv 21^{55}(mod\ 221) \equiv 200 \neq 1.$

$j = 0: a^r(mod\ n) \equiv 21^{55}(mod\ 221) \equiv 200 \neq -1.$

$j = 1: a^{2r}(mod\ n) \equiv (21^{55})^2(mod\ 221) \equiv -1,$ отже, 221 може бути простим.

Приклад показує, що число 5 є сильним свідком для 221, а 21 є сильним брехунцем для 221.

Якщо перебрати в якості *a* всі значення від 1 до 220, то можна побачити, що число 221 має 6 таких сильних брехунців: 1, 21,47, 174, 200, 220, а *sl*(221) = 6. [6]

7. Властивості брехунців

Основним показником якості перелічених тестів на простоту є кількість ітерацій, після виконання яких на складеному вхідному числі тест дасть відповідь «складене». Кожен із тестів має брехунців. Чим менша кількість брехунців існує для заданого складеного числа *n,* тим менша кількість ітерацій необхідна для визначення його складеності.

Приклад 5.

Нехай *n* = 221.

Брехунці Ферма: 1, 18, 21, 38, 47, 64, 86, 103, 118, 135, 157, 174, 183, 200, 203, 220. Кількість брехунців: *fl*(221) = 16.

Брехунці Ейлера: 1, 18, 21, 38, 47, 64, 86, 103, 118, 135, 157, 174, 183, 200, 203, 220. Кількість брехунців: *el*(221) = 16.

Сильні брехунці. 1, 21, 47, 174, 200, 220. Кількість брехунців: *sl*(221) = 6.

Таким чином, при використанні тесту Ферма ймовірність визначення складеності числа 221 з першої ітерації дорівнює 205/221, Соловай-Штрасена − 205/221, Мілера-Рабіна − 215/221.

Нехай *n* − непарне складене число. Тоді:

1) Якщо a − сильний брехунець для числа n, то a буде брехунцем Ейлера для числа n.

2) Якщо a − брехунець Ейлера для числа n, то a буде брехунцем Ферма для числа n.

Якщо для заданого числа n побудувати множини брехунців для кожного із трьох наведених ймовірнісних тестів, то вони розташуються так, як показано на рис.

Рис. 5.1 Множини брехунців одного числа для різних тестів

Твердження. Якщо $n \equiv 3 \pmod 4$, то число a є брехунцем Ейлера тоді і тільки тоді, коли воно є сильним брехунцем.

Твердження. Нехай n − непарне складене число. Тоді якщо $n \neq 9$, то кількість його сильних брехунців не більша за $\varphi(n)/4$.

Твердження. Нехай $n = p \cdot q$ − добуток двох простих чисел, $d = НСД(p-1, q-1)$. Тоді кількість брехунців числа n дорівнює $sl(n) = r^2 \cdot (2 + (4'-4)/3)$, де $d = 2' \cdot r$, r − непарне.

Приклад 6.

Обчислимо кількість сильних брехунців складеного числа $n = 221 = 13 \cdot 17$. Маємо: d= НСД(12,16) = 4 = $2^2 \cdot 1$, $r = 1$, $t = 2$. Отже, $sl(221) = 1^2 \cdot (2 + (4^2 - 4)/3) = 2 + 4 = 6$.

Твердження. Нехай $n = p \cdot q$ − добуток двох простих чисел, $p = 2 \cdot r+1$, $q=4 \cdot r+1$, r − непарне. Тоді кількість брехунців досягає своєї верхньої межі: $sl(n)=\varphi(n)/4$.

Приклад 7.

При $r = 1$ маємо: $p = 3$, $q = 5$, $n = p \cdot q = 15$. $sl(15)=\varphi(15)/4=(3-1) \cdot (5-1)/4=2 \cdot 4/4=2$. Число 15 дійсно має два сильні брехунці. [6]

Тема 6. АРИФМЕТИЧНІ ОСНОВИ КРИПТОГРАФІЇ (ч. 2)

1. Набори лишків

Числа, конгруентні за модулем *m* утворюють *клас лишків* за модулем *m*. Усі числа з одного класу мають одну і ту же остачу *r* від ділення на *m*. Будь-яке число *a* з класу лишків називається лишком за модулем *m*. Відповідний клас позначається через $\bar{a}$. Оскільки відношення $a \equiv b(mod\ m)$ є відношенням еквівалентності, маємо розбиття цілих чисел на класи залишків. Всього є *m* класів лишків за модулем *m*: ($\bar{0}, \bar{1}, \bar{2}, \ldots, \overline{m-1}$). Взявши з кожного класу по одному лишку отримаємо повний набір (систему) лишків.

m цілих чисел від 0 до *m-1* – це повний набір лишків.

Властивість 1. Будь-які *m* чисел, попарно непорівнювані за модулем *m* утворюють повний набір лишків.

Властивість 2. Якщо $НСД(a,m)=1$ і *x* пробігає повний набір лишків за модулем *m*, то $ax+b$, де *b* − будь-яке ціле, також пробігає повний набір лишків за модулем *m*.

Згідно властивості $a \equiv b(mod\ m) \Rightarrow НСД(a,m)=НСД(b,m)$ числа одного класу лишків мають з модулем *m* один і той же спільний дільник.

Розглянемо ті класи, для яких цей дільник рівний 1. Взявши з одного такого класу по одному лишку отримаємо *зведений набір лишків* (систему лишків).

Приклад 1.

Зведений набір за модулем *42* буде: 1, 5, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41.

Означення. Набір лишків, взаємно простих з модулем *m*, взятий із повного їх набору, називається *зведеним набором лишків за модулем m.*

Зведений набір лишків за модулем простого числа *m* містить *m-1* елементів і становить *{1,2,…, m-1}*.

Приклад 2.

Модуль *m=10*. *Повний набір лишків* становить $r \in \{1,2,\ldots, 9\}$. Відповідний *зведений набір лишків* містить елементи, взаємно прості з 10, і становить *{1,3, 7, 9}*. Таким чином, із повного набору лишків виключено елемент 0, елементи *2*, *4*, *6* і *8*, які мають з модулем *m=10* спільний дільник 2, а також елемент 5, який є дільником модуля.

Означення. Набором лишків цілого числа *a* за модулем *m* називається сукупність усіх різних остач від ділення чисел $a \cdot q$ на модуль *m*, де *q* послідовно приймає всі значення від *0* до *m-1*.

Приклад 3.

Набір лишків числа *a=5* за модулем *m=6* такий: *{0,5,4,3,2,1}*.

q	0	1	2	3	4	5
$a \cdot q$	0	5	10	15	20	25
$a \cdot q \bmod m$	0	5	4	3	2	1

Приклад 4.

Набір лишків числа *a=10* за модулем *m=6* такий: *{0,4,2}*

Q	0	1	2	3	4	5
$a \cdot q$	0	10	20	30	40	50
$a \cdot q \bmod m$	0	4	2	0	4	2

Отже, довжина набору лишків *(кількість)* цілого числа *a* за модулем *m* максимальна і рівна *m* у разі, якщо числа *a* і *m* взаємно прості.

Означення. Дано просте число *p*. Ціле число *g* називається *первісним коренем* за простим модулем *p*, якщо послідовність яка складається із *p-1* елементів $g^0 \bmod m = 1$, $g^1 \bmod p$, $g^2 \bmod p$, ... , $g^{p-2} \bmod p$ містить всі елементи зведеного набору лишків за модулем *p*, тобто *{1,2,…,p-1}*.

Оскільки зведений набір лишків за модулем простого числа *p* містить *p-1* елементів і така ж кількість елементів у вказаній послідовності, то це означає, що всі елементи цієї послідовності мають бути попарно різні і перший елемент послідовності – завжди рівний одиниці.

При побудові послідовності черговий елемент $g^k \ mod \ p$ доцільно отримувати шляхом домножування вже обчисленого попереднього елемента $g^{k-1} \ mod \ p$ на $g \ mod \ p$.

Приклад 5.

Число $g=3$ є первісним коренем за простим модулем $p=5$.

$3^0 \ mod \ 5$	$1 \cdot 3 \ mod \ 5$	$3 \cdot 3 \ mod \ 5$	$4 \cdot 3 \ mod \ 5$
1	3	4	2

Приклад 6.

Число $g=4$ не є первісним коренем за простим модулем $p=5$.

$4^0 \ mod \ 5$	$1 \cdot 4 \ mod \ 5$	$4 \cdot 4 \ mod \ 5$	$1 \cdot 4 \ mod \ 5$
1	4	1	4

В нижньому рядку бачимо повторення елементів. Це означає що 4 не є первісним коренем за простим модулем $p=5$. [8]

2. Дискретне піднесення до степеня

Означення. Дискретне піднесення до степеня являє собою обчислення цілочисельної функції $x=g^y mod \ p$, де модуль p – натуральне число, y – ціле невід'ємне число, $1 \leq g \leq p$.

Сучасна інтерпретація «індійського піднесення до степеня» відома під назвою *дискретне піднесення до степеня бінарним методом.*

Ефективність бінарного методу визначається різким скороченням кількості виконуваних множень.

Бінарний метод базується на тому, що будь-який показник степеня може бути розкладений за степенями двійки.

Приклад 7.

Розглянемо дискретне піднесення до степеня $7^{50} \ mod \ 11$. Розкладаємо показник степеня за степенями двійки: $50_{10} = 110010_2 = 1 \cdot 2^5+1 \cdot 2^4+0 \cdot 2^3+0 \cdot 2^2+1 \cdot 2^1+0 \cdot 2^0 = 32+16+2$.

Звідси отримуємо: $7^{50} = 7^{32} \cdot 7^{16} \cdot 7^2$

Це показує, що кожен наступний проміжний результат може бути отриманим шляхом множення самого на себе попереднього результату.

Зокрема, щоб отримати 7^{32} потрібно перемножити $7^{16} \cdot 7^{16}$. За рахунок цього як раз і скорочується кількість потрібних операцій множення.

Реальний процес дискретного піднесення до степеня $7^{50} \ mod \ 11$ можна представити у вигляді наступної таблиці:

Показник	Основа	Результат =1 (коли «Показник» mod 2=1)
50	$7^1 \bmod 11 = 7$	
25	$7^2 \bmod 11 = 5$	$1 \cdot 5 \bmod 11 = 5$
12	$7^4 \bmod 11 = 5^2 \bmod 11 = 3$	
6	$7^8 \bmod 11 = 3^2 \bmod 11 = 9$	
3	$7^{16} \bmod 11 = 9^2 \bmod 11 = 4$	$5 \cdot 4 \bmod 11 = 9$
1	$7^{32} \bmod 11 = 4^2 \bmod 11 = 5$	$9 \cdot 5 \bmod 11 = 1$
0		

Зміст таблиці циклічного процесу дискретного піднесення до степеня наступний:

- ✓ стовпчик «Показник» містить результати поступового цілочисельного ділення на 2 даного показника степеня 50; ознакою завершення процесу є нульове значення показника степеня;
- ✓ в стовпчику «Основа» здійснюється множення попереднього результату самого на себе за заданим модулем, починаючи із заданого значення 7;
- ✓ в стовпчику «Результат» здійснюється накопичення потрібного результату шляхом множення попереднього його значення на поточне значення основи в ті моменти, коли показник є непарним; початкове значення результату вважається рівним одиниці.

Особливо добре ефективність бінарного методу видно на великих числах.[8]

3. Функція Ейлера. Теорема Ейлера. Теорема Ферма

Означення. Число класів залишків у зведеному наборі залишків позначають через $\phi(m)$ і ***називають функцією Ейлера.*** Функція Ейлера визначена для всіх натуральних чисел і являє собою кількість взаємно простих з a натуральних чисел, що не перевищують a.

Приклад 8.

$\phi(1) = 1, \phi(2) = 1, \phi(3) = 2, \phi(4) = 2, \phi(5) = 4, \phi(6) = 2.$

Для *m=10* взаємно простими з ним та меншими його є натуральні числа 1, 3, 7, 9. Отже, $\phi(10) = 4$.

Очевидні наступні властивості:

1. $\phi(p) = p - 1$
2. $\phi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1),\ k \epsilon N$

Приклад 9.

При *p=3, k=4* отримуємо: $\phi(3^4) = 3^{4-1}(3-1) = 27 \cdot 2 = 54$, Дійсно, $\phi(3^4) = \phi(81) = 54$, оскільки для отримання числа елементів зведеного набору залишків по модулю 81 треба від числа елементів повного набору залишків 81 відняти 27 елементів, значення яких кратні трьом.

Лема. *(Мультиплікативність функції).* Якщо *НСД(a,b)=1* => $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$.

n= $p_1^{к1} * p_2^{к2} *_{...} * p_s^{кs}$, p_i – різні прості числа, $k_i \geq 1$ ($k_i \in N$), тоді $\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)...\left(1 - \frac{1}{p_s}\right)$.

Приклад 10.

При *a=2, b=5* і *a·b = 2·5 = 10* маємо $\phi(2 \cdot 5) = \phi(2) \cdot \phi(5) = 1 \cdot 4 = 4$

З функцією Ейлера пов'язана його ж знаменита теорема.

Теорема Ейлера. Для взаємно простих цілого *x* та натурального *m* *(НСД(x,m)=1)* справедлива конгруенція $x^{\phi(m)} \equiv 1\ (mod\ m)$ або $x^{\phi(m)}\ mod\ m \equiv 1$.

Приклад 11.

Для *x=3, m=10* маємо $3^4 \equiv 1 (mod\ p)$, тобто $3^4\ mod\ 10 = 81\ mod\ 10 = 1$.

Наслідок з теореми Ейлера (відомий під назвою *мала теорема Ферма).*

Якщо *p* – просте число і *НСД(a, p) = 1*, то $a^{p-1} \equiv (mod\ p)$ або $a^{p-1}\ mod\ p = 1$

Висновок малої теореми Ферма випливає з того, що $\phi(p) = p - 1$.

Наслідок з малої теореми Ферма: $a^p \equiv a(mod\ p)$, тобто $a^p\ mod\ p = a\ mod\ p$, при всіх цілих *a*, в тому числі і кратних *p*.

Приклад 12.

Дано взаємно прості число $a = 10$ і просте число $p = 3$, тобто $НСД(10, 3) = 1$. Тоді у відповідності з малою теоремою Ферма маємо $10^2 \equiv 1(mod\ 3)$, тобто $10^{3-1}\ mod\ 3 = 10^2\ mod\ 3 = 100\ mod\ 3 = 1$. Одночасно, у відповідності із наслідком з малої теореми Ферма маємо $10^3\ mod\ 3 = 1$ і $10\ mod\ 3 = 1$. [8]

4. Діофантове рівняння

Відомо, що взаємно прості числа – це такі натуральні a і m для яких $НСД(a,m)=1$. Для такої пари взаємно простих чисел a і m завжди можна знайти такі цілі U, V що $aU+mV=1$ або $aU-mV=1$, $a>0$, $m>0$. Такі рівняння називаються діофантовими рівнянням першого степеня.

Для вирішення цієї задачі число $\alpha = \frac{a}{m}$ перетворюють у скінченний ланцюговий дріб за допомогою алгоритму Евкліда:

$a=m\cdot q_0 + a_1,$

$m=a_1\cdot q_1+a_2,$

$a_1=a_2\cdot q_2 + a_3,$

$a_2=a_3\cdot q_3 + a_4,$

...

$a_{k-2}=a_{k-1}\cdot q_{k-1} + a_k,$

$a_{k-1}=a_k\cdot q_k + 0.$

Ланцюговий дріб має вигляд: $\frac{a}{m} = [q_0\ ,\ q_1\ ,\ \dots\ ,\ q_k]$, а послідовності $\{P_n\}$ і $\{Q_n\}$ чисельників і знаменників підходящих дробів для ланцюгового дробу визначаються рекурентно:

$P_{-2} = 0;\ P_{-1} = 1;$

$Q_{-2} = 1;\ Q_{-1} = 1;$

При

$n\geq 0 \rightarrow P_n = q_n\cdot P_{n-1} + P_{n-2};$

$n\geq 0 \rightarrow Q_n = q_n\cdot Q_{n-1} + Q_{n-2}.$

Їх обчислення зручно оформляти у вигляді таблиці:

n	-2	-1	0	1	2	...	k-1	k
q_n			q_0	q_1	q_1	...	q_{k-1}	q_k
P_n	0	1	P_0	P_1	P_2	...	P_{k-1}	P_k
Q_n	1	0	Q_0	Q_1	Q_2	...	Q_{k-1}	Q_k

Але відомо, що $P_k \cdot Q_{k-1} - P_{k-1} \cdot Q_k = (-1)^{k-1}$ і $a/m = P_k/Q_k$.

Таким чином, $(-1)^{k-1} P_k \cdot Q_k - P_{k-1} \cdot (-1)^{k-1} Q_k = 1$. Так як $НСД\ (a,m)=1$, то $P_k = a$, $Q_k = m$ іншими словами: пара чисел U, V рівна $U = (-1)^{k-1} Q_{k-1}$ та $V = (-1)^{k-1} P_{k-1}$, є цілочисельним розв'язком діофантового рівняння.

Приклад 13.

Знайти розв'язки рівняння: $aU-mV=1$, при $a=211, m=79$.

n	-2	-1	0	1	2	3
q_n			2	1	2	26
P_n	0	1	2	3	8	211
Q_n	1	0	1	1	3	79

За формулою $a=m \cdot q_0 + a_1$, можна знайти q_0 та a_1

$211=79 \cdot q_0 + a_1$, $q_0 = 211\ div\ 79=2$, отже, $a_1=53$.

При $n = 0$, маємо:

$P_0 = q_0 \cdot P_{-1} + P_{-2} = 2 \cdot 1 + 0 = 2$;

$Q_0 = q_0 \cdot Q_{-1} + Q_{-2} = 2 \cdot 0 + 1 = 1$.

За формулою $m=a_1 \cdot q_1 + a_2$, можна знайти q_1 та a_2

$79 = 53 \cdot q_1 + a_2$, $q_1 = 79\ div\ 53 = 1$, тому $a_2=26$.

При $n = 1$, маємо:

$P_1 = q_1 \cdot P_0 + P_{-1} = 1 \cdot 2 + 1 = 3$;

$Q_1 = q_1 \cdot Q_0 + Q_{-1} = 1 \cdot 0 + 1 = 1$.

Користуючись формулою $a_1=a_2 \cdot q_2 + a_3$, знаходимо q_2 та a_3

$53 = 26 \cdot q_2 + a_2$, $q_2 = 53\ div\ 26 = 2$, тому $a_3=1$.

При $n = 2$, маємо:

$P_2 = q_2 \cdot P_1 + P_0 = 2 \cdot 3 + 2 = 8$;

$Q_2 = q_2 \cdot Q_1 + Q_0 = 2 \cdot 1 + 1 = 3.$

Далі за формулою $a_2 = a_3 \cdot q_3 + a_4$, знаходимо q_3 та a_4

$26 = 1 \cdot q_3 + a_4$, $q_3 = 26 \; div \; 1 = 26$, тому $a_4 = 0$.

При $n = 3$, маємо:

$P_3 = q_3 \cdot P_2 + P_1 = 26 \cdot 8 + 3 = 211$;

$Q_3 = q_3 \cdot Q_2 + Q_1 = 26 \cdot 3 + 1 = 79.$

Отже, при $k = 3$, $P_3 = a = 211$, $Q_3 = m = 79$. Тому знаходимо пару чисел U, V: $U = (-1)^2 \cdot Q_2 = 1 \cdot 3 = 3$ та $V = (-1)^2 \cdot P_2 = 1 \cdot 8 = 8$.

Перевірка, дійсно $3 \cdot 211 - 8 \cdot 79 = 633 - 632 = 1$.

Тема 7. ШИФРИ, ЩО ВИКОРИСТОВУЮТЬ АНАЛІТИЧНІ ПЕРЕТВОРЕННЯ

В загальному випадку будь-який шифр аналітичних перетворень $y=f(x)$ за даним елементом x відкритого тексту обчислює відповідний елемент криптограми y. Тому ці шифри належать до шифрів заміни. Кожному з них відповідає певне правило обчислення f. Правила обчислень базуються на різноманітних математичних методах.

1. Афінна система моноалфавітної заміни

Афінна система моноалфавітної системи (АФМЗ) відрізняється від замін Цезаря. Вона дає суттєву відмінність у порядку слідування знаків вторинного алфавіту у порівнянні з первинним. Афінна система моноалфавітної заміни використовує модулярну арифметику. Її загальна формула має вигляд: $j_y=(a\cdot j_x+b) \bmod m$, де

j_x – *порядковий номер знака відкритого тексту в алфавіті,*

j_y – *порядковий номер знака криптограми в алфавіті,*

m - *обсяг алфавіту,*

a, b – *ключ заміни, який являє собою пару цілих чисел* $0 \le a < m$, $0 \le b < m$

Для побудови шифрувальної таблиці цю формулу застосовують послідовно до кожного знаку алфавіту, при цьому вказане криптографічне перетворення є взаємно однозначним на даному алфавіті тільки при умові, що числа a і m взаємно прості. $(НСД(a,m)=1)$.

Побудуємо, наприклад, шифрувальну таблицю для алфавіту «АБВГДЕЖЗИК» обсягом $m=10$, використовуючи ключ $(a,b)=(7,3)$. При цьому всі вказані вище умови виконуються, в тому числі $НСД(7,10)=1$.

Порядковий номер знака алфавіту	Знак первинного алфавіту (відкритого тексту)	Результат обчислення за формулою $j_y=(a\cdot j_x+b) \bmod m$	Знак вторинного алфавіту (криптограма)
0	А	$(7\cdot 0 + 3) \bmod 10 = 3$	Г
1	Б	$(7\cdot 1 + 3) \bmod 10 = 0$	А

2	В	(7·2 + 3) mod 10 = 7	З
3	Г	(7·3 + 3) mod 10 = 4	Д
4	Д	(7·4 + 3) mod 10 = 1	Б
5	Е	(7·5 + 3) mod 10 = 8	И
6	Ж	(7·6 + 3) mod 10 = 5	Е
7	З	(7·7 + 3) mod 10 = 2	В
8	И	(7·8 + 3) mod 10 = 9	К
9	К	(7·9 + 3) mod 10 = 6	Ж

Шифрувальна таблиця набуває наступного вигляду:

Номер	0	1	2	3	4	5	6	7	8	9
Знак відкритого тексту	А	Б	В	Г	Д	Е	Ж	З	И	К
Знак криптограми	Г	А	З	Д	Б	И	Е	В	К	Ж

У відповідності з наведеним прикладом відкритий текст «ЖИВЕ» → «ЕКЗИ». Дешифрування криптограми здійснюється за цією ж шифрувальною таблицею.

Для АСМЗ характерний недолік пов’язаний з відсутністю маскування частот появи різних знаків відкритого тексту. Перевагою методу є те, що вторинний алфавіт є суттєво перевпорядкованим і водночас існує проста можливість його обчислення через первинний.

Відмітимо, що вимога *НСД(a,m)=1* є надзвичайно суттєвою, оскільки саме вона забезпечує повний набір знаків вторинного алфавіту. [8]

2. Шифр Віженера

Шифр Віженера також використовує модулярну арифметику.

У загальному випадку застосування шифру Віженера полягає у наступному:

Розглянемо алфавіт відкритого тексту $A = (a_0, a_1, \dots, a_{m-1})$, що містить m знаків, ключ $K = (k_1, k_2, \dots, k_{n-1})$, знаки якого вибрано випадковим способом алфавіту A. Довжина ключа n рівна довжині відкритого тексту $X = (x_1, x_2, \dots, x_{n-}$

$_1$). Накладемо ключ на відкритий текст. Розглянемо деякий знак x_p, $p=0,1,2,\ldots,n-1$ відкритого тексту X. Він має порядковий номер i в алфавіті A. Розглянемо також відповідний знак ключа k_p. Він має в алфавіті A порядковий номер j. Тоді знак x_p відкритого тексту перетворюється у знак криптограми y_p у відповідності з правилом $y_p=a_{(i+j) \bmod m}$.

Наприклад, для алфавіту «абвгдеж» обсягом $m=7$ таблиця порядкових номерів знаків відкритого тексту має наступний вигляд:

Номер	0	1	2	3	4	5	6
Знак відкритого тексту	а	б	в	г	д	е	ж

Зашифруємо відкритий текст «вежагад» за допомогою ключа «бгбдеиг» тієї самої довжини. Ще раз підкреслимо що знаки відкритого тексту і знаки ключа вибрано із одного й того ж алфавіту. Процес застосування шифру Віженера можна представити у вигляді наступної таблиці

Знак відкритого тексту	Порядковий номер знака відкритого тексту в алфавіті *(i)*	Знак ключа шифрування	Порядковий номер знака ключа шифрування в алфавіті *(j)*	Обчислення номера знака криптограми в алфавіті	Знак криптограми
в	2	б	1	(2+1) mod 7 = 3	г
е	5	г	3	(5+3) mod 7 = 1	б
ж	6	б	1	(6+1) mod 7 = 0	а
а	0	д	4	(0+4) mod 7 = 4	д
г	3	е	5	(3+5) mod 7 = 1	б
а	0	в	2	(0+2) mod 7 = 2	в
д	4	г	3	(4+3) mod 7 = 0	а

Таким чином отримуємо криптограму «гбадбва».

Для дешифрування криптограм потрібно спочатку отриманий відповідний ключ. Тут застосовується дуже просте правило: якщо деякий знак ключа шифрування має деякий номер j в алфавіті A, то відповідний знак ключа дешифрування знаходиться в тому ж алфавіті під номером $(m-j) \bmod m$.

Для прикладу, ключ шифрування «бгбдевг» перетворимо у відповідний ключ дешифрування при умові, що використовується той же самий алфавіт.

Знак ключа	Порядковий	Обчислення	Знак ключа

шифрування	номер знака ключа шифрування в алфавіті	порядкового номеру знака ключа дешифрування	дешифрування
б	1	(7-1) mod 7 = 6	ж
г	3	(7-3) mod 7 = 4	д
б	1	(7-1) mod 7 = 6	ж
д	4	(7-4) mod 7 = 3	г
е	5	(7-5) mod 7 = 2	в
в	2	(7-2) mod 7 = 5	е
г	3	(7-3) mod 7 = 4	д

Таким чином, отримуємо ключ дешифрування «жджгвед».

Тепер дешифрування криптограми можна здійснити точно таким же способом, як і шифрування відкритого тексту.

Знак криптограми	Порядковий номер знака криптограми в алфавіті	Знак ключа дешифрування	Порядковий номер знака ключа дешифрування в алфавіті *(j)*	Обчислення номера знака криптограми в алфавіті	Знак відкритого тексту
г	3	ж	6	(3+6) mod 7 = 2	в
б	1	д	4	(1+4) mod 7 = 5	е
а	0	ж	6	(0+6) mod 7 = 6	ж
д	4	г	3	(4+3) mod 7 = 0	а
б	1	в	2	(1+2) mod 7 = 3	г
в	2	е	5	(2+5) mod 7 = 0	а
а	0	д	4	(0+4) mod 7 = 4	д

Таким чином, отримуємо відкритий текст «вежагад».

При такому підході маємо абсолютно стійкий шифр, подібний до шифру Вернама. Щоб позбутися незручності такого шифру пов’язаним з великий обсягом ключа *K*, Віженер запропонував використовувати коротший ключ *K**, який шляхом свого періодичного повторення може бути перетворений у ключ *K*, довжина якого співпадає з довжиною відкритого тексту *X*.

Наприклад, алфавіт «0123456789» довжиною *m=10*, ключ шифрування «738» відкритий текст «5742906». Тоді криптограма має вигляд «2029283», а люч дешифрування буде «372». [8]

3. Шифр з автоключем

Умови існування абсолютно стійкого шифру кажуть, властивості шифру Вернама підтверджують те, що чим більша довжина ключа, тим вища стійкість шифру. Але, з іншої сторони, як показує шифр Віженера, користуватися короткими ключами значно зручніше.

Шифр з авто ключем, не позбавляючи зручностей, пов’язаних з коротким ключем, дозволяє подовжувати ключ автоматично.

Короткий ключ, як і у шифрі Віженера, вибирається звичайним способом з букв використовуваного алфавіту.

Автоматичне подовження ключа можливе за двома схемами:

- ✓ шляхом приєднання до короткого ключа того ж самого відкритого тексту;
- ✓ шляхом поступового приєднання до короткого ключа утворюваної криптограми.

У всьому іншому *(шифрування/дешифрування)* використання шифру з автоключем повністю співпадає з шифром Віженера. [8]

Тема 8. ШИФРИ З ВИКОРИСТАННЯМ ГАМУВАННЯ

Однією з ознак абсолютно стійкого шифру є повна випадковість ключа. Зрозуміло, що використовуючи випадкові числа, можна побудувати будь-який випадковий ключ.

Одним із найпростіших механічних приладів для отримання випадкових чисел є рулетка. В сучасних умовах для отримання випадкових чисел застосовують різноманітні генератори, які поділяються на дві групи – апаратні та програмні. В апаратних генераторах джерелом випадкових чисел є шум в електронних приладах. Програмний генератор випадкових чисел являє собою програму, яка генерує послідовність чисел за деяким алфавітом. Завдяки алгоритму, така послідовність чисел цілком детермінована (визначена), тобто принципово не може бути випадковою. Але, оскільки така числова послідовність за своїм зовнішнім виглядом та властивостями дуже нагадує випадкову, то її називають *послідовністю псевдовипадкових чисел.*

В криптографії послідовність псевдовипадкових чисел називають *гамою шифру,* або просто *гамою.* Гама має свій період. Період гами – це така кількість псевдовипадкових чисел у послідовності, після якої вони починають повторюватись. [8]

1. Шифр звичайного накладання двійкової гами

Процес шифрування звичайним накладанням двійкової гами полягає у наступному [8]:

1. Відкритий текст подають у вигляді неперервної послідовності k-розрядних двійкових чисел. Для цього використовують перетворення порядкових номерів знаків відкритого тексту із алфавіту обсягом $m=2^k$.
2. Генерують гаму шифру у вигляді послідовності псевдовипадкових двійкових цифр.
3. На кожний черговий розряд відкритого тексту накладають відповідний розряд двійкової гами з використанням операції додавання по модулю 2 і таким шляхом отримують черговий двійковий розряд криптограми.

