МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Остапов С. Е.
Євсеєв С. П.
Король О. Г.

ТЕХНОЛОГІЇ ЗАХИСТУ ІНФОРМАЦІЇ

Навчальний посібник

Харків. Вид. ХНЕУ, 2013

УДК 004.056.5(076.5)
ББК 32.973я73
О-76

Рецензенти: докт. техн. наук, професор, зав. кафедри захисту інформації Національного університету "Львівська політехніка" *Дудикевич В. Б.*; докт. техн. наук, професор, зав. кафедри системного програмування Хмельницького національного університету *Поморова О. В.*; докт. техн. наук, проректор з наукової роботи та міжнародних зв'язків ПВНЗ "Буковинський університет" *Виклюк Я. І.*

Рекомендовано до видання рішенням вченої ради Харківського національного економічного університету.

Протокол № 6 від 21.01.2013 р.

Авторський колектив: докт. фіз.-мат. наук, професор Остапов С. Е. – вступ, розділи 1 – 3, 8, 9; канд. техн. наук, ст. наук. співробітник Євсеєв С. П. – розділи 4, 5, 7, 10; викладач Король О. Г. – розділи 6, 11 – 13.

Остапов С. Е.

О-76 Технології захисту інформації : навчальний посібник / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2013. – 476 с. (Укр. мов.)

Розглянуто основи сучасного захисту інформації в комп'ютерних системах, не пов'язаних із державною таємницею. Викладено основні поняття та визначення захисту інформації, формування політики безпеки, критерії оцінки захищеності комп'ютерних систем, основи криптографічного захисту інформації, захисту інформації від несанкціонованого доступу в сучасних операційних системах, а також описано комплексні системи захисту в корпоративних інформаційних системах.

Рекомендовано для студентів, які навчаються за напрямами "Комп'ютерна інженерія", "Програмна інженерія", "Інформатика" та "Комп'ютерні науки" всіх форм навчання, для студентів інших спеціальностей, де вивчається цикл навчальних дисциплін із захисту інформації, а також для самостійного опанування його основ.

ISBN

УДК 004.056.5(076.5)
ББК 32.973я73

Вступ

Захист інформації перетворюється сьогодні на одну з найактуальніших задач внаслідок надзвичайно широкого розповсюдження як власне різноманітних систем обробки інформації, так і розширення локальних та глобальних комп'ютерних мереж, якими передаються величезні об'єми інформації державного, військового, комерційного, приватного характеру, власники якої часто були б категорично проти ознайомлення з нею сторонніх осіб. Проблема набуває особливої гостроти після прийняття урядом України закону про захист персональних даних, який зобов'язує зберігати та передавати персональні дані працівників лише у захищеному вигляді в інформаційних системах (ІС).

Не менш важливим завданням вважається широке впровадження інформаційних технологій у різні сфери людської діяльності в Україні: стрімке зростання обігу пластикових карток, майбутнє введення електронних паспортів та медичних карт, студентських квитків та залікових книжок; зрештою все більше державних установ та приватних підприємств переходять на електронний документообіг, який до того ж, вимагає юридичної чинності підпису фізичної або юридичної особи. Розповсюдження таких технологій також, безперечно, вимагає добре поставленого захисту інформації.

Усі ці та багато інших задач покликані вирішувати різноманітні технології захисту інформації. Навчальний посібник, що пропонується читачеві, містить систематичний опис основ захисту інформації. Він складається із вступу та тринадцяти розділів.

Перший розділ присвячено основним поняттям та визначенням захисту інформації. Тут розглянуто властивості інформації, визначено захист інформації як галузь людської діяльності та його основні завдання, подано класифікацію основних загроз для інформації та їх джерел.

У другому розділі розглядаються основні поняття політики інформаційної безпеки, аналізуються моделі загроз і модель порушника, розглянуто методику оцінки ризиків підприємства.

Розділи 3 – 9 містять основи криптографічного захисту інформації, де розглядається симетрична та асиметрична криптографія, гешувальні алгоритми та електронний цифровий підпис, елементи криптоаналізу та основні напрями розвитку сучасної криптографії.

У розділах 10 – 12 описані механізми та протоколи керування криптографічними ключами, методи та пристрої забезпечення безпеки, а також моделі захисту та використання механізмів контролю доступу.

У 13 розділі визначені основні процедури, які пов’язані з використанням паролів і механізмів контролю за доступом у різних середовищах.

Вивчення навчальних дисциплін циклу захисту інформації дозволить студентам, які навчаються за напрямами "Комп’ютерна інженерія", "Програмна інженерія", "Інформатика" та "Комп’ютерні науки" всіх форм навчання оволодіти знаннями та вміннями, які створять теоретичне та практичне підґрунтя для отримання компетентностей щодо проведення аналізу загроз, які виникають при зберіганні, обробці та передаванні інформації; побудови систем захисту з використанням методів традиційної криптографії; демонстрації здатності критично вивчати, аналізувати і оцінювати з різних точок зору: технології, методи та процедури для проектних робіт, пов’язаних з розробленням профілю безпеки, надання порівняльної характеристики різних варіантів застосування механізмів і протоколів захисту інформації в інформаційних системах, забезпечення обґрунтованого підбору програмно-апаратних і програмних засобів для забезпечення необхідного рівня захисту інформації, проведення аналізу ефективності прийнятих технічних рішень щодо забезпечення захисту інформації в ІС.

Навчальний посібник узагальнює окремі методичні розробки з навчальних дисциплін "Захист інформації в інформаційних системах", "Технології захисту інформації", "Безпека програм та даних" і містить систематизований навчальний матеріал у галузі інформаційної безпеки для вивчення теоретичного матеріалу та набуття практичних навичок як самостійно, так і під керівництвом викладача.

Навчальний посібник призначено для студентів, які навчаються у вищих навчальних закладах за напрямами підготовки: 6.050101 – "комп’ютерні науки" (спеціалізацій "Інформаційні управляючі системи та технології" та "Комп’ютерний еколого-економічний моніторинг"); 6.050102 – "Комп’ютерна інженерія"; 6.050103 – "Програмна інженерія" та 6.040302 – "Інформатика".

Розділ 1. Огляд безпеки системи

1.1. Основні поняття

Уже давно інформація стала загальнолюдським поняттям, однак до цього часу не існує загальнонаукового визначення цього терміна. З точки зору різних галузей науки, це поняття визначається різними специфічними ознаками. Електронна енциклопедія "Вікіпедія", наприклад, дає таке визначення інформації [37; 38]: **інформація** – це відомості про щось незалежно від форми їх подання. Сучасна наука розглядає два типи інформації: об'єктивна – властивості матеріальних об'єктів та явищ, які передаються іншим об'єктам та відображаються в їх структурі; та суб'єктивна, яка відображає зміст об'єктивної інформації, сформована свідомістю людини за допомогою слів, образів і відчуттів та зафіксована на будь-якому матеріальному носієві. У побутовому розумінні інформація – це відомості про оточуючий світ та процеси, що в ньому відбуваються, які сприймаються людиною або спеціальним пристроєм.

До захисту інформації найближчими, на наш погляд, є такі означення інформації.

За Клодом Шенноном [47]: *інформація* – це відомості, за допомогою яких усувається невизначеність, що існувала у споживача до їх отримання.

За Глушковим В. М. [4]: *інформація* – міра неоднорідності розподілу матерії та енергії у просторі та часі, міра змін, якими супроводжуються всі процеси, що протікають у світі.

Згідно з визначенням, яке надає ЮНЕСКО, *інформація* – універсальна субстанція, що пронизує усі сфери людської діяльності, слугує провідником знань та думок, інструментом спілкування, взаєморозуміння та співробітництва, утвердження стереотипів мислення та поведінки.

Згідно з Законом України "Про інформацію", *інформація* – це документовані або публічно оголошені відомості про події та явища, що відбуваються у суспільстві, державі та навколишньому природному середовищі [39].

Інформація має певні властивості. Наприклад, з практичної точки зору, вони визначаються таким чином:

цінність інформації визначається можливістю забезпечення досягнення мети, поставленої її отримувачем;

достовірність – відповідність отриманої інформації реальності навколишнього світу;

актуальність – відповідність цінності та достовірності отриманої інформації поточному часу.

Визначаються також інші властивості інформації, з інших точок зору, наприклад, часові властивості тощо.

З точки зору захисту інформації нас цікавлять такі її властивості, які підлягають захисту. До них відноситься [12]:

конфіденційність – властивість інформації бути захищеною від несанкціонованого ознайомлення;

цілісність – властивість інформації бути захищеною від несанкціонованої зміни або знищення;

доступність – властивість інформації бути захищеною від несанкціонованого блокування.

Іноді ще визначають таку властивість інформації, як **спостережність** – властивість весь час (на усіх етапах обробки та передавання) знаходитись під контролем системи захисту.

1.2. Захист інформації та його основні завдання

Зрозуміло, що усю перелічену інформацію треба захищати. Принципи захисту повинні бути різними залежно від того, який тип інформації необхідно захищати. Якщо інформація, що підлягає захисту, належить до одного з класифікованих ступенів секретності, то основні зусилля системи захисту повинні бути зосереджені на захисті конфіденційності. Чи треба захищати відкриту інформацію, і якщо треба, то від яких загроз? Для того, щоб відповісти на це питання, розглянемо два приклади.

Кілька років тому хакери влаштували колективну атаку на ресурси компанії Microsoft, яка полягала в тому, що на її сервери одночасно було направлено стільки запитів, що вони перевищили можливості серверів та пропускну спроможність каналів зв’язку. Зрозуміло, що на цей період легальні користувачі були позбавлені сервісів Microsoft (до речі, від аналогічної атаки постраждала й Sun), оскільки їх запити на обслуговування не виконувалися. Така атака називається "відмовою в обслуговуванні". Вона направлена на блокування Інтернет-сервісів для легальних споживачів відкритої інформації. Отже, стає зрозумілим, що і відкриту інформацію також треба захищати, насамперед, від несанкціонованого блокування, а значить, захищати доступність такої інформації.

У кінці 80-х – на початку 90-х років XX сторіччя банківські документи, зокрема платіжні вимоги про зарахування грошей на банківський рахунок (авізо), курсували телеграфними мережами у відкри-

тому вигляді. Цей факт сприяв виникненню цілого бізнесу фальшивих авізо, причому, зараховуючи гроші на вказаний розрахунковий рахунок, працівники банків самі не знали, фальшиві авізо вони обслуговують, чи законні. Для протидії таким атакам Центробанк розробив систему шифрування банківських документів на базі мікрокалькулятора "Електроніка МК-85", який генерував так званий код підтвердження достовірності і, таким чином, захищав документи від змін, іншими словами, захищав цілісність документів [8].

Отже, як видно з наведених прикладів, відкриту інформацію також треба захищати, причому захищати її цілісність і доступність.

Діяльність, спрямована на забезпечення захисту конфіденційності, цілісності та доступності важливої для держави, суспільства та особи інформації, в тому числі відкритої, охорона якої передбачається законом, називається захистом інформації [12].

Тепер спробуємо визначити, в яких системах повинен здійснюватися захист інформації [28; 30]. Сьогодні спеціалісти використовують багато понять для визначення захисту інформації в системах обробки та передавання інформації. Це і *захист інформації в комп'ютерних системах, і захист інформації в автоматизованих системах, і захист інформації в інформаційно-телекомунікаційних мережах* тощо. Причому вважається, що усі ці поняття є синонімами. Насправді це не зовсім так. Слід зазначити [28; 30], що за останні роки ці терміни зазнали певної еволюції. Зараз в Україні загальноприйнятим є поняття *захисту інформації в інформаційно-телекомунікаційних системах,* його найбільше використовують у законодавчих та нормативних документах [12].

Розглянемо детальніше, що необхідно будемо розуміти під визначенням тієї чи іншої інформаційної системи [28; 30].

1. ***Інформаційно-телекомунікаційною*** системою називають організаційно-технічну систему, що реалізує певну сукупність технологій обробки та передавання даних шляхом їх кодування у формі фізичних сигналів.

2. ***Комп'ютерна система,*** згідно з визначенням НД ТЗІ 1.1-003099 [28], – це сукупність програмно-апаратних засобів, яку подають на оцінювання. Під оцінюванням розуміють експертне оцінювання захищеності інформації в цих програмно-апаратних засобах. Таке оцінювання є складовою частиною експертизи на відповідність чинним нормативним документам та стандартам.

3. Під ***обчислювальною системою*** розуміють сукупність програмно-апаратних засобів, призначених для обробки інформації. Обчислювальна система поєднує технічні засоби обробки та передавання інформації, а також методи і алгоритми обробки даних, реалізовані у вигляді відповідного програмного забезпечення.

4. ***Автоматизована система*** – це організаційно-технічна система, що реалізує інформаційну технологію та поєднує в собі: обчислювальну систему, фізичне середовище, персонал та інформацію, що обробляється. Зі структурою автоматизованої системи можна детальніше ознайомитися в [28].

В Україні, згідно з нормативними документами ДСТСЗІ СБУ (*Департамент спеціальних телекомунікаційних систем та захисту інформації Служби Безпеки України*), НД ТЗІ 2.5-005-99, зараз існує така **класифікація автоматизованих систем** (АС) [29]:

АС-1 – одномашинний однокористувацький комплекс, який обробляє інформацію однієї або кількох категорій конфіденційності. Приклад – ***автономний (ізольований) персональний комп'ютер***, доступ до якого здійснюється з використанням організаційних засобів безпеки.

АС-2 – локалізований багатомашинний багатокористувацький комплекс, який обробляє інформацію різних категорій конфіденційності. Приклад – ***локальна обчислювальна мережа***. Істотна відмінність від попередньої категорії – наявність категорій користувачів з різними повноваженнями доступу та наявність апаратного забезпечення, яке може одночасно обробляти інформацію різних категорій конфіденційності.

АС-3 – розподілений багатомашинний багатокористувацький комплекс, який обробляє інформацію різних категорій конфіденційності. Приклад – ***глобальна мережа***. Істотна відмінність від попередньої категорії – необхідність передавання інформації через незахищене середовище. У навчальному посібнику надалі будуть використані саме ці поняття та означення.

1.2.1. Класифікація загроз для інформації та їх джерел

"Концепція технічного захисту інформації в Україні" [55] визначає такі **джерела загроз для інформації:**

інші держави;

політичні партії;

злочинні угрупування;

суб’єкти підприємницької діяльності;

окремі фізичні особи;

навмисні та ненавмисні дії персоналу;

стихійні лиха та техногенні катастрофи.

Мотивація цих джерел може бути зовсім різною: від повної її відсутності у стихійних лих до економічних та політичних переваг (табл. 1.1).

Таблиця 1.1

Джерела загроз для інформації

Джерела	Мотивація
Інші держави	Одержання переваг у зовнішньополітичній, зовнішньоекономічній, військовій, та інших сферах
Політичні партії	Одержання переваг у політичній боротьбі, боротьбі за владу
Злочинні угрупування	Одержання політичних, економічних переваг, нанесення шкоди
Суб’єкти підприємницької діяльності	Одержання переваг у конкурентній боротьбі, економічні переваги
Фізичні особи	Самоствердження, отримання економічних переваг і винагород
Помилки персоналу	Низька кваліфікація працівників; образа, зрада, примушення
Стихійні лиха, техногенні катастрофи	Не мають мотивації

На основі поданої класифікації джерел можна скласти одну з можливих класифікацій загроз для інформації (табл. 1.2).

Таблиця 1.2

Класифікація загроз для інформації

№	Загрози	Методи боротьби
1	2	3
1	Наслідки стихійних лих і техногенних катастроф	Резервування апаратного забезпечення (дзеркальні файлові та web-сервери, географічно рознесені); резервні копії інформації
2	Відмови обладнання	Резервування апаратного забезпечення (з можливістю "гарячої" заміни); резервні копії інформації; вибір надійного постачальника апаратного забезпечення; вчасна профілактика та ремонт апаратного забезпечення
3	Наслідки помилок проектування системи захисту	Залучення ліцензованих спеціалістів (ліцензіатів) для побудови та експертизи системи захисту; обов’язкова експертиза проекту; періодичний аудит системи захисту

Закінчення табл. 1.2

1	2	3
4	Наслідки помилок персоналу	Ретельне підбирання персоналу; навчання персоналу; створення системи адміністративних стягнень за порушення; створення позитивного мікроклімату в колективі
5	Навмисні дії порушників	Залежно від способу дій (див. далі)

У теорії захисту інформації доведено [55], що якщо система захисту побудована з урахуванням усіх сучасних методів та засобів захисту, тобто повністю реалізований п. 3 табл. 1.2, а підприємство має ретельно підібраний та навчений персонал, який не допускає помилок, тобто повністю реалізовано п. 4 табл. 1.2, то навмисні дії порушників у такій системі неможливі.

Однак насправді це не зовсім правильно. З часом система захисту застаріває, персонал змінюється та втрачає пильність, зловмисники знаходять нові способи атак та методи подолання захисту, невідомі на час розробки системи захисту. Тому маючи обґрунтовані надії на стійкість системи захисту інформації, краще, все ж таки, пам’ятати основне правило захисту інформації: **жодна система захисту не може довгий час протистояти цілеспрямованим діям озброєного сучасними технологіями кваліфікованого порушника**.

Це правило вироблено роками досвіду спеціалістів захисту інформації і має універсальний характер. Воно не залежить від рівня системи захисту, сумлінності користувачів та адміністраторів, апаратного та програмного забезпечення. Воно стверджує, що *проблема полягає не в тому, чи подолають зловмисники систему захисту, а в тому, коли це відбудеться*. І завдання захисту інформації полягає в тому, щоб злам системи відбувся якомога пізніше. Про це говорить і мета захисту інформації, визначена в "Концепції технічного захисту інформації" [55]: "**метою захисту інформації є унеможливлення або суттєве утруднення реалізації загроз для інформації, що є власністю держави, сприяння реалізації законних інтересів громадян, юридичних осіб, державних органів здійсненню ними своїх завдань і функцій, загроз, реалізація яких може нанести державі, суспільству або особі політичні, економічні, моральні та інші збитки**".

Виходячи з мети, можна сформулювати **основні завдання захисту інформації:**

захист інформації з обмеженим доступом від витоку;

протидія технічним розвідкам;

захист інформації з обмеженим доступом від несанкціонованих дій під час її обробки та зберігання;

захист інформації від спеціальних впливів;

захист цілісності та доступності відкритої інформації.

1.2.2. Основні задачі, які повинні вирішуватися системою комп'ютерної безпеки

Далі коротко наведені основні задачі, які повинні вирішуватися системою комп'ютерної безпеки [10]:

керування доступом користувачів до інформаційних ресурсів з метою захисту від неправомірного випадкового або навмисного втручання у роботу системи і несанкціонованого (із перевищенням наданих повноважень) доступу до програмних і апаратних ресурсів із боку персоналу та/або сторонніх осіб;

захист даних, що передаються каналами зв'язку;

реєстрування, збереження і надання даних про події, що відбувалися у системі і мали відношення до безпеки;

контроль роботи користувачів системи з боку адміністраторів, обов'язкове повідомлення адміністратора безпеки про спроби несанкціонованого доступу до ресурсів системи;

контроль і підтримка цілісності критичних ресурсів системи захисту і середовища виконання прикладних програм;

забезпечення замкненості програмного середовища із метою захисту від безконтрольного впровадження у систему потенційно небезпечних програм і засобів подолання системи захисту;

керування засобами захисту.

1.2.3. Класифікація основних засобів протидії загрозам безпеки

Основні засоби протидії загрозам безпеки в комп'ютерних системах поділяються на правові (законодавчі), морально-етичні, організаційні (адміністративні), фізичні, технічні [10; 22; 34; 35].

До **правових** засобів захисту відноситься законодавчо-права база, на якій ґрунтуються інші засоби. Зокрема, в Україні це закони "Про інформацію", "Про технічний захист інформації", "Про державну таємницю", нормативні документи Державної служби спеціального зв'язку та захисту інформації, які регламентують правила обробки інформації.

До **морально-етичних** засобів протидії відноситься дотримання норм поведінки, що традиційно склались або складаються в інформаційному суспільстві країни та світу.

Організаційні (адміністративні) засоби захисту – це засоби організаційного характеру, що таким чином регламентують процес функціонування системи обробки даних, використання її ресурсів, діяльність персоналу, а також порядок взаємодії користувачів з системою, щоб унеможливити або максимально ускладнити здійснення загроз безпеки інформації.

Фізичні засоби захисту ґрунтуються на використанні різного роду механічних, електро- або електронно-механічних пристроїв, спеціально призначених для створення фізичних перешкод на можливих шляхах проникнення потенційних порушників, їх доступу до компонентів системи та інформації, що захищається, а також засоби візуального спостереження, зв'язку і охоронної сигналізації.

Технічні (апаратно-програмні) засоби захисту базуються на використанні різних електронних пристроїв і спеціального програмного забезпечення, що входять до складу автоматизованої системи і виконують, самостійно або в комплексі з іншими засобами, функції захисту інформації.

1.3. Поняття про інформацію з обмеженим доступом

Отже, можна зробити висновок, вивчаючи властивості інформації з Державного стандарту України, що інформацію необхідно захищати. Яку інформацію треба захищати і від чого? Аналіз властивостей, які потребують захисту, показує, що з цієї точки зору, є два види інформації: **відкрита**, тобто така, яка призначена для ознайомлення усіх бажаючих, наприклад, газети, журнали, телевізійне та радіомовлення, інформаційні сайти, реклама тощо, та **інформація з обмеженим доступом**, тобто така, ознайомитися з якою можна лише з санкції її власників або розпорядників. Згідно з Державним стандартом України, **інформація з обмеженим доступом** – це така інформація, права доступу до якої обмежено існуючими правилами та нормами.

Інформація з обмеженим доступом поділяється ще на дві категорії: **конфіденційну** і **таємну інформацію** [18; 38; 45].

Конфіденційною інформацією будемо називати таку інформацію з обмеженим доступом, якою володіють, користуються чи розпоряджаються окремі фізичні або юридичні особи чи держава, і порядок доступу до якої встановлюється ними.

Таємною інформацією слід називати таку інформацію з обмеженим доступом, яка містить відомості, що становлять державну або іншу передбачену законом таємницю. Віднесення інформації до категорії таємної здійснюється згідно із законами України.

Отже, відмінність конфіденційної інформації від таємної полягає в тому, що остання захищається законом України "Про державну таємницю", тому зловмисник, який несанкціоновано здобув або розголосив цю інформацію, автоматично завдав шкоди державі, суспільству або особі, і буде нести відповідальність згідно із законами України. Конфіденційна інформація – це інформація професійного, ділового, виробничого, банківського, комерційного та іншого характеру, яка не порушує передбаченої законом таємниці. Виняток становить інформація комерційного та банківського характеру, а також інформація, правовий режим якої встановлено Верховною Радою України за поданням Кабінету Міністрів України (з питань статистики, екології, банківських операцій, податків тощо), та інформація, приховування якої є загрозою життю і здоров'ю людей.

Згідно із Законом України "Про державну таємницю" [37], **державною таємницею** є вид інформації, що охоплює відомості у сфері оборони, економіки, науки і техніки, зовнішніх відносин, державної безпеки та охорони правопорядку, розголошення яких може завдати шкоди національній безпеці України, та які визнані у порядку, встановленому цим Законом, державною таємницею і **підлягають охороні державою.**

Тим же законом визначено ступені секретності інформації в Україні. Ступінь секретності інформації – це категорія, яка характеризує важливість таємної (секретної) інформації, степінь обмеження доступу до неї та рівень її охорони державою. Законом передбачається чотири ступені секретності (в порядку зменшення секретності): **особливої важливості**; **цілком таємно**; **таємно**; **для службового користування (ДСК)**.

До відомостей особливої важливості можуть належати детальні дані перепису населення, стратегічні плани Генерального штабу Збройних сил України; стратегічні політичні плани уряду тощо. До цілком тає-

мних – детальні тактико-технічні дані передової військової техніки та технологія їх виробництва, плани розгортання військ; стратегічні плани парламентських партій та фракцій тощо. Таємні відомості – технологія виробництва військової техніки та її компонентів, стратегічні та тактичні плани військ та армійських підрозділів тощо. Категорія "для службового користування" означає, що такою інформацією можна вільно користуватися в межах підприємства, але вони не підлягають розголошенню за його межами. До такої інформації відноситься фінансова звітність підприємства, дані про заробітну платню його працівників, про структуру постачання, можуть бути також дані про збут. Із затвердженням Закону Україні "Про захист персональних даних" (набув чинності з 1 січня 2011 року), персональні дані працівників підприємства, використання яких може однозначно ідентифікувати людину, також підлягають категоріюванню "для службового користування". Такими даними можуть бути прізвище, ім’я, по батькові, номер паспорта, ідентифікаційний код та інші.

1.4. Структура політики безпеки та її основні частини

Політика безпеки організації – сукупність принципів, правил, процедур і практичних рішень у галузі безпеки, які регулюють керування, захист та розподіл інформації, що захищається.

Політика безпеки залежить від багатьох чинників, а саме:

від рівня секретності та властивостей інформації, яка підлягає захисту;

від конкретної технології обробки інформації;

від технічних та програмних засобів, що використовуються організацією;

від розташування організації;

інших чинників, які уточнюються на етапі розробки політики безпеки.

Політика безпеки повинна враховувати сучасний стан та найближчі перспективи розвитку інформаційних технології, мету, завдання, правові основи експлуатації, режими функціонування об’єктів, містити аналіз загроз безпеці та способи їх реалізації.

Основні положення таких документів повинні розповсюджуватися на усі структурні підрозділи організації, а також на інші організації, які взаємодіють з основною організацією як постачальники або споживачі інформаційних ресурсів.

Законодавчою основою Політики безпеки служать Конституція, Громадянський та Карний кодекси держави, закони, укази, постанови,

документи Держспецзв'язку та захисту інформації (ДСТСЗІ СБУ або інших аналогічних організацій).

Політика безпеки є методологічною основою для:

формування та впровадження єдиної політики в галузі захисту інформації організації;

прийняття важливих рішень та розробки спільних практичних заходів, спрямованих на виявлення, знешкодження та ліквідацію наслідків реалізації різних типів загроз безпеці інформації;

координації діяльності структурних підрозділів організації при виконанні робіт зі створення, розвитку та експлуатації інформаційних ресурсів з дотриманням вимог із забезпечення інформаційної безпеки;

розробки пропозицій щодо вдосконалення правового, нормативного, технічного та організаційного забезпечення інформаційної безпеки в організації.

При розробці Політики безпеки враховуються основні принципи створення комплексних систем забезпечення безпеки, характеристики та можливості організаційно-технічних методів, сучасні апаратно-програмні засоби захисту та протидії загрозам безпеці інформації.

Політика безпеки, як правило, складається з таких розділів:

1. *Загальні положення*, де обговорюються призначення та правова основа документа, даються основні означення та термінологія.

2. *Об'єкти захисту*, де описано категорії інформаційних ресурсів, які підлягають захисту, подається структура, склад і розміщення основних об'єктів захисту та інформаційні зв'язки між ними.

Наприклад, тут необхідно обрати стратегію системи комплексного захисту інформації (***захисна стратегія***, тобто така, яка захищає від уже відомих загроз; ***наступальна***, тобто яка захищає від усієї множини можливих загроз; або ***попереджувальна***, тобто створення такого середовища, в якому загрози не мають умов для виникнення), проаналізувати наявні типи інформації, яка потребує захисту, класифікувати її за категоріями та оцінити збитки від витоку того чи іншого типу інформації. Які можуть бути типи цінної інформації на підприємстві? Звичайно це залежить від роду діяльності підприємства. Для виробничих підприємств існує ***технічна та технологічна інформація***: хімічна формула, рецептура, результати випробувань, дані контролю якості, методи та технологічні процеси виготовлення, виробничі показники, структура постачань тощо; ***ділова інформація***: рішення керівництва, методи реалізації

функцій підприємства, вартісні показники, списки клієнтів, результати дослідження ринку, економічні прогнози тощо; ***економічна інформація***: бюджет підприємства, економічні показники та звіти, структура прибутків та видатків, структура собівартості продукції, заробітна платня працівників, плани та угоди з контрагентами тощо. Можуть бути й інші типи закритої інформації, наприклад, науково-дослідна та дослідно-конструкторська діяльність, результати дослідження напрямів розвитку конкурентів, використання новітніх технологій, нарешті, інформація про службу безпеки підприємства.

3. *Мета та основні завдання забезпечення безпеки*. В цьому розділі описано інтереси суб'єктів інформаційних відносин, які зачіпаються розробкою даної Політики безпеки; мета та основні завдання системи забезпечення безпеки організації та шляхи вирішення поставлених завдань.

4. *Основні загрози безпеці інформації організації*. Цей розділ присвячено опису основних загроз та шляхів їх реалізації. Як правило, загрози поділяються на природні та штучні або суб'єктивні. Штучні загрози поділяються на навмисні та ненавмисні. Усі загрози, як правило, описуються в рамках так званої *моделі загроз – формалізованого* або *неформалізованого* опису можливих загроз та шляхів їх реалізації. *Неформалізований* опис загроз – опис у вигляді звичайного тексту; *формалізований* – із залученням таблиць, графіків, математичних моделей. У найбільш серйозних випадках використовують математичні моделі загроз, які повинні довести адекватність та повноту обраних типів загроз та шляхів їх реалізації.

5. *Модель можливих порушників.* У цьому розділі подається формальний або неформальний опис порушника, його мотивації, кваліфікації та можливих дій. Такий опис називається *моделлю порушника*, і буває також формалізованим або неформалізованим. У найбільш відповідальних випадках також будується математична модель порушника, яка доводить свою повноту і адекватність в умовах роботи організації.

6. *Витік інформації технічними каналами.* Цей розділ присвячений аналізу існуючих та майбутніх можливих технічних каналів витоку інформації та опису способів протидії. Як правило, тут описано пасивні та активні методи протидії, їх взаємодія, доцільність та економічна обґрунтованість.

7. *Основні принципи побудови системи інформаційної безпеки організації*. Розділ присвячено опису таких важливих принципів, як законність, системність, комплексність, неперервність захисту, модифікованість, економічна доцільність, персональна відповідальність, мінімізація повноважень, розмежування функцій, гнучкість та простота системи захисту, обґрунтування та можливість технічної реалізації розробленої політики безпеки. Зазвичай описують також методи контролю та його обов'язковість.

8. *Методи та засоби забезпечення задекларованого рівня захищеності інформаційних ресурсів*. Розділ присвячено опису засобів забезпечення інформаційної безпеки, зокрема, регламентації доступу в приміщення, допуску співробітників до інформаційних ресурсів; порядок обслуговування та модифікації апаратних та програмних ресурсів. Регламентуються також методи забезпечення фізичної цілісності (незмінності конфігурації) апаратних та програмних ресурсів. Обов'язково потрібно також описати методи підбирання та підготовки персоналу, його відповідальність за порушення встановленого порядку користування інформаційними ресурсами підприємства. Окремим підрозділом описують засоби забезпечення інформаційної безпеки підприємства: фізичні та технічні засоби захисту; засоби ідентифікації та автентифікації користувачів, засоби розмежування доступу; засоби контролю та реєстрації подій (аудит) в системі; криптографічні засоби захисту. Звертають також увагу і на керування системою захисту та контроль її ефективності.

9. *Порядок внесення змін та доповнень*. У цьому розділі регламентовано порядок внесення змін та доповнень до політики безпеки.

Зрозуміло, що залежно від типу та структури підприємства, рівня секретності інформації, яку необхідно захищати, структури самих інформаційних ресурсів, структура політики безпеки може (і повинна) змінюватися, і описана авторами лише приблизно. Однак з наведеного прикладу стає зрозумілим, які питання повинні бути висвітлені у політиці безпеки.

Основою будь-якої політики безпеки є модель загроз та модель порушника. Взагалі разом з рівнем секретності інформації, яку потрібно захищати, вони визначають і витрати на створення системи захисту, і саму політику безпеки. Необхідно розглянути їх детальніше.

Для того, щоб побудувати модель загроз, спочатку треба розглянути основні загрози інформації в комп'ютерних системах, ґрунтуючись на класифікації, наведеній в попередньому розділі.

1.5. Життєвий цикл розробки систем безпеки

1.5.1. Розробка профілю захисту і проекту безпеки об'єкта оцінки

У 1999 році практично була закінчена розробка міжнародного стандарту ISO/IEC 15408 "Критерії оцінки безпеки інформаційних технологій", які ще відомі як "Єдині критерії" [29; 34; 35; 45]. Єдині критерії – це погоджений стандарт, розроблений на основі наявних стандартів й інших нормативних документів з обліком накопиченого в різних країнах досвіду в галузі забезпечення безпеки інформації. Стандарт сприяє усуненню концептуальних і технічних розходжень, наявних у раніше розроблених документах [29; 31; 36] і характеризує новий, міждержавний рівень стандартизації інформаційних технологій.

Однією з головних цілей стандарту є створення методологічної бази для здійснення оцінки властивостей безпеки продуктів і систем інформаційних технологій (IT-продуктів і IT-систем). У цілому розробка цього документа спрямована на рішення завдання забезпечення безпеки інформаційних технологій (IT-безпеки).

При розгляді проблеми забезпечення IT-безпеки розроблювачі стандарту виходили з того, що є дві основні протиборчі (конфліктуючі) сторони – власник ресурсів, що становить певну цінність і, отже, що вимагає свого захисту, і противник, що має мотиви й можливість для незаконного використання ресурсів, що може призвести до завдання збитків (морального, матеріального, економічного тощо) власнику ресурсів. При цьому під ресурсами (assets) розуміють не тільки інформацію (інформаційні ресурси), але й інші види ресурсів, наприклад, комунікаційні або обчислювальні ресурси. Противник розглядається як джерело загроз безпеки (threat agent) – можливих подій, дій (впливів), процесів або явищ, що є потенційними небезпеками, реалізація яких може призвести до завдання збитків власнику ресурсів. У зв'язку з цим

власник ресурсів змушений вживати заходів, які спрямовані на запобігання або зменшення ступеня цих небезпек. Як основні загрози розглядаються загрози порушення конфіденційності, цілісності і доступності. Однак для ефективної протидії противнику власник ресурсів повинен скласти якомога більш повний перелік загроз, які можуть бути реалізовані в конкретних умовах. Аналіз ризику реалізації загроз здійснюється шляхом завдання моделі порушника й оцінки ймовірності реалізації тієї або іншої загрози. Отримані результати дозволяють сформулювати вимоги щодо забезпечення ІТ-безпеки, реалізація яких сприяє зменшенню ризиків до прийнятного рівня [29; 31; 36].

Вимоги ІТ-безпеки по суті містять у собі функціональні вимоги безпеки і вимоги адекватності. Функціональні вимоги безпеки визначають набір функцій безпеки, які необхідно реалізувати для забезпечення конфіденційності, цілісності і доступності. Щоб механізми безпеки і засоби захисту функції, безпеки, що реалізують, можна було визнати ефективними, потрібен високий степінь упевненості в правильності їх вибору і надійності функціонування. Така впевненість досягається шляхом пред'явлення та реалізації вимог адекватності.

Основними об'єктами застосування вимог безпеки є ІТ-продукти і ІТ-системи. Під продуктом інформаційних технологій розуміється представлення кінцевому споживачу готового до використання апаратного, програмного або програмно-апаратного засобу (або сукупність таких засобів) обробки інформації. ІТ-продукт не призначений для автономної експлуатації й інтегрується в ІТ-систему (автоматизована система обробки інформації), яка становить організаційно-технічну систему, що містить у собі:

сукупність технічних засобів передачі і обробки інформації (ІТ-продуктів), об'єднаних у функціонально повний комплекс;

сукупність методів і алгоритмів обробки інформації у вигляді відповідного програмного забезпечення;

інформаційні та інші ресурси;

персонал і користувачів, об'єднаних за організаційно-структурним, тематичним, технологічним й іншими принципами для здійснення автоматизованої обробки інформації.

Практична реалізація вимог IT-безпеки виражається в розробці продукту або системи захисту інформації. У термінах ISO/IEC 15408 такими є об'єкт оцінки (TOE – Target of Evaluation) – IT-продукт або система, а також пов'язана з ними експлуатаційна, технічна, користувальницька та інша документація, яка є основним об'єктом перевірки і оцінки [58].

Таким чином, прийняті власником ресурсів заходи призводять до розробки об'єкта оцінки і визначають політику безпеки об'єкта оцінки, тобто множину правил, які регулюють порядок управління, використання, захисту і розподілу оцінки ресурсів, що захищені об'єктом.

Основними документами, що характеризують об'єкт оцінки з погляду забезпечення IT-безпеки, є профіль захисту (protection profile) і проект забезпечення безпеки (security target). Розглянемо порядок розробки цих документів.

1.5.2. Порядок розробки профілю захисту і проекту забезпечення безпеки

Профіль захисту (ПЗ) – це реалізаційно-незалежна множина функціональних вимог безпеки та вимог адекватності, спрямованих на задоволення потреб споживача. Профіль захисту є нормативним документом, що регламентує всі аспекти безпеки IT-продуктів і систем у вигляді сукупності функціональних вимог і вимог адекватності, пропонованих функціям безпеки і, отже, до засобів і механізмів захисту. Профіль захисту також може бути використаний для класифікації об'єкта оцінки.

Проект безпеки (ПБ) становить множину вимог безпеки та специфікацій функцій безпеки. Проект безпеки використовується як основа для здійснення оцінки розглянутого об'єкта оцінки. Крім специфікацій функцій безпеки і засобів захисту ПБ містить вимоги до сертифікації об'єкта оцінки та підсумкову специфікацію системи (підсистеми) забезпечення безпеки в конкретному IT-продукті або системі.

Розробка даних документів є основним практичним додатком Єдиних критеріїв. Розробка ПЗ і ПБ здійснюється, відповідно, у три і чотири етапи [29].

І етап. Аналіз безпеки середовища експлуатації об’єкта оцінки.

ІІ етап. Визначення й формулювання завдань із забезпечення безпеки.

ІІІ етап. Розробка вимог ІТ-безпеки.

ІV етап. Розробка специфікацій функцій безпеки й підсумкової специфікації забезпечення безпеки об’єкта оцінки.

Розглянемо більш докладно кожний із цих етапів.

Аналіз безпеки середовища експлуатації об’єкта оцінки здійснюється з метою:

обмеження системи (підсистеми) забезпечення безпеки об’єкта оцінки від навколишнього середовища експлуатації;

визначення ступеня небезпеки (агресивності) середовища експлуатації для функціонування об’єкта оцінки шляхом виявлення загроз безпеки і аналізу ризиків;

формування вихідних передумов для визначення завдань із забезпечення безпеки;

середовище експлуатації становить сукупність фізичних, інформаційних, технічних об’єктів і систем, зовнішніх стосовно об’єкта оцінки, а також організаційних заходів, правових норм, умов експлуатації і технологічних особливостей застосування об’єкта оцінки, що роблять фізичний, інформаційний, енергетичний та інший вплив на функціонування об’єкта оцінки. У середовище експлуатації також входять і можливі загрози безпеки, які вже існують або можуть виникнути в даному середовищі.

При здійсненні аналізу безпеки середовища експлуатації, з погляду забезпечення фізичної безпеки об’єкта оцінки, рекомендовано звернути увагу на його фізичне оточення. При цьому аналізу піддаються фізичні засоби і міри забезпечення безпеки, у тому числі забезпечення фізичної безпеки персоналу.

Важливо врахувати ресурси, що вимагають свого захисту засобами об’єкта оцінки. Аналіз здійснюється з урахуванням основного призначення об’єкта оцінки і того, чи є об’єкт оцінки ІТ-продуктом або ІТ-системою. Якщо в якості об’єкта оцінки розглядається ІТ-продукт, то розроблювач може орієнтуватися лише на загальні припущення про середовище експлуатації, включаючи умови експлуатації, застосування і загрози безпеки. Така невизначеність пов’язана з тим, що ІТ-продукт розробля-

ється в припущенні, що він буде використовуватися в багатьох системах обробки інформації. На відміну від ІТ-продукту ІТ-система розробляється для рішення конкретних прикладних завдань, розраховуючи на вимоги кінцевих споживачів (замовників). У цьому випадку розроблювач має можливість повною мірою враховувати специфіку впливів на об'єкт оцінки з боку середовища експлуатації.

Необхідно також розрізняти об'єкт оцінки з погляду його основного призначення. Залежно від того, яка головна мета функціонування об'єкта оцінки, необхідно переломлювати кут зору на аналіз середовища експлуатації об'єкта оцінки. Так, у якості об'єкта оцінки можна розглядати продукт або систему, основним призначенням яких є забезпечення безпеки інформації або інших ресурсів. З іншого боку, в якості об'єкта оцінки можуть виступати і комунікаційні системи, обчислювальні мережі, елементи систем передачі даних і здійснення електронних платежів і т. д.