(Виключна диз'юнкція (також операція XOR, додавання за модулем два) – двомісна логічна операція, що приймає значення «істина» тоді і тільки тоді коли значення «істина» має рівно один з її операндів. Виключна диз'юнкція є запереченням логічної еквівалентності. Для запису операції використовуються позначення: $a \oplus b$).

4. Подають криптограму через даний алфавіт, виконавши попереднє розбиття криптограми на послідовні к-розрядні двійкові числа

Рівність $m=2^k$ є природнім обмеженням даного методу. Якщо $m>2^k$, то у вигляді k-розрядних двійкових чисел неможливо представити всі знаки алфавіту. Якщо $m<2^k$, то не кожне двійкове число може бути представлене у вигляді знаку даного алфавіту. В останньому випадку залишається тільки відмовитися від цієї операції.

Ключові дані, що входять до складу шифру, такі:

- ✓ алфавіт з порядковими номерами знаків, а також кількість розрядів двійкового їх подання;
- ✓ параметри обраного методу генерації двійкової гами шифру, в тому числі кількість розрядів двійкового подання кожного псевдовипадкового числа їх послідовності.

Розглянемо приклад шифрування відкритих текстів, побудованих в алфавіті обсягом $m=2^3=8$ знаки цього алфавіту можна представити у вигляді 3-розрядних двійкових чисел:

Номер	0	1	2	3	4	5	6	7
Знак відкритого тексту	А	Б	В	Г	Д	Ж	З	Е
Двійковий код	000	001	010	011	100	101	110	111

Двійкову гамму шифру згенеруємо лінійним конгруентним методом $x_{n+1}=(a \cdot x_n+b) \bmod m,\ n=0, 1, \ldots$ з параметрами $m=16,\ a=5,\ b=7,\ x_0=10$. Початок послідовності псевдовипадкових чисел має вигляд: 10, 9, 4, 11, 14, …. Оскільки тут $m=2^4$, то кожне десяткове число слід представляти не менше як 4-

рирозрядне двійкове. У випадку 4-х розрядного подання, початок двійкової гами такий: 1010|1001|0100|1011|1110 ….

Шифрування:

Відкритий текст «БАГАЖА»	001|000|011|000|101|000
Гамма шифру	101|010|010|100|101|111
Результат додавання по модулю 2	100|010|001|100|000|111
Криптограма	Д В Б Д А Е

При дешифрування криптограми на неї накладається та ж сама двійкова гамма.

Дешифрування:

Криптограма «ДВБДАЕ»	100|010|001|100|000|111
Гамма шифру	101|010|010|100|101|111
Результат додавання по модулю 2	001|000|011|000|101|000
Відкритий текст	Б А Г А Ж А

2. Шифр багаторазового накладання двійкових гам

Шифр полягає в тому, що на відкритий текст спочатку накладається перга гама, потім на результат шифрування накладається друга гама, на новий результат − третя і так далі. Використання цього шифру пов’язане з тим, що є можливість досягнути більшого періоду результуючої гами у порівнянні з періодами складових гам. Для цього, періоди складових гам мають бути різними.

При використанні такого шифру слід пам’ятати, що результат шифрування не залежить від послідовності накладання гам. Аналогічно, і результат дешифрування буде правильним не залежно від того, в якому порядку застосовувати гами. [8]

3. Комбінований шифр Френдберга

Цей шифр є комбінованим, оскільки поєднує заміну з перестановкою, яка, в свою чергу, використовує гаму шифру. Як було доведено автором шифру, даний метод дозволяє приховувати частотні характеристики алфавіту відкритого тексту, що ускладнює зламування шифру.

Основні кроки шифрування чергового знаку відкритого тексту методом Френдберга наступні:

- ✓ виконання заміни згідно шифрувальної таблиці;
- ✓ отримання чергового числа гами шифру;
- ✓ використання цього числа для виконання перестановок у шифрувальній таблиці.

Далі здійснюється перехід до шифрування наступного знаку відкритого тексту і т.д.

Як бачимо, основні складові методу традиційні – шифрувальна таблиця і гама шифру.

Але особливістю в даному випадку є те, що обсяг алфавіту повідомлень і модуль послідовності псевдовипадкових чисел мають співпадати.

Шифрування методом Френдберга найбільш доцільно розглянути на прикладі.

Розглянемо приклад шифрування відкритих текстів, побудованих в алфавіті обсягом $m = 7$. Спосіб вибору шифрувальної таблиці для виконання замін не є суттєвим і може бути довільним. У загальному випадку її початковий вигляд вважається заданим і входить до складу ключових даних. Припустимо, що шифрувальна таблиця має наступний вигляд.

Номер	0	1	2	3	4	5	6
Знак відкритого тексту	А	Б	В	Г	Д	Е	Ж
Знак криптограми	D	X	T	V	N	S	W

Спосіб отримання гами шифру для виконання перестановок теж не є суттєвим і може бути довільним. Важливо тільки, щоб якимось чином була забезпечена достатня кількість псевдовипадкових чисел у їх послідовності. Крім того, вимога співпадання обсягу алфавіту повідомлень та модуля послідовності псевдовипадкових чисел для вибраного алфавіту проявляється в тому, що псевдовипадкові числа можуть приймати значення в межах від 0 до 6 включно. Припустимо, що послідовність псевдовипадкових чисел така: 403631205314625….

Процес шифрування відкритого тексту “ВЖЕДАГЖ” представлено в таблиці. Отримуємо криптограму “TWSTTSD” . Дешифрування криптограми відбувається аналогічно. Отже, наведену таблицю можна розглядати одночасно і як таблицю дешифрування криптограми “TWSTTSD”. Звичайно, буде отримано відкритий текст “ВЖЕДАГЖ”.

Цікаво, що даний приклад дійсно показує, що даний шифр дозволяє якісно приховати частотні характеристики алфавіту повідомлень. Це підтверджується, наприклад, тим, що один і той же знак криптограми “Т” відповідає різним знакам відкритого тексту (“В”, “Д” і “А”). [8]

Таблиця процесу шифрування (дешифрування):

Крок	Видозміни шифрувальних таблиць	Знак відкритого тексту	Знак криптограми	Порядковий номер знаку криптограми в алфавіті	Псевдовипадкове число	Перестановка знаків відкритого тексту у шифрувальній таблиці
1	0123456 АБВГДЕЖ DXTVNSW	В	Т	2	4	2 ⇔ 4
2	0123456 АБДГВЕЖ DXTVNSW	Ж	W	6	0	6 ⇔ 0
3	0123456 ЖБДГВЕА DXTVNSW	Е	S	5	3	5 ⇔ 3
4	0123456 ЖБДЕВГА DXTVNSW	Д	Т	2	6	2 ⇔ 6
5	0123456 ЖБАЕВГД DXTVNSW	А	Т	2	3	2 ⇔ 3
6	0123456 ЖБЕАВГД DXTVNSW	Г	S	5	1	5 ⇔ 1
7	0123456 ЖГЕАВБД DXTVNSW	Ж	D	0	2	0 ⇔ 2
8	0123456 ЕГЖАВБД DXTVNSW					

Тема 9. СТАНДАРТ ШИФРУВАННЯ ДАНИХ DES, GOST (ГОСТ 28147-89)

На сьогодні розроблено багато симетричних криптосистем, багато з яких є національними стандартами. Найбільш відомі DES, GOST та деякі інші.

Стандарт DES (*Data Encryption Standard*) вперше був опублікований в США у 1975 р. Він був розроблений в корпорації IBM шляхом модифікації більш ранішої версії, що називається LUCIFER. Пізніше з'явилися подальші його модифікації.

ГОСТ 28147-89 – стандарт симетричного шифрування, блоковий алгоритм. Повна назва − «ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования». Алгоритм, що лежить в основі ГОСТа був створений у 8-му головному управлінні КГБ (зараз структура ФСБ), ймовірно ще в 1970 рр. в рамках проектів створення програмних та апаратних реалізацій шифру для різних комп'ютерних платформ. ГОСТ спроектований на забезпечення військового рівня безпеки на 200 років вперед. В 1989 р. він був стандартизований і вперше став офіційним стандартом захисту конфіденційної інформації (специфікація шифру залишалася секретною). В 1994 р. стандарт був розсекречений, опублікований та переведений на англійську мову.

У 2010 році ГОСТ пропонують долучити до ISO 18033 у якості всесвітнього стандарту шифрування. Дуже мала кількість алгоритмів змогли стати міжнародними стандартами. ГОСТ дозволений для захисту секретної інформації без обмежень, у відповідності з тим.

1. Опис стандарту DES

В криптосистемі DES використовується блоковий принцип шифрування двійкового тексту. Довжина блоку шифрування складає 64 біт. Розмір ключа також 64 біт. При цьому кожен восьмий біт є службовим і в шифруванні участі не буре. Кожен такий біт є двійковою сумою семи попередніх і служить лише для знаходження помилок при передачі ключа по каналу зв'язку.

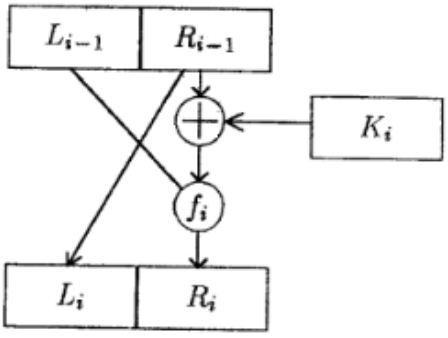
Процес крипто перетворення включає наступні три основні етапи:

1. Біти вхідного повідомлення x переставляються згідно початкової підстановки IP у відповідності таблиці:

IP	58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
	62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
	57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
	61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Це означає, що 58-й біт стає першим, 50-й – другим і т.д. Далі отриманий вектор $x_0 = IP(x)$ подається у вигляді $x_0 = L_0R_0$, де L_0 – ліва половина із 32 біт, а R_0 – права половина із 32 біт.

2. Повідомлення L_0R_0 перетворюється далі 16 разів по так званій схемі Фейстеля, показаній на мал.1.:



Мал. 9.1 Криптоперетворення Фейстеля

$L_0 = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i),\ i = 1,2,\dots 16,$ де функція f і опис ключів $K_1, K_2, \dots, K_{16}$, будуть описані детально.

3. Повідомлення $L_{16}R_{16}$ перемішується підстановкою IP^{-1}: $y = IP^{-1}(L_{16}R_{16})$ є зашифроване повідомлення.

Шифрування відбувається за схемою, наведеною на мал. 9.2. [13]

2. Функція f

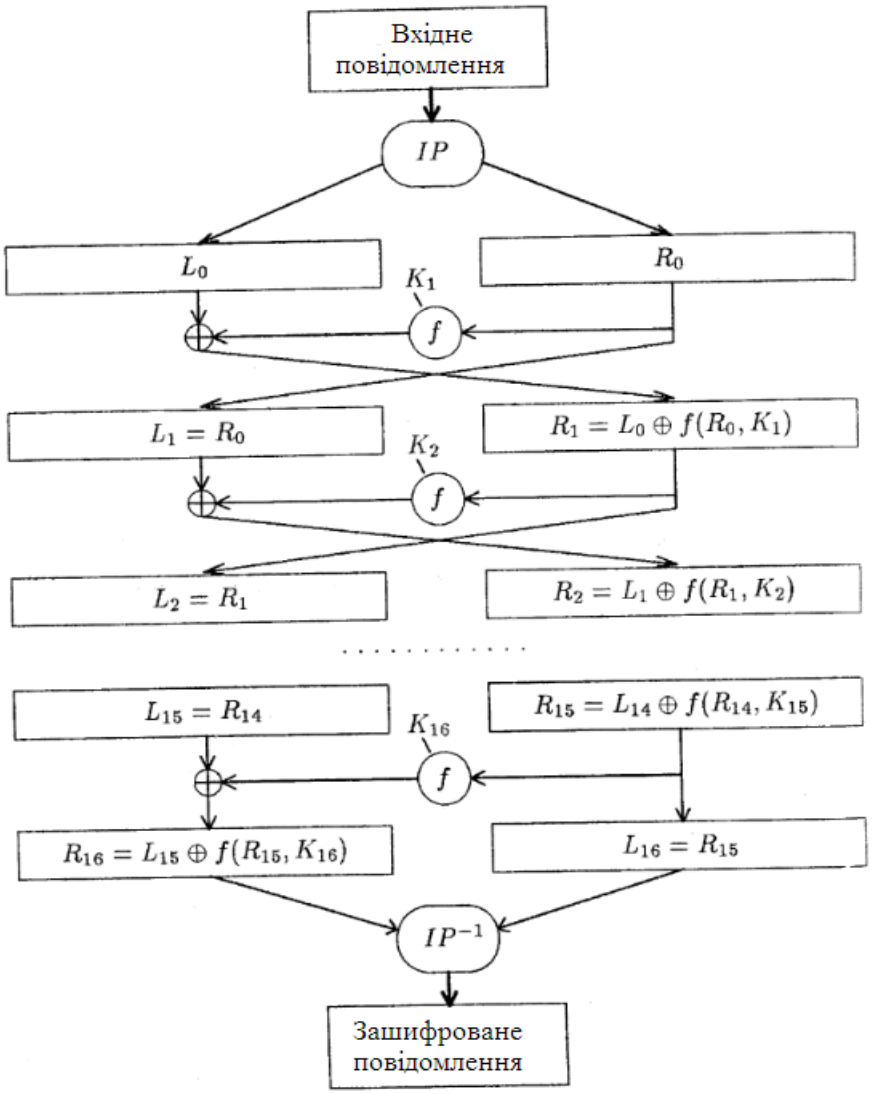
Ця функція має два аргументи *A, B*. Перший з них складається з 32 біт, а другий із 48 біт. Результат має 32 біт.

1. Перший аргумент *A*, що має 32 біт, перетворюється в 48-бітовий вектор *P(A)* шляхом перестановки з повторенням початкового вектора *A*. Ця процедура одна і та ж для всіх раундів. Вона задається таблицею:

P_1	32	1	2	3	4	5	4	5	6	7	8	9	8	9	10	11
	12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21
	22	23	24	25	24	25	26	27	28	29	28	29	30	31	30	1

2. Далі обчислюється сума $P(A) \oplus B$ і записується у вигляді конкатенації восьми 6-бітових слів: $P(A) \oplus B = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$.

3. На цьому етапі кожне слово B_i поступає на відповідний блок *S*-блок S_i. Блок S_i перетворює 6-бітовий вхід B_i в 4-бітовий вихід C_i. *S*-блок – це матриця 4 х 16 з цілими елементами в діапазоні від 0 до 16. Два перших біти слова B_i, якщо їх розглядати як двійковий запис числа, визначають номер рядка матриці *S*-блока. Чотири останніх біти визначають деякий стовпець. Тим самим знайдений деякий елемент матриці. Його двійковий запис і є виходом. [13]



Мал. 9.2 Схема криптоперетворення DES

В таблицях представлені всі 8 блоків DES.

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	13	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

4. Вихід $C = C_1 C_2 \ldots C_S$ перемішується фіксованою підстановкою P_2:

P_2																
	16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
	2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

3. Розписування ключів

1. В 64-бітовому ключі K видаляються біти 8, 16, …, 64. Біти, що лишилися, перемішуються підстановкою P_3:

P_3														
	57	49	41	33	25	17	9	1	58	50	42	34	26	18
	10	2	59	51	43	35	27	19	11	3	60	52	44	36
	53	55	47	39	31	23	15	7	62	54	46	38	30	22
	14	6	61	53	45	37	29	21	13	5	28	20	12	4

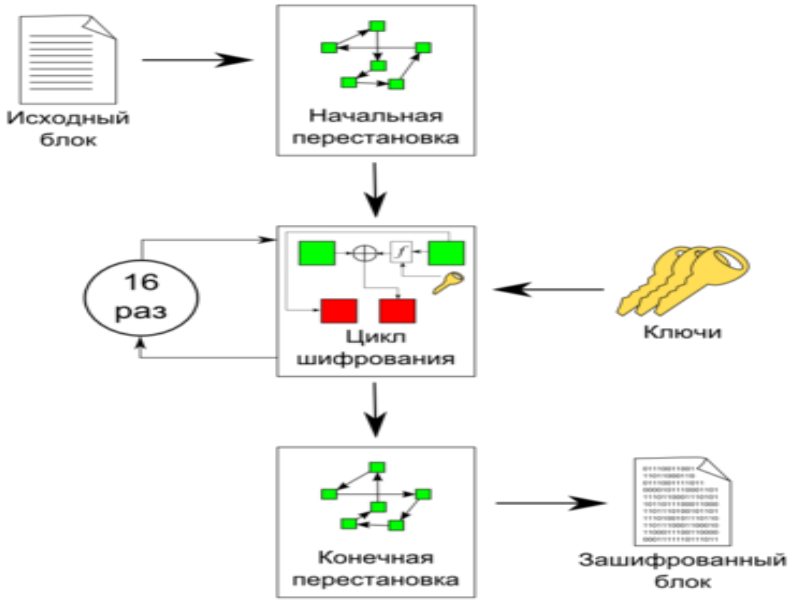
Вихід $P_3(K)$ подається у вигляді $P_3(K) = C_0 D_0$, де C_0 – ліва половина, D_0 – права половина.

2. Чергові C_i, D_i обчислюються за схемою $C_i = L_i(C_{i-1})$, $D_i = L_i(D_{i-1})$, де L_i – циклічний зсув вліво на одну позицію, якщо *i=1, 2, 3, 9, 16*. В решту випадках L_i – зсув вліво на дві позиції. [13]

3. На цьому етапі вихід перемішується підстановкою P_4:

P_4												
	14	17	11	24	1	5	3	28	15	6	21	10
	23	19	12	4	26	8	16	7	27	20	13	2
	41	52	31	37	47	55	30	40	51	45	33	48
	44	49	39	56	34	53	46	42	50	36	29	39

Схематично робота алгоритму виглядає так:

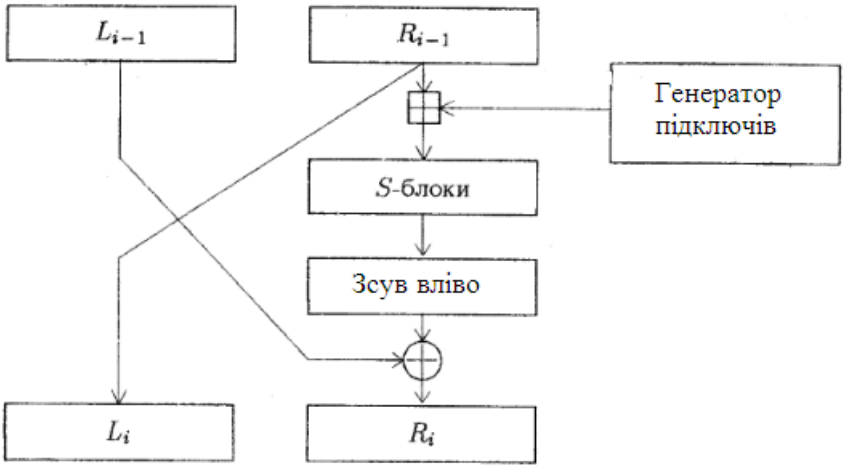


Дешифрування здійснюється тим самим алгоритмом і ключем, але в розписування ключів вносяться деякі зміни: міняють на зворотній порядок генерування ключів. Іншими словами, дешифрування відбувається зворотнім способом.

З ціллю збільшення стійкості алгоритму DES було розроблено декілька його модифікацій: double DES (2DES), triple DES (3DES), DESX, G-DES.

4. Вітчизняний аналог DES – специфікований ГОСТ 28147-89

ГОСТ є 64-бітовим 32-раундовим алгоритмом з 256-бітовим ключем. *S*-блоки, які є в алгоритмі, також можна використовувати як ключ. При шифруванні повідомлення представляється у вигляді конкатенації двох половинок *LR*. Один раунд алгоритму здійснює своє перетворення Фейстеля, мал. 3.: $L_i = R_{i-1}, R_i = L_{i-1} \oplus f\ (R_{i-1}, K_i)$.



Мал. 9.3 Один раунд ГОСТа

Відмінність від стандарту DES – лише в конструкції функції *f*. Робота алгоритму закінчується транспозицією: $LR \rightarrow RL$.

Опис функції f

Спочатку права половина R_i додається по модулю 2^{32} з підключем K_i.

Отримане 32-бітове повідомлення ділиться на вісім 4-бітових частин. Кожна з цих 4-бітових чисел перетворюється відповідним *S*-блоком в інше 4-бітове число.

Будь який *S*-блок визначається деякою 16-бітовою перестановкою на множині 16 елементів 0, 1, 2, … 15. В якості *S*-блоків можуть використовуватися різні таблиці чисел. Наприклад, в одному з банків використовуються *S*-блоки:

S_1 = (4 10 9 2 13 8 0 14 6 11 1 12 7 15 5 3),

S_2 = (14 11 4 12 6 13 15 10 2 3 8 1 0 7 5 9),

S_3 = (5 8 1 13 10 9 4 2 14 15 12 7 6 0 9 11),

S_4 = (7 13 10 1 0 8 9 15 14 4 6 12 11 2 5 3),

S_5 = (6 12 7 1 5 15 13 8 4 10 9 14 0 3 11 2),

S_6 = (4 11 10 0 7 2 1 13 3 6 8 5 9 12 15 14),

S_7 = (13 11 4 1 3 15 5 9 0 10 14 7 6 8 2 12),

S_8 = (1 15 13 0 5 7 10 4 9 2 3 14 6 11 8 12).

Після перетворення *S*-блоками отримане 32-бітове повідомлення зсувається вліво на 11 позицій.

Ключ

Початковий 256-бітовий ключ ділиться на вісім 32-бітових підключів k_1, k_2, …, k_8; вони використовуються в 32 раундах в наступному порядку: 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1.

При дешифруванні порядок використання підключів міняється на протилежний. [13]

Є дві основні модифікації алгоритму ГОСТ: GOST-H и GOSTA. За результатами аналізу був зроблений висновок про те, що GOST-H і GOSTA слабші від початкового алгоритму ГОСТ 28147-89, оскільки обидва мають класи слабких ключів.

За словами основоположника алгебраїчного криптоаналізу Ніколая Куртуа блоковий шифр ГОСТ, який найближчим часом повинен був стати міжнародним стандартом, фактично взломаний, іншими словами, знайдені дуже важливі слабкі місця. (травень 2011)

5. Порівняння шифрів ГОСТ 28147-89 і DES

Порівняння основних характеристик цих двох алгоритмів наведені в таблиці:

Параметр	ГОСТ	DES
Розмір блоку шифрування	64 біт	64 біт
Довжина ключа	256 біт	56 біт
Число раундів	32	16
Вузли заміни (S-блоки)	не фіксовані	фіксовані
Довжина ключа для одного раунду	32 біт	48 біб
Схема вироблення раундового ключа	проста	складна
Початкова та кінцева перестановка бітів	немає	є

Функція шифрування ГОСТа набагато простіша функції шифрування DES, вона не містить операцій бітових перестановок, які неефективно реалізуються на сучасних універсальних процесорах. Але не дивлячись на те, що у ГОСТа удвічі більше раундів (32 проти 16), програмна реалізація на процесорах Intel x86 більш ніж у 2 рази краща по швидкодії від реалізації DES.

На кожному раунді шифрування використовується “раундовий ключ”, в DES він 48-бітовий і виробляється за відносно слабким алгоритмом, що включає бітові перестановки і заміни за таблицею, в ГОСТі він береться як фрагмент ключа шифрування. Довжина ключа шифрування в ГОСТі 256 біт, довжина раундового – 32 біт, отримуємо, що ключ шифрування містить 256/32=8 раундових ключів. В ГОСТі 32 раунди, відповідно, кожний раундовий ключ використовується 4 рази, порядок використання раундових ключів встановлений і різний для різних режимів.

Таблиця замін в ГОСТі – аналог S-блоків DES – являє собою таблицю (матрицю) розміром 8x16, що містить числа від 0 до 15. В кожному рядку кожне з 16 чисел повинно зустрітися тільки один раз. На відміну від DES, таблиця замін в ГОСТі одна і та ж для всіх раундів і не зафіксована в стандарті, а є змінним секретним ключовим елементом.

В ГОСТі на відміну від DES, немає початкової і кінцевої бітових перестановок шифрованого блоку, які, за думкою багатьох спеціалістів, не впливають суттєво на стійкість шифру, хоча впливають (в сторону зменшення) на ефективність його реалізації.

Тема 10. ОСНОВНІ ПОНЯТТЯ АСИМЕТРИЧНОЇ КРИПТОГРАФІЇ

1. Основні поняття асиметричних криптосистем

При практичному використанні моделі Шеннона необхідність реалізації захищеного каналу для ключового обміну породжує так звану проблему безпечного розповсюдження ключів. Крім того, виникає проблема підтвердження істинності тієї чи іншої інформації.

Наприклад, це може бути необхідно при перевірці того, що ключ дійсно належить особі від імені якого він надходить у систему, оскільки існує можливість так званого нав'язування ключа.

Обидва ці завдання без використання захищеного каналу зв'язку вдалося вирішити в рамках моделі криптосистеми з відкритим ключем, запропонованої Діффі та Хеллманом у 1976 році.

Відмінність моделі Діффі-Хеллмана від моделі Шеннона в тому, що вона є асиметричною в тому сенсі, що користувачі по відношенню до секретного параметру не рівноправні. Ключ відомий повністю тільки одержувачу повідомлення і являє собою пару *(e,d)* де підключ *e* (відкритий ключ) служить ключем зашифрування, а підключ *d* (секретний, особистий ключ) служить для розшифрування. Ключ *d* відомий тільки одержувачі повідомлень, які відправники повинні шифрувати використовуючи ключ *e*. Такі криптосистеми називаються *асиметричними* або *системами з відкритими ключами*.

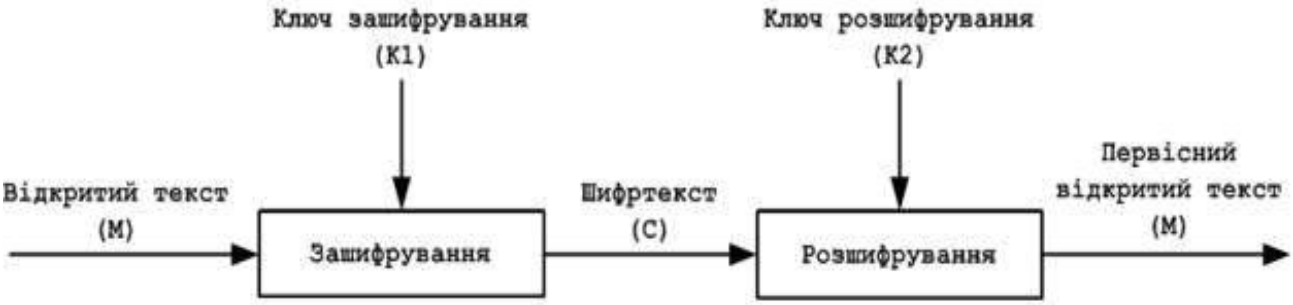
В асиметричних криптосистемах процедури прямого і зворотного криптоперетворення виконуються на різних ключах і не мають між собою очевидних і легко відсліджуваних зв'язків, що дозволяють за одним ключем визначити інший. В такій схемі знання тільки ключа зашифрування не дозволяє розшифрувати повідомлення, тому він зазвичай публікується учасником обміну для того, щоб будь-хто бажаючий міг послати йому зашифроване повідомлення.

Принцип функціонування асиметричної криптосистеми полягає у наступному:

- ✓ користувач А генерує два ключі – відкритий і секретний, і передає відкритий ключ по незахищеному каналу користувачу Б;
- ✓ користувач Б шифрує повідомлення, використовуючи відкритий ключ шифрування використовуючи відкритий ключ шифрування користувача А;
- ✓ користувач Б посилає зашифроване повідомлення користувачу А по незахищеному каналу;
- ✓ користувач А отримує зашифроване повідомлення і дешифрує його, використовуючи свій секретний ключ

Пара (відкритий ключ; секретний ключ) обчислюється за допомогою спеціальних алгоритмів, при чому жоден ключ не може бути виведений з другого.

Схема асиметричного алгоритму:



Слід зауважити, що ключі K_1 та K_2 не обов'язково мають бути різними, але в тому випадку, коли вони співпадають, втрачається багато переваг алгоритму з відкритим ключем.

Іноді повідомлення зашифровуються закритим ключем, а розшифровуються − відкритим. Від часу винайдення криптографії з відкритим ключем було запропоновано безліч криптографічних алгоритмів з відкритими ключами. Багато з них не є стійкими, а з тих які є, багато не придатних для практичної реалізації. Або вони використовують дуже великий ключ, або розмір отриманого шифротексту набагато перевищує розмір відкритого тексту.

Небагато алгоритмів є безпечними і практичними. Зазвичай ці алгоритми засновані на одній з важких проблем: деякі з них підходять тільки для

розподілу ключів, інші для шифрування, і треті корисні тільки для цифрових підписів.

Прикладами криптосистем з відкритим ключем є:

- ✓ Elgamal (названа на честь автора Ельгамаля);
- ✓ RSA (названа на честь винахідників Рона Рівеста, Аді Шаміра і Леонарда Адлмана);
- ✓ Diffie-Hellman і DSA (Digital Signature Algorithm) (винайдений Девідом Кравіцом);
- ✓ Rabin (Рабіна).

Але тільки три алгоритми добре працюють як при шифруванні так і для цифрового підпису: RSA, Elgamal, Rabin. Усі ці алгоритми повільні. Розшифровують і зашифровують данні набагато повільніше, ніж симетричні алгоритми. Зазвичай їх швидкість недостатня для шифрування великих обсягів даних.

Гібридні криптосистеми дозволяють прискорити події: для шифрування повідомлень використовується симетричний алгоритм з випадковим ключем, а для розшифрування алгоритм з відкритим ключем.

Головне досягнення асиметричного шифрування в тому, що воно дозволяє людям, що не мають існуючої домовленості про безпеку, обмінюватися секретними повідомленнями. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналі цілком відпала. [10, 12, 14]

2. RSA – криптографічна система з відкритим ключем.

Незважаючи на досить велику кількість різних асиметричних шифросистем, найпопулярніша криптосистема RSA, яка розроблена в 1977 р. і набула назву на честь Рона Рівеста, Аді Шаміра і Леонарда Ейдельмана.

Систему побудовано на факті, що добуток великих простих чисел здійснюється легко, проте розкладання на множники добутку двох таких чисел практично неможливе. Доведено (теорема Рабіна), що розкриття шифру RSA еквівалентно такому розкладу. Тому для довільної довжини ключа дають

нижню оцінку кількості операцій для розкриття шифру, а з урахуванням продуктивності сучасних комп'ютерів оцінити і необхідний для цього час.

Можливість гарантовано оцінити захищеність алгоритму RSA стала однією з причин популярності цієї криптосистеми на фоні десятків інших схем. Тому алгоритм RSA використовується в банківських комп'ютерних мережах, особливо для роботи з віддаленими клієнтами (обслуговування кредитних карток).

Сьогодні алгоритм RSA використовується в багатьох стандартах, серед яких SSL, S – HHTP, S – MIME, S/WAN, STT і PCT.

Розглянемо математичні результати, які лежать в основі цього алгоритму.

Теорема 1. (*Мала теорема Ферма, згадувалася раніше*)

Якщо p – просте число, то $x^{p-1} = 1 \ (mod\ p)$ для будь – якого x, взаємно простого з p, і $x^{p} = x \ (mod\ p)$ для будь – якого x.

Визначення. Функцією Ейлера $\varphi(n)$ називається число додатних цілих, менших від n і простих відносно n.

N	2	3	4	5	6	7	8	9	10	11	12
$\varphi(n)$	1	2	2	3	2	6	4	6	4	10	4

Теорема 2.

Якщо $n=p\cdot q$ (p і q – відмінні одне від одного прості числа), то $\varphi(n)=(p-1)(q-1)$.

Теорема 3.

Якщо $n=p\cdot q$ (p і q – відмінні один від одного прості числа), x – взаємно просте з p і q, то $x^{\varphi(n)} = 1 \pmod n$.

Наслідок.

Якщо $n=p\cdot q$, (p і q – відмінні одне від одного прості числа), e – просте відносно $\varphi(n)$, то відображення $E_{e,n}: x \to x^{e} \pmod n$ буде взаємно однозначним.

Відомий і той факт, що коли e – просте відносно $\varphi(n)$, то існує ціле d – таке, що $ed = 1 \pmod{\varphi(n)}$.

На цих математичних фактах і заснований популярний алгоритм RSA.

Нехай $n=p\cdot q$, де p і q – різні прості числа. Якщо e і d задовольняють рівняння $ed = 1 \pmod{\varphi(n)}$, то відображення $E_{e,n}$ і $E_{d,n}$ будуть інверсіями $E_{d,n}$ і $E_{e,n}$, і легко обчислюються, якщо відомі e,d,p,q. Якщо відомі e і n, а p і q невідомі, то $E_{e,n}$ є односторонньою функцією і знаходження $E_{d,n}$ за заданим n рівнозначно розкладанню n. Якщо p і q – достатньо великі прості, то розкладання n практично нездійсненне. Це і закладено в основу системи шифрування RSA.

Користувач i вибирає пару різних простих p_i і q_i та розраховує пару цілих (e_i, d_i), які є простими відносно $\varphi(n)$, де $n_i=p_iq_i$. Довідкова таблиця містить ключі $\{(e_i,n_i)\}$. Припустимо, що вихідний текст $x=(x_0,x_1,\ldots,x_{n-1})$, $x \in Z_n$, $0 \le i < n$.

Користувач i зашифровує текст при передачі його користувачу j, застосовуючи до n відображення $E_{di,ni}$: $N \rightarrow E_{di,ni}$ $n=n'$.

Користувач j проводить дешифрування n', застосувавши $E_{ei,ni}$: $N' \rightarrow E_{ei,ni}$ $n'= E_{ei,ni}E_{di,ni}n=n$.

Самоочевидно, що для того, щоб знайти інверсію $E_{di,ni}$ до $E_{ei,ni}$, потрібно знати множники $n=p_iq_i$. Час виконання найкращих з відомих алгоритмів розкладу при $n=10^{100}$ на сьогодні виходить за межі реальних технічних можливостей. [7, 9]

3. Застосування алгоритму RSA.

Приклад. Зашифруємо повідомлення CAB. Для простоти використаємо маленькі числа (на практиці застосовують значно більші).

1. Виберемо p=3 і q=11.

2. Визначимо n=3·11=33.

3. Знайдемо $(p-1)(q-1)$=20. В якості d візьмемо взаємно просте з 20, наприклад d=3.

4. Виберемо число e. В ролі такого числа виступає будь – яке число, для якого задовольняється співвідношення $(e\cdot 3) \pmod{20}$=1, наприклад 7.

5. Запишемо шифроване повідомлення як послідовність цілих чисел за допомогою відображення: A→1, B→2, C→3. Тоді повідомлення набуде вигляду (3,1,2). Зашифруємо за допомогою ключа {7,33}. ШТ1=(3^7)(mod 33)=2187(mod 33)=9,

ШТ2=(1^7) (mod 33)=1(mod 33)=1,

ШТ3=(2^7) (mod 33)=128(mod 33)=29.

6. Розшифруємо одержане зашифроване повідомлення (9,1,29) на основі закритого ключа {3,33}:

ВТ1=(9^3)(mod 33)=729(mod33)=3,

ВТ2=(1^3)(mod 33)=1(mod33)=1,

ВТ1=(29^3)(mod 33)=24389(mod33)=2.

Отже, в реальних системах алгоритм RSA реалізується в такий спосіб: кожен користувач вибирає два великих простих числа p і q та, відповідно до описаного алгоритму, вибирає два простих числа e і d. Як результат добутку перших двох чисел (p і q) встановлюється n.

Відкритий ключ утворює $\{e,n\}$, а $\{d,n\}$ – закритий (хоч можна і навпаки). Відкритий ключ публікується і доступний кожному бажаючому надіслати власнику ключа повідомлення, яке зашифроване вказаним алгоритмом. Після шифрування, повідомлення неможливо розкрити за допомогою відкритого ключа. Власник закритого ключа має можливість розшифрувати прийняте повідомлення. [7, 9]

4. Система Діффі-Хеллмана та Ель – Гамаля.

Ця система є альтернативою до RSA і при однаковому значенні ключа забезпечує таку ж криптостійкість.

На відміну від RSA метод Ель – Гамаля заснований на проблемі дискретного логарифма. Цим він подібний до алгоритму Діффі – Хеллмана. Якщо підносити число до степеня в скінченному полі досить легко, то відновити аргумент за значенням (знайти логарифм) досить складно.

Система Ель-Гамаля

Основу системи становлять параметри p і g – числа, перше з яких – просте, а друге – ціле.

Далі Аліса генерує секретний ключ a і обчислює відкритий ключ $y=g^a \bmod p$. Якщо Борис хоче надіслати Алісі повідомлення m, то він вибирає випадкове число k, менше, ніж p, і обчислює $y_1=g^k \bmod p$ та $y_2=m \oplus y^k$, де $\oplus$ означає побітове додавання за модулем 2. Потім Борис надсилає (y_1,y_2) Алісі. Аліса, одержавши зашифроване повідомлення, відновлює його: $m=(y_1^a \bmod p) \oplus y_2$.

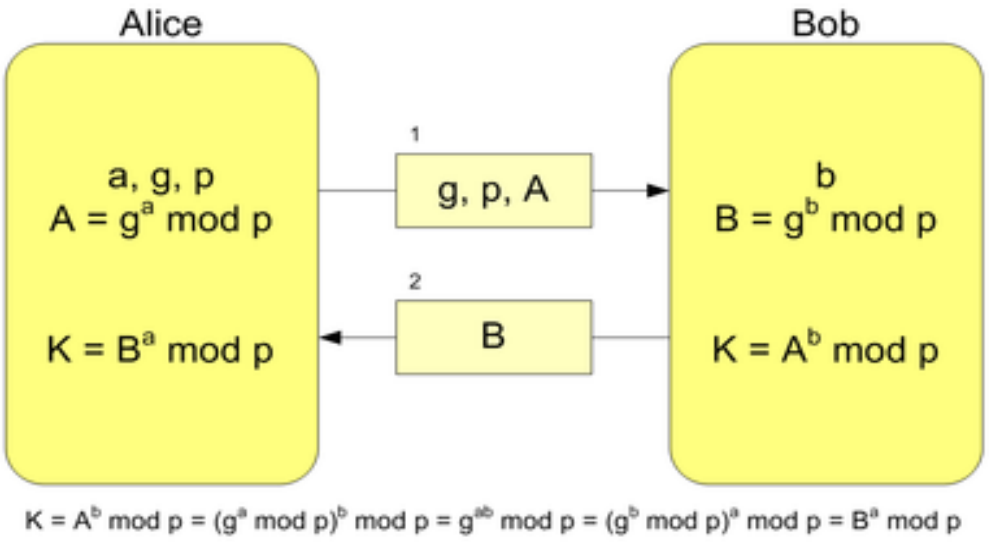
Система Діффі-Хеллмана

Схема обміну ключами Діффі-Хеллмана, винайдена в 1976 році при співробітництві Уитфилда Діффі і Мартіна Хеллмана, під сильним впливом роботи Ральфа Меркля (Ralph Merkle) про систему розповсюдження публічних ключів, стала першим практичним методом для отримання загального секретного ключа при спілкуванні через незахищений канал зв'язку. Для забезпечення стійкості, за порадою Джона Гілла (John Gill), була використана проблема дискретного логарифмування. За кілька років до цього ця ж схема була винайдена Малькольмом Вільямсоном з англійського штабу урядового зв'язку, але залишалася в секреті до 1997 року.

Опис алгоритму

Припустимо, що обом абонентам відомі деякі два числа g і p (наприклад, вони можуть бути “зашиті” в програмне забезпечення), які не є секретними і можуть бути відомі також іншим зацікавленим особам. Для того, щоб створити невідомий більш нікому секретний ключ, обидва абоненти генерують великі випадкові числа: перший абонент – число a, другий абонент – число b. Потім перший абонент обчислює значення $A = g^a mod\ p$ і пересилає його другому, а другий обчислює $B = g^b mod\ p$ і передає першому. Передбачається, що зловмисник може отримати обидва цих значення, але не модифікувати їх (тобто у нього немає можливості втрутитися в процес передачі). На другому етапі перший абонент на основі наявного в нього a і отриманого по мережі B обчислює значення $B^a mod\ p = g^{ab} mod\ p$, а другий абонент на основі наявного в нього b і отриманого по мережі A обчислює значення $A^b mod\ p = g^{ab} mod\ p$.

Неважко побачити, що в обох абонентів вийшло одне і те ж число: $K = g^{ab} mod\ p$. Його вони і можуть використовувати в якості секретного ключа, оскільки тут зловмисник зустрінеться з практично нерозв'язною (за розумний час) проблемою обчислення $g^{ab} mod\ p$ по перехоплених $g^a mod\ p$ і $g^b mod\ p$, якщо числа p, a, b вибрані достатньо великими.



Описані вище дії зображені на даному рисунку.

При роботі алгоритму, кожна сторона:

1. Генерує випадкове натуральне число a – *закритий ключ*.
2. Спільно з віддаленою стороною встановлює *відкриті параметри* p і g (зазвичай, значення p і g генеруються на одній стороні і передаються іншій), де

   p є *випадковим простим числом*

   g є *первісним коренем по модулю* p
3. Обчислює *відкритий ключ* A, використовуючи перетворення над *закритим ключем* $A = g^a\ mod\ p$.
4. Обмінюється *відкритими ключами* з віддаленою стороною.
5. Обчислює *загальний секретний ключ* K, використовуючи відкритий ключ віддаленої сторони B і свій закритий ключ a

   $K = B^a\ mod\ p$

   K виходить рівним з обох сторін, тому що:

   $B^a\ mod\ p = (g^b\ mod\ p)^a mod\ p = g^{ab} mod\ p = (g^a mod\ p)^b mod\ p = A^b mod\ p$

У практичних реалізаціях, для a і b використовуються числа порядку 10^{100} і p порядку 10^{300}. Число g не обов’язково має бути великим і зазвичай має значення в межах першого десятка.

Необхідно зазначити, що алгоритм Діффі-Хеллмана працює тільки на лініях зв'язку, надійно захищених від модифікації. Якби він був застосовний на будь-яких відкритих каналах, то давно зняв би проблему розповсюдження ключів і, можливо, замінив собою всю асиметричну криптографію. Однак, у тих випадках, коли в каналі можлива модифікація даних, з'являється можливість атаки людина посередині. Атакуючий замінює повідомлення переговорів про ключ на свої власні і, таким чином, отримує два ключі – свій для кожного з законних учасників протоколу. Далі він може перешифровувати листування між учасниками, своїм ключем для кожного, і, таким чином, ознайомитися з їх повідомленнями, залишаючись непоміченим. [2, 7, 9, 12, 13]

Тема 11. ХЕШ-ФУНКЦІЯ

1. Односторонні функції і функції з лазівками

Стійкість асиметричної криптосистеми забезпечується за рахунок особливих властивостей шифрів – перетворення, яке являє собою так звану односторонню функцію з «лазівкою». Обчислення значення такої функції (від відкритого тексту і відкритого ключа) повинно бути нескладним. У той же час, її звернення повинно бути обчислювально нереалізованим без знання секретної інформації, «лазівки», пов'язаної з секретним ключем.

Строго кажучи, не доведено, що односторонні функції існують. Однак визнано, що деякі перетворення мають властивості, близькі до властивостей односторонніх функцій. Вони широко використовуються в діючих системах криптографічного захисту інформації (ще один аргумент на користь актуальності проблеми надійності криптосистем.)

Функція $f(x) = a^x \bmod p$, при великих значеннях x і $ord_n a$, веде себе як одностороння. Обернена функція (дискретний логарифм) обчислювально нереалізовувана і завдання дискретного логарифмування є алгоритмічної проблемою.

Аналогічними властивостями володіє і степенева функція виду $g(x) = x^e \bmod n$, де $n = pq$. Для зворотної дії цієї функції досить вирішувати завдання розкладання числа n на множники, однак це завдання також є алгоритмічної проблемою.

Яким чином проблема безпечного розповсюдження ключів вирішується за допомогою асиметричних криптосистем?

Для цього кожен бажаючий передати ключ для симетричної криптосистеми своєму абоненту, перезашифровує його ключем e цього абонента (вважається, що асиметрична система створена заздалегідь і відкритий ключ опублікований). Результат шифрування передається по відкритому каналу.

Одностороння функція гарантує безпеку, тому що розшифрувати повідомлення можна лише знаючи ключ d, а його знає лише потрібний абонент.

Загальновідомо, що даний механізм все одно не є безпечним. Справа ускладнюється настільки, що на практиці виявилося необхідним вводити в глобальному масштабі систему так званих центрів сертифікації відкритих ключів.

Центр сертифікації грає роль довіреної особи, яка гарантує, що повідомлення, зашифроване даними відкритим ключем, зможе розшифрувати тільки абонент, для якого це повідомлення призначалося.

При подальшому розвитку, ідея використання односторонніх функцій в криптографії дозволила вирішити ряд проблем, пов'язаних із захистом інформації. [14]

2. Загальні поняття хеш-функцій

Часто виникають ситуації, коли одержувач повинен вміти довести достовірність повідомлення зовнішній особі. Щоб мати таку можливість, до передавальних повідомлень мають бути приписані так звані цифрові сигнатури.

Цифрова сигнатура – це рядок символів, який залежить як від індентифікатора відправника, так і від змісту повідомлення.

Використання цифрової сигнатури передбачає застосування деяких функцій шифрування:

$$S=H(k,T),$$

де S – сигнатура, k – ключ, T – вихідний текст. Функція $H(k,T)$ – хеш – функція.

Хешування – перетворення вхідного масиву даних будь-якої довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються *хеш-функціями* або *функціями згортки*, а їх результати називають *хешем, хеш-кодом* або *дайджестом повідомлення.*

По іншому, *алгоритм хешування* – це послідовність математичних перетворень, в результаті яких з деякої двійкової послідовності змінної

довжини отримується унікальна двійкова послідовність фіксованої довжини. Функція, що реалізовує даний алгоритм називається *хеш-функцією.*

Хешування застосовується для порівняння даних: якщо в двох масивах хеш-коди різні, то масиви гарантовано відрізняються; якщо однакові – масиви, швидше всього, однакові. В загальному випадку однозначної відповідності між вихідними даними та хеш-кодом немає, тому що кількість значень хеш-функцій менше, ніж варіантів вхідного масиву; існує множина масивів, що дають однакові хеш-коди – так звані *колізії*. Ймовірність виникнення колізій грає важливу роль в оцінці якості хеш-функцій.

Існує багато алгоритмів хешування з різними характеристиками (розрядність, обчислювальна складність, криптостійкість і т.д.). Вибір тої чи іншої хеш-функції визначається специфікою вирішуваної задачі. Найпростішим прикладом хеш-функції є контрольна сума.

Для того, щоб перетворення даних можна було назвати хеш-функцією, воно повинно одночасно задовольняти наступні умови:

- обробляти послідовність різної довжини;
- видавати у якості результату бітову послідовність фіксованої довжини;
- достатньо просто обчислюватися при будь-якій вхідній інформації;
- володіти властивістю незворотності, коли для отриманого результату неможливо отримати вхідне значення;
- бути однозначним (тобто для різних вхідних послідовностей не повинно бути побудовано однакові хеш-образи).

З криптографічної точки зору, останні дві властивості хеш-функції найбільш важливі, так як забезпечують неспівпадання хешованих форм, наприклад паролів різних користувачів і ускладнюють задачу розкриття цих паролів. [11, 12, 13, 14]

3. Вимоги до хеш-функцій

При практичному використанні хеш-функцій повинні виконуватися такі вимоги:

- алгоритм повинен володіти високою швидкістю обробки інформації (це особливо актуально для банківських операцій, де необхідна особлива оперативність обробки інформації);
- хеш-функція повинна бути стійкою до атаки методом “грубої сили”;
- програмна реалізація хеш-функції повинна бути оптимізована під використання на сучасній апаратно-програмній базі.

Ці вимоги повинен задовольняти як сам алгоритм вироблення хеш-функції, так і хеш-функція.

В сучасних умовах алгоритмічне підвищення швидкості вироблення хеш-значення може бути досягнуто за рахунок застосування простого перетворення, яке переводить одне повідомлення в інше за допомогою елементарної операції, наприклад видалення будь-якого блоку повідомлення. Подібними перетвореннями можна також описати залежність між двома повідомленнями, що практично не відрізняються одне від одного. Даний тип повідомлення дуже часто зустрічається у банківських справах, наприклад з ціллю заповнення бланків платіжних доручень. Звідси слідує, що для збільшення швидкості обробки необхідно, щоб алгоритм вироблення хеш-значення включав у себе також алгоритм обчислення хеш-значення одного повідомлення із хеш-значення другого повідомлення, яке отримується із початкового з допомогою елементарного перетворення.

4. Криптографічні хеш-функції. Поділ хеш-функцій

Серед багатьох існуючих хеш-функцій прийнято виділяти криптографічно стійкі, які застосовуються в криптографії. Для того, щоб хеш-функція H вважалася криптографічно стійкою, вона повинна задовольняти три основні вимоги, на яких основана більшість застосувань хеш-функції у криптографії:

- *незворотність:* для заданого значення хеш-функції M повинно бути обчислювально неможливо знайти блок даних X, для якого $H(X)=M$.

- *стійкість до колізій першого роду:* для заданого повідомлення M повинно бути обчислювально неможливо підібрати інше повідомлення *N*, для якого *H*(*N*) = *H*(*M*).
- *стійкість до колізій другого роду:* повинно бути обчислювально неможливо підібрати пару повідомлень *(M, M')*, які мають однаковий хеш-код.

Дані вимоги не є незалежними:

- зворотна функція нестійка до колізій першого і другого роду;
- функція, нестійка до колізій першого роду, нестійка до колізій другого роду; зворотне не вірно.

Слід відмітити, що не доведено існування незворотних хеш-функцій, для яких обчислення якого-небудь прообразу заданого значення хеш-функції теоретично неможливо. Зазвичай знаходження зворотного значення є лиш обчислювально складною задачею.

Для криптографічних хеш-функцій також важливо, щоб при найменшій зміні аргемента значення функції змінювалося якнайбільше. Значення хешу не повинно давати витоку інформації навіть про окремі біти аргументу. Ця вимога є основою криптостійкості алгоритмів хешування.

Основний принцип проектування хеш-функцій полягає в тому, що її значення повинні створювати лавинний ефект. Іншими словами, невелика зміна в аргументі хеш-функції повинна дуже впливати на її значення. Це забезпечує додаткову стійкість.

Щоб значення хеш-функції, що застосовується в системах з низьким рівнем стійкості, були вільні від повторень, їх довжина повинна бути приблизно рівна 128 бітам, але краще надавати перевагу значенню в 160 біт.

Хеш-функції	
Хеш-функції загального призначення	Adler-32 • CRC • FNV • Murmur2 • PJW-32 • TTH • Jenkins hash
Криптографічні хеш-функції	HAVAL • Keccak • LM-хеш • MD2 •

	MD4 • MD5 • MD6 • N-Hash • RIPEMD-128 • RIPEMD-160 • RIPEMD-256 • RIPEMD-320 • SHA-1 • SHA-2 • LanMan • NT LanMan • MySQL 3.23 • MySQL SHA1, Cisco PIX • Skein • Snefru • Tiger • Whirlpool • ГОСТ Р 34.11-94

Крім зазначеного поділу хеш-функцій, що у таблиці, розрізняють також ключові функції хешування та без ключові.

За внутрішнім перетворенням, що використовується у хеш-функції, їх поділяють на:

- функції, що використовують бітові логічні перетворення;
- функції, що використовують блочні симетричні шифри;
- функції, що використовують перетворення в групах, полях, кільцях з цілочисловим або поліноміальним базисом;
- функції, що використовують матричні перетворення. [11, 12, 13, 14]

5. Застосування хеш-функцій. Способи взлому

З хешуванням ми зустрічаємося на кожному кроці: при роботі з браузером (список web-посилань), текстовим редактором чи перекладачем (словники), мовами скриптів (Perl, PHP, Python та ін.), компілятором (таблиця символів).

Хеш-функції також використовуються у деяких структурах даних – хеш-таблицях, фільтрах Блума і декартових деревах. Вимоги до хеш-функцій у цьому випадку інші:

- хороша змішуваність даних;
- швидкий алгоритм обчислення.

Перевірка даних

В загальному випадку це застосування можна описати як перевірка деякої інформації на ідентичність оригіналу, без використання оригіналу. Для звірення

використовується хеш-значення інформації, що перевіряється. Розрізняють два основні напрями цього застосування:

Перевірка даних на помилки

- наприклад, контрольна сума може бути передана по каналу зв'язку разом з основним текстом. На пункті прийому, контрольна сума може бути розрахована заново і її можна порівняти з переданим значенням. Якщо буде виявлено розходження, це означає, що при передачі виникли спотворення і можна запросити повторно;
- бітовим аналогом хешування в даному випадку може слугувати спосіб, коли при переїздах в пам'яті тримають кількість місць багажу. Тоді для перевірки не потрібно згадувати про кожен чемодан, а досить їх порахувати. Співпадання буде означати, що жоден чемодан не загубився. Тобто, кількість місць багажу є його хеш-кодом.

Пришвидшення пошуку даних

- наприклад, при записі текстових полів у базі даних може розраховуватися їх хеш-код і дані можуть поміщатися в розділ, що відповідає цьому хеш-коду. Тоді при пошуку даних потрібно буде спочатку обчислити хеш-код тексту і зразу стане відомо, в якому розділі їх потрібно шукати, тобто, шукати не по всій базі, а тільки по одному розділу (помітно пришвидшує пошук);
- бітовим аналогом хешування в даному випадку може бути розміщення слів у словнику по алфавіту. Перша буква слова є його хеш-кодом, і при пошуку ми переглядаємо не весь словник, а тільки потрібну букву.

Також хеш-функції використовуються при обробці інформації для створення цифрового підпису.

Оскільки, застосування хеш-функцій набуло розмаху, то почали з'являтися способи взлому хеш-кодів. Єдиним відомим вразливим їх місцем є колізії, тому пошуком колізій і займаються програми для взлому хеш-кодів.

Перший варіант – це простий перебір, при якому виконується проходження по всіх можливих варіантах. Але для складних варіантів це займає дуже багато часу, тому такий спосіб вирішили допрацювати.

Тепер для розкриття паролів, перетворених за допомогою хеш-функції, використовують райдужні таблиці.

Райдужна таблиця (від англ. rainbow table) – спеціальний варіант таблиць пошуку (lookup table), що використовує механізм зменшення пам'яті, яка займається, за рахунок збільшення часу пошуку. Але таблиці можуть взломувати тільки ту хеш-функцію, для якої вони створювались.

Вперше цей метод був використаний у програмі Ophcrack для взлому хешів LanMan, що використовуються у Microsoft Windows. Пізніше була розроблена більш вдосконалена програма RainbowCrack, яка може взломувати хеші за деякими алгоритмами (LanMan, NT LanMan, MD5, MD4, MD2, SHA1, MySQL 3.23, MySQL SHA1, Cisco PIX, RIPEMD-160 та ін.). Необхідність у великих таблицях для взлому складних хешів привела до появи проектів розподіленого обчислення радужних таблиць та online-взломів хешів, оскільки обчислення таблиць такого розміру на одній машині не можливе. Сучасні програми містять порядку 1 Тб таблиць для кожного алгоритму.

На даний момент йде розробка програми The UDC, яка дозволяє будувати таблиці не по наборі символів, а по наборі словників, що дозволяє розкривати більш довгі паролі, ніж RainbowCrac.

Для захисту від райдужних таблиць використовують метод, який робить їх неефективними. Для цього використовують хеш-функції, які включають сіль (salt). Тому для розкриття пароля, взломщику необхідні таблиці для всіх значень солі. Таким чином, збільшується довжина і складність пароля та частково попереджається його розкриття. [11, 12, 13, 14]

6. Генератори превдовипадкових чисел

Призначені для отримання числових послідовностей у яких розподіл вибірок елементів поводить себе як аналогічні вибірки із сукупності з рівноймовірним і незалежним розподілом ймовірностей. Такі послідовності отримуються за допомогою математичних алгоритмів зі скінченним числом параметрів. Тому не кожний спосіб вибору елементів числової послідовності дає сукупність чисел з бажаними характеристиками. При побудові генераторів

псевдовипадкових чисел висуваються такі необхідні вимоги до вибору елементів для яких статистичні властивості відповідають властивостям рівноймовірності і незалежності. Тобто програмний генератор псевдовипадкових чисел має задовольняти наступним вимогам:

- період гами має бути досить великим;
- гама має бути практично непередбачуваною;
- гама має бути відтворюваною

Період гами – це та кількість псевдовипадкових чисел у послідовності, після якої вони починають повторюватися. Чим більший період гами, тим для довших відкритих текстів її можна застосовувати. Чим менший період гами, тим легше передбачувати числа в ній і зламувати ключі та шифри. Довжина періоду гами залежить від вибраного алгоритму генерування послідовності псевдовипадкових чисел.

Непередбачуваність гами означає неможливість передбачити наступне число гами, навіть тоді, коли відомі тип генератора і попередній фрагмент гами. Загальних методів визначення рівня непередбачуваності гами не існує. Але вважається, що рівень непередбачуваності буде достатнім, якщо гама має дуже великий період. Крім того, різні комбінації сусідніх чисел гами мають бути рівномірно розташовані вздовж неї.

Відтворюваність гами означає наявність можливості отримати ту ж саму послідовність псевдовипадкових чисел. Це потрібно для того, щоб можна було дешифрувати криптограму, отриману шляхом застосування гами шифру. Для цього застосовують такі програмні генератори псевдовипадкових чисел, в яких та чи інша їх послідовність залежить від, так званого, ключа генератора. Це означає, що для різних значень ключа генератора мають генеруватись різні послідовності. Але при одному й тому ж значені ключа має генеруватись одна й таж сама послідовність. [8]

В криптографії застосовуються так званні *криптографічно стійкі датчики (КСД)* – так називають генератори що використовують секретні параметри. Для таких генераторів потрібна властивість непедбачуваності

(відрізок вихідної послідовності відносно великої довжини не може бути продовжено як вперед так і назад без знака ключа.

Приклади КСД:

1. Генератор рекомендований стандартом *ANSI X9.17*, що використовується частково при виконанні платіжних операцій.

2. В українському стандарті на цифровий підпис *ДСТУ 4145-2002* генератор випадкових двійкових послідовностей побудований за схемою ПСЧ *ANSI X9.17* з використанням криптоалгоритму *ГОСТ 28147-89*.

3. Генератор BBS *(Algorithm Blum-Blum-Shub)* – генератор псевдовипадкових чисел, запропонований в 1986 році Ленор Блюмом, Мануелом Блюмом, Майклом Шубом. Праметром генератора є просте число $n=pq$, де співмножники великі псевдовипадкові прості числа обнакового розміру, при чому кожний співмножник порівнюваний з $3 \bmod 4$.

Тема 12. ПРИКЛАДИ ХЕШ-ФУНКЦІЙ

1. Основні ідеї хеш-функцій MD2, MD4, MD5 та MD6

Три алгоритми серії MD розроблені Рівестом (Масачусетський Технологічний інститут) у 1989, 1990, 1991 роках відповідно. Всі вони перетворюють текст будь – якої довжини в 128 – бітну сигнатуру.

Алгоритм MD2 передбачає:

- доповнення тексту до довжини, кратної 128 бітам;
- визначення 16 – бітної контрольної суми (старші розряди відкидаються);
- додавання контрольної суми до тексту;
- повторне визначення контрольної суми.

Алгоритм MD4 передбачає:

- доповнення тексту до довжини, рівної 448 біт за модулем 512;
- додавання довжини тексту в 64 – бітному вираженні;
- 512 – бітні блоки піддають процедурі Damgard – Merkle, для чого кожен блок бере участь у трьох різних циклах.

В алгоритмі MD4 доволі швидко були знайдені “дірки”, тому він був заміненим алгоритмом MD5, в якому кожний блок бере участь не в трьох, а в чотирьох різних циклах.

Після того, як MD5 був взломаний, на його заміну запропонували алгоритм MD6.