У таких системах забезпечення безпеки є одним з множини функціональних завдань, спрямованих на досягнення головної мети функціонування.

Результатами аналізу безпеки є [31; 45]:

1) аксіоматичні висновки (припущення) про безпеку середовища експлуатації конкретного об'єкта оцінки;

2) перелік загроз безпеки, оцінка ймовірності їх реалізації і модель порушника безпеки. Кожна загроза характеризується з трьох сторін. По-перше, визначається модель порушника безпеки, що здатний реалізувати конкретну загрозу; по-друге, описуються передбачувані методи нападів (атак) і можливі слабкі місця (уразливості) системи, використання яких лежить в основі реалізації атак. Нарешті визначаються ресурси, які є об'єктом конкретної атаки. Все це є основою для проведення аналізу ризиків реалізації кожної загрози, що дозволяє оцінити ймовірність загрози та можливий збиток, що може бути нанесений у випадку реалізації загрози. Особливо зупинимося на моделі порушника. Тут необхідно виходити з того, якими ресурсами володіє потенційний противник, а також які мотиви для здійснення злочинних дій. У загальному випадку в ролі противника можна розглядати фізичну особу, що здійснює нерегулярні атаки, корпоративну групу, що володіє обмеженими можливостями та ресурсами і державну спецслужбу, що володіє більшими можливостями;

3) висновки щодо застосованості організаційної політики безпеки. Під *організаційною політикою безпеки* розуміють одне або кілька правил, процедур, практичних заходів (дій) або загальних рекомендацій, спрямованих на забезпечення безпеки і експлуатації об'єкта оцінки накладаються організацією відповідно до виконуваних даною організацією функцій. Тобто організаційна політика безпеки є конкретизацією загальної політики безпеки об'єкта оцінки стосовно конкретної організації.

Таким чином, усі роботи, виконувані на даному етапі, спрямовані на забезпечення максимального обліку конкретних умов експлуатації об'єкта оцінки при його перевірці й оцінці.

На основі отриманих у ході аналізу безпеки середовища експлуатації результатів здійснюється визначення і формулювання задач із забезпечення безпеки або задач захисту (security target).

Задача захисту – це цільова постановка задачі на протидію виявленим загрозам і задоволення політики безпеки. Задача захисту повинна бути погоджена з множиною інших функціональних задач об'єкта оцінки і не суперечити основному призначенню об'єкта оцінки. Задачі визначаються як для об'єкта оцінки, так і для середовища його експлуатації. Причому останні формулюються тільки для елементів середовища експлуатації, пов'язаних з інформаційними технологіями. Технічні засоби забезпечення безпеки, організаційні міри, правові норми розглядаються як зовнішні стосовно об'єкта оцінки елементи системи захисту, застосовувані для підвищення ефективності власних засобів забезпечення безпеки об'єкта оцінки.

Таким чином, задача захисту адресована винятково для реалізації вимог забезпечення IT-безпеки.

Сформульовані і, по можливості, формалізовані задачі захисту є базою для безпосередньої розробки профілю захисту. Розробка профілю захисту здійснюється у два етапи:пошук і вибір профілю-прототипу; синтез вимог IT-безпеки.

Міжнародний стандарт передбачає створення спеціальної картотеки пакетів вимог безпеки, профілів захисту і проектів безпеки. *Пакетом вимог* називається проміжна комбінація вимог безпеки, що описує множину функціональних вимог і вимог адекватності, які забезпечують

рішення виділеної підмножини задач захисту. Ця картотека буде доступна для розроблювачів, що дозволяє мінімізувати витрати на розробку нових профілів захисту і врахувати досвід попередніх розробок.

Вимоги IT-безпеки є уточненням, конкретизацією і відображенням задач захисту в множині вимог безпеки, пропонованих об'єкту оцінки і до середовища експлуатації. *Вимоги IT-безпеки* містять у собі три компоненти: функціональні вимоги безпеки; вимоги адекватності; вимоги безпеки середовища експлуатації.

У ході розробки профілю захисту здійснюється вибір вимог безпеки специфічних для конкретного середовища. При цьому вибір ґрунтується на оцінці ефективності вимог для рішення задачі протидії загрозам безпеки [31]. Функціональні вимоги визначають властивості безпеки і характеризують функції об'єкта оцінки, які є типовими для підтримки IT-безпеки.

Декомпозиція даних вимог, їх опис приводиться в другій частині ISO/IEC 15408-2, що є каталогом функціональних вимог.

Демонстрація того, що виконання функціональних вимог веде до забезпечення необхідного рівня безпеки, здійснюється через включення в профіль захисту вимог адекватності. Адекватність містить два аспекти: ефективність функцій безпеки; коректність реалізації функцій безпеки.

При оцінці ефективності функцій безпеки необхідно визначити степінь відповідності між завданнями захисту й пропонованим набором функцій безпеки, їх функціональною повнотою, погодженістю, простотою використання і ступенем запобігання загрозам безпеки. Коректність виражається в оцінці правильності та надійності реалізації функцій безпеки.

Для конкретної множини функціональних вимог ступінь адекватності може варіюватися. Тому адекватність виражається через рівні строгості вимог адекватності. Третя частина стандарту вводить шкалу рівнів оцінки адекватності, що опирається на вимоги адекватності.

При розробці профілю захисту необхідно аналізувати зв'язки і взаємозалежності як між різними функціональними вимогами, так і між функціональними вимогами і вимогами адекватності.

Вимоги безпеки, в остаточному підсумку, повинні демонструвати як наявність бажаних характеристик і властивостей безпеки, так і відсут-

ність небажаних характеристик. Бажані характеристики можна продемонструвати через використання або тестування конкретного продукту, у той час як визначення небажаних характеристик ускладнено. Перевірка, моделювання, аналіз конструкції і реалізації профілю істотно знижує ризик присутності таких небажаних характеристик. Тому підсумком робіт з розробки профілю захисту є оцінка його повноти, коректності, несуперечності (погодженості) і реалізованості.

Четвертий етап здійснюється при розробці проекту безпеки. На даному етапі здійснюється розробка підсумкової специфікації об'єкта оцінки. Ця специфікація містить опис заявлених функцій безпеки, які задовольняють функціональним вимогам і визначення показників адекватності, що характеризують степінь задоволення вимог адекватності.

1.5.3. Структура і зміст профілю захисту

Згідно з ISO/IEC 15408-1 профіль захисту повинен містити такі розділи [49; 51]:

1. Короткий опис профілю.
2. Короткий опис об'єкта оцінки.
3. Аналіз середовища експлуатації об'єкта оцінки.
4. Завдання із забезпечення безпеки.
5. Вимоги IT-безпеки.
6. Обґрунтування завдань захисту і вимог IT-безпеки.

Короткий опис профілю містить класифікаційну інформацію, необхідну для його ідентифікації в спеціальній картотеці. Тут же характеризується основне завдання або група завдань із забезпечення безпеки, розв'язуваних за допомогою цього профілю.

Короткий опис об'єкта оцінки містить: призначення і область застосування об'єкта оцінки; короткий опис особливостей, використовуваних у об'єкті оцінки інформаційних технологій; короткий опис додатків, у яких передбачено використання об'єкта оцінки (у випадку, якщо основним призначенням об'єкта оцінки є забезпечення безпеки інформації та інших ресурсів).

Розділ аналізу середовища експлуатації містить результати аналізу безпеки середовища експлуатації об’єкта оцінки й містить, а також:

а) висновки щодо різних аспектів безпеки середовища, в якому застосовано або передбачено застосування об’єкта оцінки, а саме:

відомості про передбачуване використання об’єкта оцінки, включаючи відомості про додатки, ресурси і обмеження на використання;

відомості про середовище експлуатації об’єкта оцінки, включаючи фізичні, інформаційні, технічні, технологічні, експлуатаційні й інші аспекти;

б) повний опис загроз безпеки, які безпосередньо впливають на безпеку об’єкта оцінки. Загрози описуються з погляду: противника, здатного реалізувати загрозу (модель противника, доступні ресурси для здійснення злочинних дій, мотиви здійснення злочинних дій); атак (методи нападів, використовувані слабкі місця системи захисту й інших сприятливих умов для реалізації конкретної атаки); ресурсів, що захищаються;

в) опис організаційних політик безпеки.

У розділі про завдання із забезпечення безпеки докладно описано завдання захисту для об’єкта оцінки і його середовища експлуатації.

Завдання захисту повинні: відображати конкретні цілі, що досягаються в ході забезпечення безпеки; бути застосованими для запобігання всіх виявлених загроз безпеки; охоплювати всі певні організаційні політики безпеки.

Розділ про вимоги ІТ-безпеки містить опис функціональних можливостей засобів захисту об’єкта оцінки і визначає рівень адекватності висунутих функціональних вимог стосовно певного раніше переліку загроз безпеки. Розділ містить у собі: функціональні вимоги безпеки для об’єкта оцінки; вимоги адекватності безпеки об’єкта оцінки; вимоги безпеки для ІТ-середовища експлуатації об’єкта оцінки.

При розробці даного розділу необхідно враховувати такі умови:

всі вимоги ІТ-безпеки формуються на основі використання другої і третьої частин стандарту ISO/IEC 15408; формулювання функціональних вимог і вимог адекватності повинне бути зрозумілим і однозначим, що дозволяє в остаточному підсумку провести їх оцінку і продемонструвати погодженість вимог; якщо необхідно, використовувати для компонентів функціональних вимог установлені операції над ними з метою дозволу

або заборони використання конкретних механізмів безпеки; при синтезі функціональних вимог і вимог адекватності враховувати всі взаємозв'язки й взаємозалежності між ними.

У розділі щодо обґрунтування задач захисту і вимог ІТ-безпеки подано докази того, що профіль захисту становить повну і зв'язну множину вимог до об'єкта оцінки, що задовольняє висунуті вимоги, забезпечує ефективну протидію погрозам безпеки. Розділ містить такі відомості:

а) обґрунтування завдань захисту, що демонструє, що заявлені завдання захисту відбивають і охоплюють всі аспекти безпеки об'єкта оцінки і середовища експлуатації;

б) обґрунтування вимог безпеки, що демонструє, що множина вимог безпеки (і для об'єкта оцінки, і для середовища експлуатації) застосована для реалізації всіх завдань захисту. При цьому:

комбінація окремих функціональних вимог і вимог адекватності задовольняє конкретні завдання захисту; повна множина вимог безпеки утворює взаємно підкріплювальне і внутрішньо погоджене ціле; вибір вимог безпеки є виправданим. Крім того, виправданим повинен бути вибір вимог, що не втримуються в стандарті ISO/IEC 15408, вибір рівня адекватності, не представленого в шкалі встановлених рівнів, а також незадоволення яких-небудь залежностей між вимогами; обраний рівень захищеності погоджений із задачами захисту.

Крім перерахованих розділів профіль захисту може містити додаткові відомості щодо конструкції, оцінки й застосування об'єкта оцінки.

Контрольні запитання

1. Політика безпеки. Основні поняття та принципи.
2. Структура політики безпеки та її основні частини.
3. Життєвий цикл розробки систем безпеки.
4. Система безпеки. Основні поняття про інформацію.
5. Критерії оцінки системи безпеки відповідно до стандарту НД ТЗІ.
6. Законодавча база України відповідно до систем безпеки.
7. Поняття про інформацію з обмеженим доступом.

Розділ 2. Механізми і політики розмежування прав доступу

2.1. TCSEC ("Оранжева книга") – перший стандарт у галузі оцінки захищеності комп'ютерних систем

Стандарт Міністерства оборони США TCSEC, більше відомий як "***Оранжева книга***" (англ. – Orange Book) за кольором обкладинки, був затверджений у 1983 році та оновлений у 1985. Повна назва документа – "Department of Defense Trusted Computer System Evaluation Criteria" – "Критерії оцінки захищених комп'ютерних систем Міністерства оборони" [43]. Це був перший стандарт, який встановлював основні вимоги до оцінки ефективності засобів безпеки комп'ютерних систем. З цієї точки зору його значення важко переоцінити, оскільки з'явився загальновизнаний базис понять, без якого навіть обговорення проблем інформаційної безпеки було б складним завданням.

Необхідно зупинитися на відмінностях між поняттями "***захищена система***", яке використовують в Україні, і "***довірена система***", яке використовують в англомовній літературі. "Оранжева книга" (ОК) була першим документом, де стверджувалося, що будь-яких абсолютних систем (у тому числі й абсолютно безпечних) у нашому житті не існує. Тому було запропоновано оцінювати лише степінь ***довіри*** до цієї чи іншої системи, тобто ***наскільки можна їй довіряти***. В українській літературі оцінюють не степінь довіри, а степінь ***захищеності*** комп'ютерних систем. Так склалося історично, тому автори також будуть дотримуватися цієї відмінності.

ОК дала визначення поняттю ***безпечної системи***. Згідно з нею, ***безпечною комп'ютерною системою*** називається така, де тільки уповноважені користувачі або процеси від їх імені можуть читати та змінювати призначену для них інформацію. Це означає, що доступ до інформації повинні мати лише ***уповноважені користувачі***, причому вони повинні мати доступ не до всієї інформації, а лише до ***призначеної для них***.

2.1.1. Основні поняття Оранжевої книги

Оранжева книга виробила так звані принципи безпеки. До них належить таке:

1. Безпечні системи повинні керуватися розробленими політиками безпеки, а користувачі зобов'язані їх дотримуватися.

2. Кожен об'єкт безпечної системи повинен мати певний рівень безпеки (рівень доступу для користувачів, рівень безпеки – для інформації), який визначається так званими ***мітками безпеки***.

3. Обов'язковою складовою безпечної системи повинна бути автентифікація користувачів.

4. У безпечних системах повинен бути налагоджений аудит за усіма важливими подіями, що відбуваються в системі і мають відношення до безпеки. Журнали аудиту повинні бути захищеними від зміни.

5. Засоби захисту безпечних систем повинні знаходитися під постійним контролем деякої системи, що контролює правильність їх роботи.

6. Принцип неперервності захисту: не повинно бути моментів у роботі системи, коли засоби захисту ***неактивні***.

Проаналізуємо кожен з цих принципів безпеки, звернувши увагу на те, що пропонує ОК для того, щоб захищена система задовольняла ці принципи безпеки.

1. ***Політики безпеки***. Політики безпеки повинні бути детальними, чітко визначеними та обов'язковими для виконання усіма користувачами комп'ютерної системи. ОК пропонує два типи політики безпеки: ***мандатна політика*** – така, яка ґрунтується на порівнянні внутрішніми засобами системи рівнів доступу користувачів та рівнів секретності інформації (усі рівні задаються так званими ***мітками безпеки***) і прийнятті рішення про допуск до інформації; ***дискреційна політика*** – така, коли прийняття рішення про допуск до інформації ґрунтується на зовнішніх щодо системи захисту правилах доступу.

2. ***Мітки безпеки***. ОК вважає, і це стало вже стандартом інформаційної безпеки, що мандатна політика безпеки безпечніша за дискреційну, що, як можна спостерігати пізніше, відображено в класах безпеки.

3. ***Відповідальність***. Незалежно від обраної політики безпеки, повинна існувати індивідуальна відповідальність користувачів за усі дії в системі. ОК висуває три вимоги до відповідальності:

а) ***автентифікація*** – процес, який надає підтвердження про те, що користувач є саме тим, ким він представився системі;

б) ***авторизація*** – надання користувачеві певних повноважень у системі;

в) ***аудит*** – здатність системи контролювати усі важливі дії користувачів у захищених журналах аудиту.

4. ***Монітор звернень***. Контроль за виконанням користувачами обробки інформації повинен вестися за допомогою деякої системи, яку ОК називає ***монітором звернень***, яка має задовольняти таким умовам:

а) ***ізольованість,*** тобто неможливість відстеження його роботи;

б) ***повнота,*** тобто неможливість обійти монітор;

в) ***верифікованість***, тобто можливість аналізу та тестування.

5. ***Ядро безпеки*** – це певна реалізація монітора звернень з гарантованим рівнем безпеки.

6. ***Гарантії безпеки***. Взагалі ОК приділяє значну увагу рівню гарантованості безпеки. Вважається, що комп'ютерна система повинна містити апаратні та/або програмні механізми, які повинні незалежно підтверджувати впевненість в тому, що система має задекларований рівень захищеності, а підсистема безпеки працює тільки так, як заплановано. Для досягнення цієї мети необхідно два типи гарантій:

а) ***операційна гарантія*** – впевненість в тому, що реалізація спроектованої системи забезпечує правильне втілення розробленої системи захисту;

б) ***гарантія життєвого циклу*** – впевненість в тому, що розробка та підтримка системи виконані відповідно до формалізованих та жорстко контрольованих критеріїв функціонування; сюди відносяться: аудит спроектованої системи безпеки, технічного завдання, способів керування налаштуваннями та відповідності реальних параметрів тим, які були заявлені;

в) гарантії неперервності захисту – надійні механізми, які забезпечують неперервний захист основних засобів від несанкціонованих змін.

7. До ***гарантій неперервності захисту*** можна ще додати те, що система повинна бути спроектована таким чином, щоби користувачі не допускалися до обробки інформації не через систему захисту. Простіше кажучи, при завантаженні системи спочатку стартує підсистема захисту, а потім, коли служби захисту активні, – вже допускаються користувачі.

Згідно з таким уявленнями про безпечні та довірені системи, в ОК було розроблено вимоги для захищених систем.

2.2. Common Criteria ("Загальні критерії") – європейський стандарт у галузі оцінки захищеності комп'ютерних систем

Цей стандарт було розроблено у 1999 році, причому відчувається великий вплив Оранжевої книги. Стандарт ISO/IES 15408, повна назва якого "*Evaluation criteria for IT security*" ("*Критерії оцінки безпеки інформаційних технологій*"), у російськомовній літературі отримав назву "*Загальні критерії*" (ЗК).

Сьогодні "*Загальні критерії*" – це найбільш повний та сучасний стандарт оцінювання. Насправді – це метастандарт, який визначає інструменти оцінки безпеки інформаційних систем і порядок їх використання. ЗК містять два основних типи вимог безпеки [36; 44; 45]:

функціональні, які ставляться до функцій безпеки та механізмів, що їх реалізують;

вимоги довіри, які ставляться до технології та процесу розробки і експлуатації.

Функціональні вимоги згруповані на основі ролі, яку вони виконують або засобу безпеки, який вони обслуговують. Усього є 11 функціональних класів, розбитих на три групи, 66 сімейств, 135 компонентів.

Перша група визначає елементарні сервіси безпеки:

1. **FAU** – вимоги до сервісу аудиту;
2. **FIA** – вимоги до ідентифікації та автентифікації;
3. **FRU** – вимоги до використання ресурсів.

Друга група визначає сервіси, похідні від елементарних:

1. **FCO** – вимоги до безпеки комунікацій учасників інформаційного обміну.
2. **FPR** – вимоги до приватності.
3. **FDP** – вимоги до захисту даних користувача.
4. **FPT** – вимоги до захисту функцій безпеки самої підсистеми захисту.

Третя група пов'язана з інфраструктурою системи:

1. **FCS** – вимоги до криптографічної підтримки (обслуговування керування ключами та операцій шифрування/розшифрування).
2. **FMT** – вимоги до керування засобами безпеки.
3. **FTA** – вимоги до керування доступом.
4. **FTP** – вимоги до довіреного каналу.

Вимоги довіри (гарантій безпеки) – вимоги, що ставляться до технологій та процесу розробки та експлуатації об’єкта. Ці вимоги розділено на 10 класів, 44 сімейства та 93 компонента. Усі десять класів розбиті на дві групи.

Перша група містить класи вимог, які передують розробці та оцінці об’єктів:

1. **APE** – вимоги до оцінки профілю захисту;
2. **ASE** – вимоги до оцінки завдання безпеки.

Друга група пов’язана з етапами життєвого циклу об’єкта, що розглядається:

1. **ADV** – вимоги до розробки, проектування об’єкта;
2. **ALC** – вимоги до підтримки життєвого циклу;
3. **ACM** – вимоги до керування конфігурацією;
4. **AGD** – вимоги до посібника користувача та адміністратора;
5. **ATE** – вимоги до тестування;
6. **AVA** – вимоги до оцінки вразливостей;
7. **ADO** – вимоги до постачанню та експлуатації;
8. **AMA** – вимоги до підтримки довіри після сертифікації об’єкта.

Розглянемо далі коротко найважливіші класи ЗК.

2.2.1. Функціональні вимоги безпеки

Нагадаємо, що у цьому розділі містяться три класи функціональних вимог безпеки [44; 45]:

1. **FAU** – аудит, тобто вимоги до сервісу аудиту.
2. **FIA** – вимоги до ідентифікації та автентифікації.
3. **FRU** – вимоги до використання ресурсів.

Клас FAU складається з шести сімейств (**FAU_GEN, FAU_SEL, FAU_STG, FAU_SAR, FAU_SAA**), користуючись якими можна сформулювати вимоги до відбору, реєстрування, збереження та аналізу даних про події, що мають відношення до інформаційної безпеки.

Сімейство FAU_GEN описує генерування даних для аудиту. Ця система включає два компоненти, FAU_GEN.1 та FAU_GEN.2. Система, яка задовольняє клас FAU_GEN.1, повинна вести журнал аудиту з мінімальним набором даних (дату, час, тип та результат події, ідентифікатор суб’єкта). Система класу FAU_GEN.2 окрім цього ще повинна

в обов'язковому порядку асоціювати кожну подію з ідентифікатором користувача, що його ініціював.

Сімейство FAU_SEL визначає вимоги до засобів відбору подій для аудиту. Відбір може відбуватися на основі таких атрибутів, як ідентифікатори об'єктів або суб'єктів, ідентифікатора користувача, вузла мережі, тип події тощо. Також передбачається визначення додаткових атрибутів.

Сімейство FAU_STG визначає вимоги до зберігання даних аудиту. Воно містить дві пари компонентів: FAU_STG.1 та FAU_STG.2; FAU_STG.4 та FAU_STG.5. FAU_STG.1 визначає вимоги до захисту даних журналів аудиту; FAU_STG.2 – гарантії доступності журналів аудиту для адміністраторів. Друга пара визначає дії системи у випадку можливої втрати даних аудиту. Згідно з FAU_STG.4 і FAU_STG.5 система може ігнорувати та заборонити події без аудиту, записати нові дані, знищивши найстарші тощо.

Сімейство FAU_SAR визначає права на повне або часткове читання (на основі критеріїв з логічними функціями) журналів реєстраційної інформації уповноваженими користувачами та заборона доступу до них для решти користувачів.

Сімейство FAU_SAA встановлює вимоги до засобів автоматичного аналізу функціонування системи, які дозволяють з'ясувати можливі порушення безпеки. Це сімейство також складається з чотирьох компонентів. Базовий компонент сімейства, FAU_SAA.1 регламентує застосування набору правил накопичення або об'єднання подій, які сигналізують про імовірне порушення політики безпеки. FAU_SAA.2 підсилює FAU_SAA.1, він уводить поняття профілю поведінки, рейтингу підозрілої активності для кожного користувача, а також порогу, перевищення якого вказує на імовірне порушення політики безпеки. FAU_SAA.3 та FAU_SAA.4 визначають поняття сигнатури атаки різних ступенів складності та функції виявлення сигнатур у реальному масштабі часу. Разом вони визначають просту та складну евристику атаки.

Шосте сімейство, FAU_ARP (автоматична реакція аудиту безпеки), визначає дії, які необхідно виконати системі після виявлення імовірних порушень безпеки. Таким чином, ЗК вперше застосовують методи активного аудиту.

Шість сімейств класу FIA містять вимоги до ідентифікації та автентифікації користувачів та до пов'язування атрибутів безпеки з ідентифікаторами користувачів.

Сімейство FIA_UID відповідає за ідентифікацію користувачів, складається з двох компонентів та визначає набір дій, які дозволено виконувати до ідентифікації. Наприклад, такою дією може бути отримання довідкової інформації. Однак більш сильний компонент, а саме FIA_UID.2, не дозволяє ніяких дій до виконання ідентифікації.

Сімейство FIA_UAU (автентифікація користувача) специфікує механізми автентифікації та її атрибути. Компоненти FIA_UAU.1 (вибір моменту автентифікації) та FIA_UAU.2 (автентифікація до будь-яких дій користувача) спрямовані на безпеку простої автентифікації, а FIA_UAU.3 (захищена від підробок автентифікація) та FIA_UAU.4 (механізми одноразової автентифікації) – на реалізацію надійної автентифікації, стійкої до мережевих атак. Якщо потрібна ще сильніша автентифікація, використовують FIA_UAU.5 (використання кількох механізмів автентифікації), FIA_UAU.6 (повторна автентифікація) та FIA_UAU.7 (автентифікація із захищеним зворотним зв'язком).

Сімейство FIA_ATD (визначення атрибутів користувача) передбачає наявність у користувачів не тільки ідентифікаторів, але й інших атрибутів безпеки, які визначаються політикою безпеки.

Сімейство FIA_USB описує вимоги до зв'язування атрибутів безпеки користувача з суб'єктом доступу, тобто процесом, який діє в системі від імені цього користувача.

Виявленням та реагуванням на невдалі спроби автентифікації займається ***сімейство FIA_AFL*** (відмови автентифікації). Це означає, що воно визначає дії системи при невдалій автентифікації, а також кількість дозволених спроб автентифікації. Сімейство складається з одного компонента.

Сімейство FIA_SOS специфікує метрику якості інформації, яку сповіщає системі користувач під час автентифікації та вимоги до засобів перевірки цієї якості, наприклад, технічних обмежень на паролі користувачів або програмних генераторів паролів.

Клас FRU (використання ресурсів) складається з трьох сімейств, які призначені підтримувати високий степінь доступності інформації.

Виконання вимог ***сімейства FRU_FLT*** (стійкість до відмов) повинно забезпечити коректну роботу усіх або деяких функцій системи навіть у випадку відмов.

Сімейство FRU_PRS (пріоритет обслуговування) регламентує дії із захисту операцій високого пріоритету з боку операцій з нижчим пріоритетом.

Сімейство FRU_RSA (розподіл ресурсів) визначає квоти для досягнення високої доступності ресурсів.

Однією з переваг ЗК є увага до доступності ресурсів, однак тут не спостерігається системного підходу, оскільки доступність можна підтримувати й іншими механізмами, наприклад, балансування навантаження, проактивне керування, використання багатопроцесорних конфігурацій, організація резервних обчислювальних систем тощо. Ці питання зовсім не розглядаються у ЗК.

2.2.2. Вимоги довіри (гарантії безпеки)

Вимоги довіри – це сукупність вимог, що висуваються до технологій та процесів розробки проектів безпеки, а також до експлуатації захищених систем. Дотримання цих вимог допомагає з'ясувати, наскільки якісним буде проект захисту, чи не матиме він протиріч тощо. Ці вимоги розділено на десять класів та 44 сімейства [44; 45].

Вимоги класу APE (оцінка профілю захисту) та ASE (оцінка завдання безпеки) зосереджуються на перевірці повноти та можливості реалізації бажаного проекту безпеки. Клас APE складається з шести однокомпонентних сімейств, що відповідають структурі профілів захисту.

Сімейство APE_INT висуває вимоги до анотації та вступу до проекту безпеки. Спеціаліст повинен підтвердити, що зібрана інформація не протирічить іншим частинам проекту безпеки. Такі дії стандартні при будь-яких оцінках захищених систем.

Сімейство APE_DES визначає правила опису об'єктів, для яких проектується (або оцінюється) система безпеки. Такий опис повинен, як мінімум, тип та загальну характеристику системи.

Вимоги сімейства APE_ENV зосереджені на аналізі середовища, в якому буде функціонувати (або вже функціонує) система безпеки. Тут необхідно описати усі актуальні загрози, усі обов'язкові для виконання політики безпеки.

Сімейство APE_OBJ висуває вимоги до формування цілей безпеки для системи та її середовища, їх обґрунтування.

Основний зміст профілю захисту складають вимоги безпеки. Ці вимоги формулюються в ***класах APE_REQ та APE_SRE***. Логічне обґрунтування вимог безпеки повинно продемонструвати їх вирішальну роль у досягненні цілей безпеки.

Клас ASE влаштований аналогічно до APE, однак деякі відмінності викликані більшою конкретністю завдання з безпеки порівняно з профілем захисту та наявністю додаткових розділів. Також модифіковано вимоги шести попередніх сімейств та додано два нових.

Дотримання вимог описаних класів дуже важливе, оскільки проблеми на стадії проектування та специфікації обчислювальних систем може призвести до тяжких наслідків на стадії експлуатації, виправити які може бути надзвичайно важко та дорого.

2.2.2.1. Вимоги довіри до етапу розробки. Функціональні вимоги, політика безпеки та специфікація системи є основою для розробки функцій безпеки [44; 45]. ***Клас ADV*** (розробка) складається з семи багатокомпонентних сімейств і містить вимоги для поступового підвищення рівня деталізації проекту аж до реалізації з демонструванням відповідності заданим рівням. У цьому класі передбачено три стилі специфікацій: неформальний, напівформальний і формальний та три способи демонстрації відповідності.

Неформальну специфікацію пишуть як звичайний текст з визначенням необхідних термінів.

Напівформальна специфікація складається за допомогою мови з обмеженим синтаксисом, ***формальна*** – використовує математичну нотацію та вимагає формального доведення.

Технологічні вимоги процедурного характеру складають зміст ***класу ALC*** (підтримка життєвого циклу). Він складається з чотирьох сімейств: ALC_LCD (вимоги до моделей життєвого циклу); ALC_TAT (обґрунтування вибору інструментальних засобів та методів); ALC_DVS (вимоги до безпеки розробки); ALC_FLR (вимоги до усунення недоліків).

Клас ACM (управління конфігурацією) містить три сімейства: ACM_CAP (специфікація можливостей управління конфігурацією); ACM_SCP (представлення реалізації об’єкта); ACM_AUT (автоматизація управління конфігурації).

2.2.2.2. Вимоги довіри до етапу отримання, подання та аналізу результатів розробки. Ці вимоги складаються з трьох класів: AGD (вимоги до посібників користувача); ATE (вимоги до тестування); AVA (оцінка вразливостей).

Клас AGD складається з двох однокомпонентних сімейств, де сформульовано ***вимоги до посібників адміністратора*** (AGD_ADM) та ***користувача*** (AGD_USR).

Клас ATE висуває вимоги до тестування та складається з чотирьох сімейств: ATE_FUN (вимоги до функціонального тестування); ATE_DPT (вимоги до обґрунтування достатньої глибини); ATE_COV (вимоги до обґрунтування достатнього покриття).

Функціональне тестування виконує розробник. При функціональному тестуванні необхідно перевірити усі функції безпеки, звертаючи також увагу на відсутність небажаних (нерегламентованих) режимів функціонування. Аналіз глибини тестування повинен підтвердити достатність розроблених тестів для демонстрування того, що функції безпеки виконуються відповідно до розроблених проектів. Аналіз покриття повинен продемонструвати повну відповідність функціональній специфікації. При такому аналізі необхідно повність перекрити всі зовнішні інтерфейси функцій безпеки.

Сімейство ATE_IND (вимоги до незалежного тестування) регламентує дії тестувальників. Їм необхідно протестувати необхідну підмножину функцій безпеки та підтвердити, що система працює відповідно до специфікацій. Також треба виконати деякі (або навіть усі) тести з тестової документації, щоби підтвердити результати виробника.

Вимоги до оцінки вразливостей висуває **клас AVA**. Основою для нього служить аналіз вразливостей (сімейство AVA_VLA), який виконується і розробником, і аналітиком. Це сімейство має чотири компоненти, які підвищують вимоги: AVA_VLA.1 та AVA_VLA.2 висувають вимоги до низького потенціалу порушника; AVA_VLA.3 (середній потенціал порушника), AVA_VLA.4 – високого потенціалу.

Вимоги до аналізу стійкості функцій безпеки системи висуваються ***сімейством AVA_SOF*** і виконується на рівні механізмів. Це означає, що для кожного механізму необхідно показати, що його стійкість досягла або перевищила рівень, заданий у профілі захисту або проекті безпеки.

Вимоги ***сімейства AVA_MSU*** (неправильне застосування) спрямовані на те, щоб виключити можливість такого конфігурування або засто-

сування системи, які є небезпечними в той час, коли адміністратор або користувач вважають їх абсолютно безпечними. Для того, щоб виключити таку можливість, необхідно забезпечити повну однозначність вказівок посібників, проаналізувати їх текст на наявність необґрунтованих, протирічивих процедур. Небезпечні стани повинні легко виявлятися. Аналітики повинні повторити усі процедури конфігурування та інсталяції системи, інші процедури для підтвердження того факту, що систему можна безпечно конфігурувати та використовувати, використовуючи лише надані посібники. Крім цього, вони повинні виконати незалежне тестування та перевірити, чи зможуть адміністратор та користувач за допомогою посібника з'ясувати, що система відконфігурована або використовується небезпечним чином.

Аналіз прихованих каналів, який регламентується ***сімейством AVA_CCA***, досить складний. Розробник повинен провести вичерпний пошук прихованих каналів для кожної політики керування інформаційними потоками та надає документацію для аналізу, де ідентифіковано приховані канали та проведено оцінку їхньої пропускної здатності, описано найбільш небезпечні варіанти використання кожного прихованого каналу. Аналітик повинен вибірково підтвердити правильність зробленого виробником аналізу прихованих каналів засобами незалежного тестування.

2.2.2.3. Вимоги до постачання та експлуатації, підтримка довіри. Вимоги до постачання та експлуатації визначаються **класом ADO**, який складається з двох сімейств: трикомпонентного ADO_DEL (вимоги до постачання), ADO_IGS (вимоги до встановлення, генерування та запуску). Цей клас висуває вимоги до документації та процедур, які здатні виявити відмінності поставленого продукту від оригіналу, а також спроб підміни від імені розробника, якщо останній нічого не постачав. Щодо процедур встановлення та запуску, то процедури безпечного виконання цих операцій також мають бути однозначно визначені та описані в посібниках адміністратора та користувача.

Клас AMA (вимоги до підтримки довіри) складається з чотирьох сімейств та містить вимоги, які будуть корисні після сертифікації системи. Вони допомагають зберегти впевненість у тому, що система продовжує відповідати заданим вимогам безпеки після того, як з часом відбулися зміни у ній самій або у оточуючому середовищі. Тут мова йде про появу нових загроз або вразливостей, зміну вимог користувача або виправлення помилок.

Дії з підтримки довіри мають циклічний характер. Кожна ітерація циклу складається з двох фаз: приймання системи для підтримки та моніторингу системи. Фаза приймання включає розробку планів підтримки довіри та категоріювання компонентів системи за їх впливом на безпеку. Елементи фази моніторингу – подання опису поточної версії системи та виконання аналізу впливу змін на рівень безпеки. Можливо, наприкінці ітерації виявиться, що план або категорія системи вимагають уточнення або зміни; тоді нова ітерація почнеться з повторного приймання системи.

Однак цикл підтримки довіри не може тривати безмежно. Рано чи пізно в системі накопичиться багато мілких змін та недоліків, виникнуть серйозні проблеми, які вимагатимуть повторного оцінювання системи.

2.2.3. Рівні оцінки довіри "Загальних критеріїв"

"Загальні критерії" визначили сім впорядкованих рівнів оцінки довіри безпеки. Ці рівні містять комбінації вимог довіри, розраховані на довгострокове застосування. Така шкала надає можливість збалансувати отриманий рівень довіри з термінами, складністю, вартістю розробки.

Перший рівень довіри передбачає функціональне тестування та може застосовуватися тоді, коли вимагається деяка впевненість, що система захисту працює бездоганно, а загрози не вважаються серйозними. Його можна досягнути з мінімальними витратами без допомоги розробників аналізом функціональних специфікацій, специфікації інтерфейсів, експлуатаційної документації разом з незалежним тестуванням.

Другий рівень довіри передбачає структурне тестування і доступ до частини проектної документації та результатам тестування розробником. Цей рівень застосовують тоді, коли розробнику або замовнику необхідно отримати середній рівень довіри при відсутності доступу в повному обсязі до документації розробника.

На додаток до Першого рівня, тут необхідний аналіз проекту верхнього рівня. Аналіз повинен підтримуватися незалежним тестуванням функцій безпеки, актом розробника про випробування функціональних специфікацій, вибірковими незалежними підтвердженнями результатів тестування; аналізом стійкості функцій та свідоцтвом пошуку явних вразливостей. Також необхідний перелік конфігурації системи з унікальною ідентифікацією її елементів та свідоцтва безпечних процедур постачання.

Третій рівень довіри передбачає систематичне тестування та перевірку функцій безпеки і дозволяє досягти максимально можливої довіри при використанні звичайних методів розробки. Його застосовують тоді, коли розробникам або споживачам потрібен середній рівень довіри на основі всебічного дослідження системи та процесу її розробки.

Порівняно з Другим рівнем тут додається вимога, яка змушує розробника розробити акт про випробування з урахуванням особливостей не тільки функціональної специфікації, але й проекту вищого рівня. Крім того, вимагається контроль середовища розробки та керування конфігурацією системи.

Четвертий рівень довіри передбачає систематичне проектування, тестування та спостереження. Вони допомагають досягти максимально можливої довіри за умови стандартної практики комерційних розробок. Це максимально можливий рівень довіри, на який, напевне, економічно доцільно орієнтуватися для існуючих типів продуктів.

Четвертий рівень характеризується аналізом функціональної специфікації, повної специфікації інтерфейсів, експлуатаційної документації, проектів вищого та нижнього рівнів, а також підмножини реалізації; необхідно також застосовувати неформальну політику безпеки системи. Додаткові вимоги – незалежний аналіз вразливостей, який демонструє стійкість системи до спроб проникнення порушників з низьким потенціалом, та автоматизація керування конфігурацією.

П’ятий рівень довіри вимагає напівформального проектування та тестування. З його допомогою досягається максимально можлива довіра для строгої практики комерційної розробки, яка підтримується помірним застосуванням спеціалізованих методів забезпечення захисту. Цей рівень використовують, коли потрібен високий рівень довіри та строгий підхід до розробки, який не несе зайвих витрат. Для досягнення п’ятого рівня вимагається формальна політика безпеки системи та напівформальне представлення функціональної специфікації та проекту вищого рівня, напівформальна демонстрація відповідності між ними, а також модульна структура проекту системи. Необхідна стійкість до спроб проникнення порушників з середнім потенціалом; передбачається також перевірка правильності аналізу прихованих каналів розробником та всебічне керування конфігурацією.

Шостий рівень довіри характеризується напівформальною верифікацією проекту. Він дозволяє отримати високу довіру застосуван-

ня спеціальних методів проектування у строго контрольованому середовищі розробки, яке застосовується для виробництва високоякісних виробів інформаційних технологій та для захисту цінних активів від значних ризиків. Шостий рівень довіри має такі особливості: структуроване представлення реалізації; напівформальне представлення проекту нижнього рівня; ієрархічна структура проекту системи; стійкість до спроб проникнення порушників з високим потенціалом;перевірка правильності систематичного аналізу прихованих каналів розробником; використання структурованого процесу розробки;повна автоматизація керування конфігурацією.

Сьомий рівень довіри вимагає формальну верифікацію проекту. Він застосовується до розробки виробів інформаційних технологій для використання в ситуаціях надзвичайно високого ризику або там, де висока цінність активів виправдовує підвищені витрати.

На додаток до шостого рівня, на сьомому вимагають:

формальне представлення функціональної специфікації та проекту верхнього рівня, формальна демонстрація відповідності між ними;

модульна, ієрархічна та проста структура проекту системи;

додавання представлення реалізації як основи акту випробувань;

повне незалежне підтвердження результатів тестування розробником. Рівні довіри ЗК, на думку аналітиків [44; 45] обрані досить вдало. Тому вважається, що їх підсилення, в разі необхідності, може носити непринциповий характер. Більше дізнатися про ЗК можна в работах [22; 23; 36; 44; 45] та почитавши самі "Загальні критерії" [42].

2.3. НД ТЗІ 2.5-004-99 "Критерії оцінки захищеності інформації у комп'ютерних системах від несанкціонованого доступу"

Цей нормативний документ встановлює критерії оцінки захищеності інформації, яка обробляється комп'ютерними системами, від несанкціонованого доступу [29]. Критерії є методологічною базою для визначення вимог із захисту інформації; створення захищених комп'ютерних систем і засобів захисту від несанкціонованого доступу; оцінки захищеності інформації в комп'ютерних системах та їх придатності для обробки інформації, що потребує захисту. Критерії надають: метрику для оцінки надійності механізмів захисту інформації від несанкціонованого доступу; базу для розробки комп'ютерних систем, де повинні бути реалізовані функції захисту інформації.