MD6 – алгоритм хешування з тої самої серії MD, розроблений професором Рівестом. Використовується для перевірки цілісності і, частково, для перевірки справжності опублікованого повідомлення. Вперше був представлений на конференції Crypto 2008 в якості кандидата на стандарт. Однак, пізніше в 2009 році на цій же конференції Рівест заявив, що MD6 ще не готова хеш-функція, щоб стати стандартом. На офіційному сайті MD6 він заявив, що, хоча, формально, заявка не відізвана, в алгоритмі ще залишаються проблеми зі швидкістю і нездатністю забезпечити безпеку у версії із зменшеною кількістю раундів. В результаті MD6 не пішов на друге коло змагання. Раніше в грудні 2008, дослідник з Fortify Software відкрив помилку,

пов’язану з переповненням буферу в оригінальній реалізації MD6. 19 лютого 2009 року професор Рівест опублікував дані про помилку, а також представив виправлені реалізації.

В алгоритмі хеш-функції використані оригінальні ідеї. За один прохід обробляється 512 байт. Крім традиційного безлючового режиму, застосовується хешування з ключем 512-біт. Не лінійність функції основана на використанні простих операцій: xor, додавання і зміщення. Кількість раундів: r = 40 + (d / 4), що не традиційно багато. [2, 5]

2. Алгоритм обчислення хеш-функції MD5

Алгоритм MD5 (Message Digest) – це алгоритм хешування, розроблений у 1991, опублікований в 1992 році.

В якості вхідних даних береться повідомлення (потік даних) довільної довжини, і обчислюється 128 – розрядний хеш-код (сигнатура, дайджест). Припускається, що не існує двох повідомлень, які мають однакові сигнатури, або не можливо створити повідомлення з наперед заданою сигнатурою. Цей алгоритм застосовується в криптографії при формуванні цифрових підписів та генерації ключів шифрування. По суті, алгоритм хешування MD5 – це спосіб перевірки цілісності даних, набагато надійніший ніж контрольна сума або інші методи.

Алгоритм під час розробки був оптимізований для 32 – розрядних систем і не потребує великих об’ємів пам’яті. MD5 є дещо повільніший, ніж MD4, але більш стійкий до криптографічних атак.

Опис алгоритму

Перед тим, як розглянути детально сам алгоритм, визначимо основні терміни та поняття, що ним використовуються. Під терміном “слово” будемо розуміти кількість інформації, яка представлена 32-ома розрядами (бітами), а під терміном “байт” – 8 розрядів (біт). Послідовність біт як послідовність байт, де кожна група із 8 – ми розрядів є окремим байтом, причому старший розряд байта іде першим. Аналогічно представляється послідовність байт, тільки молодший байт іде першим.

В якості вхідного потоку будемо мати потік даних N розрядів. N - невід'ємне ціле число (можливо 0), не обов'язково кратне 8.

На рис. 12.1 зображена схема логіки виконання MD5.

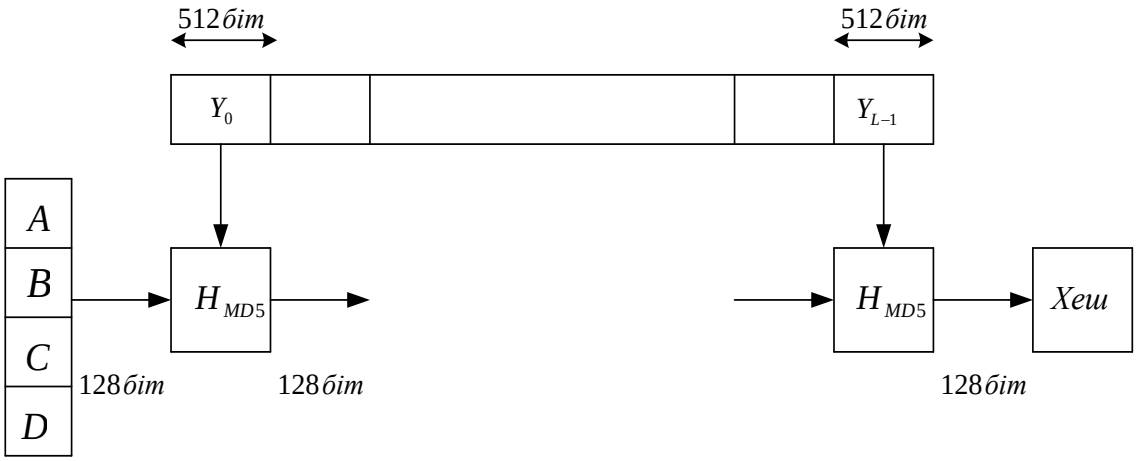


Рис. 12.1 Логіка виконання MD5.

Для обчислення MD5 хеш – функції необхідно виконати наступні 5 кроків:

Крок 1: Вирівнювання потоку

Повідомлення доповнюється таким чином, щоб його довжина була рівною 448 по модулю 512 (довжина = 448 mod 512). Це означає, що довжина доданого повідомлення на 64 розряди меньша ніж число кратне 512. Додавання розрядів проводиться завжди, навіть тоді, коли повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 розрядів, то воно доповнюється 512 розрядами до 960 розрядів. Отже число доданих розрядів знаходиться в межах від 1 до 512. Вирівнювання відбувається наступним чином: потік доповнюється одним бітом '1', а за ним біти '0' до тих пір, поки довжина потоку не буде рівною 448 по модулю 512.

Крок 2: Додавання довжини

64 – розрядне представлення довжини вихідного (до добавлення) повідомлення в бітах приєднується до результату першого кроку. Якщо початкова довжина більша ніж 2^{64}, то використовуються молодші 64 розряди. Іншими словами поле містить довжину вихідного повідомлення по модулю 2^{64}.

В результаті перших двох кроків утворюється повідомлення, довжина якого кратна 512 розрядам. Це розширене повідомлення представляється, як послідовність 512 – бітних блоків $Y_0, Y_1, ..., Y_{L-1}$, при цьому загальна довжина розширеного повідомлення рівна $L * 512$ розрядам. Таким чином довжина отриманого розширеного повідомлення кратна шістнадцяти 32 – бітним словам.

повідомлення	*доповнення від* 1 *до* 448	*довжина вихідного повідомлення*

Рис. 12.2 Структура розширеного повідомлення.

Крок 3: Ініціалізація MD-буфера

Для збереження проміжних і кінцевих результатів хеш-функції використовується 128-розрядний буфер. Він представляється, як чотири 32 – розрядні регістри (A, B, C, D). В якості ініціалізуючих значень використовуються наступні шістнадцяткові числа:

$$A = 01234567$$
$$B = 89ABCDEF$$
$$C = FEDCBA98$$
$$D = 76543210$$

Крок 4: Обробка послідовності 512 – розрядних (по 16 слів) блоків

Основою алгоритму є блок, який складається із чотирьох циклів. Він позначається як H_{MD5}. Чотири цикли мають подібну структуру, але для кожного циклу застосовується своя елементарна логічна функція f_F, f_G, f_H і f_I відповідно.

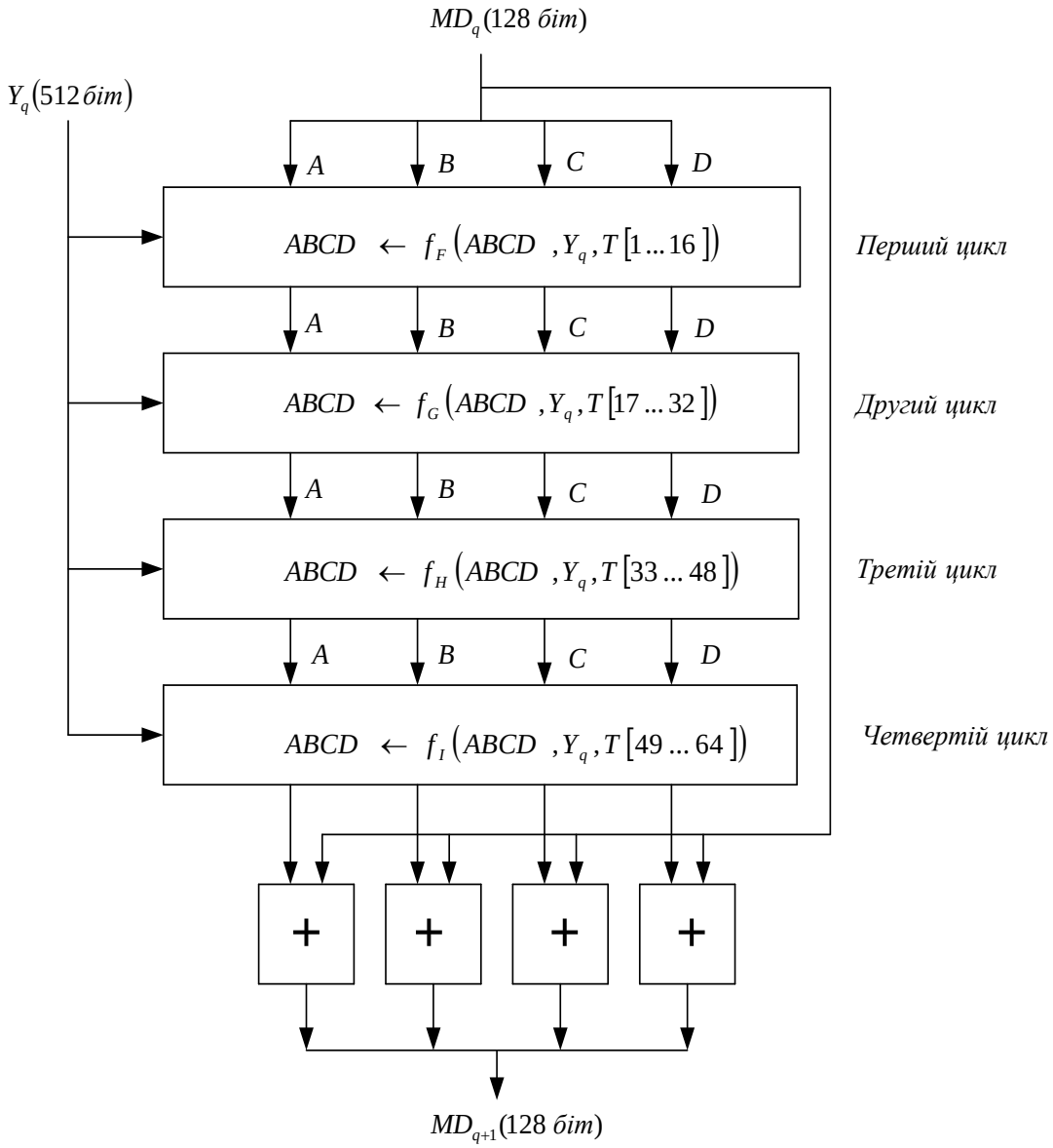


Рис. 12.3 Обробка чергового 512-розрядного блоку.

Кожний цикл отримує в якості вхідних даних поточний 512 – розрядний блок Y_q та 128 – розрядне значення буфера $ABCD$, що є проміжним значенням хеш-функції, та змінює вмістиме цього буфера. Кожний цикл використовує четверту частину 64 – елементної таблиці $T[1...64]$, побудованої на основі функції sin, $T[i] = \text{int}(4294967296 * abs(\sin(i)))$, де int( ) - ціла частина.

Наприклад: $T[1] = \text{int}(4294967296 * abs(\sin(i))) = \text{int}(3614090360{,}282...) = 3614090360$, i - ий елемент T, який позначається $T[i]$, приймає значення рівне цілій частині від $2^{32} * abs(\sin(i))$, число i задається в радіанах. Оскільки функція $abs(\sin(i))$ приймає значення, які знаходяться в проміжку від 0 до 1, то кожний елемент T є цілим, яке можна представити 32 розрядами. Для отримання MD_{q+1} вихід

чотирьох циклів додається по модулю 2^{32} з MD_q. Додавання виконується незалежно для кожного з чотирьох слів в буфері.

Крок 5: Вихід

Після обробки всіх L, 512 - розрядних блоків виходом L-ої стадії є 128 – розрядний дайджест повідомлення.

Розглянемо детальніше логіку кожного із чотирьох циклів обробки одного 512 – розрядного блоку. Цикл обробки складається із 16 кроків, які оперують з буфером $ABCD$.

Кожний крок можна подати у вигляді:

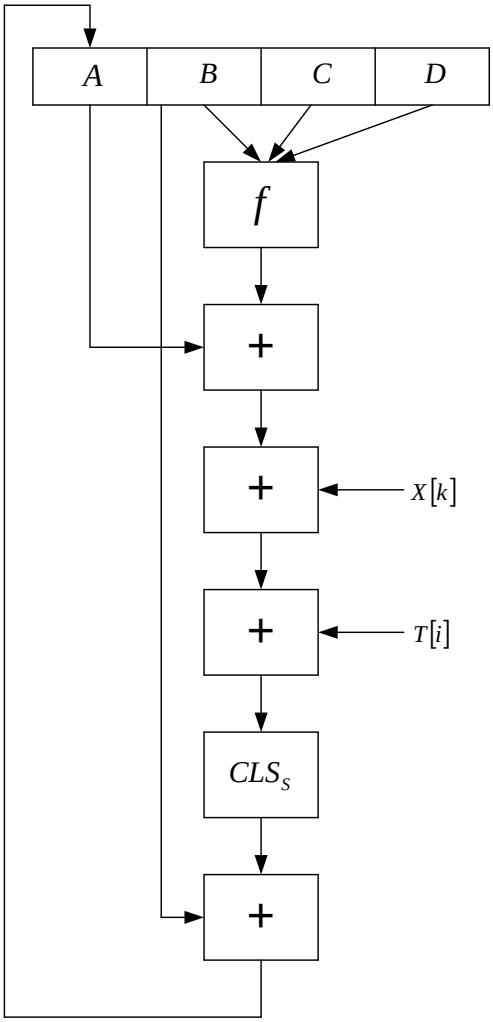


Рис. 12.4 Логіка виконання окремого кроку.

$$A \leftarrow B + CLS_s \ \left(A + f\left(B,C,D\right) + X[k] + T[i]\right),$$

де

A,B,C,D - чотири слова буфера; після виконання кожного окремого кроку відбувається циклічний зсув вліво на одне слово.
f - одна з елементарних функцій f_F, f_G, f_H, f_I.
CLS_s - циклічний зсув вліво на s розрядів 32-розрядного аргументу.
$X[k]$ - k-те 32-розрядне слово в q-му 512-розрядному блоці повідомлення.
$T[i]$ - i-те 32-розрядне слово в матриці T.
$+$ - операція додавання по модулю 2^{32}.

В кожному з чотирьох циклів алгоритму застосовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-розрядних слова на вході і на виході формує одне 32-розрядне слово. Елементарні функції мають наступний вигляд:

$$f_F = (B \& C) \vee (not\ B \& D)$$
$$f_G = (B \& D) \vee (C \& not\ D)$$
$$f_H = B \oplus C \oplus D$$
$$f_I = C \oplus (B \& not\ D)$$

В цих формулах застосовуються такі логічні операції: логічне множення (&, порозрядне І), логічне додавання ($\vee$, порозрядне АБО), додавання по модулю 2 ($\oplus$, порозрядне виключне АБО) та інверсія (not, побітове заперечення).

Масив з 32-ох розрядних слів $X[0...15]$ приймає значення поточного 512 – розрядного блоку Кожний цикл виконується 16 раз. Оскільки кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожний блок вхідного повідомлення обробляється по схемі, яка зображена на рис.4, 64 рази.

Якщо подати вхідний 512 – розрядний блок у вигляді шістнадцяти 32-розрядних слів, то кожне вхідне 32-розрядне слово використовується чотири рази, по одному разу в кожному циклі, і кожний елемент таблиці T, який складається з 64-ьох 32-розрядних слів використовується тільки один раз. Після кожного кроку циклу відбувається циклічний зсув вліво чотирьох слів A,B,C і D. На кожному кроці змінюється тільки одне з чотирьох слів буфера $ABCD$. Відповідно кожне слово буфера змінюється 16 раз, 17-ий раз в кінці для отримання остаточного виходу поточного блоку. Результат обчислення (хеш)

представлений за допомогою чотирьох 32-розрядних слів - A, B, C, D (молодші записуються в A, старші в D).

Приклад псевдокоду алгоритму обчислення MD5 хеш функції наведено нижче.

```
    // обробка вхідного потоку даних блоками по 16 слів
    for i = 0 to N/16 - 1 do
{
  // копіювання i-го блоку в X
  for j = 0 to 15 do
    X[j] = M[i * 16 + j]
  // Збереження  значень A, B, C, D
  AA = A
  BB = B
  CC = C
  DD = D
  // цикл 1
  // нехай [abcd k s i] означають операцію
  //     a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s)
  // виконати 16 наступних операцій
  [ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
  [ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
  [ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
  [ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]
  // цикл 2
  // нехай [abcd k s i] означають операцію
  //     a = b + ((a + G(b, c, d) + X[k] + T[i]) <<< s)
  // виконати 16 наступних операцій
  [ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
  [ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
  [ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
  [ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]
```

```
    // цикл 3
    // нехай [abcd k s i] означають операцію
    //     a = b + ((a + H(b, c, d) + X[k] + T[i]) <<< s)
    // виконати 16 наступних операцій
    [ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
    [ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
    [ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
    [ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]
    // цикл 4
    // нехай [abcd k s i] означають операцію
    //     a = b + ((a + I(b, c, d) + X[k] + T[i]) <<< s)
    // виконати 16 наступних операцій
    [ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
    [ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
    [ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
    [ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

    A += AA
    B += BB
    C += CC
    D += DD
} [1]
```

3. Американський стандарт хеш-функції (SHS або SHA)

Американський стандарт хеш-функції SHS (Secure Hash Standatr або SHA Secure Hash Algorithm) розроблений NIST (National Institute of Standard and Technology) на основі алгоритму MD4 та прийнятий у 1993 році. Призначений для використання сумісно з цифровим підписом, визначений у стандарті DSS. При введені повідомлення M, алгоритм виробляє 160-бітове вихідне повідомлення, що називається згорткою (Message Digest), яке і використовується при створення цифрового підпису.

Розглянемо роботу алгоритму детальніше.

Спочатку вхідне повідомлення доповнюється так, щоб його довжина стала кратна 512 бітам. При цьому повідомлення доповнюється навіть тоді, коли його довжина вже кратна вказаній. Процес відбувається наступним чином: добавляється одиниця, потім стільки нулів, скільки необхідно для отримання повідомлення, довжина якого на 64 біта менша, ніж кратна 512, і потім добавляється 64-бітове представлення довжини вхідного повідомлення.

На наступному кроці ініціалізуються п’ять 32-бітових змінних такими шістнадцятковими константами:

A=67453201

B=EFCDAB89

C=98BADCFE

D=10325476

E=C3D2E1F0

Ці змінні копіюються у нові змінні *a*, *b*, *c*, *d*, *e* – відповідно.

Головний цикл може бути досить просто описаний на псевдокоді таким чином:

For (t=0; t<80; t++) {

temp=(a<<<5)+f_t(b,c,d)+e+W_t+K_t;

e=d; d=c; c=b<<<30; b=a; a=temp;

}

Де <<< - операція циклічного зміщення вліво;

K_t – шістнадцяткові константи, визначаються за наступними формулами:

$$K_t = \begin{cases} 5A827999, & t = 0..19 \\ 6ED9EBA1, & t = 20..39 \\ 8F1BBCDC, & t = 40..59 \\ CA62C1D6, & t = 60..79 \end{cases},$$

функції $f_t(x,y,z)$ задаються виразами:

$$f_t(x,y,z) = \begin{cases} X \wedge Y \vee \neg X \wedge Z, & t = 0..19 \\ X \oplus Y \oplus Z, & t = 20..39,60..79 \\ X \wedge Y \vee X \wedge Z \vee Y \wedge Z, & t = 40..59 \end{cases},$$

значення W_t отримуються із 32-бітових під блоків 512 бітового блоку розширеного повідомлення за правилом:

$$W_t = \begin{cases} M_t, & t = 0..19 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, & t = 16..79 \end{cases}.$$

Після закінчення головного циклу значення *a, b, c, d, e* додаються з початковими *A, B, C, D, E* відповідно і здійснюється перехід до обробки наступного 512-бітового блоку розширеного повідомлення. Вихідні значення хеш-функції – це конкатенація значень *A, B, C, D, E.*[13]

Розроблені також модифікації алгоритму SHA: SHA-256, SHA-512 та SHA-384.

(Хеш-функцію ГОСТ Р 34.11-94 розглянути самостійно [13])

Тема 13. ІДЕНТИФІКАЦІЯ ТА АУТЕНТИФІКАЦІЯ ОБ'ЄКТА

1. Ідентифікація, аутентифікація та авторизація об'єкта.

З кожним об'єктом, який приймає участь у функціонуванні комп'ютерної системи (КС) обов'язково пов'язуються певні дані, які його якимось чином позначають та однозначно характеризують (ідентифікують). Ці дані, які можуть мати вигляд числа, рядка тексту або алгоритму, називаються ідентифікаційним даними об'єкту. Якщо ідентифікаційні дані зареєстровані в КС, то відповідний об'єкт вважається законним (легальним). Об'єкти відносяться до незаконних (нелегальних), якщо їх ідентифікаційні дані в КС не зареєстровані.

Коли деякий об'єкт робить спробу увійти в КС, щоб прийняти участь у її функціонуванні, система захисту КС виконує три захисні операції:

1) ідентифікація об'єкта,

2) аутентифікація об'єкта,

3) авторизація об'єкта.

Ідентифікація об'єкта. Ця операція виконується системою захисту КС в першу чергу. Перш ніж отримати доступ до КС, об'єкт повинен назвати (ідентифікувати) себе. Для цього у загальному випадку КС запитує у об'єкта його ім'я та його ідентифікаційний номер (ідентифікатор). Ім'я об'єкта використовується в КС з метою ведення діалогу з об'єктом (імена різних об'єктів можуть співпадати). Ідентифікатор об'єкта є унікальним і використовується як позначення цього об'єкта. Комп'ютерна система перевіряє, чи є указаний ідентифікатор зареєстрованим. Якщо це так, то ідентифікація вважається успішною, і КС ставить у відповідність даному об'єкту всі його ідентифікаційні дані у повному обсязі. Після цього КС переходить до виконання наступних захисних операцій, пов'язаних з цим об'єктом. Якщо ж об'єкт виявляється нелегальним, то в залежності від політики захисту в КС даному об'єкту може бути відмовлено у доступі або запропоновано перейти до процедури реєстрації.

Аутентифікація об'єкта. Ця операція виконується системою захисту КС з метою встановлення, чи є даний об'єкт дійсно тим, за кого він себе видає.

Перевірити істинність об’єкта найпростішим способом КС може шляхом запиту в нього паролю. Проте використання паролю не є безпечним для самого обєкту. У звичайній ситуації, коли є впевненість, що КС є саме тією, за кого себе видає, об’єкт просто повідомляє ій пароль. Після цього КС, порівнявши отримане значення з точним, переконується в правах об’єкту. Але в ситуації, коли є невпевненість в повноваженнях КС, такі дії об’єкту недопустимі, оскільки КС, якщо вона насправді є зловмисником, може узнати цей пароль.

Крім знання об’єктом паролю, підтвердженням його істинності можуть бути інші ідентифікаційні дані:

- елементи апаратного забезпечення об’єкту (ключі, магнітні картки, мікросхеми тощо);
- характерні риси особистості об’єкту (відбитки пальців, тембр голосу, особливості поведінки та стиль роботи, освітній рівень, вихованість, звички тощо).

Авторизація об’єкта. Ця операція виконується системою захисту КС з метою встановлення допустимих дій об’єкту в КС, а також призначення доступних йому ресурсів КС. Саме тому цю операцію називають також наданням повноважень об’єкту.

Всі ці три розглянуті операції складають в цілому процедуру, так званої, ініціалізації і відносяться до якогось єдиного об’єкта КС, тобто мають односторонній характер. Але в багатьох випадках виявляється потрібним взаємне, двостороннє встановлення істинності об’єктів, що зв’язуються між собою каналом передачі даних. [8]

2. Взаємна перевірка істинності сторін інформаційного обміну

Взаємна перевірка істинності (аутентифікація) сторін інформаційного обміну здійснюється, як правило на початку сеансу зв’язку. Мета цієї процедури – забезпечити достатній рівень впевненості абонентів по таких чотирьох складових:

- отримувач має бути впевненим в істинності відправника даних;
- отримувач має бути впевненим в істинності переданих йому даних;

- відправник має бути впевненим, що дані отримувачеві доставлені;
- відправник має бути впевненим в істинності доставлених даних.

Для взаємної перевірки істинності найбільш надійною вважається, так звана, процедура **“рукостискання”**. *Основний зміст цієї процедури полягає в тому, що абоненти, у розпорядженні яких має знаходитися один і той же секретний ключ, виконують взаємну перевірку правильності цього ключа. Інакше кажучи, сторони визначають одна одну істинними після того, як кожна доведе правильність свого ключа. Відмітимо, що звичайною зустрічною передачею секретного ключа в істинності сторін переконатись не можливо.*

Розглянемо процедуру рукостискання двох абонентів А і Б, які володіють одним і тим же секретним ключем $К_{АБ}$ деякої симетричної криптосистеми.

Процедуру рукостискання може ініціювати будь-який із абонентів. Припустимо, що її ініціює абонент А. Для цього він відправляє абоненту Б свій ідентифікатор $ІД_А$ звичайним незахищеним каналом зв’язку. Абонент Б, отримавши ідентифікатор $ІД_А$, знаходить у власній базі даних відповідний секретний ключ $К_{АБ}$. Обидва абоненти, знаючи ідентифікатор протилежної сторони, налаштовують свої криптосистеми на ключ $К_{АБ}$, після чого вони готові приступити до процедури рукостискання.

Зауважимо, що в процедурі рукостискання обидва абоненти мають використовувати одну і ту ж відкриту односторонню функцію h. (*Що таке одностороння функція?) (Основна властивість цієї функції полягає в тому, що вона дозволяє отримати таке перетворення h(x) аргумента x, знаючи яке, відновити значення аргумента не можливо).*

Нехай першим починає цю процедуру абонент А.

1). Абонент А генерує випадкову послідовність S, шифрує її і відправляє абонентові В у вигляді криптограми *E(S).*

2). Абонент Б дешифрує криптограму *E(S)* і розшифровує послідовність S, використовуючи одну і ту ж односторонню функцію h. Внаслідок цього абоненти А і Б отримують перетворення $h_А(S)$ і $h_Б(S)$ відповідно.

4). Абонент Б шифрує перетворення $h_Б(S)$ і відправляє цю криптограму абонентові А.

5). Абонент А дешифрує криптограму і порівнює отримане перетворення $h_Б(S)$ зі своїм перетворенням $h_A(S)$. Якщо ці перетворення співпадають, то абонент А переконується в тому, що абонент Б користується тим же самим секретним ключем $K_{АБ}$, тобто абонента Б можна вважати істинним.

Якщо в істинності абонента А бажає переконатися абонент Б, то він повинен повторити цю ж саму процедуру зі свого боку.

Отже,

перевагою процедури рукостискання є те, що для підтвердження істинності жоден з абонентів не повинен передавати ніяких секретних даних;

недоліком процедури рукостискання є те, що далеко не завжди абоненти, які бажають виконати взаємну перевірку істинності, мають у своєму розпорядженні спільний секретний ключ. [8]

3. Протоколи аутентифікації з нульовою передачею знань

В загальному випадку **протокол передачі даних** – це обумовлені наперед правила передачі даних між двома пристроями, чи абонентами.

Такі протоколи називають також протоколами доведення без розголошення, коли один абонент (абонент А) доводить другому (абонент Б), що він знає певні секретні дані, не розголошуючи самі дані.

Основна ідея таких протоколів полягає в тому, що наявність секретних даних у абонента А перевіряється абонентом Б без отримання самих цих даних або їх частини. Найчастіше в таких протоколах абонент Б ставить перед абонентом А ряд запитань, відповіді на які мають бути бінарного характеру (так-ні). Із збільшенням кількості правильних відповідей t у абонента Б зростає впевненість в істинності абонента А за формулою ймовірності $p=1-0.5^t$.

(*На протоколах аутентифікації з нульовою передачею знань базується використання інтелектуальних карток з мікропроцесором (старт-карток) в різноманітних комерційних, цивільних та військових застосуваннях (картки для банкоматів, картки-перепустки). При цьому головна задача полягає в тому, щоб при пред'явленні картки оперативно виявити підробку і відмовити власнику такої картки у подальшому обслуговуванні.*

Основна ідея безпечного використання інтелектуальних карток полягає в тому, що сама картка є власником деякого секретного ключа, який вважається невід'ємною ознакою її істинності. В процесі аутентифікації картка має довести своє знання цього секретного ключа, не розголошуючи його. Успішне доведення знання секретного ключа є доказом істинності картки. Т.ч., по перше, нерозголошення картками своїх секретних ключів не дає можливості зловмисникам виготовляти подібні картки, а по-друге, не приводить до негативних наслідків при користуванні підробними банкоматами.)

Перед тим як перейти до розгляду протоколу потрібні деякі поняття.

Отже, нехай маємо деяке натуральне число $m>2$, натуральне число $a<m$ називається ***квадратичним лишком за модулем m***, якщо воно взаємно просте з модулем, тобто $НСД(a, m)=1$, і крім того, існує таке ціле число x, що $x^2=a \pmod m$. Тут число x є ***квадратичним коренем за модулем m***. Якщо ж такого цілого числа x не існує, то *a* називають квадратичним не лишком за модулем *m*. *(Детальніше про властивості квадратичних лишків можна прочитати в літературі 8, Моргун)*

Протокол аутентифікації з нульовою передачею знань запропонували у 1986 році У.Фейге, А.Фіат і А.Шамир. Розглянемо його у спрощеному вигляді.

Попередню підготовку до видачі інтелектуальних карток, які будуть використовуватись у певній прикладній галузі, здійснює Центр видачі інтелектуальних карток. В першу чергу Центр вибирає значення модуля *N*, який являє собою добуток двох секретних великих простих чисел. Сам модуль *N* є відкритим і стосується всіх карток, які будуть використовуватися у цій прикладній галузі.

Інтелектуальна картка готується за заявкою користувача, який бажає таку картку отримати. Для цього користувач надає Центру певні ідентифікаційні дані: ім'я та адресу власника картки, термін дії картки, номер банківського рахунку тощо.

З урахуванням ідентифікаційних даних Центр вибирає деяке число *S*, яке, крім того, має задовольняти умовам $1<S<N$ *і НСД(S,N)=1*. Значення *S* є секретним ключем картки.

Далі Центр обчислює число *V* за формулою $V=S^2 mod\ N$. Отже, число *V* є квадратичним залишком за модулем *N*, а число *S* є квадратичним коренем із числа *V* за модулем *N*. Знайдене число *V* є відкритим ключем картки.

Розглянемо приклад. Нехай для деякої прикладної галузі Центром вибрано модуль N=35, який є добутком двох "великих" простих чисел 5 і 7.

Припустимо також, що для деякого користувача з урахуванням його ідентифікаційних даних Центр вибрав число S=19 секретним ключем картки.

Внаслідок обчислення $V=S^2 mod\ N=19^2 mod\ 35=11$ *отримано відкритий ключ картки V=11.*

Отже N та V – відкриті параметри, а S – секретний.

Як ви думаєте, чому вибрані задачі саме обчислення квадратичного залишку та квадратичного кореня? Тому, що повертаючись до понять квадратичного залишку, обчислення квадрата деякого числа x за модулем m є односторонньою функцією. Обернені задачі – задача розпізнавання, чи справді a (V) є квадратичним залишком за модулем m, а також рівноцінна задача добування квадратичного кореня із a (V) за модулем m (тобто добування S) вважаються практично нерозв'язуваними.

Надалі у протоколах аутентифікації безпосередньо беруть участь дві сторони:

- сторона А, яка являє собою картку, видану користувачеві Центром видачі інтелектуальних карток, і яка має доводити стороні Б свою істинність;
- сторона Б, яка перевіряє, надані стороною А, і приймає остаточне рішення стосовно її обслуговування.

Протокол аутентифікації має наступний вигляд:

1). Сторона А вибирає випадкове число *r*, яке має задовольняти умові $1<r<N$. Після цього сторона А підносить обране число до квадрату за модулем *N*, тобто обчислює $x=r^2 mod\ N$. Отримане значення *x* сторона А відправляє стороні Б.

Наприклад, сторона А вибирає випадкове число $r=17$. Далі вона отримує результат піднесення до квадрату $x=r^2 mod\ N=17^2 mod\ 35=9$ і відправляє число $x=9$ стороні Б.

2). Сторона Б відправляє стороні А випадковий біт b (0 або 1).

3). Сторона А обчислює значення $y=(r\cdot S^b) mod\ N$ і відправляє отримане значення стороні Б.

Тут можливі два випадки.

Якщо $b=0$, то сторона А відправляє стороні Б значення раніше вибраного випадкового числа $y=r$, тобто $y=17$.

Якщо $b=1$, то сторона А відправляє стороні Б значення $y=(r\cdot S^b) mod\ N$, тобто $y=(17\cdot 19) mod\ 35=323\ mod\ 35=8$, яке залежить від секретного ключа S.

4). Сторона Б перевіряє конгруенцію $V^b\cdot x=y^2 (mod\ N)$.

Тут можливі два випадки.

Якщо $b=0$, то сторона Б, перевіряючи конгруенцію $x=y^2 (mod\ N)$, фактично перевіряє правильність обох направлених їй чисел x та y (оскільки, $x=r^2 mod\ N$ і $y=r$).

Для нашого прикладу маємо $9=17^2 (mod\ 35)=289 (mod\ 35)$, що є вірно.

Якщо $b=1$, то сторона Б, перевіряючи конгруенцію $V\cdot x=y^2 (mod\ N)$, фактично перевіряє знання стороною А секретного ключа S.

Для нашого прикладу маємо $11\cdot 9=8^2 (mod\ 35)$, що є вірно. Дійсно, $11\cdot 9 mod\ 35=99 mod\ 35=29$ і $8^2 mod\ 35=29$.

Вказані 4 кроки утворюють цикл протоколу. Якщо результат перевірки позитивний, то цикл повторюють з іншими випадковими значеннями r та b. Якщо ж результат перевірки виявився негативним через незнання стороною А секретного ключа і невдале його вгадування, то сторона Б припиняє протокол і відмовляє стороні А в обслуговуванні.

Вважається, що, з достатньою для практики надійністю, описаний протокол слід повторити $t=[\log_2 N]$ разів, щоб бути впевненим в істинності сторони А. Якщо результат перевірки у кожному з усіх циклів був успішним, то стороні А вдається переконати сторону Б у своєму знанні секретного ключа з ймовірністю 1. Якщо ж сторона А не знає секретного ключа, то які б

винахідливі числа x та y вона не відправляла, але сторона Б викриває її неістинність з ймовірністю p=0.5 в кожному із циклів і з ймовірністю $p=1-0.5^t$ *в останньому із t циклів.*

Для того, щоб протокол працював успішно, істинна сторона А ні в якому разі не повинна повторно використовувати значення випадкового числа r, інакше зловмисна сторона Б зможе легко обчислити значення секретного ключа S. [8]

Існує ще один протокол, який відноситься до протоколів з нульовою передачею знань і був запропонований Л.Гіллоу та Ж. Куіскуотером, носить назву протокол аутентифікації Гіллоу-Куіскуотера.

(Детально даний протокол дослідити самостійно в літературі [8])

Тема 14. ЕЛЕКТРОННИЙ ЦИФРОВИЙ ПІДПИС

1. Загальні положення

Протягом багатьох століть при ведені ділової переписки, заключенні контрактів і оформленні будь-яких інших важливих паперів підпис відповідальної особи або виконавця був неодмінною умовою визнання його статусу або незаперечним свідченням його важливості.

Подібний акт переслідував дві мети:

• гарантування істинності листа шляхом звірення підпису зі зразком;

• гарантування авторства документа (з юридичної точки зору).

Виконання даних вимог ґрунтується на наступних властивостях підпису:

• підпис автентичний, тобто з його допомогою одержувачу документа можна довести, що він належить власнику (на практиці це визначається графологічною експертизою);

• підпис не підроблюваний, тобто служить доказом, що тільки та людина, чий автограф стоїть на документі, міг підписати даний документ, і ніхто інший не зміг би цього зробити;

• підпис такий, що не переноситься, тобто є частиною документа і тому перенести його на інший документ неможливо;

• документ з підписом є незмінним, тобто після підписування його неможливо змінити, залишивши даний факт непоміченим;

• підпис незаперечний, тобто людина, яка підписала документ, в разі визнання експертизою, що саме вона засвідчила даний документ, не може заперечити факт підписання;

• будь-яка особа, що має зразок підпису, може, упевнитися в тому, що даний документ підписаний власником підпису.

Один із найпростіших способів аутентифікації (підписування) електронних документів є використання шифрування. При цьому відправник А шифрує документ своїм секретним ключем і відправляє отримувачу Б. Отримувач Б дешифрує документ за допомогою відкритого ключа відправника А. Якщо це йому вдається, то документ вважається істинним. Якщо отримувач

Б не зможе дешифрувати документ, то документ істинним не вважається. Такий спосіб має надзвичайно серйозний недолік, пов'язаний з тим, що він є неефективним для підписування документів значного обсягу.

Саме тому для аутентифікації електронних документів, що передаються телекомунікаційними каналами зв'язку, використовується спеціально створюваний електронний цифровий підпис (ЕЦП).

З переходом до безпаперових способів передачі і зберігання даних, а також з розвитком систем електронного переказу грошових коштів, в основі яких – електронний аналог паперового платіжного доручення, проблема віртуального підтвердження автентичності документа набула особливої гостроти. Розвиток будь-яких подібних систем тепер немислимий без існування електронних підписів під електронними документами. Однак застосування та широке поширення електронно-цифрових підписів (ЕЦП) спричинило цілий ряд правових проблем. Так, ЕЦП може застосовуватися на основі домовленостей всередині якої-небудь групи користувачів системи передачі даних, і відповідно до домовленості усередині даної групи ЕЦП повинен мати юридичну силу. Але чи буде електронний підпис мати доказову силу в суді, наприклад при оскарженні факту передачі платіжного доручення?

Хоча ЕЦП зберіг практично всі основні властивості звичайного підпису, все-таки деякі особливості реалізації електронного автографа роблять його окремим класом підписів. Тому юридичні, правові та методологічні аспекти застосування ЕЦП повинні враховувати його специфіку.

Існує кілька методів побудови схем ЕЦП, а саме:

- шифрування електронного документа (ЕД) на основі симетричних алгоритмів. Дана схема передбачає наявність у системі третьої особи (арбітра), що користується довірою учасників обміну підписаними подібним чином електронними документами. Взаємодія користувачів даною системою здійснюється за такою схемою:

- учасник А зашифровує повідомлення своїм таємним ключем $К_А$, знання якого розділено з арбітром, потім шифроване повідомлення передається арбітру із зазначенням адресата даного повідомлення (інформація, що

ідентифікує адресата, передається також у зашифрованому вигляді);

- арбітр розшифровує отримане повідомлення ключем $К_А$, виконує необхідні перевірки і потім зашифровує секретним ключем учасника В ($К_В$). Далі зашифроване повідомлення посилається учаснику В разом з інформацією, що воно прийшло від учасника А;

- учасник В розшифровує це повідомлення й переконується в тому, що відправником є учасник А.

(Авторизацією документа в даній схемі буде вважатися сам факт зашифрування ЕД секретним ключем і передача зашифрованого ЕД арбітру. Основною перевагою цієї схеми є наявність третьої сторони, що виключає будь-які спірні питання між учасниками інформаційного обміну, тобто в даному випадку не потрібно додаткової системи арбітражу ЕЦП. Недоліком схеми є наявність третьої сторони і використання симетричних алгоритмів шифрування. На практиці ця схема не отримала широкого розповсюдження.);

• використання асиметричних алгоритмів шифрування. Фактом підписання документа в даній схемі є зашифрування документа секретним ключем його відправника. Ця схема теж використовується досить рідко внаслідок того, що довжина ЕД може виявитися критичною. Застосування асиметричних алгоритмів для зашифрування повідомлень великої довжини неефективно з точки зору швидкісних характеристик, що, наприклад, для системи валових розрахунків є одним з основних показників. У цьому випадку не потрібно наявності третьої сторони, хоча вона може виступати в ролі сертифікаційного органу відкритих ключів користувачів;

• розвитком попередньої ідеї стала найбільш поширена схема ЕЦП, а саме: зашифрування остаточного результату обробки ЕД хеш-функцією за допомогою асиметричного алгоритму.

Генерація підпису відбувається наступним чином:

1. Учасник А обчислює хеш-код від ЕД. Отриманий хеш-код проходить процедуру перетворення з використанням свого секретного ключа. Після чого отримане значення (яке і є ЕЦП) разом з ЕД відправляється учаснику В.

2. Учасник В повинен отримати ЕД з ЕЦП та сертифікований відкритий ключ учасника А, а потім провести розшифрування на ньому ЕЦП, сам ЕД піддається операції хешування, після чого результати порівнюються, і якщо вони співпадають, то ЕЦП визнається істинним, в іншому випадку помилковим.

Ефективність реалізації даної схеми в порівнянні з попередньою полягає в застосуванні повільних процедур асиметричного шифрування до хеш-коду ЕД, який значно коротше самого ЕД.

Стійкість даного типу ЕЦП заснована на стійкості асиметричних алгоритмів шифрування і застосовуваних хеш-функцій.

Крім перерахованих вище існують «екзотичні» варіанти побудови схем ЕЦП (груповий підпис, незаперечний підпис, довірений підпис і т.п.). Поява цих різновидів обумовлена різноманіттям завдань, що вирішуються за допомогою електронних технологій передачі та обробки ЕД.

У загальному випадку підписаний ЕД виглядає як пара, що складається з бінарних рядків (M, S), де M представляє собою ЕД, а S – розв’язок рівняння $F_K(S) = M$, де F_K є функцією з секретом. У зв'язку з вище описаним визначенням ЕЦП можна виділити наступні його властивості:

• є непідроблюваний, оскільки розв’язати рівняння $F_K(S) = M$ може тільки володар секрету К;

• однозначно ідентифікує автора, тобто людини, яка підписала цей документ;

• верифікація підпису (перевірка) проводиться на основі знання функції F_K;

• є таким, що не переноситься на інший ЕД; виняток становить випадок, коли для використовуваної хеш-функції знайдені колізії;

• ЕД з ЕЦП може передаватися по відкритих каналах, оскільки будь-яка зміна ЕД призведе до того, що процедура перевірки ЕЦП виявить даний факт. [8]

2. Атаки на ЕЦП

Широке застосування ЕЦП в різних областях електронних комунікацій призвело до того, що в криптографії вже склався великий теоретичний доробок, присвячений створенню схем ЕЦП, їх використання, а також питанням стійкості більшості схем ЕЦП і безпеки їх застосування.

Як вже зазначалося, стійкість більшості схем ЕЦП залежить від стійкості асиметричних алгоритмів шифрування та хеш-функцій. Тому даний розділ ми присвятимо опису існуючих на сьогоднішній день атак і загроз на схеми ЕЦП.

Наведемо **класифікацію атак на схеми ЕЦП**:

• атака з відомим відкритим ключем. Вона є, очевидно, найслабкішою з усіх перерахованих нижче, оскільки зловмисник завжди може отримати відкритий ключ користувача;

• атака з відомими підписаними повідомленнями. Противник крім відкритого ключа має ще й набір підписаних повідомлень;

• проста атака з вибором підписаних повідомлень. Противник має можливість вибирати підписані повідомлення, при цьому відкритий ключ він отримує після вибору повідомлень;

• спрямована атака з вибором повідомлень. Являє собою варіант попередньої і відрізняється тим, що, отримуючи підписані повідомленя, противник знає відкритий ключ;

• адаптивна атака з вибором повідомлень. У даній атаці противник знає відкритий ключ, може вибирати підписані повідомлення і, крім того, має підписи усіх раніше підписаних повідомлень.

Кожна атака на схему ЕЦП переслідує певні цілі (загрози), які можна розділити на наступні класи:

• повне розкриття. Противник знаходить секретний ключ користувача;

• універсальна підробка. Противник знаходить алгоритм, функціонально еквівалентний алгоритму генерації ЕЦП;

• селективна підробка. Підробка підпису для повідомлення, обраного противником;

• екзистенційна підробка. Підробка підпису хоча б для одного випадково вибраного повідомлення.

Оцінка стійкості схеми ЕЦП проводиться відносно пари (атака-загроза), тобто стійкість ЕЦП можна визначити як здатність підпису протистояти досягненню противником строго визначеної мети при проведенні атаки на схему ЕЦП.

На практиці застосування ЕЦП дозволяє запобігти або виявити такі **дії порушника** (він по відношенню до системи може бути як внутрішнім, так і зовнішнім):

• відмова одного з учасників від факту відправлення повідомлення. Учасник інформаційного обміну А заявляє, що він не посилав повідомлення учаснику В, хоча насправді посилав;

• модифікація прийнятого ЕД. Учасник В, прийнявши повідомлення, змінят його і стверджує, що саме дане повідомлення він прийняв від учасника А;

• підробка повідомлення. Учасник В створює повідомлення і стверджує, що дане повідомлення він прийняв від учасника А, хоча насправді А нічого не передавав;

• нав'язування повідомлень в процесі передачі. Зловмисник переохоплює обмін повідомленнями між А і В і модифікує їх;

• імітація надіслати повідомлення. Зловмисник намагається відправляти повідомлення від імені одного з учасників інформаційного обміну.

При цьому слід врахувати, що існують порушення, від яких неможливо захистити систему обміну повідомленнями, а саме:

• повтор передачі повідомлення. Зловмисник або один з учасників інформаційного обміну намагається знову передати раніше відправлене повідомлення. Подібні порушення особливо поширені в системі електронного переказу коштів;

• фальсифікація часу відправлення повідомлення.

Протидія таким порушенням може ґрунтуватися на використанні тимчасових вставок і суворому обліку вхідних повідомлень. [11, 12, 13, 14]

3. Алгоритми ЕЦП: Ель-Гамаля (EGSA), DSA, ГОСТ Р34.10-94

Надійний та зручний для реалізації на персональних комп'ютерах алгоритм цифрового підпису був розроблений у 1984 році американцем арабського походження Тахером Ель Гамалем. У 1991 році Національний інститут стандартів (НІСТ) США обгрунтував перед комісією Конгресу США вибір цього алгоритму як бази для відповідного національного стандарту. Найменування EGSA має походження від слів El Gamal Signature Algorithm (алгоритм цифрового підпису Ель Гамаля).

Ідея EGSA базується на тому, що для практичної неможливості фальсифікації ЕЦП може бути використана практично нерозв'язувана задача дискретного логарифмування.

Послідовно розглянемо алгоритм електронного цифрового підпису Ель Гамаля. [8]

1) Відправник, який має підписати свій документ, вибирає деяке велике просте ціле число P. Це число є відкритим і передається усім отримувачам документів відправника. Реальні значення близькі до $2^{1024}(\approx 10^{308})$

Приклад. Відправник вибирає число $P = 11$.

2) Відправник вибирає також велике ціле число G, яке має задовольняти умовам $1 < G < P$. Це число також є відкритим і передається усім отримувачам документів відправника. Реальні значення близькі до $2^{512}(\approx 10^{154})$.

Приклад. Відправник вибирає число $G = 2$.

3) Відправник вибирає ціле число X, яке має задовольняти умовам $1 < X < P$. Це число є секретним ключем відправника для підписування документів.

Приклад. Відправник вибирає секретний ключ X= *8.*

4) Відправник обчислює число $Y = G^X mod\ P$. Це число є відкритим ключем відправника, яке використовується отримувачами для

перевірки його підпису. Це число також передається усім отримувачам документів відправника.

Приклад. Відправник обчислює число $Y = G^X \ mod P = 2^8 \ mod \ 11 = 3$.

5) Відправник обчислює хеш-значення H свого документу М. Кількість біт хеш-значення має бути на одиницю меншою кількості біт значення P-1. У загальному випадку хеш-значення H документу М відправника має задовольняти умовам $1 < H < (P - 1)$.

Приклад. Відправник обчислює хеш-значення H свого документу M і отримує $H = 5$. Згадані умови стосовно кількості біт виконуються. Дійсно, значення $H = 5$ має 3 біта, а значення $P - 1 = 10$ має 4 біта.

6) Відправник вибирає випадкове ціле число K, яке має задовольняти умовам $1 < K < (P - 1)$. Крім того, числа K і P-1 мають бути взаємно простими, тобто їх найбільший спільний дільник $НСД(K, P-1) = 1$. Це число також є секретним числом відправника для підписування його документу М.

Приклад. Відправник вибирає випадкове ціле число $K = 9$.

7) Відправник обчислює ціле число $a = G^K \ mod P$. Це число є першою складовою електронного цифрового підпису його документу.

Приклад. Відправник обчислює число $a = G^K \ mod P = 2^9 \ mod \ 11 = 6$.

8) Відправник знаходить ціле число й, використовуючи рівняння $H = (X \cdot a + K \cdot b) \ mod \ (P-1)$. Це число є другою складовою електронного цифрового підпису його документу.

Приклад. Відправник знаходить ціле число b із рівняння

$H = (X \cdot a + K \cdot b) \ mod \ (P - 1)$ або $5 = (8 \cdot 6 + 9 \cdot V) \ mod \ 10$. Це число можна знайти шляхом послідовного перебору:

$b = 1$; $(8 \cdot 6 + 9 \cdot 1) \ mod \ 10 = 7$;

$b = 2$; $(8 \cdot 6 + 9 \cdot 2) \ mod \ 10 = 6$;

$b = 3$; $(8 \cdot 6 + 9 \cdot 3) \ mod \ 10 = 5$.

Отримуємо $b = 3$.

9). Відправник передає отримувачу документ М, а також його електронний цифровий підпис у вигляді пари чисел $S = (a, b)$.

Приклад. Відправник передає отримувачу документ M, а також його електронний цифровий підпис у вигляді пари чисел $S = (6, 3)$.

Отримавши документ M, а також його електронний цифровий підпис $S = (a, b)$, отримувач повинен перевірити, чи відповідає цей підпис документу.

1) Отримувач обчислює хеш-значення H отриманого документу M.

Приклад. Отримувач обчислює хеш-значення H документу M і отримує $H = 5$.

2) Отримувач обчислює ціле число $A_1 = (Y^a \cdot a^b) \bmod P$.

Приклад. Отримувач обчислює ціле число $A_1 = (Y^a \cdot a^b) \bmod P = (3^6 \cdot 6^3) \bmod 11 = 10$.

3) Отримувач обчислює ціле число $A_2 = G^H \bmod P$.

Приклад. Отримувач обчислює ціле число $A_2 = G^H \bmod P = 2^5 \bmod 11 = 10$.

4) Отримувач порівнює знайдені числа A_1 і A_2. Ці числа будуть рівні тоді і тільки тоді, коли електронний цифровий підпис відповідає документу, тобто відправником документу M є дійсно власник секретного ключа X, і що відправник підписав саме цей документ M.

Приклад. ЕЦП $S = (6, 3)$ відповідає отриманому документу M.

Використання алгоритму електронного цифрового підпису Ель Гамаля вимагає вибору щоразу іншого випадкового цілого числа K. Інакше, якщо зловмисник розкриє повторно використовуване відправником число K, то він зможе розкрити і його секретний ключ X.

Отже, алгоритм Ель-Гамаля являє собою протокол з обчисленнями і цифровим підписом вважається пара чисел. (Обчислення що проводяться в протилежну сторону базуються на практично нерозв'язній задачі дискретного логарифмування).

У 1991 році Національний інститут стандартизації і технологій (NIST) США опублікував стандарт на ЕЦП (Digital Signature Standard, DSS), в основі якого – алгоритм DSA. Він є аналогом механізму, запропонованим Ель Гамалем, але з деякими змінами, зокрема за рахунок зменшення числового порядку одного з параметрів схеми.

У даному стандарті підпис – це два великих цілих числа, отримані відповідно до процедур і параметрів, визначеним в DSS.

Алгоритм DSA є «класичним» прикладом схеми ЕЦП на основі використання хеш-функції і асиметричного алгоритму шифрування.

Стійкість системи в цілому заснована на складності знаходження дискретних логарифмів в кінцевих полях.

DSA [7, 9]

Даний алгоритм є частиною американського стандарту DSS.

В алгоритмі використовується одно направлена хеш-функція H(x).

Стандарт визначає використання алгоритму SHA-1.

Параметри:

p – просте число L бітів, де L набуває значення кратне 64 в діапазоні від 512 до 1024;

q – 160 – бітовий множник $p - 1$;

$a = g^{(p-1)/q} \bmod p$, де $g < p - 1$, для якого $g^{(p-1)/q} \bmod p > 1$;

$y = a^x \bmod p$, де $x < q$;

m – текст.

Ключ підпису: (y, p, q, a).

Ключ верифікації: (x).

Підписування:

k – випадкове число, $k < q$;

$r = (a^k \bmod p) \bmod q$ – обчислення першої частини підпису;

$s = (k^{-1}(H(m) + xr)) \bmod q$ – обчислення другої частини підпису.

Підпис: (r , s).

Верифікація:

$w = s^{-1} \bmod q$;

$u_1 = (H(m) \cdot w) \bmod q$;

$u_2 = (rw) \bmod q$;

$v = ((a^{u_1} \cdot y^{u_2}) \bmod p) \bmod q$, якщо $v = r$, то підпис справжній.

ГОСТ Р34.10-94 [7, 9]

Даний алгоритм є російським стандартом цифрового підпису.

Цей алгоритм використовує одно направлену хеш-функцію H(x). Стандарт визначає використання хеш-функції ГОСТ Р34.11-94, яка основана на симетричному алгоритмі ГОСТ 28147-89.

Параметри:

p – просте число, довжина якого лежить в діапазоні 509-512 біт, або 1020-1024 біт;

q – просте число, множник p – 1, довжиною від 254 до 256 біт;

a – випадкове число, $a < p-1$, $a^q \bmod p = 1$;

$y = a^x \bmod p$, де $x < q$;

Ключ підпису: *(p, q, a, y)*.

Ключ верифікації: *(x)*.

Підписування:

k – випадкове число, $k < q$;

$r = (a^k \bmod p) \bmod q$ – обчислення першої частини підпису;

$s = (xr + k(H(m))) \bmod q$ – обчислення другої частини підпису;

Якщо $H(m) \bmod q = 0$, необхідно встановити його в 1.

Якщо $r = 0$ (або $s=0$), то необхідно вибрати інше значення *K* і почати знову.

Підпис: $(r \bmod 2^{256}, s \bmod 2^{256})$.

Верифікація:

$v = H(m)^{q-2} \bmod q$;

$z_1 = (sv) \bmod q$;

$z_2 = ((q - r) \cdot v) \bmod q$;

$u = \left((a^{z_1} \cdot y^{z_2}) \bmod p\right) \bmod q$, якщо $u = r$, то підпис справжній.

4. Арбітраж ЕЦП

Практичне застосування ЕЦП, крім процедур генерації та перевірки електронного підпису, вимагає присутності системи розбору конфліктних ситуацій (арбітраж ЕЦП). Будь-який алгоритм ЕЦП подібних процедур у

своєму описі не містить. Це пов'язано з тим, що побудова схем арбітражу пов'язана з контекстом використання ЕЦП, так як його проведення – не стільки технічне, скільки організаційне завдання.

У більшості існуючих систем використання арбітражу ЕЦП засновано на тому, що підписати повідомлення може тільки власник секретного ключа (в цьому випадку відповідальність за його компрометацію покладається на користувача). Це твердження не можна вважати вірним у випадку, коли $h(m_1) = h(m)$, тобто один і той же підпис з'являється під двома різними повідомленнями. Арбітр у подібному випадку не зможе вирішити суперечку, що виникла, хоча очевидно, що хтось з учасників знайшов лазівки для хеш-функції. Тому в більшості випадків процедури арбітражу будуть неповними, за винятком варіанта, коли схеми ЕЦП розроблені спеціально з врахуванням проведення подібного розгляду. З існуючих на сьогоднішній день схем електронного підпису арбітраж найефективніше може бути проведений для ЕЦП, побудованих на основі симетричного шифрування за участю третьої сторони. При використання схем ЕЦП, побудованих без участі арбітра, слід з особливою старанністю вибирати процедуру підписання ЕД, щоб арбітр у випадку виникнення спірної ситуації міг вирішити, яка зі сторін є правою.

Конкретна реалізація процедур арбітражу залежить насамперед від потреб користувачів, технічної реалізації ЕЦП і від обставин, через які виникла необхідність у проведенні арбітражу. Для прикладу розглянута процедура, що виникла при відмові користувача від факту підписання документа m, при цьому в якості використовуваної ЕЦП береться алгоритм DSA.

Тут вихідними даними є загальнодоступні параметри DSA – p, q, g; ЕД – m і підпис – r, s. ЕД і підпис представляються арбітру учасником В, який хоче довести, що даний підпис належить учаснику А; в свою чергу, учасник А відмовляється визнати, що даний підпис був ним згенерований.

Насамперед арбітр вимагає від учасника А пред'явлення секретного ключа. Ця вимога для будь-якого користувача ЕЦП в конкретній описаній системі має виконуватися беззастережно. Арбітр перевіряє відкритий ключ із загальнодоступного довідника на відповідність представленого секретного

ключа. У випадку не співпадання арбітр звертається в центр сертифікації відкритих ключів та вимагає надання завіреного учасником А документа, що містить відкритий ключ. Якщо з'ясовується, що відкритий ключ в загальнодоступному довіднику не співпадає із зазначеним у документі, винним визнається центр сертифікації відкритих ключів. Коли відкриті ключі в довіднику і в документі збігаються, це означає, що пред'явлений некоректний секретний ключ, і учасник А визнається винним.

У разі, якщо відкритий і секретний ключі відповідають раніше створеним зразкам, арбітр виконує наступні обчислення:

$W = s^{-1} mod\ q$

$ul = (h\ (m)\ w)\ mod\ q$

$u2 = (rw)\ mod\ q$

$v = ((g^{ul} y^{u2})\ mod\ p)\ mod\ q$

Завершальною фазою є перевірка рівності $v = r$. Якщо вона виконується, то підпис визнається дійсним, якщо ні – помилковим.

Тема 15. ЙМОВІРНІСНЕ ШИФРУВАННЯ. КВАНТОВА КРИПТОГРАФІЯ

1. Ідея ймовірнісного шифрування

Поняття ймовірнісного шифрування було винайдено Шафі Голдвассером (Shafi Goldwasser) і Сільвією Мікалі. Хоча їх теорія дозволяє створити найбезпечнішу із винайдених криптосистем, рання реалізація була неефективною, але більш пізніші реалізації все змінили.

Ідея ймовірнісного шифрування: усунути витікання інформації у криптографії з відкритими ключами. Так як криптоаналітик завжди може розшифрувати випадкове повідомлення відкритим ключем, він може отримати деяку інформацію. При умові, що у нього є шифротекст $C=E_K(M)$, і він намагається отримати відкритий текст M, він може вибрати випадкове повідомлення M' і зашифрувати його: $C'=E_K(M')$. Якщо $C'=C$, то він вгадав правильний відкритий текст. В протилежному випадку він робить наступну спробу.

Крім того, ймовірнісне шифрування дозволяє уникнути навіть часткового витікання інформації про справжнє повідомлення. При використанні криптографії з відкритими ключами крипто аналітик може розізнати дещо про біти: XOR 5-го, 17-го та 39-го біт рівне 1, і т.д. При ймовірнісному шифруванні залишається скритою і така інформація.

Таким способом можна витягнути не багато інформації, але потенційно можливість криптоаналітика розшифрувати випадкове повідомлення вашим відкритим ключем може створити певні проблеми. Кожен раз, шифруючи повідомлення, криптоаналітик може витягнути трохи інформації. Ніхто не знає, наскільки важлива ця інформація.

Ймовірнісне шифрування намагається усунути витік такої інформації. Ціль цього методі полягає в тому, щоб ні в обчисленнях, які виконуються над шифротекстом, ні перевірка будь-яких інших відкритих текстів не могли дати криптоаналітику ніякої інформації про відповідний відкритий текст.

При ймовірнісному шифруванні алгоритм шифрування є ймовірнісним, а не детермінованим. Іншими словами, багато шифротекстів при розшифруванні

дають даний відкритий текст, і конкретний шифротекст, що використовується в будь-якому конкретному шифруванні, вибирається випадковим чином.

$C_1=E_K(M)$, $C_2= E_K(M)$, $C_3= E_K(M)$, ... $C_i= E_K(M)$,

$M=D_K(C_1)=D_K(C_2)=D_K(C_3)=...=D_K(C_i)$.