Критерії можуть застосовуватися до усього спектру комп'ютерних систем і призначаються для розробників, споживачів комп'ютерних систем, які використовуються для обробки критичної інформації, а також для органів, що виконують оцінювання захищеності такої інформації та контроль за її обробкою.

Структура Критеріїв подана на рис. 2.1.

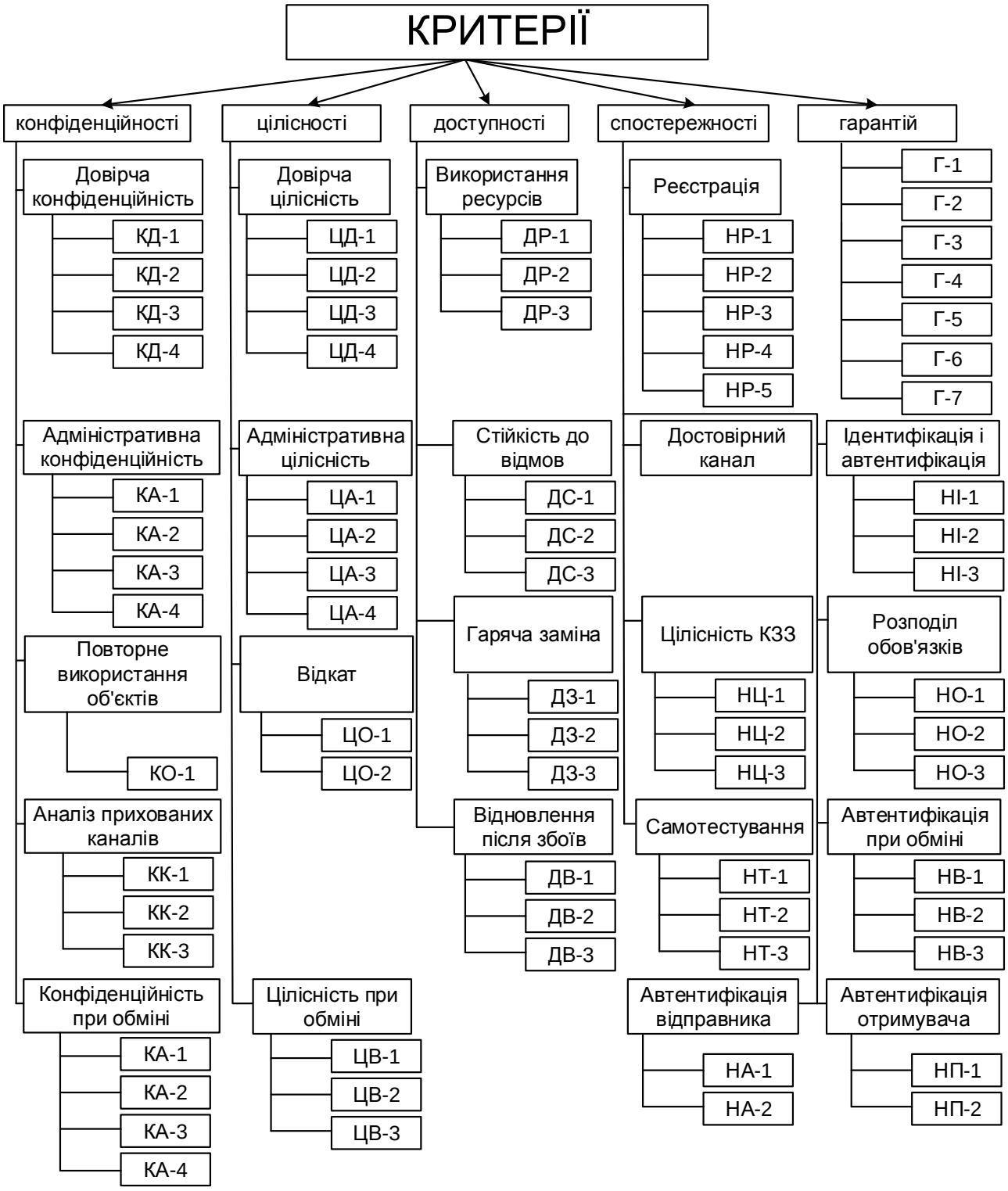


Рис. 2.1. **Структура критеріїв НД ТЗІ 2.5-004-99**

Як видно з рис. 2.1, в документі розглядаються вимоги двох типів: вимоги до функцій захисту (або послуг безпеки); вимоги до гарантій.

Комп'ютерна система розглядається як набір функціональних послуг. Кожна послуга є набіром функцій, що дозволяють протистояти певній сукупності загроз і включає кілька рівнів. Чим вищий рівень послуги, тим повніше забезпечується захист від певного типу загроз.

Функціональні критерії розбито на чотири групи, кожна з яких визначає вимоги до захисту від загроз одного з чотирьох основних типів.

Конфіденційність. Загрози, пов'язані з несанкціонованим ознайомленням з інформацією, становлять загрози конфіденційності. У цьому розділі описано такі послуги, як довірча конфіденційність; адміністративна конфіденційність; повторне використання об'єктів; аналіз прихованих каналів; конфіденційність при обміні (експорті та імпорті).

Цілісність. Загрози несанкціонованої модифікації інформації становлять загрози цілісності. В цьому розділі описано такі послуги: довірча цілісність; адміністративна цілісність; відкат і цілісність при обміні.

Доступність. Загрози, що стосуються порушення можливості використання комп'ютерних систем або інформації, що обробляється ними, становлять загрози доступності. В цьому розділі описано такі послуги, як використання ресурсів, стійкість до відмов, гаряча заміна та відновлення після збоїв.

Спостережність. Ідентифікація та контроль за діями користувачів, керування комп'ютерною системою становлять предмет послуг спостережності та керованості. У цьому розділі описано такі послуги: реєстрація, ідентифікація та автентифікація, достовірний канал, розподіл обов'язків, цілісність комплексу засобів захисту, саме тестування, автентифікація при обміні, автентифікація відправника (неможливість відмови від авторства), автентифікація одержувача (неможливість відмови від одержання).

Крім функціональних критеріїв цей документ містить критерії гарантій, що дозволяють оцінити коректність реалізації послуг. Критерії гарантій включають вимоги до архітектури комплексу засобів захисту (КЗЗ); середовища розробки; випробування КЗЗ; середовища функціонування та експлуатаційної документації. У Критеріях уведено сім ієрархічних рівнів гарантій, що нагадує Загальні критерії. Ієрархія рівнів гарантій

надає споживачеві міру впевненості в тому, що система безпеки здатна протистояти актуальним загрозам інформації. Ця впевненість зростає зі збільшенням номеру рівня гарантій.

Усі послуги до деякої міри незалежні. Якщо ж така залежність виникає, тобто реалізація деякої послуги неможлива без реалізації іншої, цей факт відображено як необхідну умову для даної послуги або рівня. За винятком послуги аналіз прихованих каналів залежність між функціональними послугами і гарантіями відсутня. Однак **рівень послуги** "Цілісність КЗЗ" **НЦ-1 – необхідна умова абсолютно для всіх рівнів усіх інших послуг**. Порядок оцінки комп'ютерної системи щодо відповідності цим Критеріям визначається відповідними нормативними критеріями, зокрема, НД ТЗІ 3.7-001-99 "Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі". Експертна комісія повинна визначити, які послуги і на якому рівні реалізовано в комп'ютерній системі, що оцінюється, і як дотримуються вимог гарантій. Результатом оцінки є рейтинг, тобто перелік комбінацій, які позначають рівні реалізованих послуг разом з рівнем гарантій. Для того, щоб до рейтингу комп'ютерної системи міг бути включений певний рівень послуги чи гарантій, необхідно виконати усі вимоги цього рівня.

Контрольні запитання

1. Основні поняття щодо системи безпеки відповідно до "Оранжевої книги".
2. Критерії європейського стандарту у галузі оцінки захищеності комп'ютерних систем.
3. Основні поняття українського стандарту у галузі оцінки захищеності комп'ютерних систем – НД ТЗІ 2.5-004-99 України.
4. Основні рівні довіри відповідно до європейського стандарту.
5. Послуги та механізми безпеки інформаційних систем.
6. Вимоги до постачання та експлуатації, підтримка довіри у відповідності до європейського стандарту.
7. Вимоги довіри (гарантії безпеки) на різних етапах життєвого циклу системи безпеки.

Розділ 3. Шифрування даних

Завдання захисту інформації в комп'ютерних системах перетворюється сьогодні в одну з найактуальніших внаслідок широкої розповсюдженості таких систем, а також розширення локальних і глобальних комп'ютерних мереж, якими передаються величезні об'єми інформації державного, військового, комерційного, приватного характеру, власники якої часто були б категорично проти ознайомлення сторонніх осіб з цією інформацією.

Не менш важливим завданням вважається широке впровадження в різні сфери діяльності людини електронного документообігу, який повинен забезпечуватися юридичною чинністю підписаних електронних документів.

Усі ці та багато інших завдань захисту інформації покликана вирішувати криптографія.

Криптографічні механізми настільки тісно пов'язані з сучасними інформаційними технологіями, що разом з підвищенням комп'ютерної грамотності необхідно опановувати основи криптографії.

Грецьке слово *cryptos* перекладається як "таємниця", а отже, криптографія означає тайнопис. Звідси випливає, що початковим завданням криптографії було розроблення методів, спрямованих на приховування змісту переданої або збереженої інформації. І хоча на цей час сфера застосування криптографічних механізмів значно розширилася, основні ідеї можна проілюструвати саме на прикладі забезпечення конфіденційності інформації.

Існує множина публікацій, які містять історичні огляди з криптографії (наприклад у роботах [2; 3; 22; 47 – 59]). Варто зазначити, що кожному етапові розвитку цивілізації властиві відповідні криптографічні пристрої. Тривалий час шифрування текстів виконувалося вручну. Їх створення можна було вважати скоріше мистецтвом, ніж якоюсь стандартною процедурою. Відомо два протилежні погляди щодо шифрів. Відповідно до першого можна створити шифр, який неможливо розкрити. Другий погляд відбивав таку точку зору: малоймовірно, що "загадку", яка лежить в основі створеного шифру, не можна розгадати. Згодом ***науку про перетворення інформації у незрозумілу для сторонніх осіб форму*** стали називати ***криптографією***. Методи пошуку "розгадки"

стали називати *криптоаналітичними методами,* а відповідну галузь досліджень – *криптоаналізом.* Отже, ***криптоаналіз – це наука, спрямована на подолання криптографічного захисту***. Тепер усе ширше використовують термін ***криптологія,*** тобто наука про шифри. Вважають, що криптологію складають дві великі частини, які доповнюють одна одну, – ***криптографія*** та ***криптоаналіз***.

Процес криптографічного перетворення інформації автори будуть називати ***шифруванням*** (або ***зашифруванням***, як це часто використовується у криптологічній літературі). Зашифровану інформацію повинні прочитати ті особи, для кого призначена ця інформація. Перш ніж прочитати, її треба перетворити у зрозумілу форму. Цей процес, який називається ***розшифруванням***, виконується за допомогою деякої секретної частини криптографічної системи – ***криптографічного ключа*** (або просто ***ключа***).

Зловмисник, який перехопив зашифровану інформацію, як правило, не має такого ключа. Тому він намагається подолати криптографічний захист за допомогою криптоаналітичних методів. Такий спосіб, тобто розшифрування повідомлення без знання ключа, необхідно назвати ***дешифруванням***. Отже, можна сказати, що розшифровують "свої", а дешифрують – "чужі".

Методи криптографічного захисту інформації можуть реалізовуватися як апаратно, так і програмно. Апаратна реалізація має суттєво більшу вартість, однак водночас і більшу продуктивність та захищеність. Програмна реалізація практичніша, дешевша та гнучкіша у використанні.

3.1. Основні поняття роботи К. Шеннона "Теорія зв’язку в секретних системах"

Клод Елвуд Шеннон, у роботі "Теорія зв’язку в секретних системах" [48] зробив визначальний внесок у сучасну криптографічну науку. Прийнято вважати, що зазначена робота, яка була спершу його секретною доповіддю "Математична теорія криптографії" (1945 рік, розсекречено після Другої світової війни, у 1949 році), визначила основи та сформувала обличчя сучасної криптографії. Дехто порівнюють його внесок з впливом на фізику робіт Ісаака Ньютона.

К. Шеннон народився у 1916 р. в м. Гейлорді (штат Мічиган, США). У 1936 році він закінчив Масачусетський технологічний інститут, спеціалізуючись одночасно на математиці та електротехніці. Саме таке поєднання дозволило у 1940 р. захистити докторську дисертацію, де він уперше застосував до описання роботи реле та перемикачів булеву алгебру. Зараз такий аналіз лежить в основі сучасної цифрової схемотехніки, а на той час це було майже революційною справою. Сам К. Шеннон на це скромно зауважував: "Просто так склалося, що ніхто інший не був знайомий з обома сферами одночасно".

У 1941 р. К. Шеннона запросили на роботу в Bell Laboratories, де у роки війни він займався розробкою криптографічних систем, що пізніше допомогло йому відкрити методи кодування з корекцією помилок. Метою К. Шеннона була оптимізація передавання інформації телефонними та телеграфними лініями. Для того, щоб вирішити цю проблему, йому довелося сформулювати, що ж таке інформація, чим визначається її кількість. У роботах 1948 – 1949 р. він визначив кількість інформації через ентропію – величину, яка використовується у термодинаміці та статистичній фізиці як міра розупорядкованості системи, а за одиницю інформації – величину, яку потім було названо "бітом", тобто вибір з двох рівноймовірних варіантів.

К. Шеннон розглядає шифрування як відображення відкритого тексту в шифрограму [48]:

$$C = F_i(M),$$

де C – шифрограма;

M – відкритий текст;

F_i – відповідне відображення, індекс i відповідає конкретному криптографічному ключу, використаному при шифруванні.

Для того, щоб існувала можливість однозначного розшифрування повідомлення, відображення F_i повинно мати єдине обернене відображення F_i^{-1}, таке, що $F_i F_i^{-1}=I$, де I – тотожне перетворення:

$$M = F_i^{-1}(C).$$

Ураховано, що джерело ключів є статистичним процесом або пристроєм, що задає відображення F_1, F_2, …, F_N з імовірностями $p_1,p_2,\ldots,p_N$.

Розглянемо [33] найпростіший шифр, де вихідний алфавіт повідомлень співпадає з множинами знаків ключа та криптограми, а шифрування виконується послідовною заміною знаків відкритого тексту знаками криптограми залежно від чергового значення знаків ключа. У такому разі відкритий текст, ключ і криптограма є послідовностями літер того самого алфавіту: $M = (m_1 m_2 m_3 \ldots m_n)$; $K = (k_1 k_2 k_3 \ldots k_n)$; $C = (c_1 c_2 c_3 \ldots c_n)$. Кожен крок шифрування визначається співвідношенням $c_i = f(m_i, k_i)$.

У практичних криптосистемах довжина ключа значно менша за довжину відкритого тексту, тому часто ключова послідовність може обчислюватися за допомогою деякого первісного ключа меншого розміру або навіть може бути періодичною.

Завдання криптоаналітика полягає в обчисленні відкритого тексту за криптограмою, знаючи множину відображень F_1, F_2, …, F_N. Існують криптосистеми, для яких будь-який об'єм перехопленої інформації недостатній для знаходження шифрувального відображення, причому ситуація не залежить від обчислювальної потужності обладнання криптоаналітика. Шифри такого типу називаються ***безумовно стійкими*** (за К. Шенноном – ***ідеально секретними***). Строго кажучи, безумовно стійкими будуть такі шифри, для яких криптоаналітик, навіть маючи безмежні обчислювальні ресурси, не зможе покращити оцінку вихідного повідомлення (відкритого тексту) M, знаючи криптограму C, порівняно з оцінкою при невідомій криптограмі. Це можливо лише тоді, коли M і C є статистично незалежними. Безумовно стійкі криптосистеми існують, що легко показати [24]. Нехай у розглянутому простому шифрі використовується алфавіт з L літер, а поточні знаки криптограми генеруються за законом:

$$c_i = f(m_i, k_i) = (m_i + k_i) \bmod L,$$

де кожному знаку c_i, m_i, k_i поставлено у відповідність їх порядковий номер у алфавіті.

Оберемо в якості ключа послідовність з n випадкових знаків $k_1 k_2 k_3 \ldots k_n$, тобто оберемо випадковий ключ, розмір якого дорівнює довжині повідомлення. Для генерування ключа використаємо деякий фізичний генератор випадкових чисел, що забезпечує рівну імовірність кожного елемента з множини чисел $\{1,2,\ldots,N\}$. Обране джерело

забезпечує рівноймовірність вибору будь-якого ключа довжини n. У такому разі імовірність вибору ключа довжини n складає:

$$P(K = K_i) = L^{-n}.$$

З цього виразу видно, що для довільних M і C виконується аналогічне співвідношення:

$$P(M = M_i/C = C_i) = L^{-n}.$$

А це, у свою чергу, означає, що криптограмі довжини n з імовірністю L^{-n} може відповідати будь-який відкритий текст довжини n. Для шифрування іншого повідомлення оберемо інший випадковий ключ. Така процедура шифрування забезпечує безумовну стійкість. Криптосистеми, що використовують рівно ймовірний випадковий ключ, що має рівну з відкритим текстом довжину, називаються шифрами зі ***стрічкою одноразового використання*** або шифрами з ***безмежною ключовою гамою***. На практиці такі системи отримали лише обмежене використання, оскільки досить незручні у використанні.

Криптосистеми іншого типу характеризуються тим, що при зростанні кількості доступної для криптоаналітика інформації, при певному значенні $n = n_0$ існує єдиний розв'язок криптоаналітичної задачі. Мінімальний об'єм криптограми, для якого існує єдиний розв'язок, називається ***інтервалом єдиності***. У випадку стрічки одноразового використання $n_0 \to \infty$. За кінцевої довжини криптографічного ключа значення n_0 кінцеве. Відомо, що за криптограмою, довжиною, більшою за інтервал єдності, можна знайти цей єдиний розв'язок. Однак для криптоаналітика з обмеженими обчислювальними ресурсами, імовірність знайти цей розв'язок за скінчений проміжок часу (поки інформація має ще якусь цінність) надзвичайно мала (10^{-30} і менша). Шифри такого типу називаються ***умовно стійкими*** (практично стійкими за К. Шенноном). Їх стійкість ґрунтується на значній обчислювальній складності розв'язку криптоаналітичної задачі.

Мета розробника умовно стійких криптосистем полягає в тому, щоб зменшити витрати на процедури шифрування та розшифрування, і одночасно задати такий рівень складності криптоаналітичної задачі, щоб для успішного розв'язку її потрібно було залучити такі ресурси, вартість яких перетворювала знаходження рішення в економічно невигідну задачу.

Завдання такого об'єму обчислень називаються важкими або обчислювально складними, а про їх розв'язок говорять, що вони обчис-

лювально нездійсненними. Шифри, які ґрунтуються на обчислювально нездійсненних задачах, називаються ***обчислювально стійкими***. Найбільшу практичну розповсюдженість мають якраз обчислювально стійкі криптосистеми.

Під стійкістю криптосистем такого роду будемо розуміти складність розв'язку криптоаналітичної задачі при певних умовах. К. Шеннон увів поняття ***робочої характеристики*** $W(n)$ шифру як середню кількість роботи для знаходження ключа за відомими n знаками криптограми з використанням найкращого алгоритму криптоаналізу. Кількість роботи можна виміряти, наприклад, кількістю операцій, які необхідно виконати для обчислення ключа. Цей параметр безпосередньо пов'язаний з алгоритмом обчислення ключа. Складність визначення $W(n)$ пов'язана зі складністю знаходження найкращого способу розкриття. Особливо цікавим є граничне значення $W(n)$ для $n \to \infty$. Сьогодні невідомо обчислювально стійких криптосистем, для яких обчислено $W(\infty)$. Внаслідок складності такої оцінки, практичні шифри характеризують досягнутою оцінкою робочої характеристики, яку отримують для найкращого з відомих сьогодні методів обчислення ключа.

К. Шеннон запропонував також модель для оцінки інтервалу єдиності, з якої отримано співвідношення:

$$n_0 = H(K)/D,$$

де $H(K)$ – ентропія ключа, яка для випадкового ключа дорівнює довжині ключа в бітах;

D – надмірність мови, що вимірюється у бітах на знак. Це співвідношення можна записати у вигляді:

$$H(K) \leq n^D,$$

де $H(K)$ характеризує кількість невідомих у двійковому представленні ключа; n^D – кількість рівнянь для обчислення ключа. Якщо кількість рівнянь менше за кількість невідомих, розв'язок системи буде неоднозначним. У таких умовах криптосистема буде безумовно стійкою. Якщо кількість рівнянь більша кількості невідомих, то існує єдиний розв'язок, а криптосистема не вважається безумовно стійкою. Однак вона може бути обчислювально стійкою, якщо $n >> n_0$. Рівень стійкості обчислювально стійких криптосистем залежить від типу шифрувальних процедур (за винятком вибору дуже малого ключа шифрування, коли складність повного перебору можливих ключів дуже мала). Конкретні процедури перетворення також визначає хід робочої характеристики, тобто явний вигляд залежності $W(n)$.

Підсумовуючи наведене, можна сказати, що К. Шеннон започаткував новий етап розвитку криптографічної науки, етап наукової криптографії, який тривав до широкого впровадження комп'ютерної техніки. Саме завдяки його роботам було сформульовано основні наукові поняття цієї науки, які використовуються й досі.

3.2. Симетричні, асиметричні та комбіновані криптосистеми. Їх переваги та недоліки

3.2.1. Симетричні криптосистеми

Розглянемо загальну схему симетричної, або традиційної, криптографії (рис. 3.1).

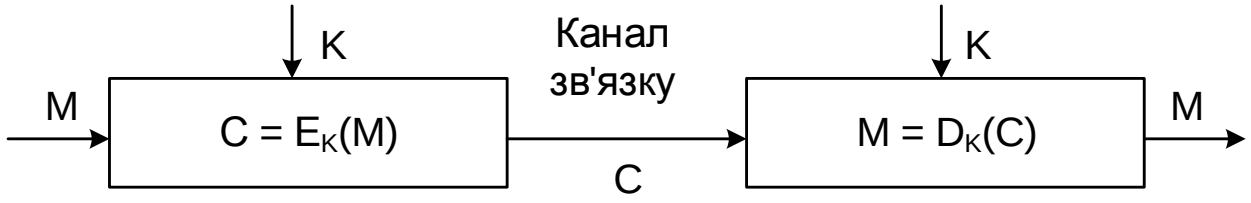


Рис. 3.1. **Загальна схема симетричного шифрування**

У процесі шифрування використовується певний алгоритм шифрування, на вхід якому подаються незашифроване повідомлення (англійською – *plaintext*) і ключ шифрування. Виходом алгоритму є зашифроване повідомлення, що називається англійською *ciphertext (шифротекст)*. Ключ шифрування є значенням, що не залежить від незашифрованого повідомлення. Зміна ключа повинна призводити до зміни зашифрованого повідомлення.

Зашифроване повідомлення передається одержувачу. Одержувач перетворює зашифроване повідомлення у вихідне незашифроване за допомогою алгоритму розшифрування і того ж самого ключа, який використовувався при шифруванні, або ключа, який можна легко одержати з ключа шифрування.

Незашифроване повідомлення будемо позначати *P* або *M*, від слів *plaintext* та *message* (англ. – повідомлення). Зашифроване повідомлення будемо позначати *C*, від слова *ciphertext*.

Стійкість, яку забезпечує традиційна криптографія, залежить від кількох факторів.

По-перше, криптографічний алгоритм повинен бути досить сильним, щоб передане зашифроване повідомлення неможливо було розшифрувати без ключа, використовуючи тільки різні статистичні закономірності зашифрованого повідомлення або які-небудь інші способи його аналізу.

По-друге, безпека переданого повідомлення повинна залежати від секретності ключа, але не від секретності алгоритму. Алгоритм повинен бути проаналізований фахівцями, і позбавлений слабких місць, через які можна відновити погано прихований зв'язок між відкритим і зашифрованим повідомленнями. До того ж для стійких алгоритмів виробники можуть створювати дешеві апаратні засоби або вільно розповсюджувані програми, що реалізують цей алгоритм шифрування.

По-третє, алгоритм повинен бути настільки довершеним, щоб не можна було обчислити ключ, навіть знаючи досить багато пар *зашифроване повідомлення-незашифроване повідомлення*, отриманих при шифруванні з використанням цього ключа.

Клод Шеннон увів поняття ***дифузії*** і ***конфузії*** для опису стійкості алгоритму шифрування [48].

Дифузія – це розсіювання статистичних особливостей відкритого тексту в широкому діапазоні статистичних особливостей зашифрованого тексту. Дифузія досягається тим, що значення кожного елемента відкритого тексту впливає на значення багатьох елементів зашифрованого тексту або, що те ж саме, будь-який елемент зашифрованого тексту залежить від багатьох елементів відкритого тексту.

Конфузія – це знищення статистичного взаємозв'язку між зашифрованим текстом і ключем.

Стандартний алгоритм шифрування повинен бути таким, щоб його можна було успішно використовувати в багатьох галузях: для шифрування даних або потоків даних; для створення певної кількості випадкових бітів; легко перетворюватися в однобічну геш-функцію. Стандартний алгоритм шифрування повинен дозволяти реалізацію на різних платформах, які висувають різні вимоги, в тому числі на спеціалізованій апаратурі шифрування/дешифрування. Додатковими вимогами до стандартних алгоритмів можуть бути: алгоритм повинен бути простим для програмної реалізації, щоб мінімізувати імовірність програмних помилок; простір ключів має бути плоским; алгоритм повинен мати можливість використання довільного випадкового рядка бітів у якості можливого

ключа; наявність слабких ключів небажана; алгоритм повинен легко модифіковуватися для різних рівнів безпеки; бажано, щоб усі операції з даними виконувалися над блоками, кратними або байту, або 32-бітному слову.

Алгоритми симетричного шифрування відрізняються способом, яким обробляється вихідний текст. Можливе шифрування блоками або шифрування потоком. Блок тексту розглядається як додатне ціле число, або як кілька незалежних додатних цілих чисел. Довжина блоку завжди дорівнює 2^n. У більшості блокових алгоритмів симетричного шифрування використовуються такі типи операцій:

таблична підстановка, коли група бітів відображується в іншу групу бітів (так звані S-box);

перестановки, за допомогою яких біти повідомлення перемішуються (так звані P-box);

операція додавання за модулем 2 (XOR або $\oplus$);

операція додавання за модулем 2^{32} або за модулем 2^{16};

циклічний зсув на певну кількість бітів.

Ці операції циклічно повторюються в алгоритмі, утворюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду і ключ, отриманий за певним алгоритмом з ключа шифрування *K*.

Критерії, використані при розробці алгоритмів.

Беручи до уваги перераховані вимоги, вважають, що алгоритм симетричного шифрування повинен:

маніпулювати даними в більших блоках, переважно розміром 16 або 32 біти;

мати розмір блоку 64 ÷ 256 бітів;

мати масштабований ключ до 256 бітів;

використовувати прості операції, ефективні на мікропроцесорах, що виключають додавання, табличні підстановки, або множення за модулем;

не повинні використовуватися зсув змінної довжини, побітні перестановки або умовні переходи;

повинна бути можливість реалізації алгоритму на 8-бітному процесорі з мінімальними вимогами до пам’яті;

використовувати заздалегідь обчислені підключі. При неможливості попереднього обчислення підключів можливе лише зменшення швидкодії. Завжди повинна бути можливість шифрування даних без яких-небудь попередніх обчислень.

Складатися зі змінного числа ітерацій. Для застосувань з маленькою довжиною ключа недоцільно використовувати велику кількість ітерацій для протистояння диференціальним й іншим атакам. Отже, повинна бути можливість зменшити число ітерацій без значної втрати стійкості.

По можливості не мати слабких ключів. Якщо це неможливо, то кількість слабких ключів повинна бути мінімальною, щоб зменшити ймовірність випадкового вибору одного з них. Проте усі слабкі ключі повинні бути заздалегідь відомі, щоб їх можна було відбракувати в процесі створення ключа.

Задіяти підключі, які є однобічним гешем ключа. Це дає можливість використовувати довші парольні фрази без шкоди для безпеки.

Не мати лінійних структур, які зменшують лінійну складність.

Алгоритм повинен бути простим для розуміння, що спрощує його аналіз та пошук слабких місць.

Більшість блокових алгоритмів ґрунтується на використанні сітки Фейстеля, всі мають плоский простір ключів і дозволяють відбраковку слабких ключів.

Ключ раунду називається підключем. Кожний симетричний алгоритм шифрування може бути поданий способом, наведеним на рис. 3.2.

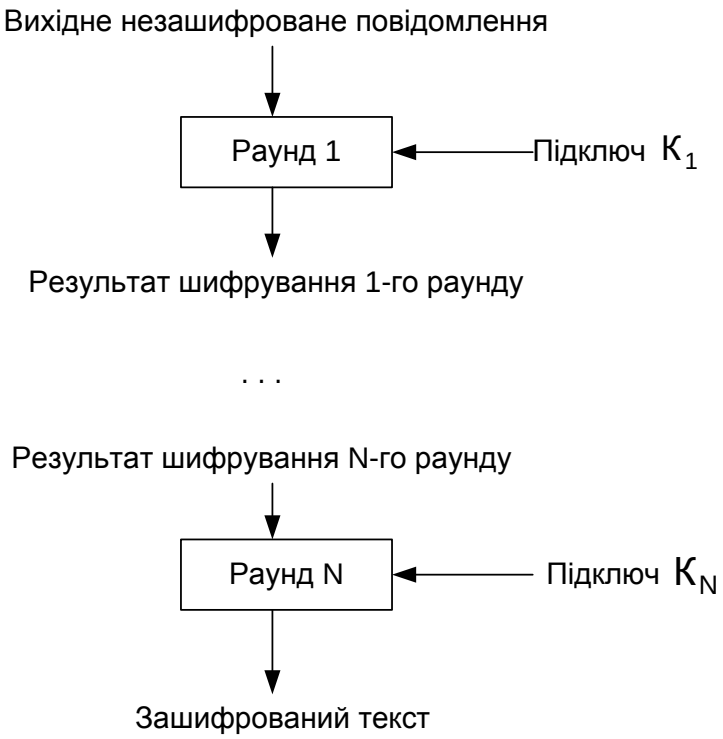


Рис. 3.2. **Структура симетричного алгоритму**

3.2.2. Асиметричні криптосистеми

Якими б не були надійними симетричні криптоалгоритми, слабким місцем їх практичної реалізації залишається проблема розподілу криптографічних ключів. Для безпечного обміну інформацією між двома суб'єктами, один з них повинен згенерувати ключ та якимось чином конфіденційно передати іншому. Таким чином, для передавання криптографічного ключа необхідно використати або існуючу криптосистему, або захищений інформаційний канал. Однак тут постає питання: якщо суб'єкти мають захищений інформаційний канал, то чи не можна використати його для передавання самої інформації? Зрозуміло, що це досить незручно та дорого.

Для вирішення цієї проблеми на основі нових результатів сучасної алгебри було запропоновано системи з публічним ключем (***асиметричні криптосистеми***).

Суть таких систем, як уже зазначалося, полягає в тому, що кожним суб'єктом інформаційного обміну генеруються два ключа, зв'язані між собою певними правилами. Один ключ оголошується ***публічним*** (***відкритим***), а інший – ***приватним*** (***секретним***). Публічний ключ розміщується на доступному усім ресурсі (публікується), тому він доступний для усіх учасників інформаційного обміну. Приватний ключ зберігається суб'єктом, який його створив, і недоступний для інших суб'єктів.

Відкритий текст зашифровується на публічному ключі та передається адресатові. Зашифрований текст не може бути розшифрований на публічному ключі (в усякому разі для досить довгих ключів це обчислювально дуже складна процедура). Розшифрування повідомлення можливо лише на відповідному приватному ключі, відомому лише безпосередньо адресату.

Асиметричні криптосистеми, як вже зазначалося, використовують так звані односторонні функції, які мають таку властивість: при заданому значенні x відносно легко обчислити значення $f(x)$, однак якщо відомо значення $y = f(x)$, то не існує простого способу обчислення значення x. Велика кількість класів незворотних функцій і породжують усю різноманітність криптосистем з відкритим ключем. Однак в самому означенні є деяка невизначеність: що означає "не існує простого способу"?

Тому для асиметричних криптосистем ставляться дві важливих вимоги: перетворення відкритого тексту повинно бути незворотним без можливості його відновлення на публічному ключі; обчислення приват-

ного ключа на основі публічного також повинно бути неможливим на сучасному технологічному рівні. При цьому бажаною є точна нижня оцінка трудомісткості розкриття шифру.

Алгоритми шифрування з відкритим ключем використовують у трьох напрямах:

1) як самостійні засоби захисту інформації;
2) як засоби автентифікації користувачів;
3) як засоби розповсюдження ключів.

Справа в тому, що внаслідок особливостей математичних розрахунків, асиметричні криптоалгоритми значно повільніші за симетричні. Тому часто на практиці раціонально використати перші для шифрування невеликої кількості інформації, а потім за допомогою симетричних алгоритмів виконувати шифрування великих інформаційних потоків.

Комбінований метод шифрування

Головною перевагою криптосистем з відкритим ключем є їх потенційно висока безпека: немає необхідності ні передавати, ні повідомляти будь-кому значення секретних ключів, ні переконуватись в їх дійсності.

У симетричних криптосистемах існує небезпека розкриття секретного ключа під час передачі.

Однак алгоритми, що лежать в основі криптосистем з відкритим ключем, мають такі недоліки:

генерація нових секретних і відкритих ключів заснована на генерації нових великих простих чисел, а перевірка простоти чисел займає багато процесорного часу;

процедури шифрування і розшифрування, пов’язані зі зведенням у степінь багатозначного числа, досить громіздкі.

Тому швидкодія криптосистем з відкритим ключем звичайно на 2 – 5 порядків разів менша за швидкодію симетричних криптосистем з секретним ключем.

Комбінований (гібридний) метод шифрування дозволяє поєднувати переваги високої таємності, надавані асиметричними криптосистемами з відкритим ключем, з перевагами високої швидкості роботи, властивими симетричним криптосистемам із секретним ключем. При такому підході криптосистема з відкритим ключем застосовується для шифрування, передачі і наступного розшифрування тільки секретного ключа симетричної криптосистеми. А симетрична криптосистема застосовується для шифрування і передачі вихідного відкритого тексту. У результаті криптосистема з відкритим ключем не заміняє симетричну

криптосистему із секретним ключем, а лише доповнює її, дозволяючи підвищити в цілому захищеність інформації, яка передається. Такий підхід іноді називають *схемою електронного цифрового конверту*.

Якщо користувач А прагне передати зашифроване комбінованим методом повідомлення М користувачу В, то порядок його дій буде такий.

1. Створити (наприклад, згенерувати випадковим чином) симетричний ключ, названий у цьому методі сеансовим ключем K_s.

2. Зашифрувати повідомлення М на сеансовому ключі K_s.

3. Зашифрувати сеансовий ключ K_s на відкритому ключі K_B користувача В.

4. Передати відкритим каналом зв'язку на адресу користувача В зашифроване повідомлення разом із зашифрованим сеансовим ключем.

Дії користувача В при одержанні зашифрованого повідомлення і зашифрованого сеансового ключа повинні бути зворотними:

5. Розшифрувати на своєму секретному ключі K_B сеансовий ключ K_s.

6. За допомогою отриманого сеансового ключа K_s розшифрувати і прочитати повідомлення М.

При використанні комбінованого методу шифрування можна бути впевненим у тому, що тільки користувач В зможе правильно розшифрувати ключ K_s і прочитати повідомлення М.

Таким чином, при комбінованому методі шифрування застосовуються криптографічні ключі як симетричних, так і асиметричних криптосистем. Очевидно, що вибір довжин ключів для кожного типу криптосистеми слід здійснювати таким чином, щоб зловмиснику було однаково важко атакувати будь-який механізм захисту комбінованої криптосистеми. У табл. 3.1 наведені розповсюджені довжини ключів симетричних і асиметричних криптосистем, для яких труднощі атаки повного перебору приблизно дорівнюють труднощам факторизації відповідних модулів асиметричних криптосистем [23; 27; 34; 35; 61].

Таблиця 3.1

Довжини ключів для симетричних і асиметричних криптосистем

Довжина ключа симетричної криптосистеми, біти	Довжина ключа асиметричної криптосистеми, біти
56	384
64	512
80	768
112	1 792
128	2 304

Комбінований метод допускає можливість виконання процедури автентифікації, тобто перевірки дійсності переданого повідомлення. Для цього користувач А на основі функції гешування повідомлення і свого секретного ключа k_A за допомогою відомого алгоритму електронному цифровому підпису генерує свій підпис і записує її, наприклад, у кінець переданого файла.

Користувач В, прочитавши прийняте повідомлення, може переконатися в дійсності цифрового підпису абонента А. Використовуючи той же алгоритм ЕЦП і результат гешування прийнятого повідомлення, користувач В перевіряє отриманий підпис. Комбінований метод шифрування є найбільш раціональним, поєднуючи в собі високу швидкодію симетричного шифрування та високу криптостійкість, яка гарантується системами з відкритим ключем.

3.3. Основні вимоги до сучасних криптосистем

Абстрактно *секретна система* визначається як деяка множина відображень одного простору (множини можливих повідомлень) в інший простір (множину можливих криптограм) [16; 21; 23; 51].

Зафіксуємо множину можливих повідомлень $M = \{M_1, M_2, \ldots M_m\}$ і множину криптограм $E = \{E_1, E_2, \ldots E_n\}$. Зафіксуємо також множину відображень $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_k\}$, де $\varphi_i : M \to E, i = 1, 2, \ldots, k$.

Якщо множини *M* та *E* рівнопотужні, тобто *n*=*m*, а відображення $\varphi_i \in \varphi$ сюр'єктивне та ін'єктивне, то існує обернене відображення $\varphi_i^{-1} : E \to M$, що кожному елементу множини *E* ставить у відповідність елемент множини *M*.

Очевидно, що φ_i та φ_i^{-1} задають взаємно однозначне відображення множин *M* та *E*.

Зафіксуємо тепер множину ключів $k = \{K_1, K_2, \ldots, K_k\}$ так, що для всіх $i = 1, 2, \ldots, k$ відображення $\varphi_i \in \varphi$ однозначно задається ключем K_i, тобто:

$$\varphi_i : M \xrightarrow{K_i} E.$$

Кожне відображення φ_i з множини відповідає способу шифрування за допомогою конкретного ключа K_i. На рис. 3.3 схематично подане відображення $\varphi_i \in \varphi$, задане ключем K_i.

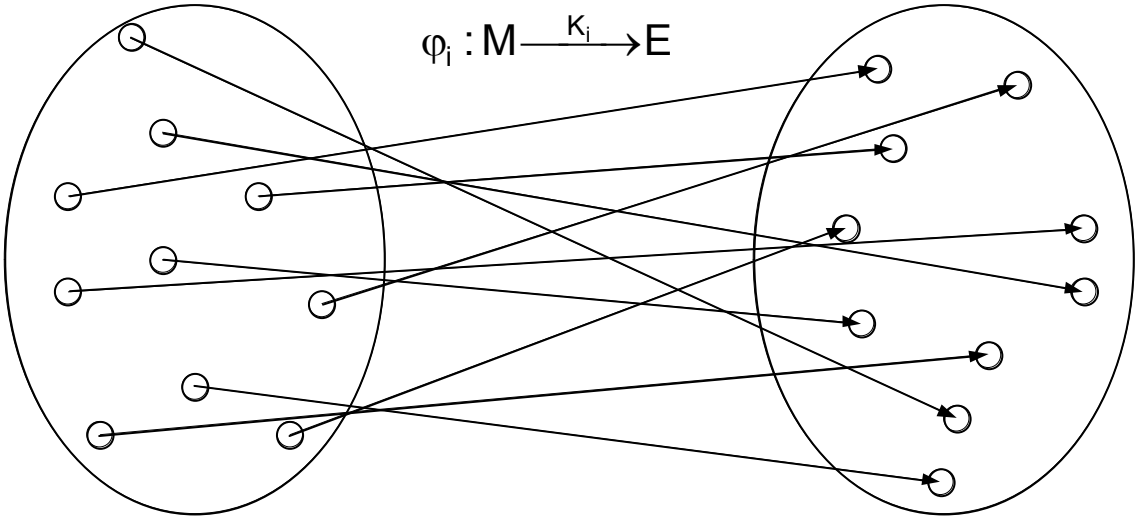


Рис. 3.3. **Відображення** $\varphi_i : M \xrightarrow{K_i} E$ **множини відкритих текстів у множину криптограм**

Зафіксуємо множину ключів $K^* = \{K_1^*, K_2^*, \ldots, K_k^*\}$ у загальному випадку $K^*_i \in K^*$. Усі елементи множини зворотних відображень $\varphi^{-1} = \{\varphi_1^{-1}, \varphi_2^*, \ldots, \varphi_k^*\}$ задаються відповідним ключем: $\varphi_i^{-1} : E \xrightarrow{K_i^*} M$.