При ймовірнісному шифруванні крипто аналітику більше не вдасться шифрувати будь-які відкриті тексти в пошуках правильного шифротекста. Для ілюстрації, нехай у криптоаналітика є шифротекст $C_i= E_K(M)$. Навіть якщо він правильно вгадає M, отриманий при шифруванні $E_K(M)$ результат буде зовсім іншим шифротекстом C: C_j. Порівнюючи C_i та C_j, він не може по їх співпаданню визначити правильність своєї здогадки.

Навіть якщо у крипто аналітика є відкритий ключ шифрування, відкритий текст і шифротекст, він не може без закритого ключа довести, що шифротекст є результатом шифрування конкретного відкритого тексту. Навіть виконавши весь пошук, він може довести тільки, що кожний можливий відкритий текст є можливим відкритим текстом.

У цій схемі шифротекст завжди буде більший за відкритий текст. Цього неможливо уникнути, оскільки це є результатом того, що багато шифротекстів розшифровуються в один і той самий відкритий текст. В першій схемі ймовірнісного шифрування шифротекст отримувався настільки більшим відкритого, що він був не корисним.

2. Ефективна реалізація ймовірнісного шифрування

Мануель Блюм (Manual Blum) та Гольдвассер отримали ефективну реалізацію ймовірнісного шифрування з допомогою генератора псевдовипадкових бітів Blum Blum Shub (BBS, тема 12, пункт 5).

Генератор BBS оснований на теорії квадратичних залишків. Існує два простих числа, p та q, конгруентних 3 по модулю 4. Це закритий ключ. Їх добуток, $n=pq$, є відкритим ключем. Безпека схеми опирається на складність розкладання n на множники.

Для шифрування повідомлення M спочатку вибирається випадкове число x, взаємно просте з n. Потім обчислюється $x_0=x^2 \ mod \ n$. x_0 – служить

початковою послідовністю для генератора псевдовипадкових бітів BBS, а вихід генератора використовується в якості потокового шифру. Побітно виконується XOR M з виходом генератора. Генератор видає біти b_i (молодший значущий біт x_i, де $x_i=x_{i-1}^2 \bmod n$), тому

$M=M_1, M_2, M_3, \ldots M_t$

$c=M_1 \oplus b_1, M_2 \oplus b_2, M_3 \oplus b_3, \ldots M_t \oplus b_t$, де t – це довжина відкритого тексту. І остання дія: добавити останнє обчислене значення, x_t, до кінця повідомлення.

Розшифрувати це повідомлення можна тільки одним способом – отримати x_0 і з цієї стартової послідовності запустити генератор BBS, виконуючи XOR виходу з шифротекстом. Тільки той хто знає p та q, зможе розшифрувати повідомлення.

Цю схему можна зробити ще швидшою, використовуючи всі відомі безпечні біти x_i, а не тільки молодший значущий біт. З таким покращенням ймовірнісне шифрування Blum-Goldwasser виявляється швидшим RSA і не допускає витікання інформації про відкритий текст. Крім того, можна довести, що складність розкриття цієї схеми рівна складності розкладу n на множники.

З іншої сторони, ця схема зовсім небезпечна по відношенню до розкриття з вибраним шифротекстом. За молодшими значущими бітами правильних квадратичних залишків можна обчислити квадратний корінь будь-якого квадратичного залишку. Якщо це вдасться, то вдасться і розкласти на множники.

3. Ймовірнісні моделі шифру

На основі відображень шифру та ймовірнісних моделях відкритого тексту і множини ключів можна побудувати ймовірнісну модель шифру. Позначимо деякі розподіли ймовірностей (р.й.):

$P_{вх}$ – р.й. на множині відкритих текстів;

$P_{кл}$ – р.й. на множині ключів;

$P_{ш}$ – р.й. на множині шифрованих текстів;

$P_{вх,к}$ – загальний р.й. на множині пар відкритих текстів та ключів;

$Р_{вх,ш}$ – загальний р.й. на множині пар відкритих та шифрованих текстів;

$Р_{вх/ш}$ – умовне р.й. на множині відкритих текстів (при умові що шифрований текст фіксований).

Нехай *a* – відкритий текст, *z* – ключ, *y* – шифрований текст, *E(a,z)* – криптограмма, отримана в результаті шифрування відкритого тексту *a* ключем *z*.

Зазвичай вважають, що при шифруванні ключ *z* вибирається незалежно від відкритого тексту *a*. Тому $P_{вх,к}(a,z)=P_{вх}(a) * P_{кл}(z)$

Загальні і умовні розподіли ймовірностей визначаються з формул:

$$P_{ш}(y) = \sum_{(a,z):E(a,z)=y} P_{вх}(a) \cdot P_{кл}(z),$$

$$P_{вх,ш}(a,y) = \sum_{z:E(a,z)=y} P_{вх}(a) \cdot P_{кл}(z),$$

$$P_{вх/ш}(a/y) = P_{вх,ш}(a,y)/P_{ш}(y),$$

де остання рівність випливає з визначення умовної ймовірності і справедливе при умові коли розподіл $P_{ш}(y)>0$.

Таким чином, маючи розподіли ймовірностей на множині відкритих текстів та ключів та знаючи сімейство шифрів, можна обчислити як розподіл ймовірності на множині шифрованих текстів, так і різні загальні і умовні розподіли ймовірностей.

Використовуючи ймовірнісну модель шифру, Шеннон вперше сформулював поняття абсолютно стійкого шифру. [12]

4. Що таке квантова криптографія?

Квантова криптографія вводить природну невизначеність квантового світу. З її допомогою можна створювати лінії зв’язку, які не можливо послухати, не вносячи перешкод у передачу. Закони фізики захищають такий квантовий канал, навіть якщо той хто підслуховує може виконувати будь-які дії, навіть якщо він має доступ до не обмежуваної обчислювальної потужності, навіть якщо *P=NP*. Шарль Бене, Жіль Брассар, Клод Крепо та інші розширили цю ідею, описавши квантовий розподіл ключів, квантову передачу з забуванням, квантові обчислення з декількома учасниками.

Ці ідеї так би й залишилися предметом обговорення фанатів криптографії, але Банне і Брассар розробили діючу модель. Тепер у нас є експериментальна квантова криптографія.

У відповідності з законами квантової механіки частинки насправді не знаходяться на одному місці, а з визначеною ймовірністю існують зразу у багатьох місцях. Однак, це так тільки до тих пір, поки не приходить вчений і не виміряє частинку, що виявилася в даному конкретному місці. Але ви міряти всі параметри частинки (наприклад, координати та швидкість) одночасно неможливо. Якщо міряти одну з цих двох величин, то акт вимірювання знищує всяку можливість виміряти другу величину. Невизначеність є фундаментальною властивістю квантового світу.

Цю невизначеність можна використовувати для генерації секретного ключа. Подорожуючи, фотони коливаються у визначеному напрямі, вверх-вниз, вліво-вправо, або, що більш ймовірно, під якимось кутом. Звичайне сонячне світло неполяризоване, фотони коливаються у всіх можливих напрямках. Коли напрям коливань багатьох фотонів співпадає, вони є поляризовані. Поляризаційні фільтри пропускають тільки ті фотони, які поляризовані у визначеному напрямку, а інші блокуються.

Нехай є імпульс горизонтально поляризованих фотонів. Якщо вони спробують пройти через горизонтальний фільтр, то у них це вийде вдало. Якщо повільно повертати фільтр на 90°, кількість фотонів, що можуть пройти, ставатиме все менше і менше, і на кінець жоден фотон не зможе пройти через фільтр. Це перечить здоровому глузду. Здається, що навіть незначний поворот фільтра повинен зупинити усі фотони, так як вони горизонтально поляризовані. Але в квантовій механіці кожна частинка з визначеною ймовірністю може змінити свою поляризацію і проскочити через фільтр. Якщо кут відхилення фільтру невеликий, то ймовірність висока, в протилежному випадку – рівна нулю.

5. Протокол передачі з використанням квантової криптографії

Поляризацію можна змінити у будь-якій системі координат: двох напрямах, що розходяться під прямим кутом. Прикладами систем координат є прямокутна – горизонтальне та вертикальне направлення, діагональна – ліва та права діагоналі. Якщо імпульс фотонів поляризований в заданій системі координат, то при вимірюванні в тій же системі координат можна взнати поляризацію. При вимірюванні в неправильній системі координат – отримається випадковий результат. Ця властивість використовується для генерації секретного ключа:

1. Відправник А посилає Б послідовність фотонних імпульсів. Кожен з імпульсів випадковим чином поляризований в одному із чотирьох напрямків: горизонтальному, вертикальному, ліво- та праводіагональному.

   Наприклад, А посилає для Б: | | / — \ — | — /
2. У Б є детектор поляризації. Він може настроїти свій детектор на вимірювання прямокутної або діагональної поляризації. Одночасно міряти ту і іншу у нього не вийде, йому не дозволить квантова механіка. Вимірювання однієї поляризації не дасть виміряти іншу. І так, він встановлює свої детектори випадковим чином:

   X + + X X X + X + +

   Тепер, якщо Б правильно налаштує свій детектор, він зареєструє правильну поляризацію. Якщо він налаштує детектор на вимірювання прямокутної поляризації, то імпульс буде поляризований прямокутно, він взнає, яку поляризацію фотонів вибрав А. Якщо він налаштує детектор на вимірювання діагональної поляризації, а імпульс буде поляризований прямокутно, то результат вимірювання буде випадковим. Б не зможе визначити різницю. У наведеному прикладі він може отримати такий результат: / | — \ / \ — / — |
3. Б повідомляє А по незахищеному каналу, які настройки він використовував.

4. А повідомляє Б, які настройки були правильними. В нашому прикладі детектор був встановлений правильно для імпульсів 2, 6, 7 і 9.
5. А та Б залишають тільки правильно виміряні поляризації. В нашому прикладі вони такі: * | * * * \ — * — *

   З допомогою раніше приготовленого коду А та Б перетворюють в біти ці результати вимірювань поляризації. Наприклад, горизонтальна і ліводіагональна можуть означати одиницю, а вертикальна та праводіагональна – нуль. В нашому прикладі вони обидва отримають 0 0 1 1.

Отже, А та Б отримали чотири біта. З допомогою цієї системи вони можуть генерувати стільки бітів скільки їм необхідно. В середньому Б правильно вгадує в 50% випадків, тому для генерації *n* біт А доведеться вислати *2n* фотонних імпульсів. Вони можуть використовувати ці біти як секретний ключ симетричного алгоритму або забезпечити абсолютну безпеку, отримавши достатньо біт для використання в якості одноразового блокноту.

Плюсом є те, що С не зможе підслухати. Так як і Б, йому доведеться вгадати тип поляризації, і як і у Б, половину здогадок буде неправильна. Так як неправильні вимірювання змінюють поляризацію фотонів, то при підслуховуванні С вносить помилки в передачу. Якщо це так, то А та Б отримають різні бітові послідовності. Отже, А та Б закінчують протокол такими діями:

6. А та Б порівнюють декілька бітів своїх рядків. Якщо є розходження вони взнають про підслуховування. Якщо рядки не відрізняються, то вони відкидають використані для порівняння біти і використовують, ті що залишилися.

Покращення цього протоколу дозволяє А та Б використовувати свої біти навіть у присутності С. Вони можуть порівнювати тільки парність бітових множин. Тоді, якщо не виявлено розходжень, їм доведеться відкинути тільки один біт підмножини. Це виявить підслуховування з ймовірністю 50%, але якщо вони звірять таким чином *n* різних бітових підмножин, ймовірність С підслухати і залишитися непомітною буде рівна $1/2^n$.

В квантовому світі не буває пасивного підслуховування. Якщо С намагатиметься розкрити усі біти, він обов’язково порушить канал зв’язку.

Тема 16. КРИПТОГРАФІЧНА СТІЙКІСТЬ ШИФРІВ

Шифри можна класифікувати за криптографічною стійкістю, тобто за можливістю протистояти атакам крипто аналітика.

Питання про стійкість – найважливіше питання криптоаналізу шифрів.

Розглянемо різні підходи до оцінки криптографічної стійкості шифрів.

1. Абсолютно стійкі шифри

Питання про теоретичну стійкість шифрів вперше було сформульоване Клодом Шенноном: “На скільки надійна криптосистема, якщо криптоаналітик противника володіє необмеженим часом і всіма необхідними засобами для аналізу криптограм?” З цим питанням тісно пов’язане наступне: “Чи існують шифри, які не міг би розкрити криптоаналітик, що володіє як завгодно великою криптограмою і необмеженими обчислювальними ресурсами?”

Відомо, що ще до Шеннона над вирішенням цих питань працював американський інженер Д. Вернам, який в 1917 році запропонував новий спосіб шифрування телеграфних повідомлень. Алгоритм шифрування Вернама полягав в тому, що представлена в двійковому коді послідовність відкритого тексту побітно складалась з ключем – випадковою двійковою послідовністю. Додавання бітів відбувалося за модулем 2.

Вернам інтуїтивно відчував, що запропонований ним шифр володіє високими криптографічними якостями але строго довести цього не зумів.

Обґрунтував високі криптографічні якості шифру Вернама зумів Шеннон, він опублікував в 1949 р. основні положення теоретичної криптографії. З використанням ймовірнісної моделі шифру Шеннон дав математичне означення стійкого абсолютно шифру і показав, що шифр Вернама є дійсно абсолютно стійким.

За Шенноном шифр є *абсолютно стійким*, якщо відкриті і шифровані тексти статистично незалежні, тобто для будь-якого відкритого тексту a і будь-якої криптограми y виконується рівність: $P_{вх}(a) = P_{вх/ш}(a/y)$ при умові $P_{ш}(y)>0$. Іншими словами, при використанні абсолютно стійкого шифру розподіл

ймовірностей на множині відкритих текстів після перехоплення криптограми не відрізняється від розподілу ймовірностей на множині відкритих текстів до отримання перехопленої криптограми *y*. Перехоплення повідомлення, зашифрованого за допомогою абсолютно стійкого шифру, не містить для криптоаналітика ніякої інформації якщо ключ йому невідомий.

Шифр називається *ідеально стійким*, якщо неможливо визначити однозначно відкритий текст при відомому зашифрованому тексті будь-якої великої довжини. Очевидно, абсолютно стійкий шифр є ідеально стійким.

Шеннон довів, що абсолютно стійкі шифри існують, наприклад, так званий шифр Вернама по модулю *m*.

Також слід відмітити, що довжина ключа в абсолютно стійких шифрах співпадає з довжиною повідомлення. Це означає, що використання таких шифрів для захисту великих об'ємів інформації потребує великих трудових затрат, пов'язаних з розподілом, збереженням і знищенням ключових матеріалів.

Тим не менше абсолютно стійкі шифри все ж знайшли практичне застосування для захисту особливо важливих ліній зв'язку з відносно невеликим об'ємом інформації, що передається. [12]

2. Системний підхід до оцінки стійкості шифрів

Питання про практичну стійкість, поставлене Шенноном, формулюється так: "Чи надійна шифросистема, якщо крипто аналітик володіє обмеженим часом і обмеженими обчислювальними можливостями для аналізу перехоплених криптограм?" Дане питання тісно пов'язане з проблемою конструювання шифросистем.

З однієї сторони, криптографічна система повинна забезпечувати надійний захист інформації, з іншої – повинна бути зручна для технічної реалізації та експлуатації.

За Шенноном, практично стійка криптосистема за своїми властивостями повинна бути близька до ідеальної системи, тобто повинна бути вдалою підробкою під ідеальний шифр.

Системний підхід до оцінки стійкості шифросистеми має на увазі визначену деталізацію поняття «стійкий шифр». В результаті цієї деталізації формується ряд критеріїв математичного і технічного характеру, яким повинна задовольняти стійка шифросистема. При розробці нового підходу до аналізу шифросистеми формується відповідний критерій якості шифросистеми, який доповняє раніше складену систему критеріїв.

Основною кількісною мірою криптографічної стійкості шифру є *обчислювальна складність* рішення задач дешифрування. Обчислювальна складність визначається декількома характеристиками. Розглянемо найважливіші з них.

Припустимо, що перед криптоаналітиком поставлена задача дешифрування шифру E за деяким набором криптограм. Нехай A_E – клас застосовних до шифру E алгоритмів дешифрування, якими володіє крипто аналітик. При цьому криптоаналітик розглядає як ймовірнісний простір W елементарних подій множину пар ключів і відкритих текстів, якщо відкриті тексти відомі, або множину ключів, якщо відкриті тексти відомі. Для алгоритму $\psi \in A_E$ позначимо через $T(\psi)$ середню трудоємкість його реалізації, яка вимірюється за допомогою деяких умовних обчислювальних операціях. При цьому величина трудоємкості зазвичай усереднюється на множині W.

Однією з основних характеристик практичної стійкості шифру E є середня трудоємкість T_E дешифрування, що визначається за формулою

$$T_E = \min_{\psi \in A_E} T(\psi).$$

При цьому необхідно відмітити наступне:

1. Існують алгоритми дешифрування, визначені не на всьому ймовірнісному просторі W, а лише на деякій його частині. Крім того, деякі алгоритми дешифрування влаштовані так, що їх реалізація приводить до успіху (вирішення де шифрувальної задачі) не на всій області визначення, а лише на деякій її підмножині. Тому до важливих характеристик алгоритму дешифрування ψ слід віднести не тільки його трудоємкість $T(\psi)$, але і

надійність $v(\psi)$, під якою розуміється середня доля інформації, що дешифрується з використанням алгоритму ψ.

Якщо надійність алгоритму дешифрування мала, то з точки зору криптографа він є безпечним, а з точки зору криптоаналітика не ефективним. Таким чином, при отриманні оцінки величини $T(\psi)$ потрібно розглядати лише ті алгоритми дешифрування, надійність яких достатньо велика. При цьому для визначення "найкращого" алгоритму дешифрування системи E можна використовувати різні критерії в залежності від конкретних умов задачі.

2. Складність дешифрування залежить від кількісних і якісних характеристик криптограм, якими володіє криптоаналітик. Кількісні характеристики визначаються кількостю перехоплених криптограм і їх довжинами. Якісні характеристики пов'язані з достовірністю перехоплених криптограм *(наявність спотворень, пропусків,…)*. Оцінюючи стійкість шифру, криптоаналітик отримує верхні оцінки граничної стійкості, так як практичне дешифрування використовує обмежену кількість шифрматеріалу і обмежений клас так званих відомих методів дешифрування.

3. Важливою характеристикою криптографічної стійкості криптосистем є часова складність її дешифрування. Оцінка часової складності дешифрування системи має на увазі більш детальну обробку реалізації алгоритмів дешифрування з врахуванням характеристик обчислювального пристрою, що використовується для дешифрування. До таких характеристик обчислювального пристрою, що реалізує алгоритми дешифрування, відносяться архітектура, швидкодія, об'єм та структура пам'яті, швидкість доступу до пам'яті та інші. Тому, час дешифрування системи E визначається наявним класом алгоритмів дешифрування A_E і обчислювальними можливостями крипто аналітика.

Вибір найкращого алгоритму дешифрування ускладнюється й тим, що різним обчислювальним пристроям можуть відповідати різні "найкращі" алгоритми дешифрування.

Питання про криптографічну стійкість шифросистеми має деякі особливості з точки зору крипто аналітика і криптографа.

Криптоаналітик атакує шифросистему, володіючи конкретними інтелектуальними, обчислювальними та економічними ресурсами. Його ціль – успішно дешифрувати систему.

Криптограф оцінює стійкість шифросистеми, імітуючи атаку на шифр зі сторони криптоаналітика противника. Для цього криптограф моделює дії криптоаналітика, оцінюючи по максимуму інтелектуальні, обчислювальні, технічні та інші можливості противника. Ціль криптографа – впевнитися у високій криптографічній стійкості розробленої шифросистеми.

Таким чином, системний підхід до оцінки практичної стійкості шифру пов’язаний з оцінкою обчислювальних трудозатрат при дешифруванні системи з позиції різних критеріїв якості шифру. [12]

3. Інші підходи до оцінки практичної стійкості шифрів

Асимптотичний аналіз стійкості

Цей підхід розвивається теорію складності обчислень. При дослідженні шифру оцінка його стійкості пов’язана з деяким параметром шифру, зазвичай це довжина ключа, і проводиться асимптотичний аналіз оцінок стійкості.

Вважається, що криптосистема має високу криптографічну стійкість, якщо вона виражається через довжину ключа експоненціально, і шфиросистема має низьку криптографічну стійкість, якщо стійкість виражається у вигляді многочлена від довжини ключа.

Оцінка кількості необхідної шифрматеріалу.

Даний підхід оснований не на складності обчислень при реалізації дешифрування, а на оцінці середньої кількості матеріалу, який необхідно проаналізувати криптоаналітику для розкриття шифру. Оцінка кількості необхідного крипто аналітику шифр матеріалу передставляє інтерес з тієї точки зору, яка є нижньою оцінкою стійкості шифру в змісті обчислювальної складності дешифрування.

Цей підхід застосовується в основному для оцінки стійкості поточних рандомізірованих шифрів. Особливістю пристроїв таких шифрів є те, що вони використовують для шифрування та дешифрування секретний ключ невеликого розміру, а також велику і загальнодоступну випадкову послідовність чисел (рандомізатор). Ключ визначає, які частини рандомізатора використовуються для шифрування, у той час як крипто аналітику, який не знає секретного ключа, доводиться аналізувати весь рондомізатор.

В якості прикладу такого шифру розглянемо шифр Діффі. В цьому шифрі рандомізатором є масив з 2^n випадкових двійкових послідовностей, занумерованих елементами множини V_n. Ключем є n-мірний двійковий вектор. При шифруванні з використанням ключа k двійкова послідовність відкритого тексту складається побітно (як у шифрі Вернама) з послідовністю рандомізатора під номером k. Таким чином, для дешифрування повідомлення противнику необхідно дослідити порядку 2^n біт.

Згодом Рюппель помітив, що приблизно такий же рівень стійкості досягається, коли рандомізатор містить n випадкових двійкових послідовностей, а ключ k задає набір коефіцієнтів, який визначає шифруючу послідовність як нетривіальну лінійну комбінацію послідовностей рандомізатора.

Вартісний підхід.

Цей підхід передбачає оцінку вартості дешифрування системи. Особливо він актуальний тоді, коли для дешифрування криптосистеми необхідно розробити та побудувати новий обчислювальний комплекс. Вартісний підхід корисний з точки зору співставлення матеріальних затрат на дешифрування системи і цінності інформації, що захищається системою

Прикладом реалізації такого підходу є детальне виведення оцінки вартості дешифрування алгоритму DES, виконаний Діффі і Хеллманом у зв’язку з алгоритмом розпаралелювання перебору усіх ключів DES.

На закінчення відмітимо динамічний характер оцінок криптографічної стійкості шифрів. Ці оцінки необхідно час від часу переглядати у зв’язку з розвитком обчислювальних засобів і прогресу в області розробки методів

дешифрування. Існує “емпіричний закон” розвитку обчислювальних засобів, згідно якого вважається, що обчислювальні можливості криптоаналітика подвоюються через кожні 18 місяців. [12]

Тема 17. ЗАГАЛЬНІ ПОНЯТТЯ КРИПТОАНАЛІЗУ

Криптоаналіз (від грец. криптос – прихований і аналіз) – наука про методи здобуття вихідного значення зашифрованої інформації, не маючи доступу до секретної інформації (ключа), необхідної для цього. Або процес відтворення відкритого тексту або ключа шифрування, або і того і іншого називається *криптоаналізом*. Термін був введений американським крипто графом Уільямом Ф. Фрідманом в 1920 році. Під цим терміном також розуміється спроба знайти уразливість в криптографічному алгоритмі або протоколі.

1. Типи криптоаналізу

Проведення криптоаналізу для давно існуючих та недавно створених криптоалгоритмів дуже актуально, оскільки вчасно можна сказати, що даний криптоалгоритм нестійкий і удосконалити його або замінити новим.

Стратегія, яку використовує криптоаналітик залежить від схеми шифрування і від інформації, яку він має у своєму розпорядженні.

Реальний криптоаналіз оснований на трьох речах:

- вивчення системи шифрування в цілому;
- вивчення особливостей початкового тексту;
- вивчення особливостей ключової системи.

У таблиці наведена загальна класифікація різних **типів** криптоаналізу в залежності від інформації, якою володіє криптоаналітик.

Типи криптоаналізу шифрованих повідомлень

Тип криптоаналізу	*Інформація – відома криптоаналітику*
Аналіз на основі тільки шифротексту (Ciphertext only)	- Алгоритм шифрування - Зашифроване повідомлення, яке підлягає розшифруванню
Аналіз з відомим відкритим текстом (Known Plaintext)	- Алгоритм шифрування - Зашифроване повідомлення, яке підлягає розшифруванню - Одна або декілька пар відповідних фрагментів відкритого і шифрованого тексту, створеного одним і тим самим

	секретним ключем
Аналіз з вибраним відкритим текстом (Chosen Plaintext)	- Алгоритм шифрування - Зашифроване повідомлення, яке підлягає розшифруванню - Вибраний криптоаналітиком відкритий текст і відповідний шифрований текст, з допомогою секретного ключа (тобто є можливість отримати результат зашифрування для довільно вибраного ним масиву відкритих даних)
Аналіз з вибраним шифрованим текстом (Chosen Ciphertext)	- Алгоритм шифрування - Зашифроване повідомлення, яке підлягає розшифруванню - Вибраний криптоаналітиком шифрований текст і відповідний відкритий текст, розшифрований з допомогою секретного ключа (тобто є можливість отримати результат розшифрування довільно вибраного ним зашифрованого повідомлення)
Аналіз з вибраним текстом	- Алгоритм шифрування - Зашифроване повідомлення, яке підлягає розшифруванню - Вибраний криптоаналітиком відкритий текст і відповідний шифрований текст, створений з допомогою секретного ключа - Вибраний криптоаналітиком шифрований текст і відповідний відкритий текст, розшифрований з допомогою секретного ключа

Самою складною задачею з усіх представлених в таблиці є випадок, коли в розпорядженні криптоаналітика (супротивника) є тільки зашифрований текст. У деяких випадках буває невідомий навіть алгоритм шифрування, але в основному потрібно вважати, що алгоритм шифрування супротивник знає. При таких умовах один з можливих підходів криптоаналізу полягає у простому переборі усіх можливих варіантів ключів. Однак, якщо множина всіх можливих ключів дуже велика, такий підхід стає нереальним. Тому супротивнику доводиться більше надіятися на аналіз самого шифрованого тексту, що, як правило, означає виявлення його різних статистичних особливостей. Для цього

супротивник повинен мати деякі загальні уявлення про вміст відкритого тексту, наприклад, на якій мові написаний, і т.д. Спробам розкриття при наявності у супротивника тільки шифрованого тексту протистояти найлегше, так як об'єм інформації супротивника – мінімальний. Однак досить часто аналітику буває відомо більше і нерідко такий спеціаліст має можливість перехватити одне або декілька відкритих повідомлень разом з відповідними їм шифрованими текстами.

Якщо у аналітика є можливість тим або іншим способом отримати доступ до системи, яка згенерувала повідомлення, то у такому випадку аналітик має можливість провести криптоаналіз з вибраним відкритим текстом. В загальному випадку, якщо криптоаналітик має можливість вибрати повідомлення і зашифрувати його, то при правильному виборі повідомлення для шифрування він може розгадати ключ.

Лише відносно слабкі алгоритми можуть бути розкриті при аналізі тільки шифрованого тексту.

Атаки з використанням відомого або підібраного відкритого тексту зустрічаються частіше, ніж можна подумати. Необхідною умовою для хорошого криптографічного алгоритму є можливість протистояти таким атакам. Це означає, що розсекречування деякої інформації, що передається по каналу зв'язку в зашифрованому вигляді, не повинно призводити до розсекречування іншої інформації. Атаки на основі підібраних текстів вважаються найбільш загрозливими. [2, 3]

2. Методи криптоаналізу

Постановка проблеми. Здійснюючи атаку, криптоаналітик може ставити за мету рішення наступних завдань [2, 3]:

1. Одержання відкритого тексту із зашифрованого.

2. Обчислення ключа шифрування.

У загальному випадку, друге з перерахованих завдань є істотно більш складним, чим перше. Однак, маючи ключ шифрування, криптоаналітик може згодом розшифровувати всі дані, зашифровані знайденим ключем. Така атака (у

випадку її успішного здійснення) називається повним розкриттям алгоритму шифрування. Тому перед тим як надати перевагу тому чи іншому методові шифрування важливо перевірити на власному досвіді його крипостійкість.

Кількість методів з кожним роком збільшується, а існуючі методи криптоаналізу постійно модернізуються.

Слід зазначити, що стосовно симетричної і асиметричної криптографії методи криптоаналізу різняться, оскільки в кожному випадку під криптоаналізом мають на увазі рішення математичної задачі, на основі якої і побудована яка-небудь симетрична чи несиметрична криптосистема.

Так, для *симетричних* схем виділяють наступні ***методи*** криптоаналізу:

- Диференціальний криптоаналіз (блокові, потокові шифри).
- Лінійний криптоаналіз (блокові, потокові шифри).
- Кореляційний криптоаналіз (потокові шифри).
- “Передбачати і визначати” (потокові шифри).
- Статистичний криптоаналіз (блокові, потокові шифри).
- XSL атака (блокові, потокові шифри).
- Атака “грубою силою”.

Для *асиметричних* виділяють наступні:

- Диференціальний криптоаналіз.
- Лінійний криптоаналіз.
- Атака “день народження”.
- Атака “людина посередині”.
- Атака “грубою силою”.

3. Популярні методи криптоаналізу

Розглянемо найпопулярніші методи детальніше [10, 11, 12].

1. Метод грубої сили (повного перебору)

Метод грубої сили припускає перебір всіх можливих варіантів ключа шифрування до знаходження шуканого ключа.

Нехай розмір ключа шифрування в бітах дорівнює b. Відповідно, існує 2^b варіантів ключа. Криптоаналітик повинен методично перебрати всі можливі

ключі, тобто застосувати як ключ значення 0, потім 1, 2, 3 і т.д. до максимально можливого (2^b - *1*). У результаті ключ шифрування обов'язково буде знайдений, причому в середньому такий пошук зажадає $2^b/2$, тобто 2^{b-1} тестових операцій шифрування.

Зрозуміло, що необхідно мати який-небудь критерій правильності знайденого ключа. З атакою з відомим відкритим текстом все достатньо просто – при тестуванні кожного ключа K_x шифротекст C розшифровується (у результаті виходить якесь значення M') і рівняється з відповідним йому відкритим текстом M; збіг $M = M'$ говорить про те, що шуканий ключ знайдений.