Кожне конкретне відображення φ_i^{-1} з множини φ^{-1} відповідає способу розшифрування за допомогою ключа K_i^*.

Якщо відомо ключ K_i^*, то в результаті розшифрування можлива єдина відповідь – елемент множини M.

Таким чином, в абстрактне *визначення криптосистеми* входять такі множини: $M, E, \varphi, \varphi^{-1}, K, K^*$ (множини відкритих текстів і криптограм, множини прямих і обернених відображень, множини ключів).

Якщо при цьому $K \neq K^*$, то система *асиметрична*. Навпаки, якщо $K = K^*$ – *симетрична*. Відповідно до принципу Кірхгофа стійкість секретної системи повинна базуватися тільки на збереженні в таємниці від противника ключа, а не на секретності алгоритму шифрування.

На рис. 3.4 подано структурну схему секретної системи. Джерело повідомлень породжує потік повідомлень із множини M. Кожне повідомлення зображується конкретною реалізацією деякого випадкового процесу, що описує роботу джерела повідомлень. Кожному повідомленню

$M_i \in M = \{M_1, M_2, \ldots, M_m\}$ відповідає ймовірність $P(M_i)$. Розподіл ймовірностей випадкового процесу задається сукупним розподілом ймовірностей випадкових величин:

$$P_M = \{P(M_1), P(M_2), \ldots, P(M_m)\}.$$

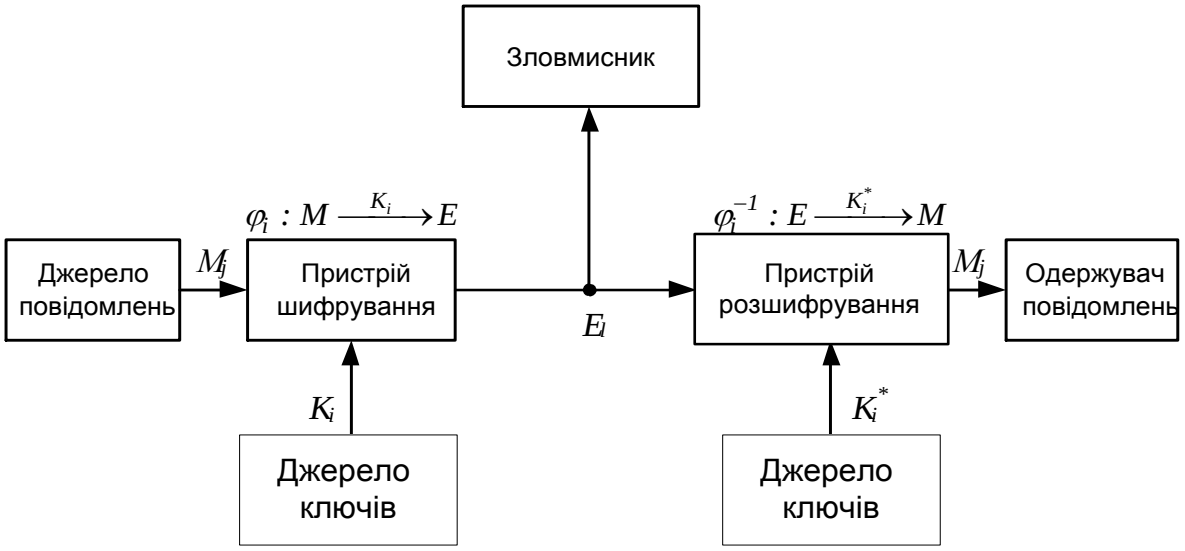


Рис. 3.4. **Структурна схема секретної системи**

Джерело ключів породжує потік ключів з множини *K* і/або *K**. Кожному ключу $K_i \in K = \{K_1, K_2, \ldots, K_k\}$ відповідає деяка ймовірність $P(K_i)$, а кожному $K_i^* \in K^* = \{K_1^*, K_2^*, \ldots, K_k^*\}$ – ймовірність $P(K_i)$. Випадковий процес генерування ключів задається множинами:

$$P_K = \{P(K_1), P(K_2), \ldots, P(K_k)\},$$
$$P_{K^*} = \{P(K_1^*), P(K_2^*), \ldots, P(K_k^*)\}.$$

Ці множини утворюють апріорні знання зловмисника про джерело повідомлень і ключів відповідно. Фактично ці множини характеризують апріорні знання супротивника щодо можливої "слабкості" секретної системи.

Вибір ключа K_i визначає конкретне відображення φ_i з множини відображень φ. Криптограма тоді буде формуватися таким чином:

$$E_l = \varphi_i(K_i, M_i).$$

Криптограма E_i передається на приймальну сторону деяким каналом й може бути перехоплена супротивником. На приймальному кінці за допомогою оберненого відображення φ_i^{-1} (заданого ключем K_i^*) із криптограми E_i відновлюється відкрите повідомлення:

$$M_i = \varphi_i^{-1}(K_i^*, E_i).$$

Якщо супротивник перехопить криптограму E_l, він може з її допомогою спробувати обчислити апостеріорні ймовірності різних можливих повідомлень:

$$P_{M|E_l} = \{P(M_1|E_l), P(M_2|E_l), ..., P(M_m|E_l)\}$$

і різних можливих ключів:

$$P_{K|E_l} = \{P(K_1|E_l), P(K_2|E_l), ..., P(K_k|E_l)\},$$

які могли бути використані при формуванні криптограми E_l.

Множини апостеріорних ймовірностей утворюють апостеріорні знання противника про ключі $K = \{K_1, K_2, \ldots, K_k\}$ й повідомлення $M = \{M_1, M_2, \ldots, M_m\}$ після перехоплення криптограми E_1. Фактично, множини $P_{K|E_l}$ та $P_{M|E_l}$ становлять множини припущень, яким приписані відповідні ймовірності. Основними показниками ефективності секретних систем прийнято вважати такі системи, наведені в роботах [13; 21 – 23; 25; 27; 29; 32; 61]:

1. Криптографічна стійкість (кількість таємності), яку оцінюють як складність розв'язку задачі дешифрування перехопленого повідомлення (без знання ключа) найкращим відомим методом. Деякі криптосистеми спроектовано з таким розрахунком, що знання супротивника про її елементи не збільшуються в результаті перехоплення будь-якої кількості зашифрованих повідомлень. Інші системи, хоча й дають супротивникові деяку інформацію при перехопленні чергової криптограми, але не допускають єдиного розв'язку, тобто не дозволяють отримати однозначне розшифрування. Системи, що допускають єдиний розв'язок, дуже різноманітні як за витратою часу й сил, необхідних для одержання цього розв'язку, так і за кількістю матеріалу, який необхідно перехопити для його одержання.

2. Обсяг ключових даних. Симетрична система використовує загальний ключ для користувачів на передавальному та приймальному кінцях. Відповідно, цей ключ потрібно передати захищеним каналом зв'язку.

Отже, він не повинен бути занадто великим, щоб його можна було легко передати, і занадто малим, щоби його не можна було легко добути повним перебиранням множини ключів.

У випадку асиметричної криптосистеми один з ключів роблять загальнодоступним, отже, він передається відкритими каналами зв'язку.

3. Складність виконання прямого й зворотного криптографічного перетворення (шифрування і розшифрування повідомлень). Ці операції повинні бути по можливості простими і такими, що легко реалізуються на практиці.

4. Розростання кількості помилок. У деяких типах шифрів помилка в одній літері при шифруванні або передаванні призводить до великої кількості помилок при розшифруванні тексту, що може призвести до значної втрати інформації та навіть до повторного передавання інформації.

5. Збільшення обсягу повідомлення. У деяких типах секретних систем обсяг повідомлення збільшується в результаті операції шифрування. Часто такі шифри називають подрібнювальними. Цей небажаний ефект потрібно мінімізувати.

Однак, незалежно від того, який алгоритм використовує криптосистема, симетричний чи асиметричний, вона повинна задовольняти такі вимоги: зашифрований текст можна прочитати лише за умови знання ключа розшифрування; трудомісткість обчислення криптографічного ключа шифрування за фрагментом криптограми та відповідним йому фрагментом відкритого тексту повинна бути занадто великою для її практичної реалізації; кількість операцій, необхідних для розшифрування інформації підбиранням ключа, повинна мати чітку нижню оцінку та виходити за межі можливостей сучасних комп'ютерних систем, а також мати достатній запас з урахуванням прогресу обчислювальної техніки; знання зловмисником алгоритму шифрування не повинно впливати на надійність системи захисту; незначна зміна ключа або відкритого тексту повинна призводити до суттєвої зміни зашифрованого тексту; в процесі шифрування необхідно забезпечувати постійний контроль за ключем шифрування та даними, що зашифровуються; довжина зашифрованого тексту не повинна бути набагато більшою довжини відкритого тексту; не повинно бути простих або таких, що легко встановити, залежностей між ключами, що послідовно використовуються в процесі шифрування; будь-який ключ з множини можливих повинен забезпечувати однаково

надійний захист інформації; додаткові біти, що додаються до криптограми в процесі зашифрування, повинні бути повністю та надійно приховані в зашифрованому тексті; криптосистема повинна забезпечувати гарантовану швидкість шифрування за довільних параметрів відкритого тексту та ключів шифрування.

Криптосистеми можуть реалізовуватися як програмно, так і апаратно. Апаратна реалізація, без сумніву, має значно більшу вартість, однак і значні переваги, зокрема високу продуктивність, простоту використання, захищеність тощо. Програмна реалізація практичніше, гнучкіше у використанні, простіше модифікується.

3.4. Сітка Х. Фейстеля, її переваги та недоліки

Блоковий алгоритм перетворює *n*-бітний блок незашифрованого тексту в *n*-бітний блок зашифрованого тексту. Кількість блоків довжини *n* рівна 2^n. Для того, щоб перетворення було зворотним, кожний з таких блоків повинен перетворюватися у свій унікальний блок зашифрованого тексту. При маленькій довжині блоку таке перетворення погано приховує статистичні особливості незашифрованого тексту. Прийнято вважати, що при довжині блоку в 64 біти, статистичні особливості вхідного тексту приховуються вже досить добре, але в цьому випадку не можна використати довільне перетворення.

З цієї точки зору широке розповсюдження набув алгоритм, який отримав назву сітки Х. Фейстеля, оскільки, з одного боку, він задовольняє усі вимоги, що ставляться до симетричних алгоритмів, а з другого – досить простий та компактний.

Сітка Фейстеля має таку структуру. Вхідний блок поділяється на підблоки однакової довжини, які називаються множинами. Наприклад, у випадку 64-бітного блоку використовуються дві множини по 32 біти кожна. Множини обробляються незалежно одна від одної, після чого виконується циклічний зсув множин вліво. Таке перетворення виконується кілька циклів або раундів. У випадку двох множин кожний сітка Фейстеля має структуру, подану на рис. 3.5. Функція *F* називається утворюючою або раундовою функцією. Кожний раунд складається з обчислення функції *F* для однієї множини і побітового виконання операції XOR результату *F* з іншою множиною. Після цього множини міняються місцями. Вважається, що оптимальна кількість раундів – від 8 до 32.

Важливо те, що збільшення кількості раундів значно збільшує криптостійкість алгоритму. Можливо, ця особливість і вплинула на таке поширення сітки Фейстеля, тому що для більшої криптостійкості досить просто збільшити кількість раундів, не змінюючи сам алгоритм. Останнім часом кількість раундів не фіксується, а лише вказуються допустимі межі.

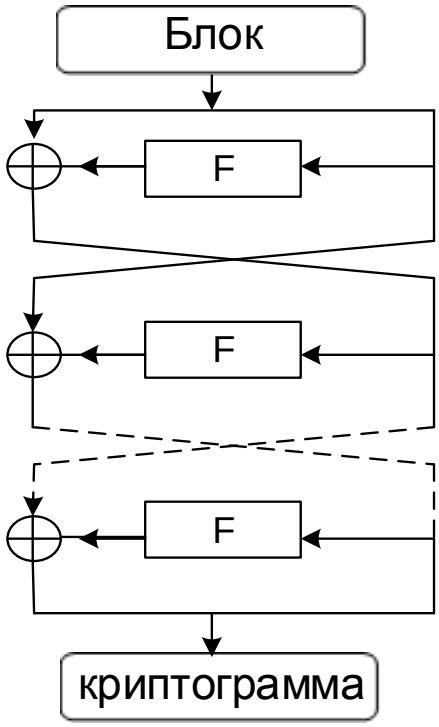


Рис. 3.5. **Структура сітки Х. Фейстеля**

Сітка Фейстеля є зворотною навіть у тому випадку, якщо функція *F* незворотна, оскільки для розшифрування не потрібно обчислювати F^{-1}. Для розшифрування використовується той же алгоритм, але на вхід подається зашифрований текст, а ключі використовуються в оберненому порядку.

Сьогодні усе частіше використовуються різні модифікації сітки Фейстеля для 128-бітного блоку із чотирма множинами. Збільшення кількості множин, а не розмірності кожної множини, пов’язане з тим, що найбільш популярними дотепер залишаються процесори з 32-розрядними словами, а оперувати ними ефективніше, ніж з 64-розрядними.

Основною характеристикою алгоритму, побудованого на основі сітки Фейстеля, є функція *F*. Існують також різні варіанти початкового і кінцевого перетворень. Подібні перетворення, називані забілюванням (англ. – whitening), здійснюються для того, щоб виконати початкове перемішування вхідного тексту.

Структура, розроблена Х. Фейстелем, має низку переваг, а саме:

процедури шифрування та розшифрування однакові, лише ключова інформація при розшифруванні подається в оберненому порядку;

для розшифрування можна використовувати ті ж апаратні або програмні блоки, що й для шифрування, що значно зменшує вартість реалізації та робить її значно зручнішою у використанні.

Недоліком мережі Фейстеля вважають те, що в кожному циклі змінюється лише половина блоку вхідного тексту. Це призводить до збільшення кількості циклів для досягнення бажаної стійкості. Стосовно вибору *F*-функції, то чітких рекомендацій немає, проте найчастіше ця функція, яка повністю залежить від ключа, складається з нелінійних замін, перестановок і зсувів.

3.5. Класифікація сучасних криптосистем та основні вимоги до них

Позначимо процес зашифрування відкритого повідомлення *M* на ключі *K* через $E_K(M)$ (від англійського *Encryption* – зашифрування). Тоді отримання шифрограми *C* можна зобразити як:

$$C = E_K(M).$$

У такому випадку процес розшифрування (*Decryption* – розшифрування), тобто отримання відкритого повідомлення *M* з криптограми *C* за допомогою відомого ключа *K* можна позначити:

$$M = D_K(C).$$

Функція D_K повинна бути оберненою до E_K, тобто $D_K = E^{-1}_K$ в тому розумінні, щоб при правильному ключі *K* дозволяла отримати відкритий текст *M*.

Отже, процес інформаційного обміну схематично можна зобразити таким чином:

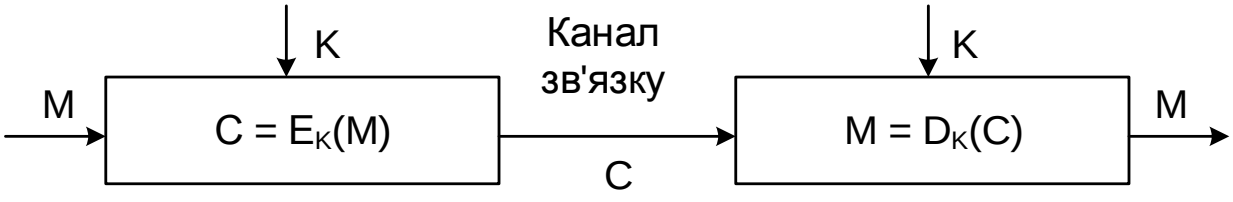


Рис. 3.6. **Схема роботи симетричної криптосистеми**

На передавальній стороні виконують шифрування відкритого повідомлення *M* за допомогою функції шифрування $E_K(M)$ на ключі *K* та отримують криптограму *C*, яку передають відкритим каналом зв'язку. На приймальній стороні, отримавши зашифроване повідомлення *C*, застосовують до нього обернене перетворення $D_K(C)$ і отримують відкрите повідомлення *M* при використанні такого самого ключа *K*, як і на передавальній стороні та якщо зашифроване повідомлення не було змінено під час проходження каналом зв'язку.

Як бачимо, для успішної роботи такої криптосистеми учасники інформаційного обміну повинні завчасно домовитися про алгоритми шифрування та розшифрування, а також про те, яким чином приймальна сторона дізнається про криптографічний ключ. Оскільки ключ, що використовується для зашифрування та розшифрування повідомлень однаковий, такі криптосистеми отримали назву ***симетричних***. Симетричні криптосистеми здебільшого використовують перетворення тексту, які є комбінацією ***перестановок*** і ***замін***. Симетричні криптосистеми використовуються вже багато років і вважаються найбільш дослідженими.

Основними *перевагами* найбільш популярних симетричних криптосистем вважають:

високу швидкість роботи (отже, можливість швидкого шифрування великих потоків інформації в каналах зв'язку);

забезпечення високого ступеня секретності, оскільки симетричні криптосистеми давно відомі та добре досліджені, ймовірність знайти нові вразливості в цих системах незначна;

можливість використання однакових апаратних засобів для зашифрування та розшифрування інформації.

Однак застосування симетричних криптосистем на практиці потребує подолання двох значних *недоліків*:

проблеми розповсюдження та зберігання криптографічних ключів, оскільки він є секретною частиною криптосистеми як на передавальній, так і на приймальній сторонах; компрометація ключа призводить до загрози для усієї системи;

великої кількості криптографічних ключів, необхідної для створення шифрованого інформаційного обміну між багатьма учасниками. Для того, щоби *n* осіб могли обмінюватися секретною інформацією, необхідно згенерувати та розповсюдити $n(n-1)/2$ криптографічних ключів.

Одним з найефективніших способів подолання недоліків симетричних криптосистем стало винайдення асиметричних способів шифрування, коли для зашифрування і розшифрування використовуються різні криптографічні ключі. Крипостійкість таких систем ґрунтується на так званих "*односторонніх функціях*", які легко обчислюються в прямому напрямі та утворюють математичну проблему надзвичайної обчислювальної складності при спробі розв'язку оберненої задачі.

Як правило, для зашифрування інформації використовують так званий ***відкритий*** або ***публічний ключ*** (хоча в деяких випадках його можна використати і для розшифрування), який непотрібно приховувати. Навпаки, цей ключ розміщують на ресурсі, доступному для всіх учасників інформаційного обміну та захищають від підміни. Кожен, хто хоче написати конфіденційного листа власникові публічного ключа, використає його для зашифрування, а власник ключа, маючи парний до цього публічного, ***приватний*** (або ***секретний***) ключ, зможе розшифрувати повідомлення. Особливістю такої криптосистеми є те, що за допомогою публічного ключа неможливо розшифрувати зашифроване на ньому повідомлення. Таким чином, ніхто, окрім власника приватного ключа, не зможе прочитати призначене йому повідомлення. Приватний ключ непотрібно розповсюджувати, він зберігається після генерування у його власника і використовується лише для розшифрування (хоча можливий і обернений варіант використання криптосистеми залежно від задач).

Отже, процес такого інформаційного обміну схематично можна зобразити таким чином (рис. 3.7):

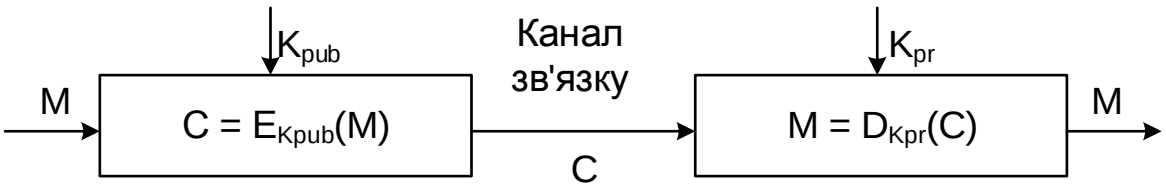


Рис. 3.7. **Схема роботи асиметричної криптосистеми**

Криптосистеми, в яких для зашифрування та розшифрування повідомлень використовуються різні криптографічні ключі, називаються ***асиметричними***.

Асиметричні криптосистеми вирішили основну проблему симетричних криптосистем, проблему розповсюдження криптографічних ключів, адже для зашифрування інформації багатьма користувачами потрібно мати лише одну пару ключів: публічний та парний йому приватний.

Однак, подолавши недоліки розповсюдження ключів, отримаємо додаткові *недоліки, що притаманні асиметричним системам*:

мала швидкість роботи асиметричних криптоалгоритмів (прийнято вважати, що ці системи приблизно в 1 000 разів повільніші за симетричні [23; 33]);

нестійкість асиметричних криптосистем проти атаки підміни публічного ключа;

досі математично строго не доведено, що не існує простого алгоритму для отримання приватного ключа з публічного.

Цей перелік потребує коментарів. Перший пункт є наслідком специфічних математичних перетворень, на базі яких ґрунтуються усі асиметричні криптосистеми та великої довжини ключів: такі задачі досить повільно виконуються на комп'ютері.

Розміщення публічного ключа на відкритому ресурсі може призвести до наступної атаки. Зловмисник генерує свою пару ключів та заміняє відкритий ключ системи своїм. Зрозуміло, що тепер він може читати усі повідомлення, які зашифровані його відкритим ключем, контролюючи канал обміну інформацією. Прочитавши повідомлення, він ***перешифровує*** їх старим відкритим ключем та відправляє адресату. Той, у свою чергу, розшифровує повідомлення своїм приватним ключем, і оскільки сеанс розшифрування проходить успішно, не помічає ***підміни публічного ключа***. Для подолання такого значного недоліку створено так звану ***інфраструктуру публічних ключів***, яка ґрунтується на системі сертифікатів, де вказано, кому належить той чи інший публічний ключ. Про структуру та принципи організації такої системи піде мова далі.

Стосовно останнього недоліку варто зазначити, що усі асиметричні криптосистеми, як вже згадувалося, ґрунтуються на односторонніх функціях. Отже, генерування пари криптографічних ключів відбувається легко, а при спробі криптоаналітика обчислити приватний ключ, знаючи публічний, він наштовхується на математичну проблему надзвичайної обчислювальної складності. Цікаво, що таке твердження фактично немає ніякого математично строгого доведення, воно ґрунтується лише на досвіді практичного використання односторонніх функцій. Не виключено, що з часом, при подальшому розвитку математичної науки, знайдеться спосіб легкого знаходження приватних ключів за парними до них публіч-

ними, що може призвести до повного краху асиметричної криптографії [21; 49].

Асиметрична криптографія, звичайно, має і значні ***переваги***. До них можна віднести:

не існує проблеми розповсюдження криптографічних ключів;

для створення багатокористувацької системи обміну зашифрованою інформацією не потрібно великої кількості ключів;

асиметрична криптографія дозволила створити зовсім нові криптографічні процедури, зокрема ***електронний цифровий підпис***, які принципово неможливі з використанням симетричної криптографії.

Підсумовуючи наведене, можна сказати, що симетричні криптосистеми, завдяки високій швидкодії та надійності, якнайкраще підходять для захисту інформації в комп'ютерних системах. Вони можуть застосовуватися як для шифрування мережевого трафіку, так і локальної інформації. Однак симетричним криптосистемам притаманний суттєвий недолік, проблема розповсюдження ключів.

З іншого боку, цього недоліку позбавлені асиметричні криптосистеми. Вони використовують для шифрування публічний ключ, який не потрібно розповсюджувати. Однак швидкодія цих систем, не в останню чергу внаслідок використання дуже довгих ключів, занадто мала (приблизно у 1 000 разів менша порівняно із симетричними), щоб їх можна було застосовувати для шифрування потоків інформації. Асиметричну криптосистему було б доцільно використати для обміну короткою інформацією, коли швидкодія системи не відіграє визначальної ролі, наприклад, криптографічних ключів. Таким чином, приходимо до ідеї комбінованого використання криптосистем: безпосереднє шифрування інформації виконують симетричною криптосистемою, а для розповсюдження криптографічних ключів використовують асиметричну. Сеанс зв'язку при цьому виглядає приблизно так. Після встановлення сеансу передавальна сторона вибирає публічний ключ асиметричної криптосистеми приймальної сторони й зашифровує ним сеансовий ключ симетричного алгоритму. Зашифрований сеансовий ключ відправляється відкритим каналом зв'язку на приймальну сторону, де розшифровується приватним ключем. Тепер можна починати шифрування каналу зв'язку за допомогою симетричної криптосистеми, оскільки ключ шифрування вже відомий

обом учасникам інформаційного обміну. Схематично роботу такої системи можна зобразити так (рис. 3.8):

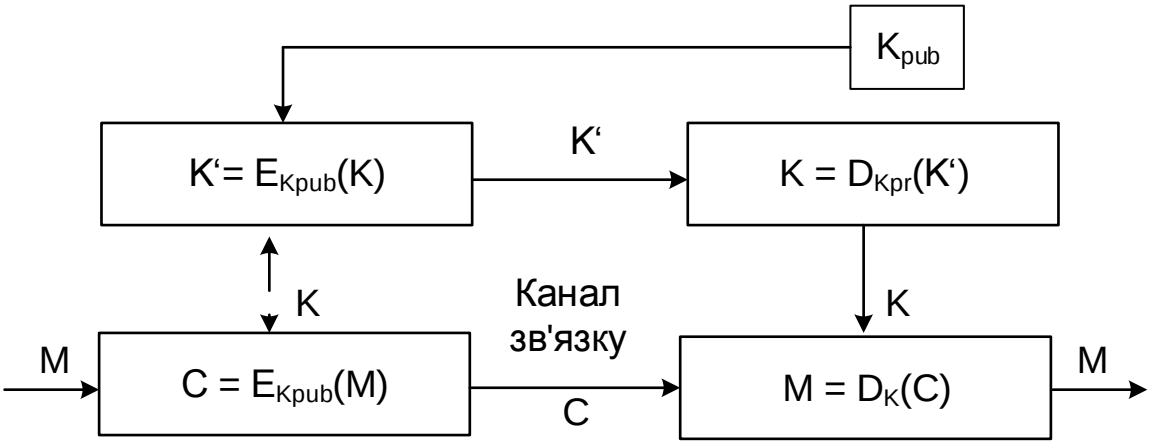


Рис. 3.8. **Схема роботи комбінованої криптосистеми**

Перевагами такої криптосистеми будуть як найвища швидкодія симетричних алгоритмів, так і зручність використання асиметричних криптосистем. Необхідно сказати, що комбіновані криптосистеми внаслідок своєї універсальності сьогодні дуже популярні. Наприклад, вони використовуються в захищених протоколах SET, SSL, TLS та інших.

3.6. Математичні основи асиметричної криптографії

Нехай n – довільне натуральне число, x та y – цілі числа. Будемо називати числа x та y конгруентними за модулем n, якщо залишки від їх ділення на число n однакові, тобто x mod n ≡ y mod n. Наприклад, 2 mod 7 ≡ 9 mod 7 або 11 mod 8 ≡ 33 mod 30, оскільки залишок від ділення у цих чисел однаковий.

Операція взяття числа за модулем має три основні властивості:

адитивності: (a+b) mod n = ((a mod n) + (b mod n)) mod n;

мультиплікативності: (a×b) mod n = ((a mod n) × (b mod n)) mod n;

збереження степеня: (a mod n)k mod n = ak mod n.

Найбільший спільний дільник (НСД) чисел A і B – це найбільше з чисел, на яке обидва цих числа діляться без залишку. Наприклад, НСД(56,98) = 14; НСД(76,190) = 38; НСД(150,19) = 1.

Взаємно прості числа – це такі числа A та B, які самі по собі, можливо, не є простими, але не мають іншого спільного дільника, на який вони діляться без залишку, окрім 1. Наприклад, числа 32 і 13 –

взаємно прості, хоча 32 саме по собі не є простим, оскільки ділиться ще на 2, 4, 8, 16.

Лишок числа A за модулем n – цілий залишок від ділення числа A на число n.

Набір лишків числа A за модулем n – це сукупність усіх різних цілих залишків від ділення числа A×i на число n, де i набуває значень від 1 до n-1. Наприклад, набір лишків числа 3 за модулем 5 буде таким: {3,1,4,2}. Зазначимо, що довжина вектора лишків максимальна і дорівнює n-1, якщо числа A та n взаємно прості.

Обчислення мультиплікативного оберненого числа.

Мультиплікативним оберненим числом до числа A за модулем n будемо називати таке число A^{-1} таке, що $(AA^{-1}) \bmod n = 1$. Зазначимо, що розв'язок такої задачі існує не завжди, а лише тоді, коли A та n – взаємно прості числа. Операція знаходження мультиплікативного оберненого дуже часто використовується у двоключовій криптографії.

Мультиплікативне обернене число можна знайти (для невеликих чисел) безпосередньо з розв'язку рівняння $(AA^{-1}) \bmod n = 1$. Це рівняння можна записати таким чином: $AA^{-1} - 1 = i \times n$, де i = 1,2,3,… – натуральне число. Тоді $AA^{-1} = 1 + i \times n$, звідки $A^{-1} = (1 + i \times n)/A$.

Наприклад, треба знайти мультиплікативне обернене числа 32 за модулем 29. Для цього отримаємо: $A^{-1} = (1+11\times29)/32 = 10$.

Таким чином, отримаємо, x = 10, тобто $(32)^{-1} \bmod 29 = 10$.

Перевірка дає $10 \times 32 \bmod 29 = 320 \bmod 29 = 1$.

Класичним алгоритмом знаходження мультиплікативного оберненого є розв'язок еквівалентного рівняння Діофанта за допомогою методики Евкліда.

Уведемо функцію Ейлера φ (n), яка визначає кількість цілих чисел, взаємно простих з n, з множини [1,n – 1]. Доведено, що для простого nφ(n)= n-1. Продемонструємо це для множини [2; 21;23]:

n	2	3	4	5	6	7	8	9	10	11
φ(n)	1	2	2	4	2	6	4	6	4	10

Як бачимо з таблички, для чисел 3, 5, 7,11 φ(n)= n – 1.

Теорема 1. Мала теорема Ферма. Якщо n – просте число, то (xn-1mod n) = 1 для будь-яких x, взаємно простих з n.

Теорема 2. Нехай число n – просте. Для будь-якого A та 1 ≤ B ≤ (n – 1) знайдеться таке 1 ≤ X ≤ (n – 1), що Ax mod n = B. Іншими словами, стверджується, що функція Ax mod n = B однозначна на проміжку 0…n – 1.

Виняток складають лише ступені числа 2 (тобто A2, A4,…), які не утворюють однозначного перетворення.

Для прикладу розглянемо деяку функцію $y = a^x \bmod n, n = 7$. У таблиці подано усі можливі значення y для a та x від 0 до 6. Тут є кілька варіантів. Для a = 1 усі значення y = 1. Для a = 2, 4 та 6 бачимо багатозначне відображення {x}→{y} (y – періодичне). У той же час для a = 3 і a = 5 існує взаємно однозначне відображення {x}→{y}. Це означає, що такі числа, 3 та 5, можна використати для практичних потреб. Такі числа називаються первісними коренями за модулем n.

a\x	0	1	2	3	4	5	6
0							
1		1	1	1	1	1	1
2		2	4	1	2	4	1
3		3	2	6	4	5	1
4		4	2	1	4	2	1
5		5	4	6	2	3	1
6		6	1	6	1	6	1

Зазначимо, що 3 та 5 взаємно прості з 7. Крім того, 5 – взаємно просте число з 6, тобто з n-1. Ця табличка також ілюструє малу теорему Ферма: a6 mod 7 = 1 (тобто an-1 mod n = 1).

Зрозуміло, що у випадку таких малих чисел, які подано в таблиці, розв'язати обернену задачу знаходження числа x, якщо відомо y та n дуже просто. Однак для великих цілих чисел розв'язок оберненої задачі, яка називається задачею дискретного логарифмування у кінцевому полі, стає обчислювально складним. Така функція називається односторонньою. Аналогічна одностороння функція використовується у криптосистемі Ель-Гамаля.

Іншим представником односторонніх функцій є розкладання великого цілого числа на прості множники. Обчислити добуток двох простих чисел дуже просто. Однак для розв'язку оберненої задачі, розкладання заданого числа на прості множники, ефективного алгоритму сьогодні не існує. Така одностороння функція використовується у криптосистемі RSA.

В усіх двоключових криптосистемах використовують так звані односторонні функції з пасткою. Це означає, що публічний ключ криптосистеми визначає конкретну реалізацію функції, а приватний ключ дає певну інформацію про пастку. Будь-хто, хто знає приватний ключ, може легко обчислювати функцію в обох напрямах, але той, хто не має такої інформації, може обчислювати функцію лише в одному напрямі. Прямий напрям використовується для зашифрування інформації та верифікації ового підпису. Обернений же напрям застосовують для розшифрування та створення електронного цифрового підпису.

В усіх криптосистемах з відкритим ключем чим більша довжина ключів, тим більша різниця у зусиллях, необхідних для обчислення функції у прямому та оберненому напрямах (для тих, хто не має інформації про пастку). Усі практичні криптосистеми з відкритим ключем використовують функції, які вважаються односторонніми, однак цю властивість строго математично не доведено для жодної з них. Це означає, що теоретично можливо створення алгоритму, що дозволить легко обчислити обернену функцію без знання інформації про пастку. Однак так само імовірним може бути теоретичне доведення неможливості такого обчислення. В першому випадку це призведе до повного краху відповідної криптосистеми, а в другому – значно зміцнить її позиції.

Контрольні запитання

1. Основні показники ефективності секретних систем.
2. Класифікація сучасних криптосистем та основні вимоги до них.
3. Класифікація симетричних криптосистем та основні вимоги щодо їх безпеки.
4. Класифікація асиметричних криптосистем та основні вимоги щодо їх безпеки.
5. Основні поняття роботи К. Шеннона "Теорія зв’язку в секретних системах".
6. Комбіновані криптосистеми. Їх переваги та недоліки.
7. Сітка Х. Фейстеля, її переваги та недоліки.
8. Основні математичні операції щодо побудови криптосистем.
9. Математична модель секретної системи. Основні вимоги щодо забезпечення криптостійкості інформаційних систем.
10. Основні вимоги щодо криптостійкості секретного (особистого) ключа у симетричних та асиметричних криптосистемах.

Розділ 4. Алгоритми з секретним ключем

4.1. DES (Data Encryption Standard) – стандарт шифрування даних США 1977 року

4.1.1. Принципи розробки

Найпоширенішим і найбільш відомим алгоритмом симетричного шифрування є DES (Data Encryption Standard – Стандарт Шифрування Даних). Алгоритм був розроблений у 1977 році, в 1980 році був прийнятий NIST (National Institute of Standards and Technolody США) у якості стандарту (FIPS PUB 46).

DES є класичною сіткою Фейстеля з двома множинами (рис. 4.1).

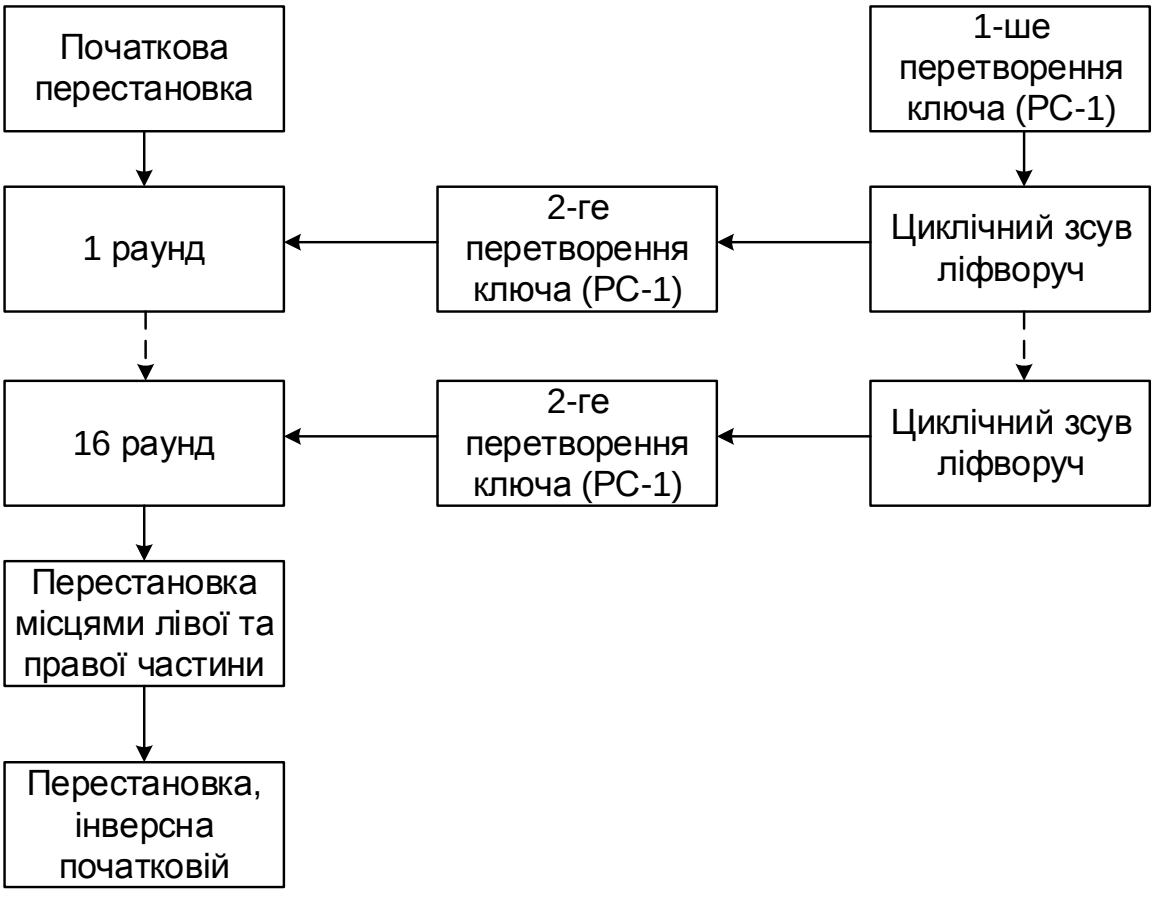


Рис. 4.1. **Загальна схема DES**

Дані шифруються 64-бітними блоками з використанням 56-бітного ключа. До секретних 56 бітів додається 8 бітів парності, тобто загальна довжина ключа дорівнює 64 біти.

Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти перемішуються відповідно до стандартної таблиці. Наступний етап складається з 16 раундів однієї й тієї ж функції, яка використовує операції зсуву і підстановки. На третьому етапі ліва і права половини виходу останньої (16-ї) ітерації міняються місцями. Нарешті на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} обернена до початкової перестановки IP.

На рис. 4.2 показаний спосіб використання 56-бітного ключа.

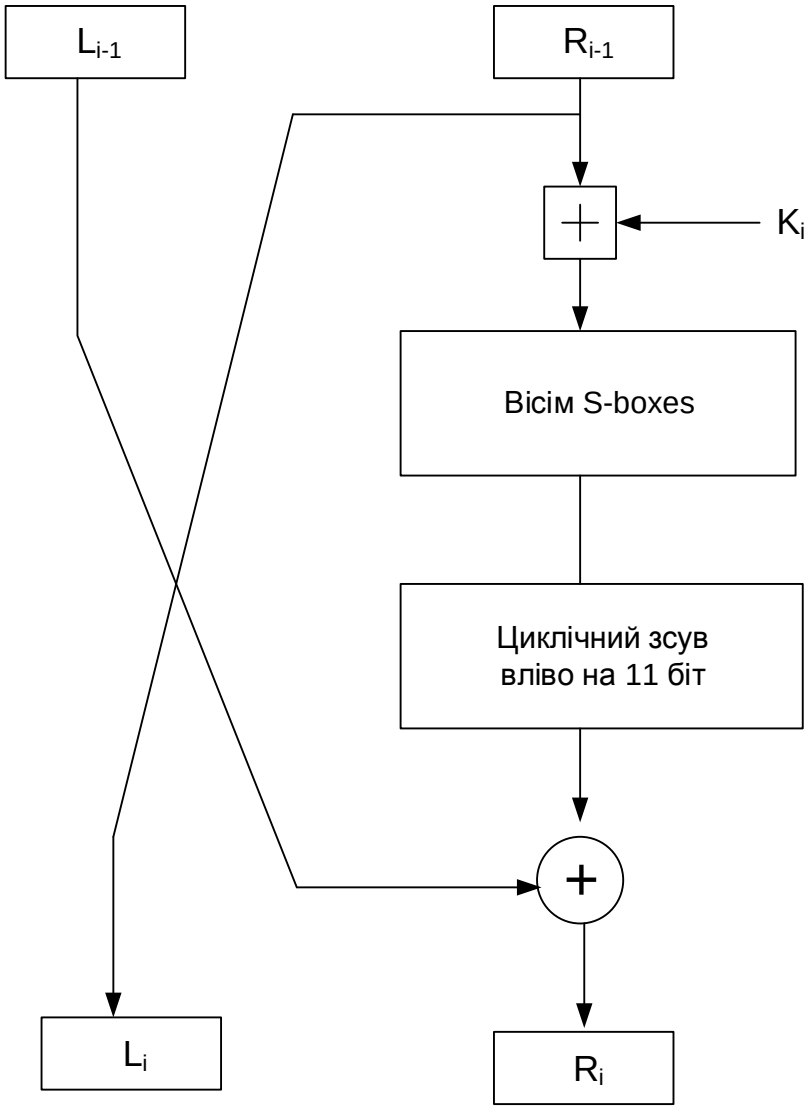


Рис. 4.2. **Один раунд DES**

Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i утворюється як комбінація лівого

циклічного зсуву і перестановки. Функція перестановки однакова для кожного раунду, але підключі K_i для кожного раунду отримуються різними внаслідок зсуву бітів ключа.

4.1.2. Шифрування. Початкова перестановка

Початкова перестановка та її інверсія визначаються стандартною таблицею. Якщо *M* – це довільні 64 біти, то $X = IP(M)$ – переставлені 64 біти. Якщо застосувати обернену функцію перестановки $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, то вийде початкова послідовність бітів. Стандартні таблиці IP та IP-1 подано в табл. 4.1 та 4.2.

Таблиця 4.1

Початкова IP-перестановка

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

Таблиця 4.2

Кінцева перестановка IP^{-1}

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

Табл. 4.1 та 4.2 читаються таким чином: перший біт переставляється на 58-ме місце, другий – на 50-те і т. д. Обернена перестановка: перший біт переставляється на 40-ве місце, другий – на 8-ме і т. д.

4.1.3. Послідовність перетворень окремого раунду

Тепер розглянемо послідовність перетворень, яка використовується в кожному раунді. 64-бітний вхідний блок проходить через 16 раундів обробки, при цьому на кожній ітерації виходить проміжне 64-бітне значення. Ліва і права частини кожного проміжного значення трактуються як окремі 32-бітні значення, позначені L і R. Кожну ітерацію можна описати в такий спосіб:

$$L_i = R_{i-1.}$$

$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$, де $\oplus$ позначає операцію XOR.

Таким чином, вихід лівої половини L_i дорівнює входу правої половини R_{i-1}. Вихід правої половини R_i є результатом застосування операції XOR до L_{i-1} і функції F, що залежить від R_{i-1} і K_i.

Розглянемо функцію F докладніше.

Блок R_i, який подається на вхід функції F, має довжину 32 біти. Спочатку R_i розширюється до 48 бітів, використовуючи таблицю, яка визначає перестановку і розширення на 16 бітів (табл. 4.3). Розширення відбувається в такий спосіб: 32 біти розбиваються на групи по 4 біти і потім розширюються до 6 бітів, приєднуючи крайні біти із двох сусідніх груп.

Таблиця 4.3

Перестановка з розширенням

31	0	1	2	3	4
3	4	5	6	7	8
7	8	4	10	11	12
11	12	13	14	15	16
15	16	17	18	19	20
19	20	21	22	23	24
23	24	25	26	27	28
27	28	29	30	31	0

Читати таблицю треба таким чином. На вхід перестановки подаються 32 біти тексту (біти перенумеровано від 0 до 31 – центральна частина табл. 4.3). До цих бітів додаються додаткові біти, як це показано в табл. 4.3 у клітинках, виділених жирними границями. Отримуємо масив довжиною в 48 бітів.

У тексті розширення виглядає таким чином. Якщо частина вхідного повідомлення

. . . efgh ijkl mnop . . .,

то в результаті розширення виходить повідомлення

. . . defghi hijklm lmnopq . . .

До отриманого в такий спосіб масиву бітів додається за правилами XOR 48-бітний раундовий ключ K_i. Результат подається на вхід блоку заміни, який складається з восьми S-боксів, тобто таблиць 4x16, в яких певним чином розміщено десяткові числа від нуля до п'ятнадцяти (у двійковому представленні). Розміщення чисел було оптимізовано Агентством Національної Безпеки США при розробці стандарту для більшої стійкості алгоритму до диференціального і лінійного крипто-аналізу.

Підстановка виконується у такий спосіб. Масив у 48 бітів розбивається на вісім частин по шість бітів кожна. Кожну частину подають на "свій" S-бокс, номер якого визначається її номером. Перший і останній біт 6-бітової частини визначає номер рядка S-бокса у двійковому представленні, а чотири середні біти – номер стовпчика. На перетині рядка та стовпчика читаємо 4-бітове число. Воно і буде результатом заміни.

Розглянемо приклад. Припустимо, що перша 6-бітова частина 48-бітового вхідного блоку має значення 110110. Оскільки вона перша, то заміна буде виконуватися на першому S-боксі. Він має вигляд, наведений у табл. 4.4.

Таблиця 4.4

Перший S-бокс DES

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

Перша "1" та останній "0" вхідної частини разом (10 дають двійку в десятковому представленні) вказують, що для заміни буде використовуватися рядок № 2 (він затемнений у табл. 4.4). Середні чотири біти (1011) дають в десятковому представленні число 11. Отже, для заміни буде використано стовпчик № 11 (він також затемнений у табл. 4.4).

На перетині рядка № 2 та стовпчика № 11 знаходиться комірка з числом 7. Воно, точніше його двійкове представлення – 0111, і буде результатом застосування S-боксу. Таким чином, замість 6-бітного числа 110110 отримаємо 0111. Аналогічним чином виконуються й заміни інших 6-бітних частин вхідного 48-бітного числа.

Далі отримане 32-бітне значення обробляється за допомогою перестановки *P*, метою якої є максимальне перемішування бітів, щоб у наступному раунді шифрування з великою ймовірністю кожний біт оброблявся іншим S-боксом (табл. 4.5).

Таблиця 4.5

P-перестановка алгоритму DES

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

4.1.4. Операція розгортання ключа

Раундовий ключ створюється за таким алгоритмом.

Крок 1. Із загального ключа шифрування вилучається кожен восьмий біт (під номерами: 8, 16, 24, 32, 40, 48, 56, 64 – біти парності). Довжина ключа таким чином зменшується до 56 бітів.

Крок 2. Біти ключа розділяються на два блоки C_0 і D_0 відповідно до стандартної таблиці PC-1 (Permuted Choice-1), яку подано у табл. 4.6.

Таблиця 4.6

Таблиця перемішування бітів ключа PC-1

Блок C_0							Блок D_0						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

Крок 3. На кожному *i*-му раунді C_i та D_i циклічно зсуваються вліво на 1 або 2 позиції, залежно від номера раунду (табл. 4.7).

Таблиця 4.7

Параметри раундового зсуву регістрів *C* і *D*

Номер циклу	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Зсув вліво (шифрування)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Зсув вправо (розшифрування)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Крок 4. Після зсуву підблоки C_i і D_i об'єднуються та з них за допомогою функції PC-2 (Permuted Choice-2) вибирається 48 бітів раундового підключа K_i. Таблицю PC-2 подано у табл. 4.8.

Таблиця 4.8

Таблиця PC-2 для отримання раундового ключа DES

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	32

Вибір бітів виконується таким чином. Підблоки розглядаються як послідовність рядків табл. 4.6, записаних один за одним, починаючи з першого. Біти отриманого таким чином блоку даних перенумеровуються зліва направо, починаючи з одиниці. Кожен елемент S таблиці розглядається як номер біта b_S в отриманому блоці даних. Перетворенням є заміна усіх $S \rightarrow b_S$.

Таким чином, блок-схема формування раундових ключів DES має такий вигляд, як це показано на рис. 4.3.

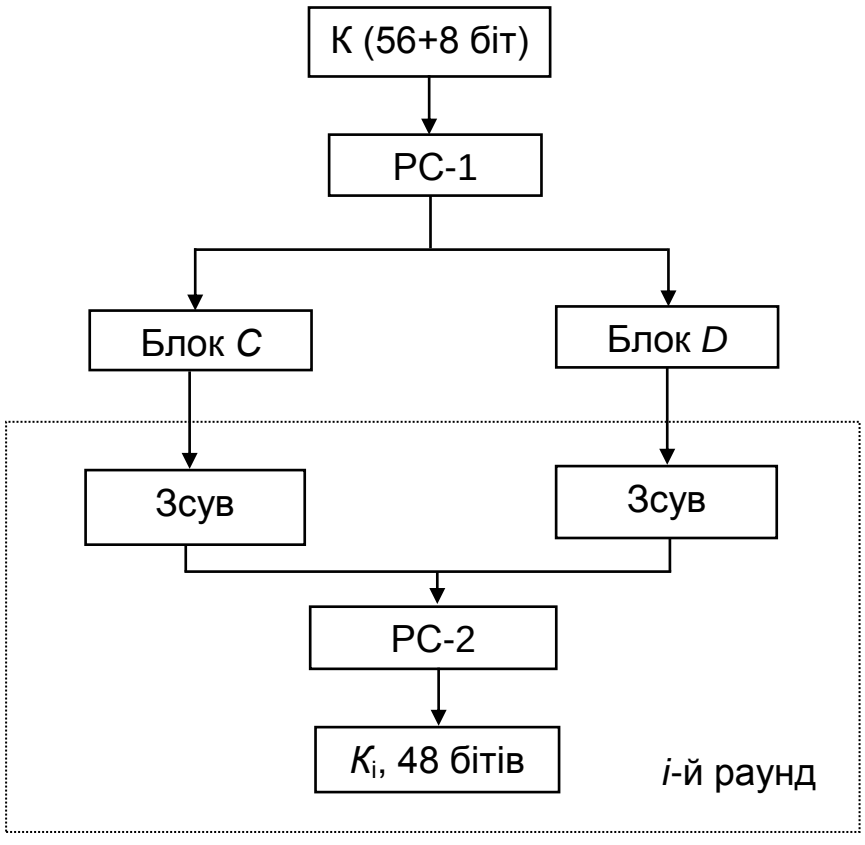


Рис. 4.3. **Блок-схема формування раундових ключів DES**

4.1.5. Операція розшифрування

Процес розшифрування аналогічний процесу шифрування. На вхід алгоритму подається зашифрований текст, але ключі K_i використовуються в оберненій послідовності: K_{16} використовується на першому раунді, K_1 – на останньому раунді. Нехай виходом *i*-го раунду шифрування буде $L_i||R_i$. Тоді відповідний вхід 16-ого раунду розшифрування буде $R_i||L_i$.

Після останнього раунду процесу розшифрування дві половини виходу міняються місцями так, щоб вхід заключної перестановки IP^{-1} був $R_{16}||L_{16}$. Виходом цієї стадії є незашифрований текст.

Перевіримо коректність процесу розшифрування. Використаємо зашифрований текст і ключ як вхідні параметри алгоритму. На першому кроці виконаємо початкову перестановку *IP* і одержимо 64-бітне значення $Ld_0||Rd_0$. Відомо, що *IP* і IP^{-1} протилежні. Отже,

$Ld_0||Rd_0 = IP$ (зашифрований текст).

Зашифрований текст = $IP^{-1}(R_{16}||L_{16})$

$$Ld_0||Rd_0 = IP(IP^{-1}(R_{16}||L_{16})) = R_{16}||L_{16}.$$

Таким чином, вхід першого раунду процесу розшифрування еквівалентний 32-бітному виходу 16-ого раунду процесу шифрування, у якого ліва і права частини записані у зворотному порядку.

Тепер необхідно показати, що вихід першого раунду процесу розшифрування еквівалентний 32-бітному входу 16-ого раунду процесу шифрування. По-перше, розглянемо процес шифрування:

$$L_{16} = R_{15}$$
$$R_{16} = L_{15} \oplus F(R_{15}, K_{16}).$$

При розшифруванні:

$$Ld_1 = Rd_0 = L_{16} = R_{15}$$
$$Rd_1 = Ld_0 \oplus F(Rd_0, K_{16}) = R_{16} \oplus F(Rd_0, K_{16}) = (L_{15} \oplus F(R_{15}, K_{16})) \oplus F(R_{15}, K_{16}).$$

XOR має наступні властивості:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$
$$D \oplus D = 0$$
$$E \oplus 0 = E.$$

Таким чином, маємо $Ld_1 = R_{15}$ і $Rd_1 = L_{15}$. Отже, вихід першого раунду процесу розшифрування є $L_{15}||R_{15}$, який є перестановкою входу

16-го раунду шифрування. Легко показати, що ця відповідність виконується всі 16 раундів. Можна описати цей процес у загальних термінах. Для *i*-ого раунду шифрувального алгоритму справедливо:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i).$$

Ці рівності можна записати по-іншому:

$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i) = R_i \oplus F(L_i, K_i).$$

Таким чином, описані входи *i*-ого раунду як функція виходів.

Вихід останньої стадії процесу розшифрування є $R_0||L_0$. Щоб входом IP^{-1} стадії було $R_0||L_0$, необхідно поміняти місцями ліву і праву частини.

Але $IP^{-1}(R_0||L_0) = IP^{-1}(IP$ (незашифрований текст)) = незашифрований текст. Отже, одержуємо незашифрований текст, що і демонструє можливість розшифрування DES.

4.1.6. Переваги та недоліки DES

Через те, що довжина ключа рівна 56 бітам, існує 2^{56} можливих ключів. На сьогодні така довжина ключа недостатня, оскільки допускає успішне застосування атак повного перебирання. Альтернативою DES можна вважати потрійний DES, IDEA, а також алгоритм Rijndael, прийнятий у якості нового стандарту США на алгоритми симетричного шифрування [16; 22; 23; 33; 61].

Без відповіді поки залишається питання, чи можливий криптоаналіз алгоритму DES із використанням існуючих характеристик. Основою алгоритму є вісім таблиць підстановки, або S-бокси, які застосовуються в кожній ітерації. Існує небезпека, що ці S-бокси конструювалися таким чином, що криптоаналіз можливий для супротивника, який знає їх слабкі місця. Протягом багатьох років обговорювалася як стандартна, так і несподівана поведінка S-боксів, але все-таки нікому не вдалося виявити їх фатально слабкі місця. Більше того, коли Елі Біхам та Аді Шамір у 1990 році опублікували результати своїх досліджень з диференціального криптоаналізу, з'ясувалося, що блоки заміни DES значно стійкіші до цього типу криптоаналізу, ніж можна було очікувати при випадковому виборі їх структури [33]. Це дозволяє стверджувати, що Агентству

Національної Безпеки США цей тип атак на симетричні криптоалгоритми був відомий ще у 70-х роках XX сторіччя.

Сформулюємо **переваги та недоліки DES**. Перевагами цієї криптосистеми вважаються:

висока швидкодія як в апаратній, так і в програмній реалізації;

можливість використання одних і тих самих апаратних або програмних блоків як для шифрування, так і для розшифрування інформації.

Основними **недоліками DES** на сьогодні вважають:

невелику довжину ключа, усього 56 бітів. При сучасному рівні розвитку комп'ютерних засобів така довжина ключа не може забезпечувати потрібний рівень захисту для деяких типів інформації;

наявність "слабких" ключів, викликана тим, що для генерування ключової послідовності виконується два незалежних регістри зсуву. Прикладом слабкого ключа може служити 1F1F1F1F0E0E0E0E. При цьому результатом генерування будуть ключові послідовності, однакові з вихідним ключем, в усіх 16 раундах. Існують також різновиди слабких ключів, що дають усього чотири ключові послідовності та "зв'язані" ключі, які отримуються один з одного інверсією одного біта;

надмірність ключа, що має біти контролю парності. Елі Біхам і Аді Шамір запропонували досить ефективну атаку на реалізацію DES у смарт-картах або банківських криптографічних модулях, що використовують EEPROM-пам'ять для зберігання ключів. Наявність бітів контролю парності дозволяє відновити ключ при втраті його частини внаслідок збоїв комірок пам'яті.

4.2. Основні модифікації DES (3DES, DESX)

З попереднього матеріалу зрозуміло, що основним недоліком алгоритму DES є мала довжина ключа. В принципі, аналітики на це звертали увагу одразу після опублікування стандарту. Однак наприкінці 70-х років XX століття не існувало таких комп'ютерних потужностей, які дозволили б реалізувати атаку "грубою силою", тобто повного перебирання ключів. Проте, вже у 1998 році некомерційній правозахисній організації США Electronic Frontier Foundation з використанням спеціалізованого комп'ютера DES-cracker вартістю у 250 тисяч доларів вдалося здійснити таку атаку за три дні.

Одразу постало питання збільшення крипостійкості DES щодо атаки "грубою силою". Найбільш вдалими рішеннями прийнято вважати так званий Потрійний DES (Triple DES, 3DES) та DESX Рона Рівеста. Обидві модифікації значно підсилюють стійкість алгоритму до атаки повного перебирання ключів. Розглянемо їх детальніше.

4.2.1. Потрійний DES

Існує багато різних варіантів потрійного DES [61]. Найбільш популярними з них є два: 3DES EDE2 (Encrypt-Decrypt-Encrypt з двома ключами) та 3DES EDE3 (Encrypt-Decrypt-Encrypt з трьома ключами).

Потрійний DES з двома ключами.

У цьому алгоритмі використовується два ключі по 56 біт, тобто загальна довжина ключа дорівнює 112 біт. Шифрування цим алгоритмом передбачає етапи, показані на рис. 4.4.

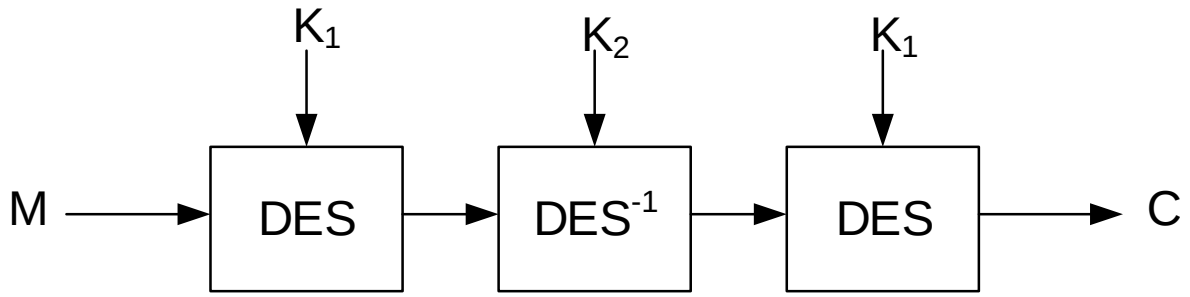


Рис. 4.4. **Схема шифрування алгоритму 3DES EDE2**

Як бачимо з рис. 4.4, відкрите повідомлення *M* спочатку шифрується звичайним однократним DES на ключі K_1, потім розшифровується на ключі K_2, після чого знов шифрується – на K_1. У цьому випадку зростання криптостійкості досягається як збільшенням довжини загального ключа (до 112 біт + біти парності), так і кількістю циклів обробки. На відміну від однократного, потрійний DES еквівалентний 48 раундам обробки відкритого тексту. Очевидно, що потрійний DES рівно втричі повільніший за звичайний, хоча і не такий повільний, як асиметричні алгоритми. Однак швидкий розвиток комп'ютерної техніки деякою мірою згладжує цей недолік. Етап розшифрування на ключі K_2 подано для сумісності з однократним DES у разі $K_1 = K_2$.

Розшифрування відбувається протилежним чином: на вхід алгоритму подають зашифрований текст (*C*), на першому етапі

розшифровують на ключі K_1, на другому – шифрують на K_2, на третьому – знов розшифровують на K_1. У результаті отримуємо розшифровану інформацію (M).

Потрійний DES з трьома ключами.

Відмінність від попереднього алгоритму полягає в тому, що тут використовується три ключі шифрування, отже, стійкість системи до атаки "грубою силою" ще зростає. Загальна довжина ключа досягає 56x3=168 біт + біти парності. У випадку $K_1 = K_2 = K_3$ 3DES EDE3 перетворюється в однократний DES, що правда, втричі повільніший.

Шифрування цим алгоритмом показано на рис. 4.5.

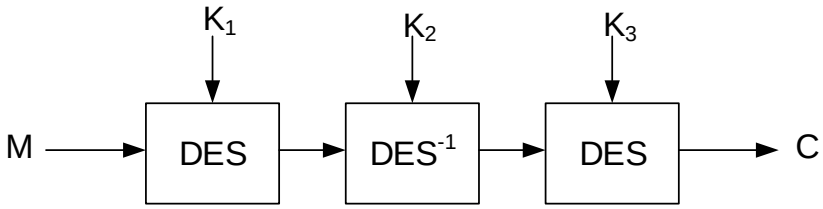


Рис. 4.5. **Потрійний DES з трьома ключами**

Розшифрування виконується аналогічно: спочатку шифроване повідомлення розшифровується на ключі K_3, потім зашифровується на ключі K_2, і, врешті решт, знов розшифровується, але на ключі K_1.

Падіння швидкодії при роботі 3DES іноді дуже помітне, і, наприклад, у режимі зчеплення блоків це сповільнення не вдається компенсувати додатковим апаратним обладнанням. У багатьох випадках, наприклад, при шифруванні критичних каналів зв'язку, таке падіння продуктивності неприпустиме.

4.2.2. Алгоритм DESX

У 1984 році Рон Рівест запропонував модифікацію DES, яка отримала назву DESX (DES eXtended), і була вільна від недоліків 3DES.

DESX визначається як $DESX_{K,K_1,K_2} = K_2 \oplus DES_K(K_1 \oplus M)$.

Як видно з наведеної формули, повний ключ DESX складається з трьох: першого зашумлюючого K_1, який додається за правилами XOR до відкритого повідомлення; ключа DES K; другого зашумлюючого ключа K_2, який додається за XOR до результатів шифрування DES. Таким чином, загальна довжина ключа DESX становить 56 + 64 + 64 = 184 біти, що навіть більше, ніж у 3DES.

Що стосується збільшення часу обробки, то він усього на дві операції додавання за модулем 2 більший за звичайний DES.

Суттєвим для DESX є те, що цих дві операції XOR роблять шифр менш вразливим до атаки "грубою силою", проти чого і була спрямована ця розробка. DESX, однак, збільшує й стійкість простого DES проти диференціального та лінійного криптоаналізу, збільшуючи потрібну кількість проб з обраним відкритим текстом до 2^{60} [55].

Таким чином, практично з усіх боків DESX кращий за DES. Це простий алгоритм, сумісний з DES, ефективно реалізується апаратно, може використовувати існуюче апаратне забезпечення DES.

4.3. Алгоритм криптографічного перетворення ГОСТ 28147-89

Стандарт шифрування даних Радянського Союзу ГОСТ 28147-89 було прийнято на озброєння у 1989 році. В основі цього стандарту лежить сітка Фейстеля. Параметри цього алгоритму такі: довжина блоку – 64 біти; довжина ключа – 256 бітів; кількість раундів – 32.

Алгоритм є класичною сіткою Фейстеля:

$$L_i = R_{i-1};$$
$$R_i = L_i \oplus F(R_{i-1}, K_i).$$

Функція *F* проста. Спочатку права половина та *i*-ий підключ додаються за модулем 2^{32}. Потім результат розбивається на вісім 4-бітових частин, кожна з яких надходить на вхід свого S-блока. ГОСТ 28147 використовує вісім різних S-блоків, кожний з яких має 4-бітовий вхід і 4-бітовий вихід. Виходи всіх S-блоків об’єднуються в 32-бітне слово, яке потім циклічно зсувається на 11 бітів вліво. Нарешті за допомогою XOR результат додається до лівої половини, у результаті чого виходить нова права половина (рис. 4.6). Генерування ключів виконується дуже просто. 256-бітний ключ розбивається на вісім 32-бітних підключів. Алгоритм має 32 раунди, тому кожний підключ використовується в чотирьох раундах за схемою, наведено в табл. 4.9, S-блоки алгоритму наведені у табл. 4.10.

Таблиця 4.9

Використання підключів у ГОСТ 38147-89

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

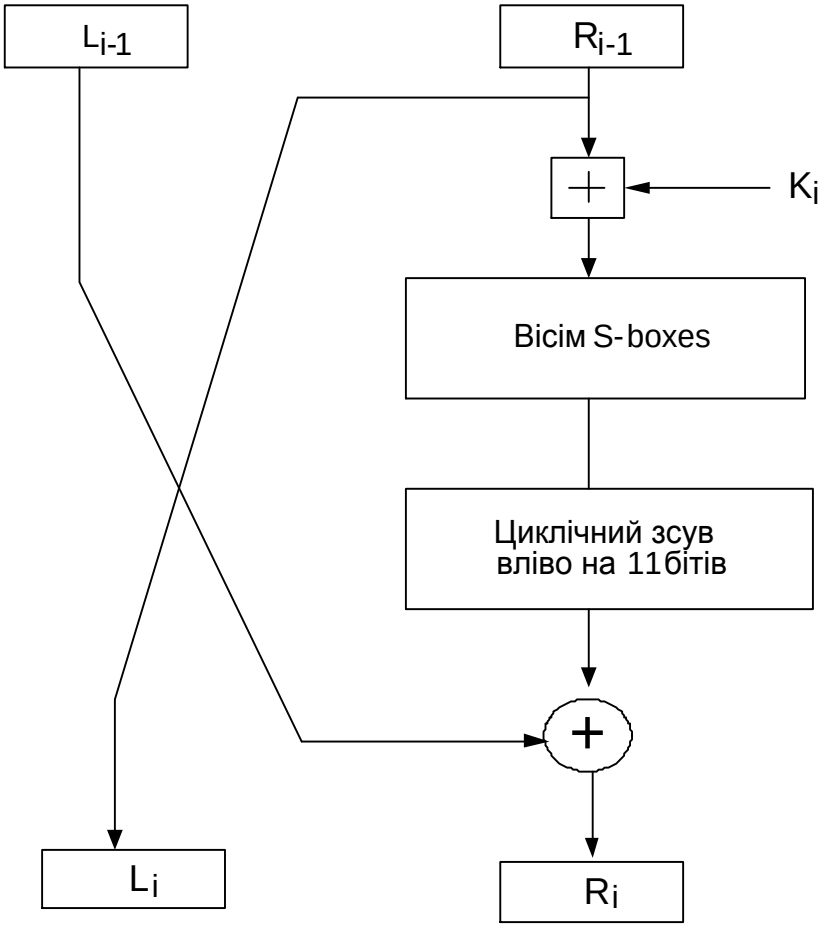


Рис. 4.6. **Один раунд алгоритму ГОСТ 28147-89**

Таблиця 4.10

S-блоки алгоритму ГОСТ 28147-89

1-ий S-блок	4	10	9	2	13	8	0	14
	6	11	1	12	7	15	5	3
2-ий S-блок	14	11	4	12	6	13	15	10
	2	3	8	1	0	7	5	9
3-ий S-блок	5	8	1	13	10	3	4	2
	14	15	12	7	6	0	9	11
4-ий S-блок	7	13	10	1	0	8	9	15
	14	4	6	12	11	2	5	3
5-ий S-блок	6	12	7	1	5	15	13	8
	4	10	9	14	0	3	11	2
6-ий S-блок	4	11	10	0	7	2	1	13
	3	6	8	5	9	12	15	14
7-ий S-блок	13	11	4	1	3	15	5	9
	0	10	14	7	6	8	2	12
8-ий S-блок	1	15	13	0	5	7	10	4
	9	2	3	14	6	11	8	12

Вважається, що стійкість алгоритму ГОСТ 28147 багато в чому визначається структурою S-блоків. Довгий час їх структура у відкритій літературі не публікувалась. На сьогодні відомі S-блоки, які використовуються в програмному забезпеченні Центрального Банку Російської Федерації і вважаються досить сильними. Входом і виходом S-блоків є 4-бітні числа, тому кожний S-блок може бути поданий у вигляді рядка чисел від 0 до 15, розташованих у деякому порядку. Заміна виконується в такий спосіб. 32-бітний блок розбивається на вісім 4-бітних підблоків, кожен з яких подається на вхід свого блоку заміни. Номер S-box визначається порядковим номером 4-бітного підблоку. Двійкове значення, що містить підблок, визначає номер комірки S-блоку, звідки треба взяти значення, яке і буде заміною.

Основні відмінності між DES і ГОСТ 28147 такі:

DES використовує більш складну процедуру створення підключів, ніж ГОСТ 28147. У ГОСТ ця процедура дуже проста: загальний 256-бітний ключ просто ділиться на вісім 32-бітних підключів.

При виборі сильних S-boxes ГОСТ 28147 вважається дуже стійким. У S-boxes DES 6-бітові входи й 4-бітові виходи, а в S-boxes ГОСТ 4-бітові входи й виходи. В обох алгоритмах використовується по вісім S-boxes, але розмір S-box ГОСТ суттєво менший від розміру S-box DES.

У DES застосовуються нерегулярні перестановки *P*, у ГОСТ використовується 11-бітний циклічний зсув вліво. Перестановка DES збільшує лавинний ефект, коли зміна одного біту на вході призводить до зміни багатьох бітів на виході. В ГОСТ зміна одного вхідного біта впливає на один S-box одного раунду, який потім впливає на два S-boxes наступного раунду, три S-boxes наступного і т. д.

Таким чином, для того, щоби зміна одного біту на вході вплинула на кожен вихідний біт, DES вистачає 5 раундів, а ГОСТ для цього потрібно 8 раундів [23]. Однак DES має 16 раундів, а ГОСТ – 32, що робить його більш стійким до диференціального і лінійного криптоаналізу. Лише у 2011 році з’явилися повідомлення про те, що за допомогою алгебраїчного криптоаналізу в ГОСТ 28147-89 було знайдено серйозні вразливості [22; 23; 40]. До практичного застосування цього типу атак ще далеко, але безпека ГОСТ 28147-89 постраждала досить суттєво.

4.4. Алгоритм Rijndael

2 січня 1997 року Національний Інститут Стандартів і Технологій США (NIST) оголосив конкурс симетричних криптоалгоритмів на заміну DES, який вичерпав свої можливості в якості стандарту шифрування. У жовтні 2000 року було оголошено переможця цього конкурсу. Ним виявився алгоритм Rijndael (вимовляється – Рейндал), авторами якого були бельгійські криптографи Вінсент Реймен та Йоан Даймен. У листопаді 2001 року цей алгоритм був прийнятий як новий стандарт шифрування США під назвою AES (Advanced Encryption Standard – вдосконалений стандарт шифрування) (див. FIPS 197 [23; 35; 58]). Розглянемо цей алгоритм детальніше.

Одним з яскравих недоліків алгоритмів на основі сітки Фейстеля є те, що за один раунд шифрується лише половина вхідного блоку, а решта бітів просто переміщується без зміни в іншу половину. Як мінімум, це призводить до збільшення кількості раундів алгоритму.

Алгоритм Rijndael не належить до фейстелівських. Замість цього раундова функція складається із чотирьох різних перетворень, які називаються *шарами*, і обробляють весь вхідний блок разом.

Кожний шар розроблявся з урахуванням протидії лінійному та диференціальному криптоаналізу. В основу кожного шару покладена своя власна функція:

1. Нелінійний шар складається з паралельного застосування S-блоків для оптимізації нелінійних властивостей.
2. Шар лінійного переміщення рядків гарантує високий степінь дифузії для невеликої кількості раундів.
3. Шар лінійного переміщення стовпців також гарантує високий степінь дифузії для невеликої кількості раундів.
4. Додатковий шар підмішування ключа складається з простого XOR проміжного стану із ключем раунду.

Перед першим раундом застосовується додаткове забілювання з використанням ключа. Причина цього полягає в тому, що будь-який шар після останнього або до першого додавання ключа може бути просто знятий без знання ключа і тим самим не додає безпеки в алгоритм (наприклад, початкова і кінцева перестановки в DES).

Початкове або кінцеве додавання ключа застосовується також у деяких інших алгоритмах, наприклад, IDEA, SAFER і Blowfish. Для

спрощення структури алгоритму шар лінійного переміщення останнього раунду відрізняється від шару переміщення інших раундів. Можна показати, що це в жодному разі не підвищує і не знижує безпеку аналогічно відсутності операції swap в останньому раунді DES.

Шифр є послідовністю ітерацій, які виконуються над деякою проміжною структурою, яка називається *станом*. Стан може бути представлений у вигляді прямокутного масиву байтів. Масив має чотири рядка, а кількість стовпчиків, яку позначають N_b, дорівнює довжині блока, поділеній на 32.

Ключ шифрування аналогічно подається у вигляді прямокутного масиву байтів з чотирьох рядків. Кількість стовпчиків (N_k) визначається довжиною ключа також поділеною на 32. Вхідні та вихідні дані подаються як одновимірні масиви байтів відповідної довжини.

Стан і ключовий масив заповнюються з цих масивів спочатку за стовпчиками, а потім за рядками так, як показано в табл. 4.11.

Таблиця 4.11

Порядок байтів у стані

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

Кількість раундів N_r залежить від N_b та N_k відповідно до табл. 4.12.

Таблиця 4.12

Кількість раундів як функція довжин блоку і ключа

N_r	$N_b = 4$ (128 біт)	$N_b = 6$ (192 біти)	$N_b = 8$ (256 біт)
$N_k = 4$ (128 біт)	10	12	14
$N_k = 6$ (192 біти)	12	12	14
$N_k = 8$ (256 біт)	14	14	14

Раундова функція складається з чотирьох перетворень: застосування таблиці заміни (ByteSub); зсуву рядків (ShiftRow); перемішування стовпчиків (MixColumn) та підмішування раундового ключа (AddRoundKey). Останній раунд відрізняється від усіх попередніх, не використовуючи перемішування стовпчиків.

Нагадаємо, що на відміну від фейстелевських алгоритмів, раундова функція діє на весь вхідний блок разом.

Блок-схема процесу шифрування алгоритму AES подано на рис. 4.7.

Опишемо окремі перетворення раундової функції.

1. Перетворення SubBytes. Суть цього перетворення полягає в заміні кожного байта {*xy*} стану (де *x* та *y* – шістнадцяткові значення) на інші відповідно до таблиці заміни, яку подано в табл. 4.13. Зрозуміло, що під час розшифрування необхідно використати обернену таблицю, яка зображена в табл. 4.14.

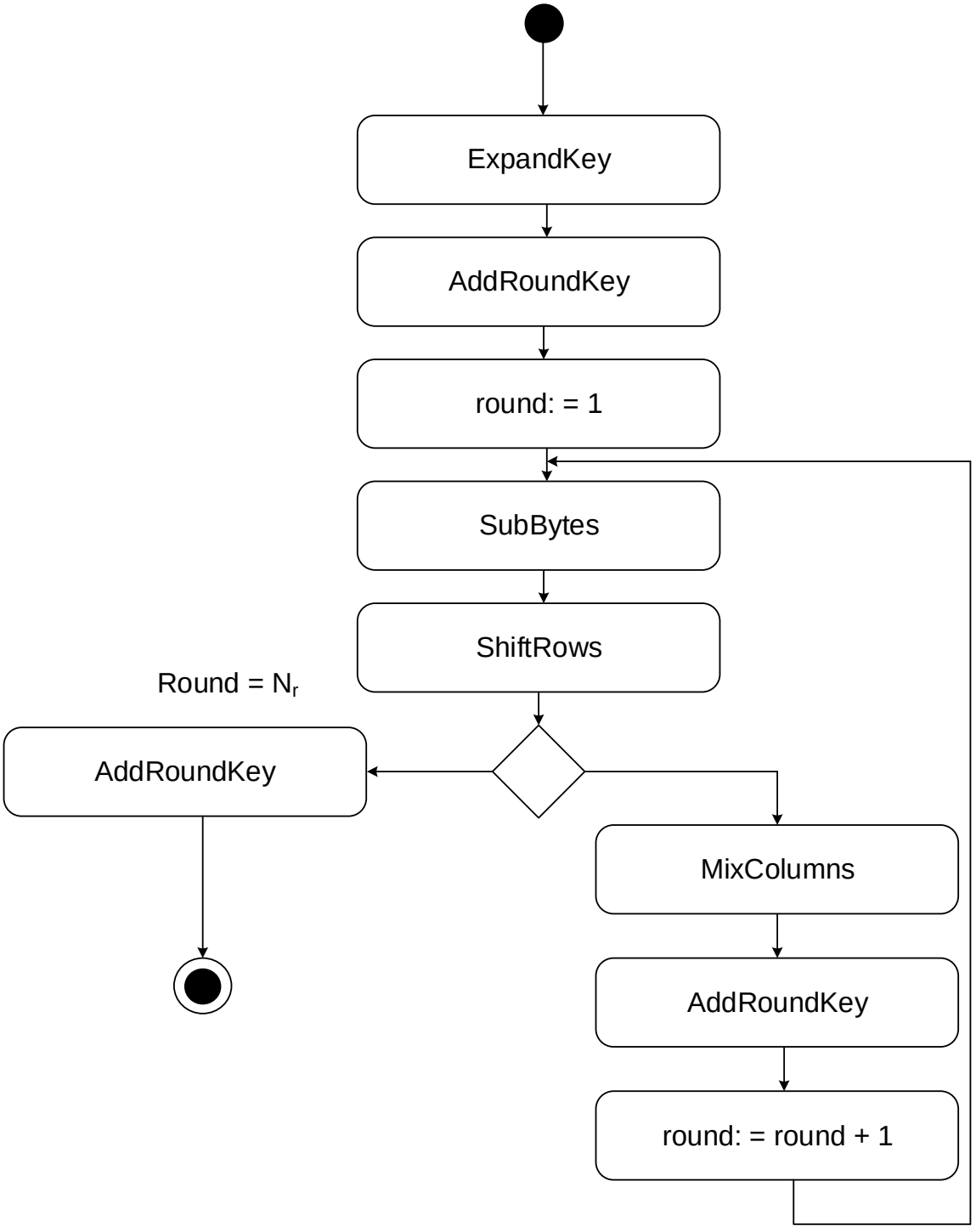


Рис. 4.7. **Блок-схема процесу шифрування алгоритму AES**

Таблиця 4.13

Таблиця заміни для шифрування алгоритму Rijndael

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Наприклад, байт {fe} буде замінено на {bb}.

Таблиця 4.14

Обернена таблиця заміни для процесу розшифрування

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Наприклад, байт {bb} буде замінений {fe}, що якраз і свідчить про взаємну оберненість таблиць заміни.

З математичної точки зору заміна еквівалентна таким операціям в полі Галуа $GF(2^8)$, яке використовується в цьому алгоритмі:

а) кожний байт замінюють на обернений елемент за операцією множення, визначеній в цьому полі ({00} відображується сам на себе);

б) потім застосовується афінне перетворення, визначене таким чином:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Множення та додавання виконується за правилами, визначеними в полі Галуа $GF(2^8)$ [57].

2. ***ShiftRows (зсув рядків)***. Суть перетворення полягає в циклічному зсуві рядків стану. Схематично такий зсув показано на рис. 4.8.