Трохи складніше з атакою на основі шифротекста. У цьому випадку необхідна наявність якої-небудь додаткової інформації про відкритий текст, наприклад:

- Якщо відкритий текст є осмисленим текстом на якій-небудь мові, перехоплений шифротекст повинен мати достатній розмір для однозначного розшифрування в осмислений текст (мінімально достатній для цього розмір називається крапкою одиничності).
- Якщо відкритий текст є бінарними даними, необхідна яка-небудь інформація про те, що він із себе представляє. Якщо перехоплюється архів, то при переборі ключів кожне значення M' повинне розглядатися як можливий заголовок архіву. При іншому потенційному M це може бути PE-заголовок файлу, що використовується в Windows, заголовок графічного файлу і т.д.
- Варто відзначити, що багато засобів шифрування інформації впроваджують у формат зашифрованого об'єкта контрольну суму відкритого тексту для перевірки його цілісності після розшифрування. Головне, що така контрольна сума може бути ідеальним еталоном в криптоаналізі, що цілком підходить для визначення вірного ключа.

2. Метод "зустрічі посередині"

Даний метод криптоаналізу заснований на "парадоксі днів народження". Відомо, що якщо вважати, що дні народження розподілені рівномірно, то в групі з 24 чоловік з імовірністю 0,5 у двох чоловік дні народження збігаються.

У загальному виді цей парадокс формулюється так: якщо $a\sqrt{b}$ предметів вибираються з поверненням із деякої сукупності розміром b , то ймовірність того, що два з них збіжаться $1-e^{-\frac{a^2}{2}}$.

Якщо множина ключів криптоалгоритму замкнуто щодо композиції, тобто для будь-яких ключів k_i і k_j найдеться ключ k_r такий, що результат шифрування будь-якого тексту послідовно на k_i і k_j однакова криптограмі зашифрована ключем k_r, тобто $F(k_j, F(k_i, x)) = F(k_r, x)$, то тоді можна скористатися цією властивістю. Нехай нам потрібно знайти ключ k_r. Тоді для знаходження ключа k_r необхідно знайти еквівалентну йому пари ключів k_i і k_j.

Нехай відомий відкритий текст x і його криптограма y. Для тексту x будуємо базу даних, що містить випадкова безліч ключів k' і відповідних криптограм $w=F(k',x)$, і впорядковуємо її по криптограмах w. Обсяг бази даних вибираємо $O(\sqrt{|\{k`\}|})$, де $|\{k`\}|$ - потужність безлічі ключів k'. Потім підбираємо випадковим образом ключі k'' для розшифровки текстів y і результат розшифровки $v=F(k'',y)$ порівнюємо з базою даних. Якщо текст v виявиться рівним однієї із криптограм w, то ключ $k'k''$ еквівалентний шуканому ключу k. Цей метод також застосовується, якщо множина ключів містить досить велику підмножину, що є напівгрупою.

Позначимо $\alpha = |\{k`\}|$ загальну кількість можливих ключів k . Тимчасова складність методу становить $O(\sqrt{\alpha}\log\alpha)$. Множник $\log\alpha$ враховує складність сортування. Необхідна пам’ять рівна $O(\sqrt{\alpha}\log\alpha)$ біт або $O(\sqrt{\alpha})$ блокам.

3. Диференційний криптоаналіз

Диференційний метод криптоаналізу був запропонований Э.Біхамом й А.Шаміром в 1990 р. Диференціальний криптоаналіз – це спроба розкриття секретного ключа блокових шифрів, які засновані на повторному застосуванні криптографічно слабкої цифрової операції шифрування r раз. При аналізі передбачається, що на кожному циклі використається свій підключ шифрування. Диференціальний криптоаналіз може використати як обрані, так і

відомі відкриті тексти. Конкретний спосіб диференціального криптоаналізу залежить від аналізуємого алгоритму шифрування.

Успіх таких спроб розкриття *r*-циклічного шифру залежить від існування диференціалів *(r-1)*-го циклу, які мають велику ймовірність. Диференціал *i*-го циклу визначається як пара $(a,b)_i$ така, що пари різних відкритих текстів *x*, *x`* с різницею *a* може привести до пари вихідних текстів *y*, *y`* після *i*-ого циклу, що мають різницю *b* (для відповідного поняття різниці). Ймовірність *i*-циклового диференціала $(a,b)_i$ – це умовна ймовірність $P(D\ y(i)=b \mid D\ x=a)$ того, що різниця D y(i) пари шифротекстів *(y,y`)* після *i*-ого циклу дорівнює *b* за умови, що пара текстів *(x,x`)* має різницю $D\ x=a$; відкритий текст x і підключи циклів $k^{(1)}$, $k^{(2)}$,...., $k^{(i)}$ незалежні і рівно ймовірні.

Основна процедура диференціального криптоаналізу *r*-циклічного шифру з використанням обраних відкритих текстів може бути наступною:

1. Шукаємо множину *(r-1)*-циклових диференціалів $(a_1,b_1)_{r-1}$, $(a_2,b_2)_{r-1}$,....,$(a_s,b_s)_{r-1}$. Впорядковуємо цю множину диференціалів по величині їхньої ймовірності.

2. Вибираємо відкритий текст *x* довільним чином і обчислюємо *x`* так, щоб різниця між *x* і *x`* була рівна a_1. Тексти *x* і *x`* шифруються на справжньому ключі і після r циклів одержуємо пари шифротекстів *y(r), y`(r)*. Припускаємо, що на виході передостаннього *(r-1)*-ого циклу різниця шифротекстів дорівнює найбільш ймовірній: $D\ y(r-1)=b\ 1$. Для трійки *(D y(r-1), y(r), y`(r))* знаходимо кожне можливе значення підключа останнього циклу $k^{(r)}$. Додаємо його до кількості появ кожного такого значення підключа $k^{(r)}$.

3. Повторюємо п.2 доти, поки одне або кілька значень підключа $k^{(r)}$ не стане з'являтися частіше інших. Беремо цей підключ або множину таких підключів як криптографічне рішення для підключа $k^{(r)}$.

4. Повторюємо пункти 1-3 для передостаннього циклу, при цьому значення *y(r-1)* обчислюються розшифруванням шифротекстів на знайденому підключі останнього циклу $k^{(r)}$. Далі діємо аналогічно, поки не будуть розкриті ключі всіх циклів шифрування.

4. Лінійний криптоаналіз

Лінійний криптоаналіз винайшов японський криптолог Міцуру Мацуі (Mitsuru Matsui). Цей метод використовує лінійні наближення перетворень, що виконуються алгоритмом шифрування. Даний метод дозволяє знайти ключ, маючи досить велику кількість пар (незашифрований текст, зашифрована текст). Розглянемо основні принципи, на яких базується лінійний криптоаналіз. Лінійний криптоаналіз базується на тому, що існує можливість замінити нелінійну функцію її лінійним аналогом.

Метою лінійного криптоаналізу є пошук лінійного рівняння виду $P_{i1} \oplus P_{i2} \oplus \ldots P_{ia} \oplus C_{j1} \oplus C_{j2} \oplus \ldots \oplus C_{jb} = K_{k1} \oplus K_{k2} \oplus \ldots \oplus K_{kc}$ (1), *де* P_n, C_n *і* K_n - *n*-і біти відкритого тексту, шифротекста й ключа відповідно.

Для довільно обраних біт відкритого тексту, шифротекста і ключа ймовірність з справедливості такого співвідношення становить біля 1/2. У тому випадку, якщо криптоаналітику вдається знайти такі біти, при яких імовірність P помітно відрізняється від 1/2, даним співвідношенням можна скористатися для розкриття алгоритму.

Це рівняння означає, що якщо виконати операцію XOR над деякими бітами незашифрованого повідомлення й над деякими бітами зашифрованого повідомлення, вийде біт, що представляє собою XOR деяких бітів ключа. Це називається лінійним наближенням, що може бути вірним з імовірністю *P*.

Рівняння складаються в такий спосіб. Обчислюються значення лівої частини для великої кількості пар відповідних фрагментів незашифрованого й зашифрованого блоків. Якщо результат дорівнює нулю більш ніж у половині випадків, то вважають, що $K_{k1} \oplus K_{k2} \oplus \ldots \oplus K_{kc} = 0$. Якщо в більшості випадків виходить $1 - K_{k1} \oplus K_{k2} \oplus \ldots \oplus K_{kc} = 1$. У такий спосіб одержують систему рівнянь, рішенням якої є ключ. Як й у випадку диференціального криптоаналізу, результати лінійного криптоаналізу повинні враховуватися при розробці алгоритмів симетричного крипто аналізу.

Досить часто лінійний криптоаналіз використовується в сукупності з атакою методом "грубої сили" – певні біти ключа виявляють за допомогою лінійного криптоаналізу, після чого виконується вичерпний пошук за можливим значенням інших біт

Лінійний криптоаналіз має одну досить корисну властивість: за певних умов співвідношення (1) може бути перетворене до наступного:

$$C_{j1} \oplus C_{j2} \oplus \ldots \oplus C_{jb} = K_{k1} \oplus K_{k2} \oplus \ldots \oplus K_{kc}$$

У даному співвідношенні повністю відсутні біти відкритого тексту, тобто за допомогою лінійного криптоаналізу можна побудувати атаку на основі тільки шифротекста, що ще більше розширює область застосування лінійного криптоаналізу, оскільки атака, що вимагає тільки перехоплений шифротекст, є найбільш практичною.

Тема 18. ДЕШИФРУВАННЯ КЛАСИЧНИХ ШИФРІВ

1. Дешифрування шифру простої заміни

Введемо математичні моделі відкритих текстів, що використовуються при криптографічному аналізі шифрів, а також виділимо певні властивості відкритих текстів. Далі використаємо деякі з цих властивостей і акцентуємо увагу на ймовірнісно-статистичних закономірностях, які мають місце у відкритому тексті й використовуються при дешифруванні шифрів простої заміни та перестановки.

Існують стійкі закономірності відкритого тексту, які слід враховувати при дешифруванні шифрів простої заміни й перестановки. Можливість дешифрування певного шифру значною мірою залежить від того, наскільки криптографічні перетворення руйнують ймовірнісно-статистичні закономірності, наявні у відкритому тексті. До найбільш стійких закономірностей відкритого повідомлення належать такі:

1) в осмислених текстах кожної природної мови різні літери зустрічаються з різною частотою, при цьому відносні частоти вживання літер у різних текстах однієї мови є близькі поміж собою. Те ж саме можна стверджувати й по частоті вживаності пар, трійок літер відкритого тексту;

2) будь-яка природна мова має так звану надлишковість, що дозволяє з великою ймовірністю "вгадувати" зміст повідомлення, навіть якщо частина літер у повідомленні є невідома.

У таблиці 18.1 зазначено відносну частоту вживаності літер абетки російської мови.

Таблиця 5.1 - Частота вживаності літер абетки російської мови

1	а - 0,062	11	к - 0,028	21	ф - 0,002
2	б - 0,014	12	л - 0,035	22	х - 0,009
3	в - 0,038	13	м - 0,026	23	ц - 0,004
4	г - 0,013	14	н - 0,053	24	ч - 0,012
5	д - 0,025	15	о - 0,090	25	ш - 0,006
6	е - 0,072	16	п - 0,023	26	щ - 0,003
7	ж - 0,007	17	р - 0,040	27	ы - 0,016
8	з - 0,016	18	с - 0,045	28	ъ,ь - 0,014
9	и - 0,062	19	т - 0,053	29	э - 0,003
10	й - 0,010	20	у - 0,021	30	ю - 0,006
				31	я - 0,018

Подібні таблиці наводяться в багатьох джерелах. Їх побудовано па підставі підрахунків частоти вживаності певних літер на великих обсягах відкритого тексту. З огляду на те, що для експериментів залучається різний вихідний матеріал, значення ймовірностей дещо відрізняються поміж собою.

Якщо упорядкувати літери за спаданням ймовірностей, то дістанемо варіаційний ряд **О, Е, А, И, Н, Т, С, Р, В, Л, К, М, Д, П, У, Я, З, Ы, Б, Ь, Г, Ч, Й, Х, Ж, Ю, Ш, Ц, Щ, Д, Ф.**

Як запам'ятати перші 10 найчастіше вживаних літер російської абетки? Пригадуєте, як запам'ятовують основні кольори у фізиці? Треба запам'ятати фразу: *Каждый охотник желает знать, где сидит фазан*. Перші літери слів фрази вказують на основні кольори. У криптографії треба запам'ятати слово СЕНОВАЛИТР — у ньому наявні всі 10 найбільш вживаних літер.

Частотна діаграма вживаності літер, зрозуміло, залежить від мови. У таблиці 18.2 подано у відсотках відносні частоти найбільш уживаних літер деяких мов.

Таблиця 18.2 - Відносні частості найбільш уживаних літер деяких мов

Мова	Частість вживаності літер, %					
Англійська	e – 12,75	t – 9,25	r – 8,50	i – 7,75	h – 7,75	o – 7,50
Французька	e – 17,75	a – 8,25	s – 8,25	i – 7,25	n – 7,25	r – 7,25
Німецька	e – 18,50	n –11,50	I – 8,00	r – 7,50	s – 7,00	a – 5,00
Арабська	a – 17,75					
Грецька	a – 14,25					
Японська	p – 15,75					
Латина	a – 11,00					
Малайська	a – 20,25					
Санскрит	a – 31,25					

Частоти вживаності знаків абетки залежать не лише від мови, але й від характеру тексту. Наприклад, у тексті з криптографії буде вища ймовірність використання літер Ф, Ш (через часто вживані слова "шифр", "криптографія"). У математичному тексті ймовірніше буде більша частота використання літери Ф (через слова "функція, "функціонал" і т. п.).

У стандартних текстових файлах найчастіше вживається символ "прогалина"; в exe-файлах найчастіше вживається символ 0; в текстах,

написаних у текстовому процесорі ChiWriter, зручному для оформлення математичних текстів, перше місце посів символ "\" — backslash.

Частотна діаграма є стійкою характеристикою тексту. З теорії ймовірностей випливає, що за дуже слабких обмежень на ймовірнісні властивості випадкового процесу є справедливий закон великих чисел, тобто відносні частоти знаків збігаються за ймовірністю зі значеннями їхніх ймовірностей

$$P\left\{\left|\frac{\vartheta_k}{N} - p_k\right| > \varepsilon\right\} \xrightarrow[N\to\infty]{} 0.$$

Це справедливо для послідовності незалежних випробувань, дія кінцевого регулярного однорідного ланцюга Маркова. Експерименти засвідчують, що це справедливо й для відкритих текстів.

З позицій сучасної криптографії, шифри перестановки і простої заміни мають істотний недолік: вони не цілковито руйнують ймовірнісно-статистичні властивості, наявні у відкритому повідомленні.

При дешифруванні тексту, зашифрованого шифром простої заміни, використовують частотні характеристики відкритого тексту. Власне, якщо підрахувати частоти вживаності знаків у шифрованому тексті, упорядкувати їх за спаданням та порівняти з варіаційним рядом ймовірностей відкритого тексту, то ці дві послідовності будуть близькі. Найімовірніше перше місце посяде прогалина, далі слідуватимуть літери О, Е, А, И.

Зрозуміло, якщо текст не є надто довгий, то не є неодмінним повний збіг. На другому місці може бути літера О, а на третьому – Е, але в кожному разі в перших і других рядах однакові літери розташовуватимуться неподалік одна від одної, й що ближче до початку (що більше ймовірність вживання знаків), тим менше буде відстань поміж знаками.

Аналогічна картина спостерігається і для пар сусідніх літер (біграм) відкритого тексту (найчастіше зустрічається біграма російського відкритого тексту – СТ). Однак для здобуття стійкої картини довжина послідовності має бути істотно більше. На порівняно невеликих відрізках відкритого тексту ця картина є трохи розмита. Більш стійкою характеристикою біграм є відсутність в

осмисленому тексті, як говорять, певних біграм; наявність заборонних біграм, котрі мають імовірність, рівну практично нулю.

Чи бачили ви коли-небудь у відкритому тексті біграми ЪЬ, “ голосна” Ь, "прогалина" Ь? Знання й використовування зазначених особливостей відкритого тексту значно полегшує дешифрування шифрів перестановки й заміни.

Розглянемо приклад дешифрування шифру простої заміни. Нехай маємо такий шифротекст:

ДОЧАЛЬ ИЬЦИО ЛИОЙО ВНЫИЮШ ХЕМВЛНХЕИ ДОСОЛЬ ЧСО ИА" ТЬЖАТСР ЬАС АКЕИОЙО ДОКЩОКЗЖАЙО КПЗ РТАЩ ТПЬЧНАР ТДОТОУН ХЕМВОРНИЕЗ ЕИМОВЛНЯЕЕ РЮУОВ БВЕД СОЙВНМЕЧАТБОГ ТЕТСАЛЮ ЫНРЕТЕС ОС ОТОУ АИИОТСАГ ЕИМОВЛНЯЕЕ АА ЯАИИОТСЕ Е РОЫЛОЦИОТСАГ РПНКАПШЯАР ДО ЫНЖУСА ТРОАГ ЕИМОВЛНЯЕЕ ДВАЦКА РТАЙО ДОКЧАВБИАЛУОПШХОА ВНЫИООУ ВНЫЕА РЕКОР ЫНЖЕЖНАЛОГ ЕИМОВЛНЯАА КОБЬЛАИСНПШИНЗ САПАМОИИНЗ САПАРЕЫЕОИИНЗ БОЛДШЭСАВИНЗ БНЦКЮГ РНК ЕИМОВЛНЯЕЕ УЛААС ТРОЕ ТДАЯЕМЕЧАТБЕА ОТОУАИИОТСЕ Е ФСК ОТОЧАИИОТСЕ ТЕПШИО РПЕЗЭС ИН РЮУОВ ЛАСОКОР ХЕМВОРЙЙВЗ ЕИМОВЛНЯЕЕ УОПШХОА ЫИНЧАИЕА ЕЛАЭС ОУЪАЛЮ Е СВАУЬАЛНЗ ТБОВОТСШ ДАВАКНЧЕ ХЕМВОРНИИОГ

Роботу слід розпочати з підрахунку частоти вживаності символів у шифрованому тексті. Після того як здійснено підрахунок, упорядкуємо символи за спаданням частот: b_1, b_2, b_3, b_4, b_5 ...

Під цим рядом слід підписати варіаційний ряд ймовірностей знаків у відкритому тексті:

О Е А И Н Т С Р В Л К М Д П У Я З Ы Б Ь Г Ч Й Х Ж Ю Ш Ц Щ ЭФ

За дуже великої довжини шифротексту, для того аби з шифрованого отримати відкритий текст, потрібно замінити B на 0, B_2 – на Е, B_3 – на А й т. д. Принаймні саме така ситуація матиме місце для найбільш імовірних літер. У

нас матеріалу недостатньо. Переглянувши шифротекст після такої заміни, бачимо, що він не читається, – отже, матеріалу насправді є замало.

Що нам залишається? Вгадувати заміну. При цьому маємо все ж враховувати статистичні особливості відкритого тексту. У шифротексті через прогалину найімовірніше позначено прогалину; через літеру О – О чи А; через Е – О чи Е, чи А; через А – Е чи А, чи И й т. д.

Можна рекомендувати виписати шифротекст, а під ним у стовпець найбільш імовірні заміни для цих літер. У нашому прикладі заміну підібрано у такий спосіб, що літеру абетки замінено саме на найбільш імовірне для неї позначення. Тому сам шифротекст у даному прикладі виписувати немає потреби. Він збігається із середнім рядком послідовності стовпців найбільш імовірних замін.

Що рідше зустрічається літера, тим більшої глибини треба брати стовпець, аби була впевненість, що в стовпці міститься знак відкритого тексту. У нашому випадку стовпці взято однаковою глибиною в п'ять символів, але виписано їх лише для найчастіше вживаних літер.

При дешифруванні без використання засобів автоматизації далі треба вгадувати заміну. Якщо пильно придивитися до тексту, то зробити це не є дуже складно. В тексті є слово АА з двох часто вживаних літер. Що це може бути за слово? У російській мові немає слів ОО, ИИ, НН тощо. Так, перебираючи можливі слова, ми віднайдемо одне слово ЕЕ і дійдемо висновку, що літеру Е було замінено па А. Надто часто в шифротексті слова закінчуються біграмами ЕЕ. В російській мові типовими закінченнями є сполучення ЕЕ, ИИ. З огляду на те, що заміну для літери Е ми вже вгадали, дійдемо висновку, що в шифротексті літери И замінено на Е. Тоді усюди в шифротексті можна провести зворотну заміну. Тепер ми вже можемо вгадувати окремі слова. У такий спосіб, добряче поламавши голову, ми, як у грі "Поле чудес", зрештою відновимо увесь текст.

У стовпцях найбільш імовірних замін літери, котрі відповідають правильним зворотним замінам, має бути позначено як великі. Прочитавши відкритий текст, ви переконаєтесь, що він являє собою одну з кількох

пропозицій з книги С.А. Дориченка та В.В. Ященка «25 этюдов о шифрах», М., 1994.

Для дешифрування в автоматизованому режимі насамперед слід занести до пам'яті комп'ютера словник відповідної мови.

Програма дешифрування повинна, переглядаючи шифротекст, здійснювати спробні зворотні заміни, у припущенні, що на даному фіксованому місці у відкритому тексті перебувало слово, що перевірялося. Після кожної заміни програма частково відновлює ключове підставляння, частково розшифровує текст і відкидає варіант підставляння, якщо в певному місці відновленого тексту виникає літеросполучення, якого не може бути у відкритому тексті розглядуваної мови. Після відновлення певної кількості замін у тексті виникають ділянки, в яких віднайдено значну кількість літер. Решта літер добираються шляхом перебирання словника і підставляння до тексту слів, які не суперечать відновленим попередньо замінам.

Слід зазначити, що практичних навичок з дешифрування шифру простої заміни можна набути лише після проведення самостійних дослідів з дешифрування. [5]

2. Дешифрування шифру перестановки

Для відновлення відкритого тексту шифру перестановки слід переставити стовпці у такий спосіб, аби в рядках з'явився осмислений текст.

Розглянемо приклад дешифрування шифру перестановкою восьми стовпців.

Нехай шифротекст має вигляд, наведений в таблиці 18.3.

Таблиця 18.3 – Шифротекст російською мовою

1	2	3	4	5	6	7	8
п	а	я		в		и	м
0	ч	ш	г		У		е
е	б	ж	л		е		0
м		ч	–	0	т	0	я
	е	г	е		У	с	Щ
	а	к	ь	З	а	т	т
я	Р	е		е	п		ь
	ю	З	в	а	н	в	
0	и	а	в	е	ш	л	
	е	е	я	м		п	н
ь	Р	Р	н	З	е	е	е
З	а	м	а	н		а	к
ч	с	т	а		ь	а	н
0	я	л	м		а	л	
0	ь	ч	х	т	а	т	
в		е	0	а	я	е	я
0	е	Р	м	т	ь	е	
д	с	г	ы		0	а	т
е	б	в	н		ы		
	а	у	и	н	З	н	л
	г	и	а	0	к	к	д
	а	0	б	д	г	н	
	ж	а	У	е	д	я	д
х	л	и		е	м	0	а
к	Р	т	д		ь	0	е
	ь	х	в	т	0	н	
р	л	е		е	д	а	ю
р		З	е	в		е	д
ш		в	а	е	н	е	н
т	и	и	е	в			д
		в		с	д		

Зіставимо перестановці стовпців таблицю 8х8; при цьому поставимо на перетині і-того рядка й j-того стовпця одиницю, якщо j-ий стовпець після зворотної перестановки має слідувати за і-тим. Наше завдання – відновити таблицю, відповідну правильному переставленню стовпців.

Далі попарно прилаштовуватимемо один стовпець до одного. Якщо при цьому в певних рядках виникатимуть заборонені біграми, то стовпці не зможуть у відкритому тексті слідувати один за одним, – й відповідна клітинка закреслюється. В нашому прикладі шостий стовпець не може слідувати за четвертим, тому що інакше в тексті в першому рядку йтимуть дві прогалини

поспіль. Переглянемо, наприклад, шостий рядок. Якби четвертий стовпець слідував за першим, то в тексті були б слова, які розпочинаються з Ь. Після перегляду всіх рядків ми отримаємо таблицю 18.4.

Таблиця 18.4 – Дешифрування шифру перестановки

	1	2	3	4	5	6	7	8
1	X	X		X	X	X		X
2		X		X		X		
3			X					X
4	X	X		X		X	X	X
5	X				X	X	X	X
6	X	X		X		X	X	
7	X			X	X	X	X	X
8	X	X			X	X	X	X

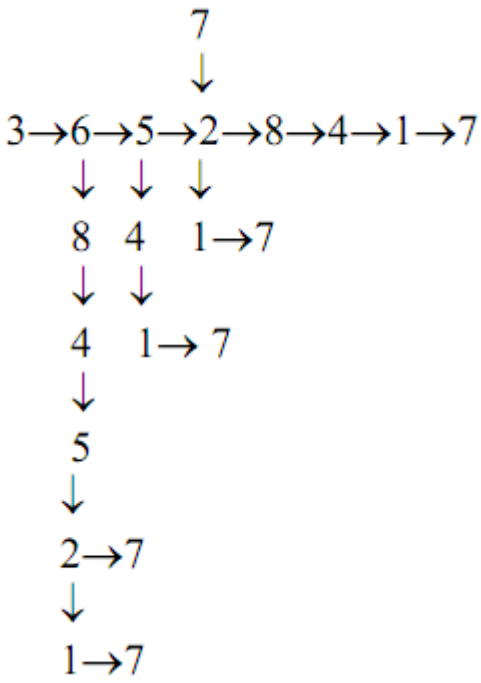
Якби текст був довшим і мав більше рядків, то в кожному рядку й у кожному стовпці залишилося б лише по одній незакресленій клітинці – і перестановку було б відновлено. Стосовно нашої таблиці ми можемо лише стверджувати, що шостий стовпець слідує за третім (позначимо цю подію як 3 —» 6), якщо шостий стовпець не є останнім. Для шостого стовпця можливі два варіанти продовження:

$$\begin{array}{ccccc} & & 8 & & \\ & & \downarrow & & \\ 3 & \rightarrow & 6 & \rightarrow & 5 \end{array}$$

Нам треба розглянути обидва варіанти й спробувати відкинути хибний. Якщо це не вдасться, то слід продовжувані обидва варіанти:

$$\begin{array}{ccccccccc} & & 8 & \rightarrow & 4 & & 1 & & \\ & & \uparrow & & & & \uparrow & & \\ 3 & \rightarrow & 6 & \rightarrow & 5 & \rightarrow & 2 & \rightarrow & 7 \end{array}$$

У підсумку отримаємо певне дерево можливого слідування стовпців у відкритому тексті:



Кожній гілці дерева відповідає певна перестановка стовпців. Далі перевіряємо кожен варіант на осмисленість і дістаємо правильний варіант :

$$3\to6\to5\to2\to8\to4\to1\to7$$

Зауважимо, що не обов'язково було будувати дерево до кінця. Наприклад, гілку $3\to6\to8\to4\to5$ можна було відкинути одразу. Хіба можна визнати за осмислений фрагмент тексту

3	6	8	4	5
я		м		в
ш	У	е	г	
ж	е	0	л	
ч	т	я		0
г	У	Щ	е	3
к	а	т	ь	е
е	а	т		а

Така процедура відкидання гілок була б необхідна, якби рядків було дещо менше й дерево було, відповідно, набагато більш розгалуженим. Запропоновану процедуру легко автоматизувати і зробити придатною для реалізації на ЕОМ.

Алгоритм дешифрування має складатися з таких етапів.

1. Попередня робота. Аналізуючи потужний обсяг відкритих текстів, побудувати множину заборонених біграм.
2. Попередня робота. Скласти словник усіляких *v*-грам для v = 2, 3, *d*, які можуть зустрітися у відкритому тексті. Число *d* обирається виходячи з

можливостей обчислювальної техніки. Побудувати таблицю 8x8. При цьому перебираються послідовно всі помітні біграми і для кожної спробуваної біграми – послідовно всі рядки. Якщо хоча б в одному рядку перший символ біграми зустрічається в і-тому стовпці, а другий – у *j*-тому, то клітинка *іxj* таблиці закреслюється.

3. Обрати певний стовпець за вихідний.
4. Розпочати процедуру побудови дерева шляхом прилаштовування до вихідного стовпця усіх варіантів стовпців.
5. Для кожного отриманого варіанта додати ще один з решти стовпців. Якщо хоча б в одному з рядків таблиці зустрінеться триграма, відсутня у словнику розміщених триграм, – то варіант від кидається.
6. Для кожного з не відкинутих варіантів додаємо ще одного стовпця і проводимо відкидання хибних варіантів за словником дозволених чотириграм.

Якщо словник побудовано лише для $d < 3$, то відкидання проводиться шляхом перевірки на припустимість триграм, які зустрілися в останніх трьох стовпцях кожного рядка. Продовжуємо цей процес до здійснення повної перестановки. Нижче наведено відновлений для нашого прикладу текст.