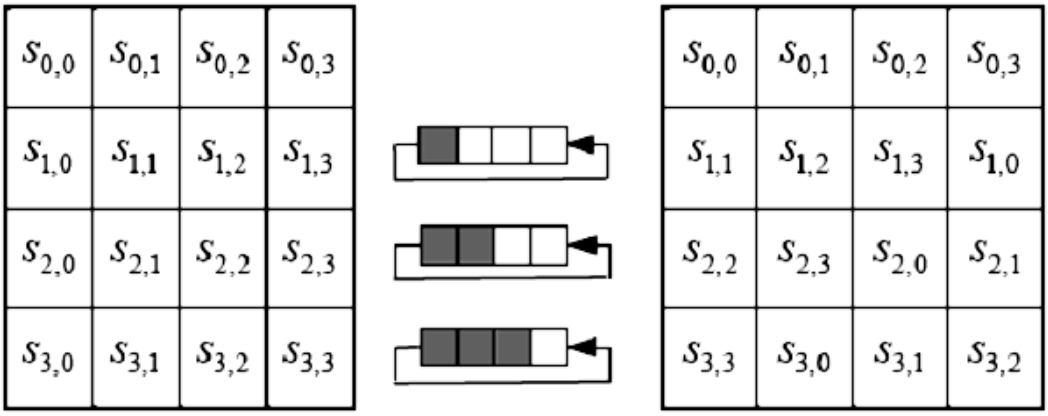


Рис. 4.8. **Перетворення ShiftRows**

Перший рядок залишається незмінним. Другий – зсувається вліво на один байт, а перший байт записується в кінець рядка. Третій зсувається на два байти, а четвертий – на три. Обернене перетворення – зсув вправо.

3. ***Перетворення MixColumns***. Це перетворення виконується як множення квадратної матриці четвертого порядку на кожен стовпчик стану:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

По суті, це означає, що стовпчики стану розглядаються як поліноми в $GF(2^8)$ і множаться за модулем $x^4 + 1$ на фіксований поліном (коефіцієнти даються у 16-ковій формі):

$$c(x) = \{03\}\, x^3 + \{01\}\, x^2 + \{01\}\, x + \{02\}.$$

Цей поліном взаємно простий з $x^4 + 1$ і, отже, існує обернений до нього поліном. Інверсія MixColumn визначається таким чином:

$$(\{03\}\, x^3 + \{01\}\, x^2 + \{01\}\, x + \{02\}) \otimes d(\mathrm{x}) = \{01\}.$$

Добуток виконується за правилами поля Галуа $GF(2^8)$. У результаті одержимо:

$$d(x) = \{0\mathrm{B}\}\, x^3 + \{0\mathrm{D}\}\, x^2 + \{09\}\, x + \{0\mathrm{E}\}.$$

У матричному вигляді будемо мати обернену операцію до MixColumn, яка використовується при розшифруванні:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

Тут множення також відбувається за правилами $GF(2^8)$.

4. ***Перетворення AddRoundKey*** зводиться до побітового додавання за модулем два кожного біта раундового ключа з бітами стану. Оберненим перетвореннv до AddRoundKey служить воно саме.

5. ***Процедура ExpandKey*** приймає на вході ключ шифрування і перетворює його на ключі для усіх раундів. Загальна кількість бітів ключового масиву дорівнює довжині блоку, помноженій на кількість раундів плюс 1, тобто $N_b \times (N_r + 1)$. Наприклад, для довжини блоку 128 бітів і 10 раундів необхідно 1408 бітів ключового масиву.

Операція розгортання ключа виконується в такий спосіб. Перші N_k позицій розширеного ключа заповнюються ключем шифрування. Наступні чотирибайтні слова, $w[i]$, утворюються як результат операції XOR між $w[i\text{-}1]$ та $w[i\text{-}N_k]$ словами. Для слів, позиція яких кратна N_k, додатково перед XOR'ом до $w[i\text{-}1]$ застосовується трансформація, яка складається

з циклічного зсуву байтів у слові, додавання за правилами XOR з константою раунду $R_{con}[i]$ та виконання табличної підстановки для усіх байтів слова. Слід зазначити, що операція розгортання ключа має два варіанти: для $N_k \leq 6$ та $N_k > 6$. Відмінність для $N_k > 6$ полягає в тому, що таблична підстановка виконується до XOR з $R_{con}[i]$.

Константи раунду не залежать від N_k і визначаються в такий спосіб:

$R_{con}[i] = (\{02^{i-1}\}, \{00\}, \{00\}, \{00\})$. Раундові ключі вибираються з ключового масиву від $w[N_b \times i]$ до $w[N_b \times (i+1)]$.

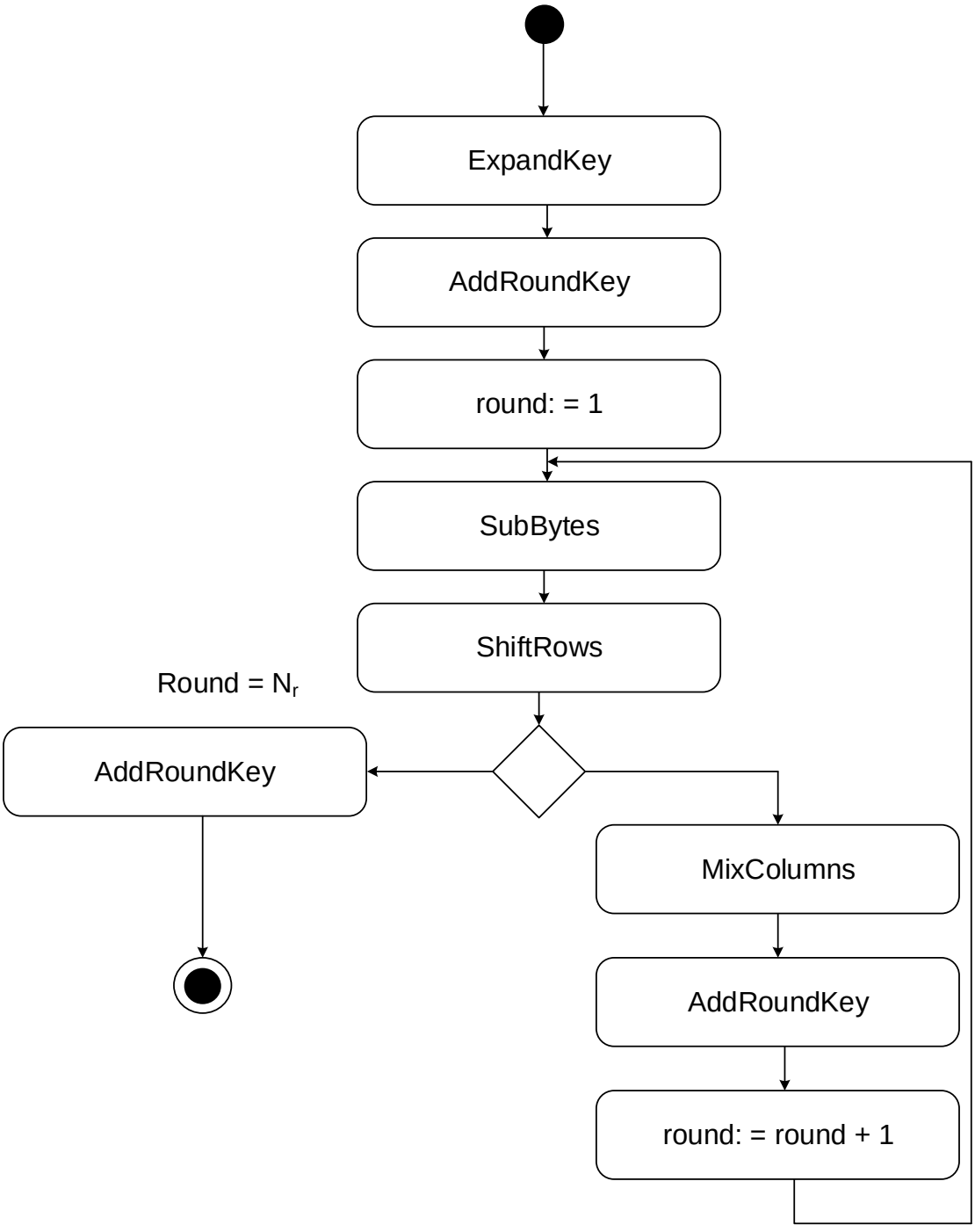


Рис. 4.9. **Алгоритм розшифрування Rijndael**

На рис. 4.9 подано алгоритм розшифрування. Як бачимо, усі перетворення виконуються в оберненому порядку. Замість шифрувальних перетворень використовуються розшифрувальні.

1. InvSubBytes – операція заміни байтів, яка використовує обернену таблицю заміни.

2. InvShiftRows – циклічний зсув байтів вправо, протилежно до такого при шифруванні.

3. InvMixColumns – перемішування стовпчиків з використанням оберненої матриці.

4. Процедури ExpandKey та AddRoundKey залишаються без змін.

5. Раундові ключі для розшифрування використовуються в оберненому порядку.

Переваги та недоліки алгоритму

За оцінками розробників алгоритм Rijndael вже на чотирьох раундах має криптостійкість, достатню для сучасних застосувань. Теоретичною межею вважаються 6 – 8 раундів. Отже, 10 – 14 раундів, які пропонують розробники, мають значний запас криптостійкості.

Переваги, які відносяться до аспектів реалізації, такі:

1. Rijndael може виконуватись швидше, ніж традиційний блоковий алгоритм шифрування на основі сітки Фейстеля, оскільки розробниками проведено оптимізацію між розміром таблиці і швидкістю виконання.

2. Rijndael придатний для реалізації в смарт-картах, оскільки він може працювати з невеликим RAM і є невелику кількість раундів.

3. Раундові перетворення добре розпаралелюються, що є важливою перевагою для майбутніх процесорів і спеціалізованої апаратури.

4. Алгоритм шифрування не використовує арифметичні операції, тому тип архітектури процесора не має значення.

5. Алгоритм шифрування повністю самодостатній. Він не використовує ніяких інших криптографічних компонентів, *S*-boxів, взятих з добре відомих алгоритмів, бітів, отриманих зі спеціальних таблиць.

6. Безпека алгоритму не базується на ітераціях арифметичних операцій, які складно зрозуміти, проста розробка краще аналізується, і крім того, в компактній структурі практично неможливо приховати "люки" або дефекти.

Структура алгоритму дозволяє використовувати шифр з різними довжинами блоку, від 128 до 256 бітів з кроком в 32 біти. Крім того, операція розгортання ключа може працювати з ключами, довжина яких

кратна 4 байтам. Отже, вона необов'язково повинна дорівнювати 128, 192 або 256 бітів. Для іншої довжини ключа необхідно знати лише кількість раундів алгоритму, яка визначається так:

$$N_r = \max(N_k, N_b) + 6.$$

Таким чином, авторам вдалося спроектувати надзвичайно вдалий та гнучкий алгоритм, криптостійкість якого можна змінювати залежно від ситуації, налаштовуючи її зміною довжини ключа, блоку та кількістю раундів.

Відносним недоліком цього алгоритму може бути те, що його структура ще недостатньо вивчена, і, можливо, містить невідомі зараз недоліки. Хоча, слід зазначити, що більшість аналітиків з сумнівом відносяться до такої можливості [33; 40].

4.5. Інші відомі блокові шифри

4.5.1. Алгоритм RC2

Криптоалгоритм RC2 є блоковим шифром із ключем змінної довжини, розроблений Р. Рівестом на замовлення компанії RSA Data Security, Inc. Абревіатура "RC" означає "Код Рональда" (Ron's Code), або "Шифр Рівеста" (Rivest's Cipher). Криптоалгоритм розроблявся як альтернатива стандарту DES. RC2 працює із блоками по 64 біти, програмна реалізація криптоалгоритму приблизно у два – три рази швидша, ніж DES. Змінна довжина ключа дозволяє домогтися адекватної криптостійкості з урахуванням можливостей силової атаки. Уряд США не забороняє експортувати апаратні і програмні реалізації криптоалгоритмів RC2 і RC4 – при дотриманні 40-бітового обмеження на довжину ключа. Американські компанії за межами США і Канади можуть використовувати крипто-алгоритми з довжиною ключа 65 бітів. При шифруванні за крипто-алгоритмом RC2 до секретного ключа методом конкатенації додається деякий допоміжний ключ від 40 до 88 бітів. Для виконання дешифрування допоміжний ключ передається одержувачу зашифрованого повідомлення у відкритому вигляді.

4.5.2. Алгоритм RC5

Криптоалгоритм RC5 також розроблений Р. Рівестом на замовлення компанії RSA Data Security, Inc. Це блоковий шифр зі змінною

довжиною блоку (32, 64 і 128 бітів), ключа і кількістю циклів криптографічного перетворення (від 0 до 255). Довжина ключа варіюється від 0 до 2048 бітів. Така параметризація дозволяє гнучко налаштувати криптоалгоритм із урахуванням конкретних вимог до криптостійкості та ефективності реалізації. Криптоалгоритм RC5 складається із трьох основних процедур: розгортання ключа, шифрування і розшифрування.

У процедурі розгортання ключа заданий секретний ключ піддається спеціальному перетворенню з метою заповнення ключової таблиці, причому розмір таблиці залежить від кількості циклів криптографічного перетворення. Ключова таблиця використовується потім для шифрування і розшифрування. Процедура шифрування складається із трьох основних операцій: цілочисельного сумування, додавання за модулем 2 і циклічного зсуву. Безумовна перевага криптоалгоритму полягає в простоті реалізації. Непередбачуваність результату операції циклічного зсуву, що залежить від конкретних вхідних даних при шифруванні, забезпечує необхідний рівень криптостійкості. Дослідження криптостійкості RC5 показали, що варіант криптоалгоритму з розрядністю блоку 64 біти та 12 (і більше) циклами перетворення гарантує адекватну криптостійкість до диференціального та лінійного криптоаналізу.

4.5.3. Алгоритм IDEA

IDEA (International Data Encryption Algorithm) є блоковим симетричним алгоритмом шифрування, розробленим Сюдзя Лай (Xuejia Lai) і Джеймсом Мессі (James Massey) зі швейцарського федерального інституту технологій. Перша версія була опублікована в 1990 році. Переглянута версія алгоритму, посилена засобами захисту від диференціальних криптографічних атак, була подана в 1991 р. і докладно описана в 1992 р.

IDEA претендував на роль одного з симетричних алгоритмів, якими передбачалось замінити DES.

Принципи розробки

IDEA використовує 128-бітовий ключ для шифрування даних блоками по 64 біти. Метою його розробки було створення стійкого криптографічного алгоритму з досить простою реалізацією.

Криптографічна стійкість.

Наступні характеристики IDEA характеризують його криптографічну стійкість:

1. Довжина блоку повинна бути достатньою, щоб приховати усі статистичні характеристики вхідного повідомлення. З іншого боку, складність реалізації криптографічної функції зростає експоненціально з розміром блоку. Використання блоку розміром в 64 біти в 90-ті роки вважалося достатнім компромісом. Подальше посилення крипостійкості досягалося режимами використання алгоритму.

2. Довжина ключа повинна бути досить великою для того, щоб запобігти атаці повного перебирання ключів. При довжині ключа 128 бітів IDEA вважається з цього боку досить безпечним.

3. Конфузія: зашифрований текст повинен залежати від ключа складним і заплутаним способом.

4. Дифузія: зміна одного біту відкритого тексту повинна впливати на усі біти зашифрованого. Це ефективно приховує статистичну структуру відкритого тексту. IDEA з цієї точки зору вважається дуже ефективним алгоритмом.

В IDEA два останні пункти задовольняються трьома операціями, на відміну від DES, який використовує операцію XOR та маленькі нелінійні таблиці заміни.

Кожна операція виконується над двома 16-бітними входами і створює один 16-бітний вихід.

Цими операціями є:

1. Побітовий XOR.

2. Додавання за модулем 2^{16} (за модулем 65536); при цьому входи і виходи вважаються беззнаковими 16-бітними цілими числами. Цю операцію будемо позначати "+".

3. Множення за модулем $2^{16} + 1$ (за модулем 65537); при цьому входи і виходи вважаються без знаковими 16-бітними цілими числами за винятком того, що блок з нулів трактується як 2^{16}. Цю операцію будемо позначати "•".

Ці три операції несумісні в такому розумінні:

1. Не існує жодної пари з трьох операцій, що задовольняють дистрибутивному закону:

$$a \bullet (b + c) <=> (a \bullet b) + (a \bullet c).$$

2. Не існує жодної пари з трьох операцій, що задовольняють асоціативному закону. Наприклад:

$$a + (b \oplus c) <=> (a + b) \oplus c.$$

Використання комбінації цих трьох операцій забезпечує комплексну трансформацію входу, значно ускладнюючи криптоаналіз порівняно з алгоритмом DES, який ґрунтується винятково на функції XOR.

Шифрування

Розглянемо загальну схему шифрування IDEA. Як і в будь-якому іншому алгоритмі він має два входи: блок відкритого тексту і ключ. У цьому випадку незашифрований блок має довжину 64 біти, ключ – 128 бітів.

Алгоритм IDEA складається з восьми раундів, які передують заключному перетворенню. Алгоритм розділяє блок на чотири 16-бітних підблоки. Кожний раунд і заключне перетворення отримують на вході чотири 16-бітних підблоки і створюють чотири 16-бітних вихідних підблоки. Раунд використовує шість 16-бітних підключів, заключне перетворення – чотири підключі, отже, разом в алгоритмі використовується 52 підключа (рис. 4.10).

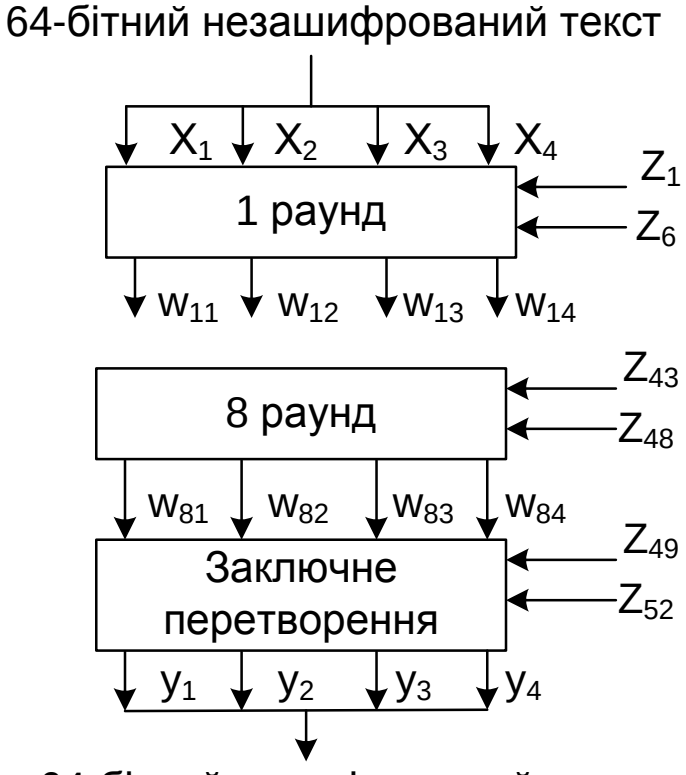


Рис. 4.10. **Коротка блок-схема алгоритму IDEA**

Послідовність перетворень окремого раунду.

Розглянемо послідовність перетворень окремого раунду детальніше.

Одним з основних елементів алгоритму, що забезпечують дифузію, є структура, яку називають МД (множення/додавання).

На вхід цієї структури подаються два 16-бітних значення і два 16-бітних підключа, на виході створюються два 16-бітних підблоки. Аналіз показує, що кожний біт виходу цієї структури залежить від кожного біта входів незашифрованого блоку і підключів.

Ця структура повторюється в алгоритмі вісім разів, забезпечуючи високоефективну дифузію (рис. 4.11).

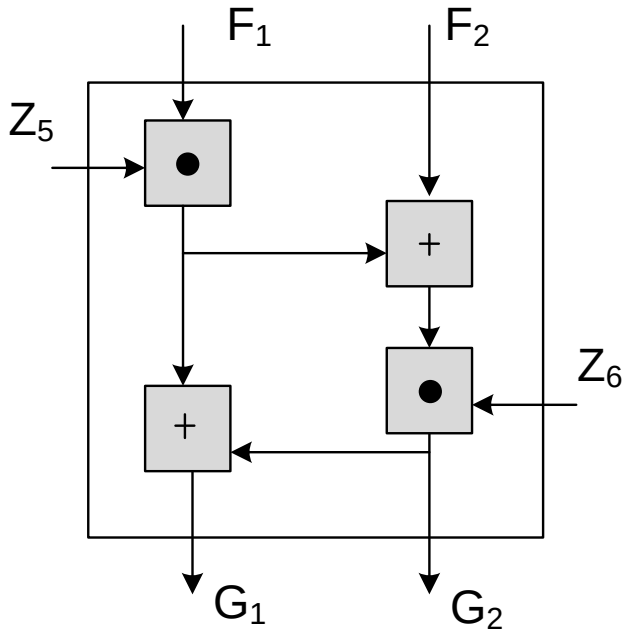


Рис. 4.11. **Структура МД (множення/додавання)**

Раунд починається з перетворення, яке комбінує чотири вхідних підблоки із чотирма підключами, використовуючи операції додавання і множення. Чотири вихідні блоки цього перетворення додаються за правилами XOR для формування двох 16-бітних блоків, які є входами МД-структури. Крім того, МД-структура отримує на вході ще два підключі і в результаті створює два 16-бітних виходи, які додаються за XOR з результатами першого перетворення для створення чотирьох вихідних підблоків цієї ітерації.

Зазначимо, що два виходи, які частково створюються другим і третіми входами (X_2 і X_3), міняються місцями для створення другого і третього виходів (W_{12} і W_{13}). Це покращує перемішування бітів і робить алгоритм більш стійким до диференціального криптоаналізу.

Дев’ятий раунд алгоритму, який називається заключним перетворенням, відрізняється від восьми попередніх тим, що другий і третій входи міняються місцями. Це робиться для того, щоб дешифрування мало ту ж структуру, що і шифрування. Зазначимо, що дев’ятий раунд вимагає тільки чотири вхідні підключі, у той час, як для перших восьми необхідно по шість вхідних підключів (рис. 4.12, 4.13).

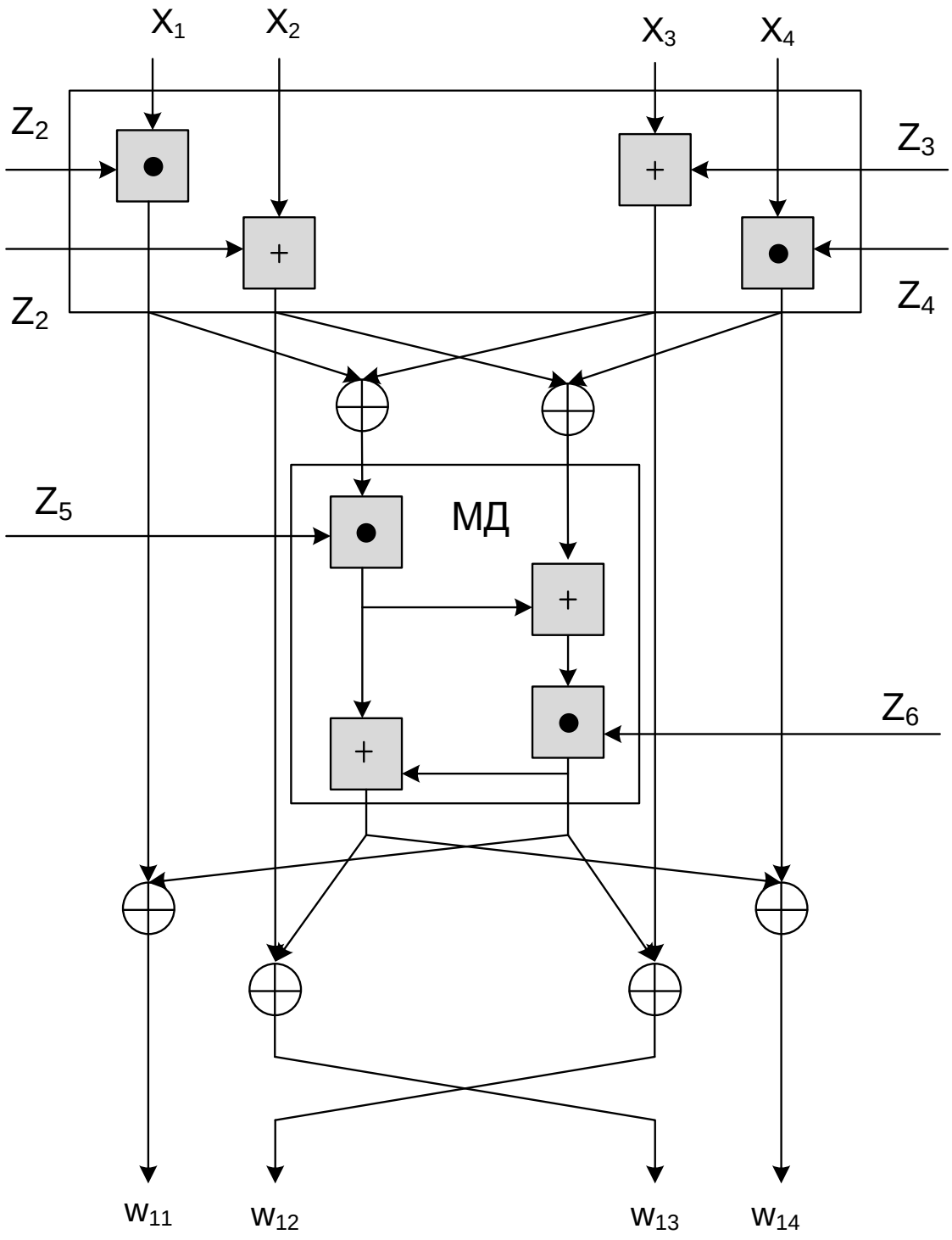


Рис. 4.12. **І-й раунд алгоритму IDEA**

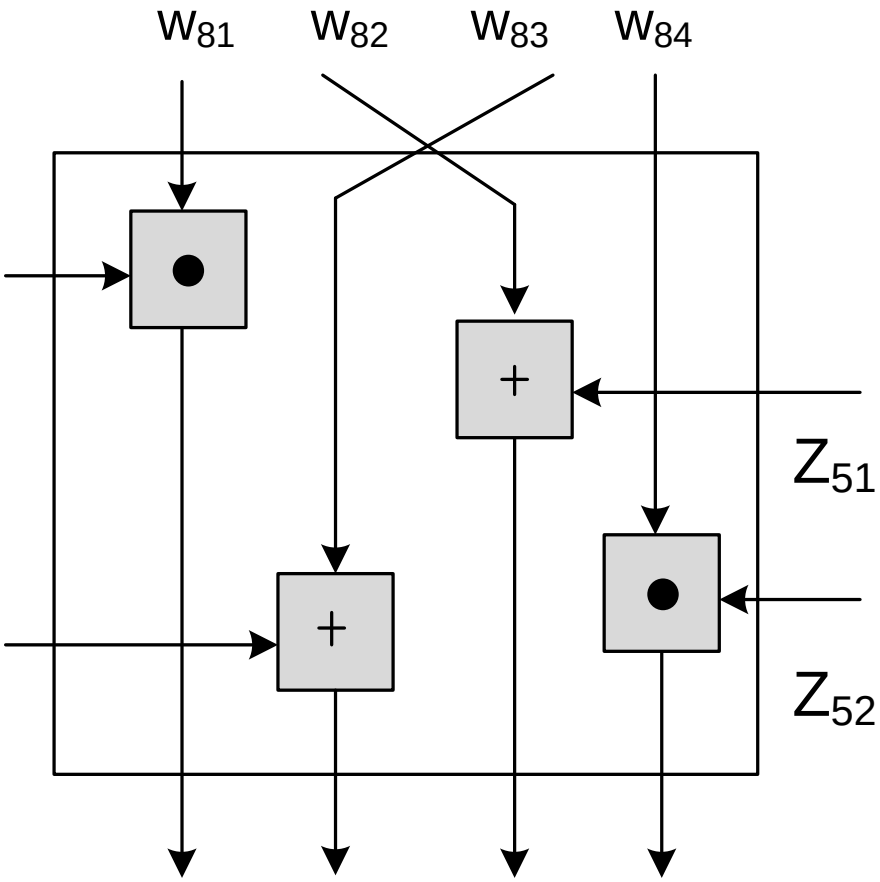


Рис. 4.13. **Заключне перетворення алгоритму IDEA**

Операція розгортання ключа

П’ятдесят два 16-бітних підключа розгортаються зі 128-бітного ключа шифрування в такий спосіб. Перші вісім підключів, які позначають Z_1, Z_2, ..., Z_8, отримуються безпосередньо з ключа, при цьому Z_1 дорівнює першим 16 бітам, Z_2 – наступним 16 бітам і т. д. Потім виконується циклічний зсув ключа шифрування вліво на 25 бітів, і створюються наступні вісім підключів. Ця процедура повторюється доти, поки не будуть створені всі 52 підключі.

Зазначимо, що кожний перший підключ раунду отримується зі своєї підмножини бітів ключа. Якщо весь ключ позначити як $Z_{[1..128]}$, то першими ключами у восьми раундах будуть:

$$Z_1 = Z_{[1..16]}; \quad Z_{25} = Z_{[76..91]}$$
$$Z_7 = Z_{[97..112]}; \quad Z_{31} = Z_{[44..59]}$$
$$Z_{13} = Z_{[90..105]}; \quad Z_{37} = Z_{[37..52]}$$
$$Z_{19} = Z_{[83..98]}; \quad Z_{43} = Z_{[30..45]}$$

У кожному раунді за винятком першого і восьмого використовуються тільки 96 бітів підключа, причому множина бітів ключа на кожній ітерації не перетинається, і не існує простих взаємовідносин, наприклад, типу простого зсуву, між підключами різних раундів. Це відбувається завдяки тому, що в кожному раунді використовується тільки шість підключів, у той час, як при кожній ротації ключа виходить вісім підключів.

Операція розшифрування

Процес розшифрування аналогічний процесу шифрування і полягає у використанні зашифрованого тексту як входу в ту ж саму структуру IDEA, але з іншим набором ключів. Ключі, які використовуються при розшифруванні, U_1, ..., U_{52} отримують з підключів шифрування у такий спосіб:

1. Перші чотири підключі першого раунду розшифрування отримуються як перші чотири підключі 10-го раунду шифрування, якщо вважати стадію заключного перетворення 9-м раундом. Перший і четвертий підключі розшифрування еквівалентні мультиплікативній інверсії за модулем $(2^{16} + 1)$ відповідно першого і четвертого підключів шифрування. Для раундів з 2 по 8 другий і третій підключі розшифрування еквівалентні адитивній інверсії за модулем 2^{16} відповідно третього та другого підключів шифрування. Для раундів 1 і 9 другий і третій підключі розшифрування еквівалентні адитивній інверсії за модулем 2^{16} відповідно другого і третього підключів шифрування.

2. Для перших восьми раундів останні два підключа і раунди розшифрування еквівалентні останнім двом підключам 9-го раунду шифрування. Для мультиплікативної інверсії використовується нотація Z_j^{-1}, тобто:

$$Z_j \bullet Z_j^{-1} = 1 \bmod (2^{16} + 1).$$

Через те, що $(2^{16} + 1)$ є простим числом, кожне ненульове ціле $Z_j \leq 2^{16}$ має унікальну мультиплікативну інверсію за модулем $(2^{16} + 1)$. Для адитивної інверсії використовується нотація $-Z_j$, таким чином маємо:

$$-Z_j + Z_j = 0 \bmod 2^{16}.$$

Для доведення коректності алгоритму розшифрування розглянемо його одночасно з процесом шифрування. Кожний з восьми раундів розбитий на дві стадії перетворення, перша з яких називається трансформацією, а друга – шифруванням (рис. 4.14, 4.15).

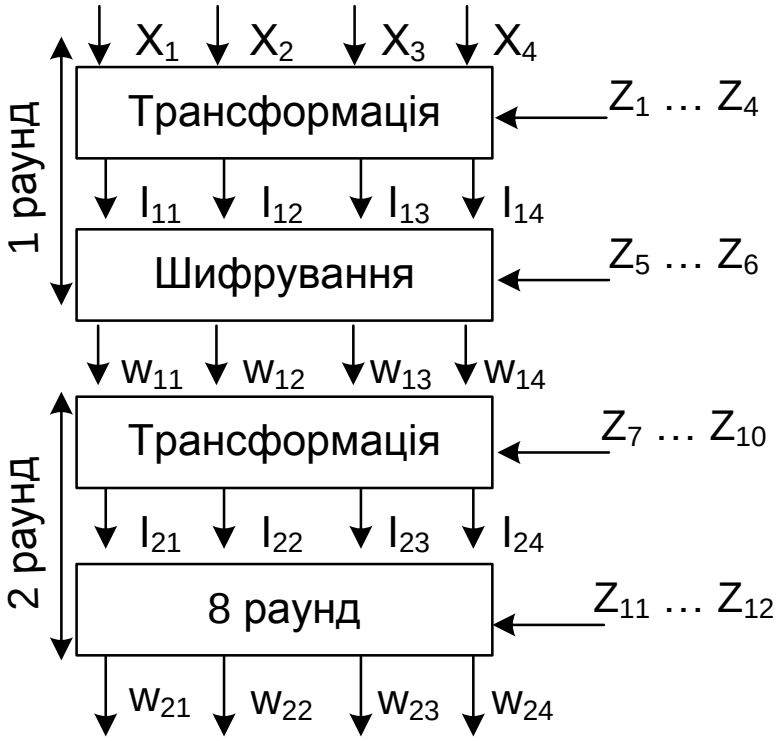


Рис. 4.14. **Блок-схема процесу шифрування IDEA**

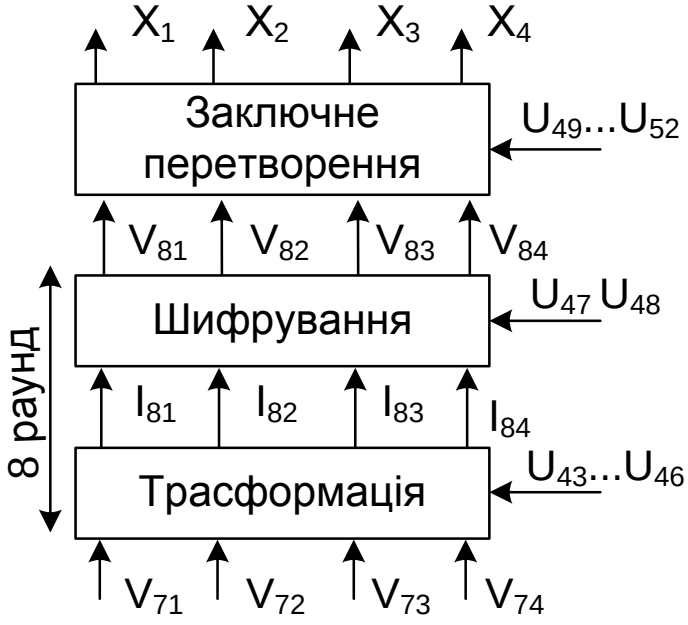


Рис. 4.15. **Блок-схема процесу розшифрування IDEA**

При шифруванні підтримуються такі співвідношення на виході трансформації:

$$Y_1 = W_{81} \bullet Z_{49};\ Y_3 = W_{82} + Z_{51};$$
$$Y_2 = W_{83} + Z_{50};\ Y_4 = W_{84} \bullet Z_{52}.$$

Для першої стадії першого раунду процесу розшифрування справедливі такі співвідношення:

$$J_{11} = Y_1 \bullet U_1;\ J_{13} = Y_3 + U_3;$$
$$J_{12} = Y_2 + U_2;\ J_{14} = Y_4 \bullet U_4.$$

Підставляючи відповідні значення, одержимо:

$$J_{11} = Y_1 \bullet Z_{49}^{-1} = W_{81} \bullet Z_{49} \bullet Z_{49}^{-1} = W_{81};$$
$$J_{12} = Y_2 + -Z_{50} = W_{83} + Z_{50} = W_{83} + Z_{50} + -Z_{50} = W_{83};$$
$$J_{13} = Y_3 + -Z_{51} = W_{82} + Z_{51} + -Z_{51} = W_{82};$$
$$J_{14} = Y_4 \bullet Z_{52\text{-}1} = W_{84} \bullet Z_{52} \bullet Z_{52}^{-1} = W_{84}.$$

Таким чином, вихід першої стадії процесу розшифрування еквівалентний входу останньої стадії процесу шифрування за винятком перестановки другого і третього блоків. Тепер розглянемо такі відношення:

$$W_{81} = I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84})$$
$$W_{82} = I_{83} \oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84})$$
$$W_{83} = I_{82} \oplus \text{MAL}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84})$$
$$W_{84} = I_{84} \oplus \text{MAL}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}),$$

де MAR(X, Y) – правий вихід МД-структури із входами X та Y;

MAL(X, Y) – лівий вихід МД-структури із входами X та Y.

У результаті одержимо:

$V_{11} = J_{11} \oplus \text{MAR}\ (J_{11} \oplus J_{13},\ J_{12} \oplus J_{14}) = W_{81} \oplus \text{MAR}(W_{81} \oplus W_{82},\ W_{83} \oplus W_{84}) = I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}) \oplus \text{MAR}\ [I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}) \oplus I_{83} \oplus\oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}) \oplus I_{82} \oplus \text{MAL}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}) \oplus I_{84} \oplus \text{MAL}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84})\ ] = I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}) \oplus \text{MAR}(I_{81} \oplus I_{83},\ I_{82} \oplus I_{84}) = I_{81}.$

Аналогічно будемо мати:

$$V_{12} = I_{83},\ V_{13} = I_{82},\ V_{14} = I_{84}.$$

Через те, що вихід другої стадії процесу розшифрування еквівалентний входу передостанньої стадії процесу шифрування за винятком перестановки другого і третього підблоків, аналогічно можна показати, що:

$$V_{81} = I_{11},\ V_{82} = I_{13},\ V_{83} = I_{12},\ V_{84} = I_{14}.$$

Таким чином, легко показати, що вихід всього процесу шифрування еквівалентний входу процесу шифрування.

4.5.4. Алгоритм SAFER

Криптоалгоритм SAFER (Secure And Fast Encryption Routine) є блоковим шифром, розробленим на замовлення корпорації Cylink [61]. Довжина ключа шифрування в одній з версій дорівнює 64 біти. Криптоалгоритм орієнтований на байтову обробку блоків по 64 біти, і має змінну кількість раундів криптографічного перетворення (для описаної специфікації – 6). Цей шифр, на відміну від більшості блокових шифрів, має різні процедури шифрування і розшифрування. Перша версія SAFER з довжиною ключа 64 біти відома за назвою SAFER K-64. Друга версія – SAFER K-128 має 128-бітний ключ. Результати криптоаналізу продемонстрували криптостійкість SAFER K-64 стосовно диференціаль-ного та лінійного криптоаналізів при дотриманні однієї умови – кількість циклів перетворення повинна бути більше шести. У результаті досліджень було виявлено слабке місце процедури перетворення ключа. У нових модифікаціях SAFER SK-64 і SAFER SK-128 виявлений недолік усунутий.

4.5.5. Алгоритм FEAL

Криптоалгоритм FEAL був розроблений як альтернатива DES [61]. Оригінальний криптоалгоритм (FEAL-4) був розрахований на програмну реалізацію і мав чотири цикли перетворення при обробці 64-бітного блоку та 64-бітного ключа шифрування. Криптоаналітичне дослідження продемонструвало слабкість криптоалгоритму – для розкриття ключа при атаці на основі обраного відкритого тексту досить 20 зразків [32]. Успіхи в криптоанализі FEAL-8 (вісім циклів перетворення) призвели до появи модифікації зі змінною кількістю циклів перетворення FEAL-N. Опубліковані результати успішного диференціального криптоаналізу FEAL-N при $N \leq 31$. Для розкриття ключа FEAL-8 методом лінійного криптоаналізу досить 2^{25} різних відкритих текстів.

4.5.6. Алгоритм Blowfish

Алгоритм Blowfish є 16-раундовою сіткою Фейстеля з довжиною блоку в 64 біти. Ключ може мати довільну довжину в межах 448 бітів.

Хоча перед початком шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко. Алгоритм призначений, в основному для застосувань, де ключ міняється нечасто, до того ж існує фаза початкової автентифікації сторін, під час якої відбувається узгодження загальних параметрів шифрування. При реалізації на 32-бітних мікропроцесорах з великим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розгортання ключа і шифрування даних. Розгортання ключа перетворює ключ довжиною, принаймні 448 бітів, у кілька масивів підключів загальною довжиною 4168 байтів. Кожний раунд шифру складається з перестановки, залежної від ключа, і заміни, яка залежить, і від ключа, і від даних. Раундовими операціями є XOR і додавання 32-бітних слів.