Таблиця 18.5 – Відновлений текст

	1	2	3	4	5	6	7	8
1	я		в	а	м		п	и
2	ш	у		ч	е	г	0	
3	ж	е		б	0	л	е	
4	ч	т	0	1	я		М	0
5	г	У		е	Щ	е		с
6	к	а	З	а	т	ь		т
7	е	п	е	Р	ь		я	
8	З	н	а	ю		в		в
9	а	ш	е	и		в	0	л
10	е		м	е	н	я		п
11	Р	е	З	Р	е	н	ь	е
12	м		н	а	к	а	З	а
13	т	ь		с	н	а	ч	а
14	л	а		я		м	0	л
15	ч	а	т	ь		х	0	т
16	е	л	а		п	0	в	е
17	Р	ь	т	е		м	0	е
18	г	0		с	т	ы	д	а
19	в	ы		б		н	е	
20	у	З	н	а	л	и		н
21	е	к	0	г	д	а		к
22	0	г	Д	а		б		н
23	а	д	е	ж	д	У		я
24	и	м	е	л	а		х	0
25	т	ь		Р	е	д	к	0
26	х	0	т	ь		в		н
27	е	д	е	л	ю		Р	а
28	З		в		д	е	Р	е
29	в	н	е		н	а	ш	е
30	и		в	и	д	е	т	ь
31	в	а	с					

Зауваження стосовно ступеня неоднозначності відновлення відкритого тексту. В обох прикладах нам вдалося, хоча й за певних зусиль, відновити відкриті тексти. Робота з дешифрування значно полегшується зі збільшенням довжини шифротексту (чи, як говорять, обсягу наявного в розпорядженні матеріалу перехоплення). Якби обсяг матеріалу було збільшено кількаразово, то в другому прикладі в таблиці 8x8 з великою ймовірністю було б закреслено всі клітинки, окрім відповідних справжньому порядку слідування один за одним стовпців у відкритому тексті. У першому прикладі варіаційний ряд частоти вживаності літер відкритого тексту майже повністю збігався б з варіаційним рядом ймовірностей знаків відкритого тексту і нам залишалося підписати один варіаційний ряд під іншим і здійснити зворотну заміну. Окремі

помилки легко усувалися б при першому ж прочитуванні дешифрованого варіанта тексту. З іншого боку, при зменшенні обсягу матеріалу завдання дешифрування істотно ускладнюється. Насправді, уявіть собі, що вам запропоновано дешифрувати зашифрований за допомогою простої заміни шифртекст, складений з одного першого слова нашого першого прикладу – ДОЧАЛЬ. Для кожного слова з шести різних літер віднайдеться проста заміна, застосовуючи яку ми отримаємо даний шифротекст. Проблема полягає не в тому, як віднайти ключову підстановку, а в тому, як обрати з численної кількості варіантів відновлених відкритих текстів саме той, котрий було зашифровано.

Отже, важливою характеристикою ефективності криптографічного захисту є ступінь неоднозначності відновлення відкритою тексту за шифрованим. Подамо якісне означення цього поняття. Під ступенем неоднозначності природно розуміти математичне сподівання кількості варіантів відкритих текстів, які може бути отримані за допомогою застосовування алгоритму дешифрування. Якщо припустити, що за випадкового вибору хибного ключа (ключа, розбіжною зі справжнім ключем, із застосуванням якого було отримано шифротекст, відновлений при розшифровуванні) і розшифровування на цьому випадковому хибному ключі, розшифрований текст не відрізняється від випадкового тексту, то ймовірність того, що отриманий текст довжини N буде осмисленим (за класичним означенням ймовірностей), дорівнює відношенню кількості сприятливих завершень – кількості $A(N)$ осмислених відкритих текстів довжини N до загальної кількості текстів довжини N, котра дорівнює n^N, де n – кількість літер в абетці відкритих текстів. Для того аби дістати ступінь неоднозначності, слід помножити отримане відношення на кількість K варіантів ключів, звідки дістаємо формулу

$$r = r(N) = KA(N)n^{-N}$$

для обчислення ступеня неоднозначності L відновлення відкритого тексту. Отже, шифрування забезпечує надійний захист повідомлення, якщо неоднозначність відновлення відкритого тексту за шифрованим є надто велика.

Ступінь неоднозначності дозволяє класифікувати шифри за ступенем їхньої криптографічної стійкості. Якісні міркування тут є такі. Якщо неоднозначність дешифрування (можлива кількість отриманих відкритих текстів) є *r(N),* а мета дешифрування визначена отриманням відкритого справжньою тексту, то ймовірність отримання саме його оцінюється величиною *l/r(N)*. Шифри, для яких *r(N) = A(N),* називаються ***абсолютно (теоретично) стійкими*** (для абсолютно стійкого шифру застосування будь-якого алгоритму дешифрування не надає жодної інформації про відкритий текст). До останньої формули, її висновку та її трактувань слід ставитися надто обережно. Насправді для отримання абсолютної стійкості шифру досить мати n^N, ключів.

Шифри, які теоретично не є стійкими, для яких надійність криптографічного захисту забезпечується надто великою складністю реалізувування відомих алгоритмів дешифрування, називають ***практично стійкими.***

Шифри, котрі не належать до класів теоретично стійких чи практично стійких шифрів, називають ***шифрами тимчасової стійкості.***

Кількість осмислених відкритих текстів довжини *N* для великих значень *N* зазвичай вважається 2^{HN}, де *H* – ентропія (в бітах) на знак відкритого тексту. Для літературного тексту, як засвідчують експериментальні розрахунки, величина *H* є наближена до одиниці. Для більш формалізованих текстів (організаційно-розпорядчих листів, документів тощо) величина *H* перебуває в межах від 0,5 до 0,8.

Повертаючись до аналізу шифру простої заміни, зауважимо, що ступінь неоднозначності відновлення відкритого тексту для цього шифру має вигляд

$$r(N) = \frac{n!2^{HN}}{n^N},$$

де *H* – ентропія на літеру відкритого тексту.

Шифр простої заміни є теоретично стійким при шифруванні літературного тексту *(n* = 32) довжиною в одне-два слова і забезпечує лише тимчасову стійкість за великого обсягу матеріалу,

Експерименти засвідчують, що практичне дешифрування шифру простої заміни для літературного тексту є можливе в разі, якщо довжина шифротексту вдвічі-втричі перевищує розмір абетки відкритою тексту, тобто дорівнює 60... 100 літерам. [5]

Тема 19. ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ КРИПТОГРАФІЧНИХ СИСТЕМ

На сучасному етапі розвитку існують наступні проблеми криптографічних систем:

- перетворення повідомлень великого об’єму;
- застосування методів стиснення і кодування даних;
- розподіл сеансових ключів;
- знаходження способів вирішення існуючих *NP*-повних задач.

Перейдемо до розгляду суті проблеми. [4]

1. Перетворення повідомлень великого об’єму

Розвиток мереж передачі даних і мультимедійних засобів гостро поставив питання захисту повідомлень великого об’єму.

Мова йде про криптографічне перетворення текстових повідомлень. Тепер починають застосовуватися інформаційні технології, при яких відбувається передача великих об’ємів даних в реальному масштабі часу. До таких технологій можна віднести системи відео конференцій, відео- та голосову пошту, факсимільну та модемну пошту.

В подібних системах необхідно забезпечити надійний захист інформації, що передається з однієї сторони, та високу швидкість перетворення та передачі – з іншої.

Вирішенням такої проблеми може бути застосування методів шифрування даних. Ці методи володіють цілим рядом переваг.

По-перше, вони можуть працювати з блоками даних будь-яких розмірів. При використанні блокових методів шифрування могла виникнути штучна затримка, викликана необхідністю очікування заповнення блока перед початком його перетворення.

По-друге, потокові методи шифрування володіють високою швидкістю перетворення даних. [4]

2. Застосування методів стиснення і кодування даних

Коли розглядалися питання теоретичної та практичної стійкості криптосистем Шеннон показав, що чим вища збитковість тексту, тим більша ймовірність того, що він буде розкритий криптоаналітиком противника.

Відомо, що будь-яка мова володіє певною збитковістю. Наприклад, російська мова менш збиткова ніж англійська. Знаючи заздалегідь частоту появи окремих символів у відкритому тексті, криптоаналітик зможе прочитати шифрований текст достатньо великого об’єму.

Для вирішення цієї проблеми необхідно використовувати в комплексі з методами шифрування також методи стиснення та кодування. Перераховані методи перетворення даних ефективно доповнюю один одного, а їх сумісне застосування дозволить використати відкриті канали зв’язку для передачі повідомлень.

Методи стиснення дозволять значно зменшити об’єм даних, що передаються або даних, що зберігаються, методи кодування – захистять інформацію, що передається, від перешкод і помилок в каналах зв’язку. Шифрування захистить інформацію від прочитання її посторонніми особами.

Таким чином, спочатку відкритий текст повинен стискатися, потім його необхідно зашифрувати, а далі уже закодувати.

Крім того, застосування подібного підходу здатне вирішити проблему криптографічного перетворення повідомлень великого об’єму. [4]

3. Розподіл сеансових ключів

У великих інформаційних системах на сьогоднішній день високу актуальність має проблема розподілу сеансових ключів. Частково ця проблема може бути вирішена за рахунок використання криптосистем з відкритими ключами. Однак, сучасні асиметричні алгоритми володіють великою обчислювальною складністю і їх застосування не завжди виправдане.

Існує два підходи до вирішення даної проблеми: використання центра розподілу ключів (ЦРК) та використання “блукаючих ключів”. Кожен з підходів має свої переваги та недоліки. Перейдемо до їх розгляду.

Перший підхід пропонує наявність у складі інформаційної системи окремого компонента, який відповідає за генерацію і видачу ключів сеансу кожному абоненту. Перед початком роботи сеансу обидві сторони звертаються до такого компонента за ключами, необхідними для роботи.

Переваги такого підходу полягають в наступному: зосередження способів генерації сеансових ключів в одному місці, спрощення адміністрування роботи системи.

Недоліками є висока уразливість системи у випадку виходу з ладу ЦРК. Крім того, зловмисник може створити засіб, який імітує роботу ЦРК,

Другий підхід пропонує заміну ключів взаємодії за деяким правилом, яке відоме лише двом сторонам, що обмінюються. Ключі можуть змінюватися як на початку нового сеансу, так і періодично протягом одного сеансу.

Переваги підходу – більш простий спосіб встановлення зв'язку сторонами.

Недоліки – у випадку порушення правил заміни ключів на одній із сторін, взаємодія сторін буде перервана. Крім того, правила заміни ключів зберігаються у багатьох місцях і можуть бути обчисленні криптоаналітиком противника. [4]

4. Знаходження способів вирішення існуючих *NP*-повних задач

За останнє десятиліття криптографія з відкритим ключем перетворилася з нової концепції в опору криптографічної технології. В майбутньому збережеться тенденція росту кількості випадків застосування асиметричних алгоритмів перетворення. Однак, висока популярність подібних систем притягує до них увагу багатьох криптоаналітиків.

На жаль, технологічна база криптографії з відкритим ключем є недостатньо розвиненою. За виключенням схеми Макеліса, яка була розроблена з ціллю протистояння відомим методам криптоаналізу, фактично всі алгоритми цифрового підпису і криптографії з відкритим ключем в якості *NP*-повної задачі використовують операцію піднесення до степеня по модулю добутку простих чисел.

Таким чином, враховуючи великі досягнення у вирішенні задач розкладу і обчисленні дискретних алгоритмів, вони стають все більш вразливими.

Виходячи з вище сказаного слідує, що при розробці нових криптосистем великого значення потрібно приділяти вибору *NP*-повній задачі. [4]

5. Заключення

На сьогоднішній день існують апробовані і гарно зарекомендувавші себе методи криптографічного захисту даних. Їх стійкість до різного роду атак доведена математично або ж зводиться до вирішення складної математичної задачі. Здавалося б, застосування таких методів в інформаційних системах повинно забезпечити конфіденційність інформації, що захищається.

Тим не менш, у засобах масової інформації періодично з'являються повідомлення про знайдені "дирки" в системах захисту інформації або факти "взлому" таких систем.

Чому так відбувається? Намагатимемося пояснити це.

Як було відмічено раніше, методи криптографічного захисту є лише малою частиною систем захисту інформації. Використання в таких системах стійких методів зовсім не гарантує того, що подібна не може бути взломана порушником.

Для того, щоб побудувати захищену інформаційну систему розробнику необхідно бути підготовленим не гірше потенційного порушника. Розробник повинен знати про всі вразливі місця інформаційних систем та причини їх виникнення. Іншими словами, знаючи способи перемоги систем захисту інформації, легше протидіяти подібним намаганням. Тому особливої важливості набуває аналіз проведених атак на захищені системи з ціллю визначення причин їх ненадійності.

До основних причин ненадійності методів криптографічного захисту інформації можуть бути віднесені:

- експортні обмеження крипто алгоритмів;
- використання несертифікованих засобів захисту інформації;

- помилки, закладені при проектуванні та реалізації систем захисту інформації;
- людський фактор.

Розглянемо більш детально основні причини ненадійності криптографічних методів захисту інформації.

Експортні обмеження криптоалгоритмів

Ця причина пов'язана з експортом крипто алгоритмів або з необхідністю купівлі на них патенту. Наприклад, в США заборонений експорт крипто алгоритмів, що використовують довжину ключа більш ніж 40 біт. Очевидно, що такі алгоритми не можуть вважатися надійними в теперішній час.

В якості прикладів програмних продуктів, що підтвердженні експертним обмеженням можуть бути версії браузерів Інтернет Netscape Navigator фірми Netscape Communications та Internet Explorer фірми Microsoft. У цих продуктах реалізований метод шифрування зі 128-бітним ключем для користувачів всередині США і з 40-бітним ключем для всіх інших.

Інформація, зашифрована таким продуктом з довжиною ключа 40 біт, може бути отримана шляхом повного перебору ключів.

Використання несертифікованих засобів захисту інформації

Сьогодні на багатьох сайтах Інтернету користувачу пропонується велика кількість засобів криптографічного захисту інформації. Рекламні проспекти стверджують, що запропонований продукт (найчастіше вільно розповсюджуваний) забезпечить надійний та гарантований захист ваших даних.

Виникає питання: яким чином можна визначити, чи є алгоритм надійним? Адже результатом роботи як стійкого, так і недостатньо стійкого крипто алгоритму буде незрозуміла для користувача послідовність символів. За зовнішнім виглядом вихідної послідовності практично неможливо визначити, який із алгоритмів був використаний у пропонованому засобі.

Розробник подібного засобу міг використовувати замість перевіреного стійкого алгоритму перетворення свій власний. Крім того, у засіб могли бути внесені не документовані можливості, універсальні ключі, що відкривають будь-який текст і т.п.

З вищевикладеного стає очевидним, що використання несертифікованих засобів захисту інформації може бути фактором ненадійності захищаючої системи.

Помилки, закладені при проектуванні та реалізації систем захисту інформації

Застосування криптографічних методів захисту інформації передбачає використання ключів перетворення даних. Тільки за допомогою ключа перетворене повідомлення може бути прочитане.

При використанні методів криптографії у інформаційних системах гостро постає питання про збереження ключової інформації. Адже подібна інформація буде являти собою великий інтерес для потенційних порушників.

Багато інформаційних систем були взломані тільки тому, що не було виділено необхідної уваги питанням зберігання ключів, паролів і т.д. Можна навести такий приклад: поряд із закритими на замок дверима на цвяшку висить ключ від замка. Неважко здогадатися, що порушник зможе подолати таку систему захисту.

Необхідно також пам’ятати, що проектування та реалізація систем захисту інформації реалізовується людиною, а людині властиво помилятися. Людина може невірно реалізувати алгоритм перетворення даних, спростити алгоритм та наробити багато інших, не менш критичних помилок.

Яскравим прикладом системи, при розробці якої було закладено багато помилок (в тому числі й у підсистему захисту інформації) може бути операційна система Windows.

Людський фактор

В будь-якій критичній системі помилки обслуговуючого персоналу є самими дорогими та приносять найбільшу шкоду. Що стосується методів криптографії, то непрофесійні дії користувачів зводять нанівець стійкий крипто алгоритм і саму коректну його реалізацію.

В першу чергу це стосується використання паролів. Зазвичай, користувачі хочуть використовувати короткі та осмислені паролі. Їх легше запам’ятати користувачу, але їх і легше розкрити противнику. Застосування ж довгих і

беззмістовних паролів призводять до виграшу з точки зору криптостійкості, однак, дуже часто людина не може запам’ятати такий пароль, записує його на папері, який потім може бути загублений чи викрадений порушником.

Крім того, яка б не була надійна система захисту інформації, самим вразливим її місцем є надто довірливий персонал. Зараз порушники не намагаються взламати систему в “лоб”, а стараються застосовувати прийоми соціальної інженерії.

Розглянувши основні принципи ненадійності інформаційних систем можна зробити висновок, що застосування одних тільки методів криптографії не гарантує надійного захисту інформації. Методи криптографічного захисту повинні використовуватися у комплексі з іншими мірами захисту інформації. [4]

ЛАБОРАТОРНІ РОБОТИ (теми та завдання)

Лабораторна робота № 1, 2

Тема: Класичні шифри перестановки

Завдання:

Рівень 1 (зашифрувати слово)

1. Розглянути звичайну перестановку, продемонструвати застосування на прикладі.
2. Розглянути звичайну рядково-стовпчикову табличну перестановку, продемонструвати застосування на прикладі.
3. Розглянути рядково-стовпчикову табличну перестановку із застосуванням ключа стовпців, продемонструвати застосування на прикладі.
4. Розглянути рядково-стовпчикову табличну перестановку із застосуванням ключа рядків, продемонструвати застосування на прикладі.
5. Розглянути рядково-стовпчикову табличну перестановку з двома ключами, продемонструвати застосування на прикладі.
6. Розглянути табличну перестановку з використанням трафарету, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати рядково-стовпчикову табличну перестановку з двома ключами.
2. Запрограмувати табличну перестановку з використанням трафарету.

Лабораторна робота № 3, 4

Тема: Класичні шифри заміни

Завдання:

Рівень 1 (зашифрувати слово)

1. Розглянути шифр Гронсфельда, продемонструвати застосування на прикладі.

2. Розглянути гомофонічну заміну, продемонструвати застосування на прикладі.
3. Розглянути шифруючи таблицю Трисемуса, продемонструвати застосування на прикладі.
4. Розглянути біграмний шифр Плейфера, продемонструвати застосування на прикладі.
5. Розглянути біграмний двотабличний шифр, продемонструвати застосування на прикладі.
6. Розглянути координатні заміни, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати біграмний шифр Плейфера.
2. Запрограмувати біграмний двотабличний шифр.

Лабораторна робота № 5, 6

Тема: Прості числа. Тестування числа на простоту

Завдання:

Рівень 1

1. Розглянути утворення послідовності простих чисел.
2. Як тестуються числа на простоту?
3. Розглянути та продемонструвати на прикладі ймовірнісний тест Ферма.
4. Розглянути та продемонструвати на прикладі ймовірнісний тест Соловай-Штрасена.
5. Розглянути та продемонструвати на прикладі ймовірнісний тест Мілера-Рабіна.
6. Розглянути властивості брехунців.

Рівень 2

1. Запрограмувати ймовірнісний тест Ферма.
2. Запрограмувати ймовірнісний тест Соловай-Штрасена.
3. Запрограмувати ймовірнісний тест Мілера-Рабіна.

Лабораторна робота № 7

Тема: Арифметичні основи криптографії

Завдання:

Рівень 1

1. Визначити набір лишків для числа.
2. Обчислити вираз використовуючи правила спрощеного обчислення остачі.
3. Виконати пошук НСД, утворюючи таблицю.
4. Виконати дискретне піднесення до степеня.
5. Розв’язати діофантове рівняння.

   (Варіанти завдань містяться у додатку 1.)

Рівень 2

1. Запрограмувати дискретне піднесення до степеня.
2. Запрограмувати розв’язання діофантового рівняння.

Лабораторна робота № 8

Тема: Шифри, що використовують аналітичні перетворення

Завдання:

Рівень 1

1. Розглянути афінну систему моноалфавітної заміни, продемонструвати застосування на прикладі.
2. Розглянути шифр Віженера, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати афінну систему моно алфавітної заміни.
2. Запрограмувати шифр Віженера.

Лабораторна робота № 9

Тема: Шифри з використанням гамування

Завдання:

Рівень 1

1. Розглянути шифр звичайного накладання двійкової гами, продемонструвати застосування на прикладі.
2. Розглянути комбінований шифр Френдберга, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати шифр звичайного накладання двійкової гами.
2. Запрограмувати комбінований шифр Френдберга.

Лабораторна робота № 10, 11

Тема: Стандарти шифрування даних DES, GOST (ГОСТ 28147-89)

Завдання:

Рівень 1

1. Розглянути стандарт шифрування даних DES, продемонструвати застосування на прикладі.
2. Розглянути стандарт шифрування даних GOST (ГОСТ 28147-89), продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати стандарт шифрування даних DES.
2. Запрограмувати стандарт шифрування даних GOST (ГОСТ 28147-89).

Лабораторна робота № 12

Тема: Основні поняття асиметричної криптографії. Система Діффі-Хеллмана та Ель – Гамаля

Завдання:

Рівень 1

1. Розглянути асиметричну систему Діффі-Хеллмана, продемонструвати застосування на прикладі.
2. Розглянути асиметричну систему Ель-Гамаля, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати асиметричну систему Діффі-Хеллмана.
2. Запрограмувати асиметричну систему Ель-Гамаля.

Лабораторна робота № 13

Тема: Хеш-функція та її застосування

Завдання:

Рівень 1

1. Розглянути хеш-функцію MD5, продемонструвати застосування на прикладі.
2. Розглянути хеш-функцію SHA, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати будь-яку хеш-функцію на вибір.

Лабораторна робота № 14

Тема: Ідентифікація та аутентифікація об'єкта

Завдання:

Рівень 1

1. Розглянути протокол аутентифікації з нульовою передачею знань (У.Фейге, А.Фіат А.Шамир), продемонструвати застосування на прикладі.
2. Розглянути протокол аутентифікації Гіллоу-Куіскуотера, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати протокол аутентифікації з нульовою передачею знань (У.Фейге, А.Фіат А.Шамир).
2. Запрограмувати протокол аутентифікації Гіллоу-Куіскуотера.

Лабораторна робота № 15

Тема: Електронний цифровий підпис

Завдання:

Рівень 1

1. Розглянути цифровий підпис Ель-Гамаля, продемонструвати застосування на прикладі.
2. Розглянути цифровий підпис DSA, продемонструвати застосування на прикладі.
3. Розглянути цифровий підпис ГОСТ Р34.10-94, продемонструвати застосування на прикладі.

Рівень 2

1. Запрограмувати створення цифрового підпису Ель-Гамаля.
2. Запрограмувати створення цифрового підпису DSA.
3. Запрограмувати створення цифрового підпису ГОСТ Р34.10-94.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Алгоритм обчислення хеш-функції MD5: Методичні вказівки до лабораторної роботи з курсу „Проектування засобів захисту інформації” для студентів напрямків 7.091501, 8.091501 “Комп’ютерні системи та мережі”, 8.091502 “Системне програмне забезпечення”, 8.091503 “Спеціалізовані комп’ютерні системи” / Укладачі: В.М. Грига, В.М. Сокіл – Львів: Національний університет “Львівська політехніка”, 2006, 11 с.
2. Баричев С.Г., Серов Р.Е. Основы современной криптографии. – М.: “Горячая линия – Телеком”, 2001. – 198 с.
3. Введение в криптографию / Под общ. ред. В.В. Ященко. – СПб.: Питер, 2001. – 288 с.: ил.
4. Грездов Г.Г. Современные методы криптографической защиты информации (Обзор по материалам открытой печати). – Киев. 2002.– 38 с.
5. Захарченко М.В., Йона Л.Г., Щербина Ю.В., Онацький О.В. Розвинення криптології та її місце в сучасному суспільстві: Навч. посібник. – Одеса: ОНАЗ ім. О.С. Попова, 2003. – 80 с.
6. Медведев М.Г. Ймовірнісні тести на простоту
7. Монастриський Л. С. Методичні вказівки до виконання лабораторних робіт з розділу комп’ютерна криптологія з курсу “Системи і методи захисту інформації” для студентів університету природничих спеціальностей. – Львів: Видавничий центр ЛНУ ім. Івана Франка, 2003. – 42 с.
8. Моргун О.М. Кріптографічні методи захисту інформації: Навч. посібник. – Черкаси: АПБ ім. Героїв Чорнобиля, 2008. – 97с.
9. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. – М.: ДМК, 2000. – 448 с., ил.
10. Саломаа А. Криптография с открытым ключом: Пер. с англ. – М.: Мир, 1995. – 318 с.
11. Смарт Н. Криптография. – М.: Техносфера, 2005, – 528 с.

12. Фомичев В.М. Дискретная математика и криптология. Курс лекций/ Под общ. ред. д-ра физ.-мат. н. Н.Д. Подуфалова. – М.: Диалог-Мифи, 2003. – 400 с.
13. Харин Ю.С. и др. Математические основы криптологии: Учебн. пособие / Ю.С. Харин, В.И. Берник, Г.В. Матвеев. – Мн.: БГУ, 1999. – 319 с.: ил.
14. Шнайер Б. Прикладная криптография, 2-е издание, Протоколы, алгоритмы и исходные тексты на языке С. – М.: “Триумф”, 2001. – 610 с.

ДОДАТОК 1

1. Визначити набір лишків для числа
   1) a=15 по модулю m=7;
   2) a=12 по модулю m=8.
2. Визначити набір лишків для числа
   1) a=14 по модулю m=6;
   2) a=11 по модулю m=7.
3. Визначити набір лишків для числа
   1) a=18 по модулю m=11;
   2) a=17 по модулю m=8.
4. Визначити набір лишків для числа
   1) a=19 по модулю m=5;
   2) a=22 по модулю m=3.
5. Визначити набір лишків для числа
   1) a=19 по модулю m=5;
   2) a=22 по модулю m=3.
6. Визначити набір лишків для числа
   1) a=19 по модулю m=5;
   2) a=22 по модулю m=3.
7. Обчислити використовуючи правила спрощеного обчислення остачі.
   1) (25+39) mod 4;
   2) (11*7) mod 3;
   3) 8^3 mod 4.
8. Обчислити використовуючи правила спрощеного обчислення остачі.
   4) (37+41) mod 6;
   5) (13*9) mod 7;
   6) 11^5 mod 4.
9. Обчислити використовуючи правила спрощеного обчислення остачі.
   1) (18+27) mod 5;
   2) (19*14) mod 2;
   3) 6^4 mod 3.

10. Обчислити використовуючи правила спрощеного обчислення остачі.
    1) (29+28) mod 6;
    2) (21*6) mod 5;
    3) 7^3 mod 2.
11. Обчислити використовуючи правила спрощеного обчислення остачі.
    1) (20+9) mod 2;
    2) (19*11) mod 6;
    3) 9^2 mod 4.
12. Обчислити використовуючи правила спрощеного обчислення остачі.
    1) (27+12) mod 5;
    2) (22*17) mod 3;
    3) 7^2 mod 4.
13. За алгоритмом Евкліда знайти НСД (побудувати таблицю):
    1) 123 і 56;
    2) 219 і 102;
    3) 201 і 46.
14. За алгоритмом Евкліда знайти НСД (побудувати таблицю):
    4) 160 і 121;
    5) 213 і 142;
    6) 216 і 24.
15. За алгоритмом Евкліда знайти НСД (побудувати таблицю):
    1) 145 і 113;
    2) 203 і 102;
    3) 216 і 91.
16. За алгоритмом Евкліда знайти НСД (побудувати таблицю):
    1) 85 і 35;
    2) 146 і 184;
    3) 225 і 45.
17. За алгоритмом Евкліда знайти НСД (побудувати таблицю):
    1) 84 і 31;
    2) 141 і 193;

3) 205 і 42.

18. За алгоритмом Евкліда знайти НСД (побудувати таблицю):

1) 97 і 36;

2) 124 і 235;

3) 57 і 28.

19. Виконати дискретне піднесення до степеня бінарним методом

$$9^{41} \bmod 15$$

20.Виконати дискретне піднесення до степеня бінарним методом

$$6^{31} \bmod 13$$

21. Виконати дискретне піднесення до степеня бінарним методом

$$7^{32} \bmod 11$$

22.Виконати дискретне піднесення до степеня бінарним методом

$$11^{58} \bmod 9$$

23.Виконати дискретне піднесення до степеня бінарним методом

$$5^{45} \bmod 7$$

24.Виконати дискретне піднесення до степеня бінарним методом

$$8^{50} \bmod 9$$

Розв’язати діофантові рівняння:

1. $7x - 5y = 1;$
2. $21x - 4y = 1;$
3. $7x - 4y = 1;$
4. $8x - 11y = 1;$
5. $16x - 7y = 1;$
6. $20x - 17y = 1;$
7. $22x - 17y = 1;$
8. $14x - 19y = 1;$
9. $37x - 21y = 1;$
10. $6x - 19y = 1;$
11. $40x - 13y = 1;$
12. $13x - 19y = 1;$
13. $39x - 12y = 1;$
14. $34x - 19y = 1;$
15. $26x - 17y = 1;$
16. $25x - 4y = 1;$
17. $22x - 19y = 1;$
18. $10x - 13y = 1;$
19. $37x - 19y = 1;$
20. $26x - 31y = 1;$
21. $39x - 21y = 1;$
22. $10x - 19y = 1;$
23. $17x - 3y = 1;$
24. $13x - 25y = 1;$
25. $41x - 24y = 1;$
26. $11x - 19y = 1;$
27. $27x - 7y = 1;$
28. $8x - 9y = 1;$

29. $23x - 21y = 1;$
30. $31x - 18y = 1;$
31. $21x - 13y = 1;$
32. $11x - 15y = 1;$
33. $52x - 47y = 1;$
34. $6x - 11y = 1;$
35. $29x - 15y = 1;$
36. $51x - 46y = 1;$
37. $15x - 23y = 1;$
38. $5x - 14y = 1;$
39. $9x - 28y = 1;$
40. $13x - 33y = 1;$
41. $20x - 13y = 1;$
42. $29x - 45y = 1;$
43. $22x - 9y = 1;$
44. $27x - 16y = 1;$
45. $35x - 44y = 1;$
46. $14x - 29y = 1;$
47. $11x - 60y = 1;$
48. $19x - 33y = 1;$
49. $7x - 15y = 1;$
50. $20x - 17y = 1;$
51. $19x - 52y = 1;$
52. $67x - 50y = 1;$
53. $42x - 13y = 1;$
54. $38x - 29y = 1;$
55. $37x - 46y = 1;$
56. $6x - 31y = 1;$
57. $17x - 12y = 1;$
58. $11x - 29y = 1;$
59. $19x - 21y = 1;$
60. $45x - 113y = 1;$
61. $11x - 105y = 1;$
62. $19x - 69y = 1;$
63. $91x - 45y = 1;$
64. $41x - 32y = 1;$
65. $8x - 19y = 1;$
66. $58x - 61y = 1;$
67. $30x - 17y = 1;$
68. $31x - 18y = 1;$
69. $49x - 36y = 1;$
70. $14x - 5y = 1;$
71. $39x - 8y = 1;$
72. $41x - 27y = 1;$
73. $51x - 99y = 1;$
74. $49x - 68y = 1;$
75. $17x - 24y = 1;$
76. $27x - 11y = 1;$
77. $22x - 7y = 1;$
78. $9x - 35y = 1;$
79. $60x - 53y = 1;$
80. $31x - 9y = 1;$