Blowfish використовує велику кількість підключів, які необхідно обчислити заздалегідь, до початку процесів шифрування або розшифрування даних. Алгоритм складається:

1. P – масив, що містить вісімнадцять 32-бітних підключів: P_1, P_2, ..., P_{18}.

2. Чотири 32-бітних S-бокси з 256 входами кожний. Перший індекс означає номер S-боксу, другий індекс – номер входу:

$$S_{1,0}, S_{1,1}, \ldots S_{1,255};$$
$$S_{2,0}, S_{2,1}, \ldots S_{2,255};$$
$$S_{3,0}, S_{3,1}, \ldots S_{3,255};$$
$$S_{4,0}, S_{4,1}, \ldots S_{4,255}.$$

Метод, який використовується для обчислення цих підключів, описано далі.

Процес шифрування

Входом алгоритму є 64-бітний блок даних X, який ділиться на дві 32-бітні половини: X_L і X_R.

$$X_L = X_L \text{ XOR } P_i$$
$$X_R = F(X_L) \text{ XOR } X_R$$
$$\text{Swap } X_L \text{ and } X_R.$$

Структура функції F

Розділити X_L на чотири 8-бітних елементи A, B, C, D.

$F(X_L) = ((S_{1,A} + S_{2,B} \bmod 2^{32}) \text{ XOR } S_{3,C}) + S_{4,D} \bmod 2^{32}$.

Розшифрування відрізняється від шифрування тим, що P_i використовуються у зворотному порядку.

Генерування підключів

Підключі обчислюються з використанням самого алгоритму Blowfish:

1. Ініціалізувати перший *P*-масив і чотири *S*-бокси фіксованим рядком.

2. Виконати операцію XOR P_1 з першими 32 бітами ключа, операцію XOR P_2 із другими 32 бітами ключа і т. д. Повторювати цикл доти, доки увесь *P*-масив не буде побітово підсумований з усіма бітами ключа. Для коротких ключів виконується конкатенація ключа із самим собою.

3. Зашифрувати нульовий рядок алгоритмом Blowfish, використовуючи підключі, описані в пунктах (1) і (2).

4. Замінити P_1 і P_2 виходом, отриманим на кроці (3).

5. Зашифрувати вихід кроку (3), використовуючи алгоритм Blowfish з модифікованими підключами.

6. Замінити P_3 і P_4 виходом, отриманим на кроці (5).

7. Продовжити процес, замінюючи всі елементи *P*-масиву, а потім всі чотири *S*-бокси виходами відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібна 521 ітерація.

4.6. Огляд алгоритмів українського конкурсу на сертифікований симетричний криптоалгоритм

У грудні 2000 р. Департамент спецзв'язку та захисту інформації СБУ та Інститут кібернетики імені В. Глушкова оголосили відкритий конкурс симетричних криптоалгоритмів з метою розробити український стандарт криптографічного захисту інформації на заміну ГОСТ 28147-89, який діє в Україні й досі.

Організатори висували такі вимоги до криптолгоритму [61]:

1. ***Параметри***: криптоалгоритм повинен бути симетричним блоковим; розмір блоку даних – 128, 256, 512 бітів; розмір разового (сеансового) ключа – 128, 256, 512 бітів.

2. ***Принципи побудови***: здатність протистояти відомим методам криптографічного аналізу та мати запас стійкості з урахуванням тенденцій розвитку засобів електронної обчислювальної техніки та криптографічної науки; застосовані криптографічні перетворення повинні базуватись на надійній та прозорій математичній базі та не мати прихованих вразливостей; швидкодія криптоалгоритму повинна бути не меншою, ніж швидкодія існуючого державного стандарту шифрування.

3. ***Реалізація криптоалгоритму***: криптоалгоритм повинен дозволяти реалізацію на 32-х або 64-х розрядних процесорах; зазначені в криптоалгоритмі операції повинні мати ефективну програмну та апаратну реалізацію; необхідний для роботи об'єм пам'яті має враховувати можливість реалізації криптоалгоритму у мікропристроях; за можливості необхідно передбачити можливість паралельного виконання декількох операцій.

4. ***Ключова система***: криптоалгоритм повинен передбачати наявність довгострокового ключа; довжина синхропосилки – не менше 64 бітів.

5. У криптоалгоритмі повинні бути передбачені ***такі режими шифрування***: проста заміна; зчеплення блоків шифрованого тексту; зворотній зв'язок за входом; зворотній зв'язок за виходом; режим вироблення гами ("довгого циклу").

Розглянемо основні поняття алгоритмів, які брали участь у цьому конкурсі. Детальніше з ними можна ознайомитися у книзі [22].

4.6.1. Алгоритм RSB-32

Алгоритм *RSB*-32 (назва походить від англійських слів Round, Step, Block – Раунд, Крок, Блок) – ітераційний блоковий шифр зі змінними довжинами ключа, блоку та елементів блоків, над якими виконуються нелінійні операції заміни. На відміну від інших симетричних шифрів у *RSB* таблиці заміни змінюються залежно від стану секретного ключа. Алгебраїчні перетворення у *RSB* виконуються над кільцем лишків за деяким модулем *m*. Загальні параметри шифру такі:

Довжина раундового ключа – 32 біти.

Довжина загального (крокового) ключа: $r*64$, $r = 1, 2,...$

Кількість кроків шифрування:$s = 1, 2,...$

Кількість раундів шифрування:$r*s$.

Розмір блоку: 256, 512, 1024 біти.

Розмір елементів заміни: 8, 16, або 32 біти, тобто *RSB* – криптосистема, орієнтована на роботу з 8- ,16-, або 32-розрядними процесорами.

Загальний ключ (*Common Key*) у *RSB*-шифрі утворюється конкатенацією *r* 32-розрядних раундових ключів (Round Key). Узагальнена структурна схема *RSB*-алгоритму показана на рис. 4.16.

Дана схема показує процес перетворення текстів як для алгоритму зашифрування, так і розшифрування (природно з урахуванням виконання вимог зворотності перетворень).

Рис. 4.16. **Узагальнена структурна схема RSB-алгоритму**

Перед початком процедури зашифрування вхідний відкритий текст розбивається на блоки, розмір яких може бути рівним 256, 512 або 1024 біт. Якщо останній блок виявився менше обраного розміру, то він доповнюється пробілами до повного блоку. Назвемо такий текст *розширеним файлом*. Об'єм розширеного файла при зашифруванні не змінюється, тому об'єм шифротексту завжди буде кратним розміру блоку.

Як видно зі структурної схеми криптоалгоритму (див. рис. 4.16), спочатку виконуються послідовні перетворення усіх блоків розширеного файла з раундовим ключем RK_1, потім з ключем RK_2 і, нарешті, з ключем *RKr*. На цьому закінчується обробка тексту на першому кроці зашифрування. За умови, що кількість кроків шифрування *s* більше одиниці, відбувається часткове відновлення раундових ключів за рахунок циклічного зсуву загального ключа СК на сім розрядів праворуч.

Узагальнена структурна схема RSB алгоритму в режимі зашифрування наведена на рис. 4.17, у якому використані такі позначення:

RC (*Round Code*) – операції зашифрування тексту з раундовим ключем; RK_{ij} – *j*-й раундовий ключ на *i*-му кроці зашифрування.

Таким чином, алгоритм RSB (як і більшість сучасних симетричних блокових шифрів) складається з великої кількості перетворень, що повторюються – раундів.

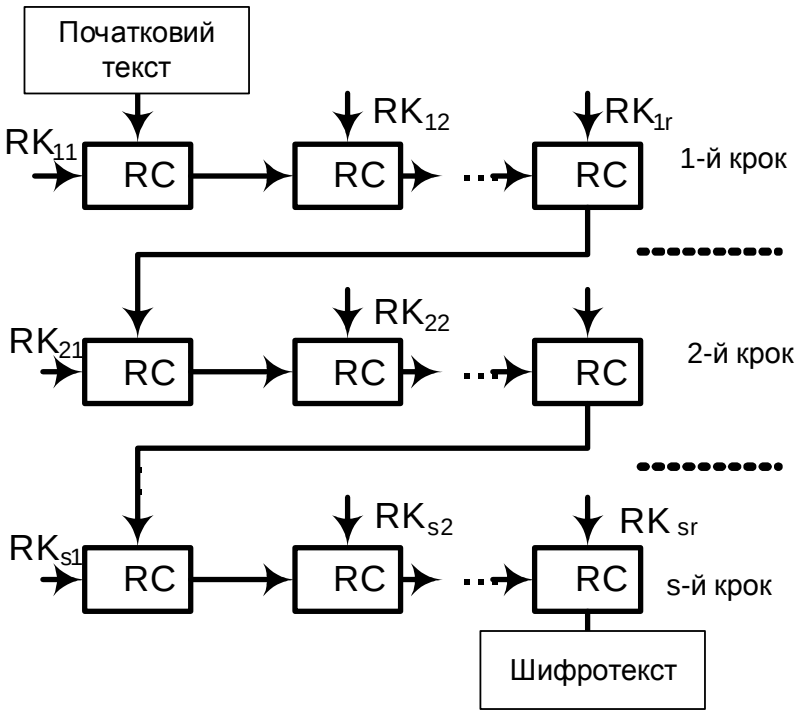


Рис. 4.17. **Узагальнена структурна схема алгоритму RSB**

Основні операції шифрування такі.

Операція циклічного зсуву (С). За допомогою даної операції здійснюється стохастичний циклічний зсув вліво (для операції зашифрування) *N*-розрядного блоку (*N* = 256, 512 або 1024 біти) на випадкове непарне число розрядів у межах від 1 до 255.

Операція перестановки (П) бітів у блоці виконується в такий спосіб.

Біт, розташований у розряді блоку під номером *x*, переміщається в розряд під номером *y*, який обчислюється за формулою:

$$y = (xM_{П})_m,$$

де $M_{п}$ – матриця перестановки, що отримується із загального масиву зворотних модульних матриць та випадкової адреси, що задається восьмирозрядною шиною сектора *П* раундового ключа.

Операція заміни (підстановки) елементів блоку (3) є зворотною нелінійною матричною операцією та служить не тільки для розсіювання елементів алфавіту вихідного тексту, але й для підвищення криптостійкості самого *RSB*-алгоритму до лінійного і диференціального криптоаналізу. Заміна елементів блоку *x* (у ролі яких виступають вісім 8-, 16- або 32-розрядних двійкових кодових комбінацій) на елементи *y* здійснюється перетворенням:

$$y = ((x \cdot M_3)_m \oplus B)_m,$$

у якому M_3 – матриця заміни четвертого порядку, що вибирається з бази за адресою, яка міститься в секторі 3 раундового ключа; B – чотирирозрядна функція (назвемо її бета-функцією), що надає нелінійності цьому перетворенню. Оператор $\oplus$ означає порозрядне сумування за модулем *m*.

Функції шифрування. У *RSB*-системі передбачені такі функції шифрування (табл. 4.15).

Таблиця 4.15

Функції шифрування

№ п/п	Назва функції	Позначення
1	Базова	*B*/*b*/*z*
2	Розширена	*R*/*b*/*z*
3	Стохастична	*S*/*b*/*z*
4	Гамування	*G*/*b*/*z*
5	Комбінована	*C*/*b*/*z*

У табл. 4.15 прийняті такі позначення: великі літери характеризують варіант шифрування; малими літерами *b* та *z* (які замінюються цифрами в реальній назві) позначаються розміри блоку та елемента заміни у блоці, відповідно. Далі наведені приклади повного позначення *RSB*-криптосистем: RSB-32S/512/32; RSB-32R/1024/8 і т. д.

Перейдемо до опису раундових функцій шифрування (F), скориставшись символами, якими позначаються такі перетворення:

C – стохастичний зсув блоку;

$\widetilde{C}$ – стохастичний зсув розширеного файла;

П – операція перестановки бітів у блоці;

З – проста заміна елементів блоку (вісім 8-, 16- або 32-розрядних слів), при якій до всіх елементів блоку застосовується та сама матриця заміни $M_З$;

$\widetilde{З}$ – кожний елемент блоку замінюється своєю індивідуальною матрицею заміни $M_З$;

(F)$\widetilde{С}$ – раундова функція *F* застосовується до всіх блоків розширеного файла, після чого файл піддається стохастичному зсуву.

Стохастичний зсув розширеного файла (операція $\widetilde{С}$) здійснюється наприкінці кожного кроку перетворення вліво (при зашифруванні) або вправо (при розшифруванні) на непарну кількість розрядів від 1 до 255. Операція $\widetilde{С}$ уведена для того, щоб уникнути жорсткої розбивки тексту на блоки, що ускладнює роботу криптоаналітика при дешифруванні зашифрованого тексту.

Отже, символічний опис раундових функцій (*F*) для варіантів зашифрування *B*, *R*, *S*, *G* і *C*, відповідно, мають вигляд:

$$FB = СП_1ЗП_2;$$

$$FR = СП_1\widetilde{З}П_2;$$

$$FS = (СП_1\widetilde{З}П_2)\widetilde{С};$$

$$FG = СП\widetilde{З}G;$$

$$FC = (СП\widetilde{З}G)\widetilde{С}.$$

Кожний варіант шифрування у криптосистемі *RSB* може бути реалізований у так званій матричній формі, що призводить до збільшення швидкості обробки текстів, але із втратою гнучкості зміни параметрів шифрування. У цьому випадку ідентифікатор системи доповнюється літерою *M*.

Ідея швидкісної матричної форми шифрування полягає в такому. Виберемо для приклада варіант шифрування *RSB*-32R/256/16/M з параметрами *r* = 6 та *s* = 8, тобто шифруються 256-розрядні блоки розширеного тексту 192-розрядним ключем (що складається із шести 32-розрядних раундових ключів). Розмір елементів заміни дорівнює 16 бітів, а весь процес шифрування завершується за 8 ітерацій.

Раундова функція зашифрування має вигляд: $FR = СП_1\widetilde{З}П_2$, тобто кожен блок розширеного тексту піддається спочатку стохастичному циклічному зсуву, потім виконується операція перестановки бітів у блоці

($П_1$), після цього для усіх 16-розрядних елементів блоку виконується операція заміни на індивідуальних матрицях і, нарешті, знову проводиться операція перестановки бітів у блоці ($П_2$). Коли закінчується обробка останнього блоку розширеного файла, аналогічна обробка блоків проводиться спочатку на другому, потім на третьому, четвертому, п'ятому і, нарешті, на шостому раундових ключах. Після цього раундові ключі поновлюються за рахунок циклічного зсуву вліво загального ключа на сім розрядів і процедура обробки блоків триває за вище описаною схемою протягом восьми кроків шифрування.

4.6.2. Алгоритм "Мухомор"

В основу алгоритму "Мухомор" покладено математичні та структурні ідеї схеми Лая – Мессі [22], яка також використовує деякі ідеї, реалізовані в шифрі IDEA NXT. Але в цьому шифрі, порівняно з IDEA NXT, використано ефективніші як з точки зору стійкості, так і з точки зору продуктивності, оригінальні базові криптографічні алгоритми – функції ускладнення базових перетворень *M*-64, *M*-128 та *M*-256, розширено параметри для більших довжин блоків та ключів, запропоновано відповідні реалізації, а також удосконалено схеми розгортання підключів. Усе це дозволило уникнути низки потенційних вразливостей та підвищити продуктивність шифру для апаратної, програмної та апаратно-програмної реалізацій.

Структура алгоритму

Вхідними даними алгоритму "Мухомор" є відкритий текст та ключ шифрування, вихідними – шифротекст.

Вхідні, вихідні блоки даних і ключ шифрування алгоритму "Мухомор" подаються у вигляді бітових (байтових) рядків довжиною 128, 256 і 512 бітів (відповідно 16, 32 і 64 байти).

Байтові рядки довжиною *n* байтів подані у такій формі:

$$B_0B_1B_2...B_{n-1}...,$$

де B_0 відповідає першим восьми бітам, B_1 – другим і так далі.

Операції лінійного розсіювання в алгоритмі "Мухомор" виконуються над полем Галуа GF(2^8), яке побудовано на незвідному поліномі:

$$m(x) = x^8 + x^4 + x^3 + x^2 + 1.$$

Алгоритм підтримує довжину блоку та ключа в 128, 256 і 512 бітів, причому розмір ключа шифрування не може бути менше розміру блоку. Дозволені комбінації розміру блоку та довжини ключа шифрування наведено в табл. 4.16.

Таблиця 4.16

Дозволені комбінації розміру блоку і довжини ключа

Розмір блоку, бітів	Довжина ключа, що підтримується, біт
128	128, 256, 512
256	256, 512
512	512

Алгоритм шифрування є ітеративною процедурою, що складається з попередньої та фінальної рандомізації а також раундового перетворення. Мінімальна кількість циклів шифрування (N_r) залежно від довжини блоку наведено в табл. 4.17.

Таблиця 4.17

Кількість циклів шифрування алгоритму (N_r)

Довжина блоку (бітів)	Кількість циклів шифрування
128	11
256	13
512	18

Процедура зашифрування. На вхід процедури подається відкритий текст і підключі шифрування (рис. 4.18).

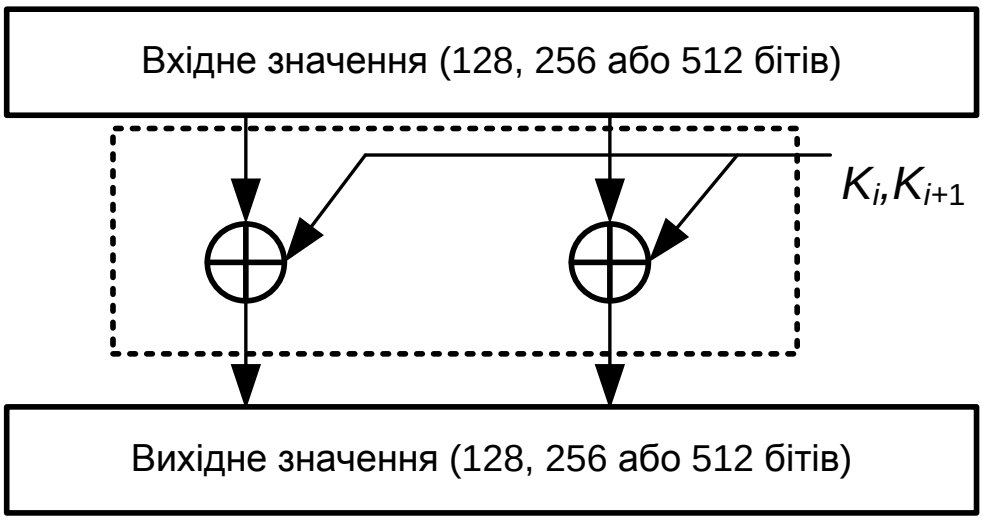


Рис. 4.18. **Схема початкової та фінальної рандомізації**

При цьому блок відкритого тексту або блок, оброблений цикловою функцією, задану кількість разів додається за модулем 2 з відповідними підключами (див. розділ схеми розгортання ключів, що відповідає довжині блоку та розміру ключа шифрування). Це і складає основу процедури початкової рандомізації (або, як кажуть, "відбілювання").

Схема циклового перетворення наведена на рис. 4.19.

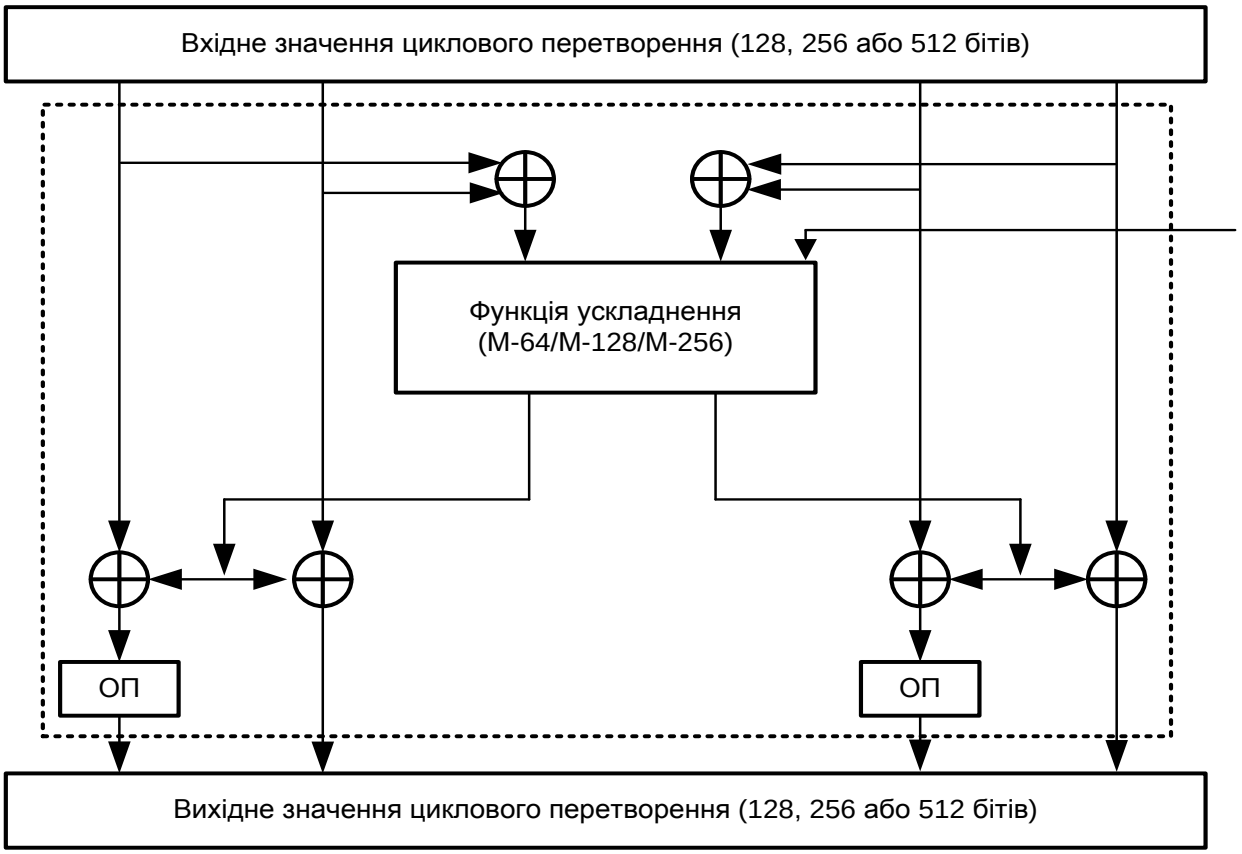


Рис. 4.19. **Блок-схема циклового перетворення**

Вхідний блок розбивається на 4 однакових підблоки, різниця між якими подається на вхід функції ускладнення (*M*-64, *M*-128 або *M*-256 для розмірів блоку 128, 256 і 512 бітів відповідно). Вихідне значення функції ускладнення додається за модулем 2 із вхідними значеннями, після чого перший і третій підблоки піддаються операції ортогонального перетворення (ОП). У процесі шифрування виконується циклове перетворення за базовою схемою Лая – Мессі.

Функція ускладнення. Основним перетворенням у функції ускладнення є SL-перетворення, що виконує перетворення 32-бітових блоків даних. Схема SL-перетворення наведена на рис. 4.20.

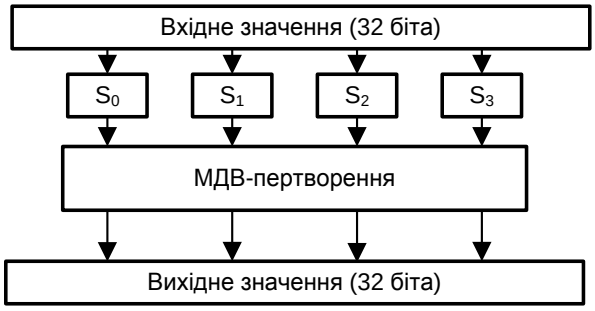


Рис. 4.20. **Схема SL-перетворення**

При SL-перетворенні вхідне 32-бітове значення ділиться на 4 байти, кожний з яких заміняється відповідно до заданої таблиці підстановки. Після заміни 4 байти (a_0, a_1, a_2, a_3) подаються на вхід МДВ-перетворення, що виконує матричне множення такого вигляду:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 02 \cdot a_0 \oplus 03 \cdot a_1 \oplus 01 \cdot a_2 \oplus 01 \cdot a_3 \\ 01 \cdot a_0 \oplus 02 \cdot a_1 \oplus 03 \cdot a_2 \oplus 01 \cdot a_3 \\ 01 \cdot a_0 \oplus 01 \cdot a_1 \oplus 02 \cdot a_2 \oplus 03 \cdot a_3 \\ 03 \cdot a_0 \oplus 01 \cdot a_1 \oplus 01 \cdot a_2 \oplus 02 \cdot a_3 \end{bmatrix}.$$

Матриця МДВ-перетворення шифру "Мухомор" збігається з матрицею алгоритму Rijndael/AES, але операція множення виконується над іншим незвідним поліномом. Вихідний 32-бітовий вектор МДВ-перетворення (b_0, b_1, b_2, b_3) і є вихідним значенням SL-перетворення.

Функції ускладнення M-64, M-128 та M-256 влаштовані інакше. З деталями їх внутрішньої організації можна ознайомитися у [22].

Ортогональне перетворення (ОП) виконується на виході раундового перетворення для обробки першого та третього підблоків. Схема ОП наведена на рис. 4.21.

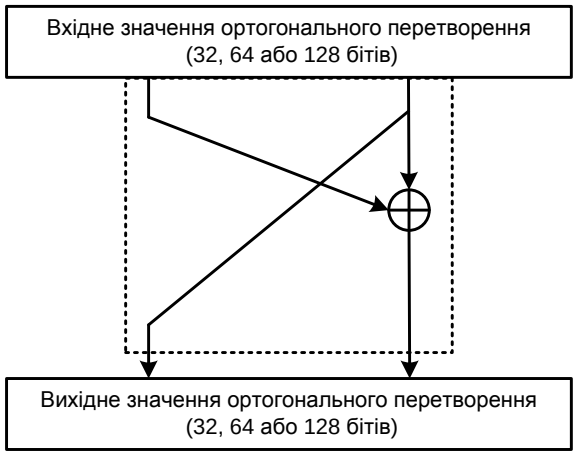


Рис. 4.21. **Схема ортогонального перетворення**

Блок, розміром 128, 256 або 512 бітів, залежно від довжини блоку відкритого тексту, що подається на вхід ОП, розбивається на дві половини, з яких права на виході стає лівою, а ліва на виході формується як сума за модулем 2 лівої та правої половин на вході.

Відмінності алгоритму розшифрування. Алгоритм розшифрування є оберненим до алгоритму зашифрування. На вхід подаються блоки шифротексту і підключі розшифрування. На початку розшифрування виконується зняття фінальної рандомізації, після чого отриманий блок даних задану кількість разів обробляється цикловою функцією, потім виконується зняття рандомізації з блоку відкритого тексту. В результаті отримується блок відкритого тексту. Процес розшифрування збігається з процесом зашифруванням за винятком таких деталей: підключі (у тому числі початкової та фінальної рандомізації) подаються у зворотному порядку; замість операції ортогонального перетворення використовується обернена до неї операція $ОП^{-1}$.

4.6.3. Алгоритм ADE

Основну увагу при розробці ADE приділялося можливості динамічно керувати процесом лінійного розсіювання й нелінійної заміни в ході криптографічного перетворення інформації. Із цією метою в базову структуру алгоритму AES уведено блоки (криптопримітиви) лінійного розсіювання й нелінійних замін, що динамічно змінюються, а правила функціонування їх задаються значеннями раундових ключів. Властивості уведених примітивів криптоалгоритму ADE не поступаються криптопримітивам алгоритму AES, а за рахунок їх динамічної зміни на кожному раунді ітеративної обробки вдається динамічно керувати процесом перетворення інформації. Це, з одного боку, не погіршує (порівняно з AES) стійкість ADE до відомих методів криптоаналізу, з іншого боку, істотно ускладнює процес формування алгебраїчних рівнянь, що аналітично зв’язують стан відкритого тексту, шифротексту і ключа.

Структура алгоритму

Операції у шифрі ADE, як і в шифрі "Мухомор", визначені у полі Галуа над незвідним многочленом: $m(x) = x^8 + x^4 + x^3 + x^2 + 1$.

За своєю конструкцією шифр ADE найбільш близький до AES. Раундове перетворення шифру ADE, як і AES, не використовує структуру Фейстеля. Замість цього воно складається з 3-х окремих інвертованих "однорідних" перетворень, які називаються шарами. Під "однорідністю" мається на увазі те, що кожний біт стану обробляється подібним шляхом.

Обґрунтування структурної схеми алгоритму шифрування. ADE, як і AES, є ітеративним блоковим шифром з різною довжиною блоку та різною довжиною ключа.

Алгоритм ADE побудований за класичною схемою підстановок-перестановок та використовує всі переваги алгоритму AES.

З деяким спрощенням можна вважати, що ADE зводиться до відповідних блоків алгоритму AES, тобто при фіксуванні деяких системних параметрів алгоритм ADE легко трансформується в AES. Структурна схема алгоритму ADE у загальному виді представлена на рис. 4.22.

На схемі позначені:

"SubByte" – блок заміни байтів, що залежить від раундового ключа;

"ShiftRow" – блок функціонального перетворення (на основі циклічного зсуву), що залежить від раундового ключа;

"MixColumn" – блок лінійного перетворення (аналог перестановки), що функціонує під управлінням раундового ключа;

"AddRoundKey" – блок функціонального перетворення (додавання з ключем за модулем 2).

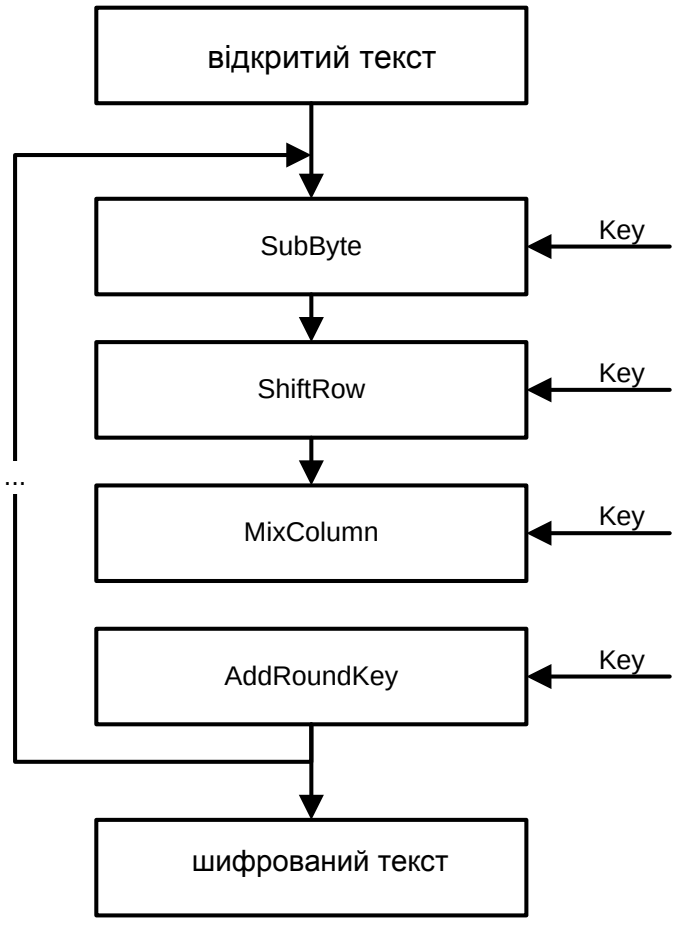


Рис. 4.22. **Структурна схема алгоритму шифрування ADE**

Розширення ключа

Раундові ключі формуються із ключа шифрування за допомогою процедури формування ключів, аналогічної до такої в AES. Вона складається з двох компонентів: розширення ключа і вибір раундового ключа.

Основний принцип полягає в такому:

загальна кількість бітів раундового ключа дорівнює довжині блоку, помноженій на кількість раундів плюс 1 (наприклад, для довжини блоку в 128 бітів і 10 раундів необхідно 1 408 раундових ключів);

ключ шифру розширюється в так званий розширений ключ;

раундові ключі отримуються з розширеного ключа в такий спосіб: перший раундовий ключ складається з перших N_b слів, другий – з наступних N_b слів, і т. д.

4.6.4. Алгоритм "Калина"

Алгоритм "Калина" подібний до Rijndael, однак він має деякі вдосконалення з метою збільшення криптостійкості.

Основні відмінності "Калини" від Rijndael полягають в такому:

збільшена кількість циклів шифрування; використано додавання за модулем 2^{32} і за модулем 2 для введення ключової інформації для покращення захисту від алгебраїчних атак, лінійного та диференціального криптоаналізів, інтерполяційної атаки тощо; використано 8 блоків нелінійного перетворення (S-блоків) замість одного для додаткового захисту від алгебраїчних атак, покращення властивостей розсіювання шифру; використання випадково сформованих S-блоків, відібраних за критеріями стійкості до диференціального, лінійного криптоаналізу та степені нелінійності булевих функцій; принципово нова схема розгортання ключа для захисту від всіх відомих атак на схеми вироблення підключів; досить висока продуктивність.

Усі модифікації спрямовані на збільшення стійкості та перекриття потенційних вразливостей Rijndael, що виявлено останніми роками.

Структура алгоритму

Структура алгоритму аналогічна структурі Rijndael, забезпечує гарне розсіювання та перемішування, відмінні криптографічні та статистичні властивості, високу продуктивність.

У цьому алгоритмі використовуються цикли перетворення 2-х типів: із введенням ключа за модулем 2 і за модулем 2^{32}, що покращує нелінійність шифру, вводить додаткові залежності між результуючими значеннями.

Таблиці підстановки (вузли нелінійного перетворення).

У шифрі "Калина" застосовуються випадкові таблиці заміни, які відбираються за критеріями стійкості до диференціального та лінійного криптоаналізу та степені нелінійності булевих функцій.

Використання випадкових S-блоків дозволяє значно ускладнити знаходження залежностей між елементами блоків заміни та змушує перейти до ймовірнісних рівнянь опису підстановки в алгебраїчних атаках, причому застосування 8 підстановок додатково знижує можливість побудови та ймовірність знаходження правильного розв’язку для такої системи рівнянь.

Можливе використання довільних S-блоків (відмінних від наведених у специфікації), що задовольняють дані вимоги, при цьому всі криптографічні властивості та стійкість шифру зберігаються.

Крім того, можливе використання S-блоків як додаткового секретного ключа.

Блок лінійного перетворення.

У якості лінійного перетворення використовується МДР-перетворення, що добре зарекомендувало себе як з точки зору розсіювання, так і з точки зору продуктивності на сучасних 64-бітних процесорах. Так, вже на двох раундах шифрування забезпечується повна залежність кожного біта від вхідних даних. Ці характеристики кращі, ніж у Rijndael 256/256, де потрібна більша кількість циклів для поширення різниці на весь блок.

До недоліків такого підходу слід віднести менш ефективну реалізацію на 32-бітних процесорах.

Операція розгортання ключа

Оскільки процедура розгортання ключа Rijndael має певні значні недоліки [21], то в шифрі "Калина" розроблено зовсім нову процедуру вироблення раундових підключів. Вона задовольняє такі вимоги: нелінійність залежності кожного біта підключа від кожного біта головного ключа повинна забезпечити усі необхідні лавинні властивості та відсутність "проміжних точок"; забезпечує стійкість до всіх відомих атак на схеми вироблення підключів; не має слабких ключів, які призводять до погіршення криптографічних властивостей шифру; відновлення майстер-ключа за одним або декількома підключами неможливе; процедура розгортання дозволяє просту реалізацію з використанням циклового перетворення шифру; обчислювальна складність генерації всіх підключів не повинна перевищувати складності однієї операції шифрування;

можлива генерація підключів у будь-якому порядку (як для зашифрування, так і для розшифрування).

4.6.5. Алгоритм "Лабіринт"

Алгоритм "Лабіринт" є ітеративним шифром, тобто основу його процедури шифрування становить циклове перетворення, що повторюється задану кількість разів (позначимо число циклів шифрування символом *r*). Кожен цикл складається із двох абсолютно ідентичних ітерацій, однак, з огляду на те, що для зашифрування одного блоку потрібно, як мінімум, дві ітерації, поняття циклу не співпадає з поняттям ітерації. Крім циклового перетворення процедура шифрування включає початкове (IT) і кінцеве (FT) перетворення. Властивість інволютивності шифру досягається за рахунок застосування класичної конструкції напівблокової сітки Фейстеля (рис. 4.23).

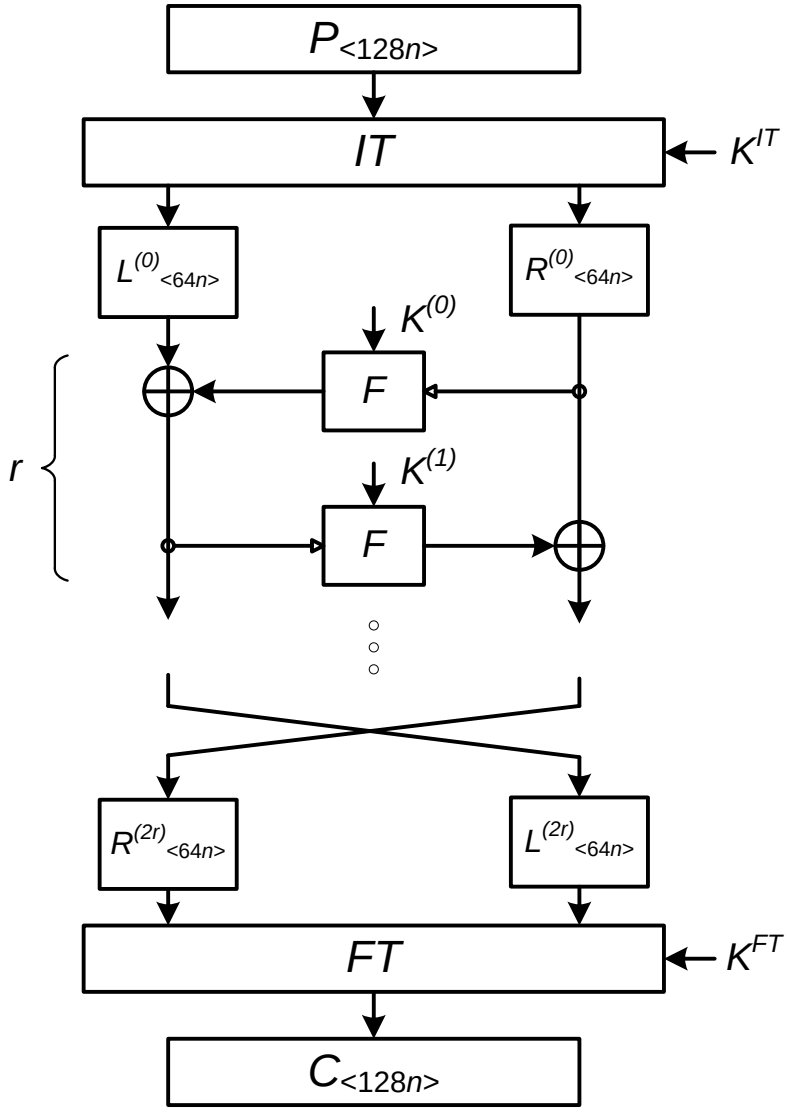


Рис. 4.23. **Структура процедури шифрування EF**

Алгоритм "Лабіринт", незалежно від довжини блоку й ключа, використовує фіксовану кількість ітерацій шифрування – 16 (або, інакше кажучи, 8 циклів, тобто $r = 8$).

Процедура шифрування EF складається із трьох кроків:

1. Вихідний блок даних P (довжиною $128n$ бітів) обробляється початковим (IT) перетворенням на ключі K^{IT}.

2. Результат 1-го кроку розбивається на два напівблоки довжиною по $64n$ біти: лівий L ("старший") і правий R ("молодший"). Отримана пара напівблоків перетвориться у шифротекст за 16 ітерацій (8 циклів) сітки Фейстеля. Всі ітерації повністю ідентичні й побудовані на базі загального нелінійного перетворення – F-функції, керованої ключем ітерації $K^{(i)}$.

3. Два напівблоки L і R, отримані в результаті 8-циклового ітеративного перетворення, міняються місцями й обробляються кінцевим (FT) перетворенням на ключі K^{FT}. Отриманий після FT перетворення двійковий вектор C (довжиною $128n$ бітів) є криптограмою.

Базова F-функція.

Циклова F-функція складається з чотирьох елементарних перетворень:

додавання слів напівблоку зі словами ключа ітерації за модулем 264;

фіксована байтова перестановка P;

нелінійна підстановка байтів напівблоку;

фіксоване перетворення лінійного перемішування.

Докладно ці перетворення розглянуті у [22].

Процедура розгортання ключа.

Процедура розгортання ключа складається із двох етапів:

ініціалізація буфера робочого ключа на основі ключа шифрування;

вибірка підключена з буфера робочого ключа.

Процедура ініціалізації полягає в завантаженні вихідного ключа в буфер робочого ключа та доповнення цього ключа до повного заповнення буфера ключовим матеріалом, сформованим з вихідного ключа. Для формування "додаткового" ключового матеріалу використовується базова F-функція шифратора, на базі якої побудований простий криптографічний генератор псевдовипадкових послідовностей – вихідний ключовий матеріал циклічно зашифровується в режимі "зв’язування шифроблоків" (CBC-режим).

Однією з переваг такої схеми розгортання ключа і всієї конструкції шифру в цілому є можливість застосування матеріалу "розгорнутого" ключа як для зашифрування, так і розшифрування.

Алгоритм "Лабіринт" задовольняє вимоги максимального класу стійкості, відповідно до умов проекту NESSIE (довжина блоку не менше 128 бітів, довжина ключа не менш 256 бітів). Структура алгоритму "Лабіринт" дозволяє ефективно його реалізувати як на 64-бітних, так і на 32-бітних мікропроцесорах. З огляду на останні досягнення у сфері спеціалізованих мікроконтролерів, а саме тенденцію переведення смарт-карт на 32-бітні архітектури та оснащення їх криптоприскорювачем, даний алгоритм може бути ефективно реалізований у тому числі і на смарт-картах. З іншого боку, обрана конструкція алгоритму забезпечує можливість ефективної апаратної реалізації, при цьому можливо істотне зниження витрат енергонезалежної пам'яті за рахунок апаратної реалізації операції множення в полі Галуа $GF(2^8)$ і зниження обчислювальної складності за рахунок відсутності операцій фіксованого зсуву. Крім того, алгоритм побудований на базі класичного сітки Фейстеля, що забезпечує інволютивність шифруючого перетворення, і можливість використання загального апаратного рішення для процедур зашифрування та розшифрування.

Конструкція циклового перетворення забезпечує можливість збільшення довжини блоку відкритого тексту без збільшення кількості циклів шифрування, а конструкція схеми розгортання ключа забезпечує незалежність кількості циклів шифрування від довжини вихідного ключа. Ці дві властивості алгоритму "Лабіринт" дозволяє одержати практично постійну продуктивність шифру при різних значеннях показників стійкості – довжинах блоку й ключа.

Можливість довільного вибору параметрів вузла заміни (S-блоку) і лінійного MBN-перетворення, у межах строго визначених обмежень до їх криптографічних показників, дозволяє, з одного боку, усунути потенційну небезпеку уведення "закладок" з боку автора алгоритму, а з іншого боку – забезпечує можливість введення додаткового секретного параметра, якщо параметри зазначених елементів становлять довгостроковий секретний ключ. Такі довгострокові ключі можуть формуватись за спеціальною методикою, що забезпечує контроль, не тільки мінімальних, а й додаткових критеріїв "фільтрації". Формування довгострокових ключів для державних організацій може виконуватися відповідним уповноваженим органом і поставлятися користувачам у встановленому порядку.

Алгоритм "Лабіринт" може використовуватися в кожному з п'яти стандартних режимів застосування блокових шифрів, а також в посиле-

них режимах потокового шифрування. Саме в посилених режимах застосування блокових шифрів сьогодні можуть використовуватися більші довжини блоку, 256 або навіть 512 бітів.

4.7. Основні режими роботи DES (ECB, CBC, CFB, OFB, режим генерування імітовставки)

Для будь-якого симетричного блокового алгоритму шифрування визначено чотири режими виконання.

ECB – Electronic Codebook – кожний блок з 64 бітів незашифрованого тексту шифрується незалежно від інших блоків із застосуванням алгоритму симетричного шифрування. Типові додатки – безпечна передача одиночних значень (наприклад, криптографічного ключа).

CBC – Cipher Block Chaining – вхід криптографічного алгоритму є результатом застосування операції XOR до наступного блоку незашифрованого тексту і попереднього блоку зашифрованого тексту. Типові додатки – загальна блок-орієнтована передача, автентифікація.

CFB – Cipher Feedback – при кожній ітерації алгоритму обробляється J бітів вхідного значення. Попередній зашифрований блок використовується як вхід в алгоритм; до J бітів виходу алгоритму і наступного незашифрованого блоку з J бітів застосовується операція XOR, результатом якої є наступний зашифрований блок з J бітів. Типові додатки – потокоорієнтована передача, автентифікація.

OFB – Output Feedback – аналогічний CFB, за винятком того, що на вхід алгоритму при шифруванні наступного блоку подається результат шифрування попереднього блоку; тільки після цього виконується операція XOR із черговими J бітами незашифрованого тексту. Типові додатки – потокоорієнтована передача за зашумленим каналом (наприклад, супутниковий зв’язок).

4.7.1. Шифрування в режимі Electronic Code Book, ECB

Цей режим є найпростішим режимом, при якому незашифрований текст обробляється послідовно, блок за блоком. Кожний блок шифрується, використовуючи той самий ключ. Якщо повідомлення довше, ніж довжина блоку відповідного алгоритму, то воно розбивається на блоки відповідної довжини, причому останній блок доповнюється, якщо буде

потреба, фіксованими значеннями. При використанні цього режиму однакові незашифровані блоки будуть перетворені в однакові зашифровані блоки.

ECB-режим ідеальний для невеликої кількості даних, наприклад, для шифрування ключа сесії.

Далі будемо використовувати такі позначення. Символ E будемо використовувати для позначення операції шифрування *n*-бітного блокового шифру, де n – кількість бітів у блоках відкритого й закритого текстів, а символом D будемо позначати операцію дешифрування для того ж шифру.

Перетворення відкритого тексту Ру закритий текст C_i здійснюється за формулою:

$$C = E_k(P),$$

де C – *n*-бітний блок шифортексту;

K – секретний ключ для блокового шифру;

P – *n*-бітний блок відкритого тексту.

Аналогічним чином маємо зворотне перетворення:

$$P = D_k(C),$$

також справедливе співвідношення виду:

$$P = D_k(E_k(P)).$$

Режим електронної кодової книги. Схема режиму подана на рис. 4.24.

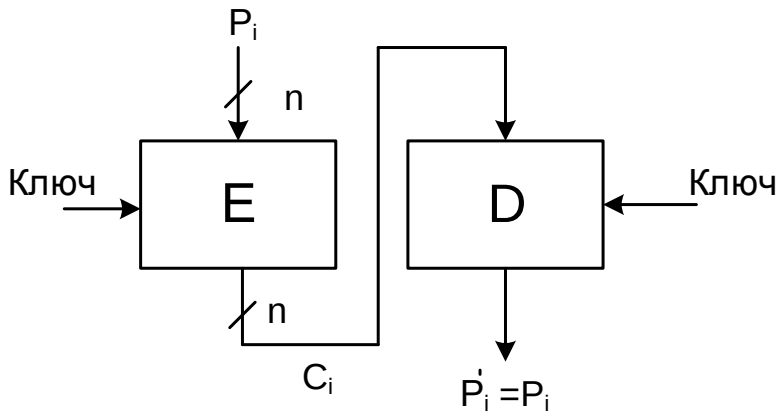


Рис. 4.24. **Режим електронної кодової книги**

На вхід алгоритму надходить *k*-бітний ключ K і *n*-бітні блоки відкритого тексту P = p_1, p_2, …, p_t... На виході формуються *n*-бітні блоки шифротексту C = c_1, c_2, …, c_t...

Шифрування: для $1 \le i \le t$:

$$c_i = E_k(p_i).$$

Дешифрування: для $1 \le i \le t$: $p_i = D_k(c_i)$.

Зазначимо такі властивості *ECB* режиму:

1. Шифрування ідентичних відкритих текстів. Блокові шифри в такому режимі не забезпечують приховання шаблонів даних – ідентичні блоки відкритого тексту (при тому самому ключі) перетворяться в ідентичний шифротекст.

2. Вплив зчеплення. Блоки шифруються незалежно один від одного. Переупорядкування блоків шифротексту призведе до переупорядкування блоків відкритого тексту. Оскільки блоки шифротексту незалежні, навмисна підстановка *ECB*-блоків (наприклад, вставка блоку, що часто зустрічається) не має впливу на дешифрування суміжних блоків.

3. Розмноження помилок. Один і більше помилкових бітів в одному блоці шифротексту впливають на дешифрування тільки цього блоку. Результат дешифрування блоку з помилками є випадковою величиною, близько 50 % відновлювальних бітів відкритого тексту.

4. Істотним недоліком ECB є те, що той самий блок незашифрованого тексту, що з'являється більше одного разу в повідомленні, завжди має той самий зашифрований вигляд. Внаслідок цього для більших повідомлень ECB-режим вважається небезпечним. Якщо повідомлення має багато однакових блоків, то при криптоаналізі ця закономірність буде виявлена.

Із перерахованих особливостей випливає і використання *ECB*-режиму. *ECB*-режим не рекомендується використовувати для шифрування повідомлення більше, ніж один блок, або якщо повторно використовуються ключі для шифрування більше, ніж одного одноблокового повідомлення. Таємність ключа може бути забезпечена шляхом включення випадкових додаткових бітів у кожний блок.

4.7.2. Шифрування в режимі Cipher Block Changing, CBC

Для подолання недоліків ECB використовують спосіб, при якому однакові незашифровані блоки перетворюються в різні зашифровані.

Для цього на вхід алгоритму подається результат застосування операції XOR до поточного незашифрованого блоку і попереднього зашифрованого блоку.

Режим зчеплення блоків шифру включає використання *n*-розрядного вектора ініціалізації, позначуваного IV. У схему додається регістр або буфер зворотного зв'язку, у який спочатку записується значення IV, а потім – наступні значення блоків шифротексту. На рис. 4.25 показана схема *CBC*. Для одержання першого блоку зашифрованого повідомлення використовується вектор ініціалізації (IV), для якого виконується операція XOR з першим блоком незашифрованого повідомлення. При розшифруванні для IV виконується операція XOR з виходом алгоритму розшифрування для одержання першого блоку незашифрованого тексту.

IV повинен бути відомий як відправнику, так і одержувачу. Для максимальної безпеки IV повинен бути захищений так само, як і ключ.

На вхід алгоритму надходить k-бітний ключ K, *n*-бітний IV, послідовність n-бітних блоків відкритого тексту $P = p_1, p_2, \ldots, p_i, \ldots, p_t$...

Послідовність блоків шифротексту $C = c_1, c_2, \ldots, c_t$ обчислюється за правилом: $c_i = E_k(p_i \oplus c_{i-1})$; $i = 1,\ldots,t; c_0 = IV$.

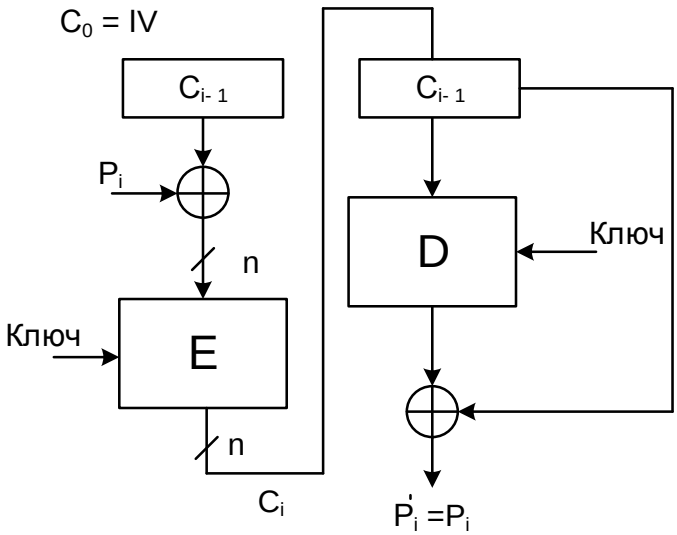


Рис. 4.25. **Режим зчеплення блоків шифру**

Розшифрування здійснюється як: $p_i = c_i - 1 \oplus D_k(c_i)$; $i = 1,\ldots,t$; $c_0 = IV$.

Властивості режиму:

1. Шифрування ідентичних відкритих текстів. Ідентичні блоки шифротексту формуються тоді, коли шифрується той самий відкритий текст на

тих самих ключах і векторі ініціалізації. Зміна вектора ініціалізації, ключа або першого блоку відкритого тексту (наприклад, використовуючи лічильник або випадкову величину) призводить до різних шифротекстів.

2. Вплив зчеплення. Механізм зчеплення приводить до того, що шифротекст c_i залежить від p_i і всіх попередніх блоків відкритого тексту. У цьому випадку вхідна залежність попередніх блоків утримується в значенні попереднього блоку шифротексту. Отже, перевпорядкування порядку проходження блоків шифротексту впливає на розшифрування. Для того, щоб правильно розшифрувати блок шифротексту, необхідно мати правильний попередній блок шифротексту.

3. Розмноження помилок. Одинична бітова помилка в блоці шифротексту c_i впливає на дешифрування як мінімум двох блоків c_i і c_{i+1} (оскільки p_i залежить від c_i і c_{i+1}). Блок p'_i відновлений із c_i звичайно є повністю випадковим (50 % бітів помилкові), у той час, як відновлений відкритий текст p'_{i+1} матиме неправильних бітів рівно стільки, скільки їх має блок c_i. Таким чином, противник може викликати передбачувані зміни бітів у p'_{i+1}, змінюючи відповідні біти блоку c_i.

4. Відновлення помилок. CBC-режим є реалізацією самосинхронізуючого шифру або шифрування з автоключем за шифротекстом в тому розумінні, що якщо має місце помилка в блоці c_i (включаючи втрату одного або більше вхідних блоків), але немає помилок в c_{i+1}, c_{i+2}, то можливе коректне дешифрування p_{i+2}.

З погляду практичного використання блокових шифрів у даному режимі необхідно також ураховувати такі рекомендації:

1. Розмноження помилок при шифруванні. Хоча розшифрування в CBC-режимі дозволяє відновити відкритий текст із помилок у блоках шифротексту, модифікація блоку відкритого тексту p_i під час шифрування змінює всі наступні блоки шифротексту. Це ускладнює (якщо не виключає) застосовуваність режимів зчеплення для додатків, що вимагають випадкового доступу (читання/запису) до шифрованих даних. Альтернативою є режим ECB.

2. Самосинхронізація і групування помилок. Незважаючи на те, що самосинхронізація можлива при наявності помилок у бітах, самосинхронізація при "загублених" бітах, що є причиною помилки визначення меж блоку (помилка цілісності кадру), неможлива ні в CBC-режимі, ні в інших режимах.

3. Цілісність IV в CBC. Незважаючи на те, що вектор ініціалізації IV в CBC-режимі не є секретним, повинна бути забезпечена цілісність, оскільки навмисна модифікація IV дозволяє противнику виконувати передбачувані зміни бітів у відновлюваному першому блоці відкритого тексту. Використання секретних IV є одним з методів запобігання таких модифікацій. Однак якщо потрібно забезпечити цілісність повідомлення, повинні використовуватися інші механізми, тому що механізми шифрування зазвичай гарантують тільки конфіденційність.

4.7.3. Шифрування в режимі Electronic Feedback, CFB

Блоковий алгоритм призначений для шифрування блоків встановленої довжини. Однак можна перетворити блоковий алгоритм у потоковий алгоритм шифрування, використовуючи два останніх режими. Потоковий алгоритм шифрування усуває необхідність розбивати повідомлення на цілу кількість блоків досить великої довжини, отже, він може працювати в реальному часі. Таким чином, якщо передається потік символів, кожний символ може шифруватися і передаватись відразу, з використанням символьно-орієнтованого режиму блокового алгоритму шифрування.

У CFB-режимі на вхід подається *k*-бітний ключ K, *n*-бітний IV, послідовність *r*-бітних блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$, де $1 \le r \le n$. На виході формуються *r*-бітні блоки шифротексту $C = c_1, c_2, \ldots, c_u$...

Шифрування здійснюється таким способом. Спочатку здійснюється запис у буфер зворотного зв'язку, записується значення IV– I_1=IV. Потім для всіх $1 \le i \le u$ виконуються такі операції:

1. Обчислення функції шифрування:

$$O_i = E_k(I_i).$$

2. Вибірка r крайніх ліворуч бітів результату шифрування O_i:

$$t_i = Oi \sim r,$$

де O_i~r означає оперцію вибірки.

3. Формування і передача r-бітного блоку шифротексту c_i:

$$c_i = p_i \oplus t_i.$$

4. Формування змінного зворотного зв'язку із блоку шифротексту, відповідно до виразу:

$$I_{i+1} = 2rI_i \oplus c_i \bmod 2^n.$$

Ця операція еквівалентна зсуву значення буферного регістра I_i на r-бітів вправо і запису в кінець регістра *r*-бітного блоку c_i. Розшифрування виконується відповідно:

$$P_i = c_i \oplus t_i; \quad i = 1,\ldots,u,$$

де t_i, O_i і I_i обчислюються відповідно до виразів, наведених вище і I_1= IV.

На рис. 4.26 подана схема CFB-режиму.

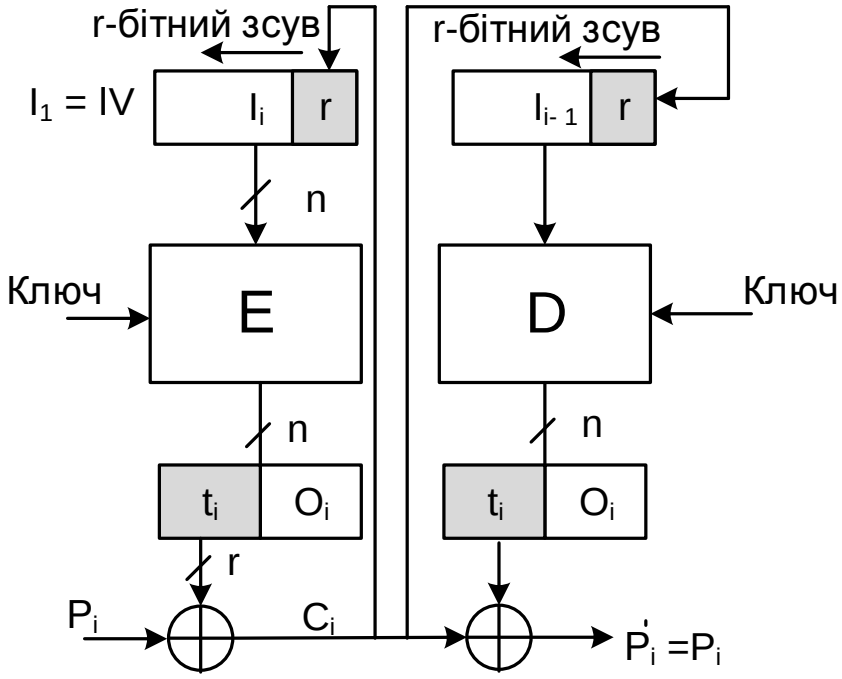


Рис. 4.26. **Режим зворотного зв'язку за шифротекстом**

Зазначимо, що фактично необхідно мати ще й буфер зворотного зв'язку для зберігання c_{i-1}, що звичайно вибирається рівним n.

CFB-режим має такі властивості:

1. Шифрування ідентичних відкритих текстів. Як і в CBC-режимі шифрування, зміна IV приводить до різних результатів шифрування того самого відкритого тексту. Значення IV не обов'язково повинне бути секретним, хоча непередбачуваність IV є бажаною в деяких додатках.

2. Вплив зчеплення. Аналогічно з CBC-режимом шифрування механізм зчеплення приводить до того, що блок шифротексту c_i залежить від двох блоків відкритого тексту p_i і p_{i-1}. Отже, перевпорядкування блоків шифротексту впливає на дешифрування. Правильне дешифрування правильних блоків шифротексту вимагає одержання $\lceil n/k \rceil$ правильних попередніх блоків шифротексту для того, щоб регістр зворотного зв'язку містив правильне значення.

3. Розмноження помилок. Одна або більше помилок у будь-якому одиничному *r*-бітному блоці шифротексту c_i впливає на дешифрування наступних $\lceil n/k \rceil$ блоків шифротексту, тобто поки не буде закінчена обробка n бітів шифротексту, після яких помилковий блок c_i буде висунутий з регістра зворотного зв'язку. Відновлений відкритий текст p_i' буде відрізнятися від p_i точно в тих же позиціях, у яких відбулися помилки в c_i. Інші некоректно відновлені блоки відкритого тексту звичайно будуть випадковими векторами, тобто матимуть 50 % помилкових бітів. У такий спосіб противник може здійснити передбачувані зміни бітів у p_i шляхом зміни відповідних бітів у c_i.

4. Відновлення помилок. CFB-режим, як і CBC-режим, що самосинхронізується, але вимагає одержання $\lceil n/k \rceil$ блоків шифротексту для відновлення синхронізації.

5. Продуктивність. Для $r < n$ продуктивність зменшується в n/r разів (порівняно з CBC), тому що кожне шифрування E забезпечує формування тільки r бітів шифротексту на виході.

Розглянутий режим є режимом роботи з *r*-бітними вхідними символами і *r*-бітним зворотним зв'язком. Цей режим схожий з режимом CFB, стандартизованим для DES (NBS FIPS Pub 81 і ANSI X3.106) і позначається як CFB *r*-бітний символ/*r*-бітний зворотний зв'язок. Стандарт ISO/IEC 10118:1991 визначає більш загальний CFB-режим. При використанні цього режиму може здійснюватися обробка *j*-бітних блоків відкритого тексту при *r*-бітному зворотному зв'язку, причому $j \le r$. Таким чином, для реалізації режиму необхідно вибрати величину зворотного зв'язку r ($1 \le r \le n$) і величину блоку відкритого тексту j($1 \le j \le r$).

На вхід надходить *k*-бітний ключ K, n-бітний IV, послідовність j-бітних блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$. На виході алгоритму формуються *j*-розрядні блоки шифротексту $C = c_1, c_2, \ldots, c_u$.

Шифрування здійснюється відповідно до таких операцій.

Перед шифруванням здійснюється запис у регістр зворотного зв'язку значення вектора ініціалізації $I_1 = IV$.

Для всіх $1 \le i \le u$ виконуються такі операції:

1. Обчислення функції шифрування:

$$O_i = E_k(I_i).$$

2. Вибірка j лівих бітів величини O_i:

$$ti = Oi \sim j.$$

3. Формування блоку шифротексту:

$$ci = pi + ti.$$

4. Формування змінного зворотного зв’язку із блоку шифротексту c_i шляхом його доповнення ліворуч r-j одиницями:

$$Fi = F(1)(r-j) || ci,$$

де $F^{(1)}(r-j)$ – блок з r-j одиниць;

$||$ – операція конкатенції.

5. Зсув регістра I_i на *r*-бітів вліво і запис F_i у регістр:

$$Ii+1 = 2rIi+Fi \bmod 2n.$$

Зазначимо, що для i = u операції 4 і 5 не виконуються.

Розшифрування виконується в аналогічному порядку, за винятком того, що: Pi = ci⊕ ti.

На рис. 4.27 подано схематичне зображення функціонування алгоритму шифрування в CFB-режимі за ISO/IEC 10118:1991.

І, нарешті, у другій редакції стандарту ISO/IEC 10116:1997 була запропонована узагальнена версія CFB-режиму. Стандартизований режим забезпечує "конвеєрну обробку даних".

У раніше розглянутих версіях CFB-режиму результат шифрування одного блоку відкритого тексту є входом для шифрування наступного блоку. Це означає, що неможливі "конвеєрні" обчислення, тобто неможливо приступити до шифрування одного блоку до закінчення обробки попереднього блоку. Щоб уникнути цього недоліку, у нову версію CFB-режиму уведений уже *m*-розрядний буфер зворотного зв’язку, де $2^n \geq m \geq n$. Також для роботи необхідний *m*-розрядний IV.

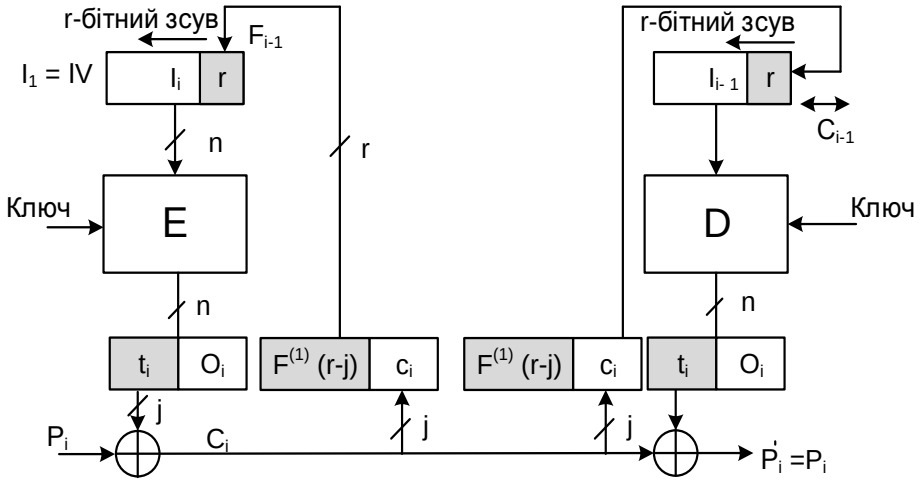


Рис. 4.27. **Режим CFB – *j*-бітний відкритий текст /*r*-бітний зворотний зв’язок (j/r CFB-режим)**

На вхід алгоритму надходять: *k*-розрядний ключ K, *m*-розрядний IV, послідовність *j*-розрядних блоків відкритого тексту P = p_1, p_2, …, p_u. На виході алгоритму формується послідовність *j*-розрядних блоків закритого тексту C = c_1, c_2 ,…,c_u. Спочатку здійснюється запис у буфер зворотного зв'язку (регістр зсуву I_i). Значення вектора ініціалізації I_1=IV.

Далі для всіх i = $\overline{1, u}$ виконуються такі операції:

1. Вибірка n кратних ліворуч біт з регістра зсуву I:$X_i = I_i \sim n$.
2. Обчислення функції шифрування E:$O_i = E_k(X_i)$.
3. Вибірка j кратних ліворуч бітів з величини O_i:$t_i = O_i \sim j$.
4. Формування і передача блоку закритого тексту: $C_i = p_i \oplus t_i$.
5. Формування змінного зворотного зв'язку із блоку шифротексту c_i шляхом його доповнення ліворуч r – j одиницями: $F_i = F^{(1)}(r - j) \| c_i$.
6. Зрушення вмісту буфера зворотного зв'язку I на r розрядів вліво і запис F_i у регістр: $I_{i+1} = 2^r I_i + F_i \bmod 2^n$.

Дешифрування виконується аналогічним чином, за винятком операції 4, а саме: $P_i = c_i \oplus t_i$.

На рис. 4.28 схематично поданий CFB-режим за ISO/IEC 10116. Зазначимо також, що в другій редакції ISO/IEC 10116 рекомендується обирати j = r.

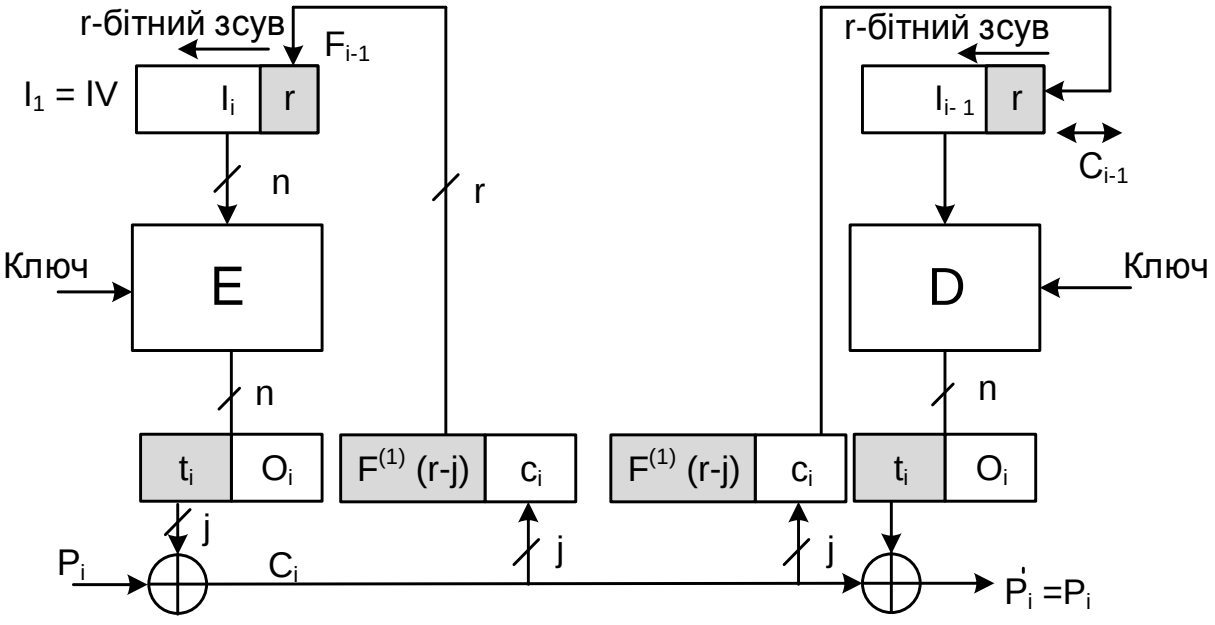


Рис. 4.28. **Режим CFB за ISO/IEC 10116 з конвеєрною обробкою даних**

4.7.4. Шифрування в режимі Output Feedback, OFB

Цей режим подібний режиму CFB. Різниця полягає в тому, що вихід алгоритму в режимі OFB подається назад у регістр зсуву, тоді як у режимі CFB у регістр зсуву подається результат застосування операції XOR до незашифрованого блоку і результату алгоритму.

Основна перевага режиму OFB полягає в тому, що якщо при передачі відбулась помилка, то вона не поширюється на наступні зашифровані блоки, і тим самим зберігається можливість дешифрування наступних блоків. Наприклад, якщо з'являється помилковий біт у C_i, то це призведе тільки до неможливості дешифрування цього блоку і одержання P_i. Подальша послідовність блоків буде розшифрована коректно. При використанні режиму CFB C_i подається в якості входу в регістр і є причиною наступного викривлення потоку.

Недолік OFB у тому, що він більш уразливий до атак модифікації потоку повідомлень, ніж CFB.

Режим OFB може бути використаний для додатків, у яких повинне бути виключене будь-яке розмноження помилок. Режим схожий з режимом CFB і дозволяє шифрувати блоки різної довжини, але як зворотний зв'язок використовується не блок шифротексту, а шифрований блок з виходу функції E.

Поширено дві версії *OFB*-режиму роботи *n*-розрядного блокового шифру. Версія ISO/IEC 10116: 1997 (ISO/IEC 10118: 1991) вимагає n-розрядного зворотного зв'язку (повний зворотній зв'язок) і є більш стійкою. Раніше була прийнята версія NBS FIPS 81, що дозволяє $r \leq n$ розрядний зворотний зв'язок. На рис. 4.29 подана схема OFB-режиму за ISO/IEC 10116: 1997.

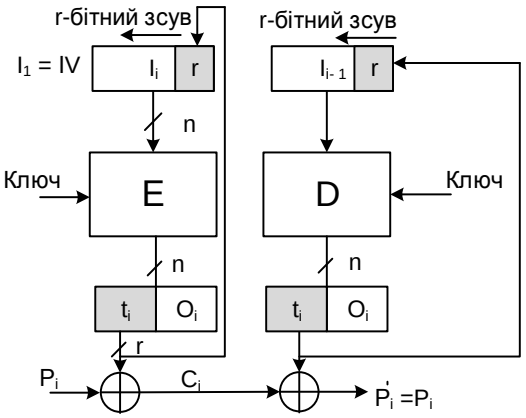


Рис. 4.29. **Режим OFB *r*-бітний відкритий текст/*n*-бітний зворотний зв'язок (r/n – *OFB*-режим)**

У цій версії OFB-режиму на вхід алгоритму надходять: *k*-бітний ключ K, *n*-розрядний IV, послідовність *r*-розрядних блоків відкритого тексту P = p_1, p_2, …, p_u, $1 \le r \le n$ ... На виході формується послідовність *r*-розрядних блоків шифротексту. При дешифруванні відновлюється відкритий текст.

Шифрування здійснюється в такий спосіб.

Спочатку в регістр I_i записується значення IV.

Потім для $1 \le i \le u$ виконуються:

1. Обчислення шифрованого блоку:$O_i = E_k(I_i)$.

2. Вибірка r крайніх ліворуч бітів величини O_i. Допускають, що крайній біт визначений як 1:

$$t_i = O_i \sim r.$$

3. Передача r-розрядного блоку шифротексту c_i:$c_i = p_i \oplus t_i$.

4. Відновлення змісту регістра зсуву I_i: $I_{i+1} = O_i$.

При розшифруванні встановлюють I_1=IV і для всіх $1 \le i \le u$ після одержання блоку c_i здійснюють ті ж дії за винятком операції 3, що має вигляд: $p_i = c_i \oplus t_i$.

При реалізації режиму OFB за NBS FIPS B1 на вхід алгоритму надходять: *k*-бітний ключ K, *n*-розрядний IV, послідовність *r*-розрядних блоків відкритого тексту P = p_1, p_2,…,p_u, де $1 \le r \le n$. На виході формується послідовність *r*-розрядних блоків шифротексту C = c_1, c_2 ,…,c_u.

Алгоритм шифрування і розшифрування аналогічний розглянутим вище, за винятком того, що операція відновлення регістра $I_{i+1} = O_i$ замінюється на операцію зсуву змісту регістра і запису в нього значення t_i:

$$I_{i+1} = 2^r I_i + t_i \bmod 2^n.$$

Розглянемо властивості OFB-режиму:

1. Шифрування ідентичних відкритих текстів. Як і в CBC і в CFB-режимах шифрування, зміна IV приводить до різних результатів шифрування того самого відкритого тексту.

2. Вплив зчеплення. Ключовий потік є незалежним від відкритого тексту.

3. Розмноження помилок. Одна або більше бітових помилок у будь-якому символі шифротексту c_i, впливає на дешифрування тільки цього символу, і точно в тих же бітових позиціях, у яких виявилися помилки в c_i. У такий спосіб забезпечується повне відновлення бітів відкритого тексту.

4. Відновлення синхронізації. OFB-режим відновлюється після помилок у бітах шифротексту, але не може самосинхронізуватися після втрати бітів шифротексту, які руйнують вирівнювання (синхронізацію) ключового потоку, що розшифровує (у цих випадках необхідна пересинхронізація шифру).

5. Продуктивність. Для r < n продуктивність зменшується в n/r разів (як і в CFB-режимі). Однак у всіх випадках, оскільки ключовий потік не залежить від відкритого й закритого тексту, він може бути обчислений заздалегідь за заданим ключем й IV.

6. Зміна IV в OFB. Вектор ініціалізації, що не є секретним, повинен змінюватися, якщо в OFB-режимі повторно використовується ключ K. У протилежному випадку буде сформований ідентичний ключовий потік, і шляхом додавання за модулем 2 (XOR) відповідних шифртекстів противник може прийти до криптоаналізу, у якому шифр буде з нескінченним ключем, де як ключ використовується відкритий текст.

У такий спосіб, виходячи із властивостей режиму, на практиці OFB-режим використовується тільки для забезпечення конфіденційності. У спрощеному OFB-режимі виводять відновлення вхідного блоку як функцію лічильника, тобто: $I_{i+1} = I_{i+1}$.

Це дозволяє уникнути проблему так званого короткого циклу, і забезпечити відновлюваність після помилок в обчисленні E. Більше того, це забезпечує властивість випадкового доступу: не обов'язково дешифрувати *i*-ий блок шифротексту для того, щоб дешифрувати i + 1-ий блок.

Раніше версія ISO-режиму була більш стійка, ніж версія NBS FIPS. В OFB-режимі з повним n-розрядним зворотним зв'язком ключовий потік генерується з використанням ітеративної функції $O_i = E_k(O_i – 1)$. Оскільки E_k є перестановкою й при припущенні, що k є випадковою величиною, E_k дійсно є випадковим вибором з множини $(2^n)!$ перестановок за n елементами. Може бути показано, що для фіксованих (випадкових) значень ключа й вектора ініціалізації очікувана довжина циклу перед повторенням будь-якого значення O_i дорівнюватиме $2^n – 1$. З іншого боку, якщо кількість бітів зворотного зв'язку дорівнює r < n, як визначено в FIPS 81, то ключовий потік формуватиметься з використанням ітерації $O_i = f (O_i – 1)$, де f не є функцією перестановки. Допускаючи її поводження як випадкової функції, очікувана довжина циклу складе величину близько $2^{n/2}$. Отже, кращим є використання OFB-режиму з повним *n*-розрядним зворотним зв'язком. Нарешті необхідно відзначити, що і OFB-режим

з повним зворотним зв'язком і режим лічильника забезпечують можливість застосування блокового шифру як генератора ключового потоку для потокового шифру.

Аналогічним чином і в CFB-режимі здійснюється шифрування потоку символів, при цьому блоковий шифр використовується як генератор ключового потоку (залежного від відкритого тексту).

Режим CBC також можна розглядати як потоковий шифр із n-розрядними блоками, що відіграють роль дуже великих символів. У такий спосіб режими роботи дозволяють побудувати потокові шифри на основі блокових шифрів.

4.7.5. Шифрування в режимах удосконаленого OFB і PCBC

Відомі недоліки привели до появи вдосконаленого варіанта шифрування в режимі OFB [14 – 17; 56]. Основні зміни стосуються методу генерації незалежної послідовності блоків: для одержання чергового блоку пропонується шифрувати не s_i, а s_i + IV(mod 264), де IV – деякий вектор ініціалізації.

Режим шифрування POBC (Propagating Cipher Block Chaining) застосовується в протоколі Kerberos (версія 4) і дозволяє виявляти помилки. Цей режим шифрування не є федеральним або міжнародним стандартом. Режим POBC – варіант режиму CBC, що володіє специфічною властивістю, – у результаті розшифрування одинична помилка поширюється на весь шифротекст (вирішується зворотне завдання з погляду режиму OFB). Ця властивість дозволяє з високою надійністю виявляти помилки, що виникають при передачі повідомлень каналами з шумом. Шифрування в режимі POBC виконується за правилом: $c_i = E_k(p_i \oplus p_{i-1} \oplus c_{i-1})$, розшифрування: $p_i = Dk(c_i) \oplus c_{i-1} \oplus p_{i-1,}$ де $p_0 \oplus c$ – вектор ініціалізації.

4.8. Сучасні потокові шифри, їх переваги та недоліки

Міцні шифри перетворюють відкритий текст у шифротекст по одному біту за операцію.

Генератор потоку ключів видає потік бітів: k_1, k_2, k_3,..., k_i. Цей потік бітів (іноді називаний ключем, що біжить) і потік бітів відкритого тексту p_1,p_2,p_3,...,p_i зазнають операції XOR, в результаті якої виходить потік бітів шифротексту: $c_i = p_i \oplus k_i$.

При розшифруванні для відновлення бітів відкритого тексту операція XOR виконується над бітами шифротексту і тим самим потоком ключів: $p_i = c_i \oplus k_i$.

Безпека системи повністю залежить від властивостей генератора потоку ключів. Якщо генератор потоку ключів видає нескінченний рядок нулів, шифротекст збігатиметься відкритим текстом і перетворення буде безглуздим. Якщо генератор потоку ключів видає повторюваний 16-бітовий шаблон, криптостойкість буде зневажливо мала. У випадку нескінченного потоку випадкових бітів криптостойкість потокового шифру буде еквівалентна криптостійкості одноразового блокноту. Генератор потоку ключів створює бітовий потік, який схожий на випадковий, але в дійсності детермінований і може бути безпомилково відтворений при розшифруванні. Чим ближче вихід генератора потоку ключів до випадкового, тим вища трудоємність криптоаналітичної атаки.

Блокові і потокові шифри реалізуються по-різному. Потокові шифри, що розшифровують дані за одним бітом, не дуже підходять для програмних реалізацій. Блокові шифри легше реалізовувати програмно, тому що вони дозволяють уникнути трудоємних маніпуляцій з бітами і оперують зручними для комп'ютера блоками даних. З іншого боку, потокові шифри більше підходять для апаратної реалізації.

4.8.1. Регістри зсуву зі зворотним зв'язком

Більшість реальних потокових шифрів заснована на *регістрах зсуву зі зворотним зв'язком*.

Регістр зсуву застосовують для генерації ключової послідовності. Регістр зсуву зі зворотним зв'язком складається із двох частин: регістру зсуву і функції зворотнього зв'язку (рис. 4.30).

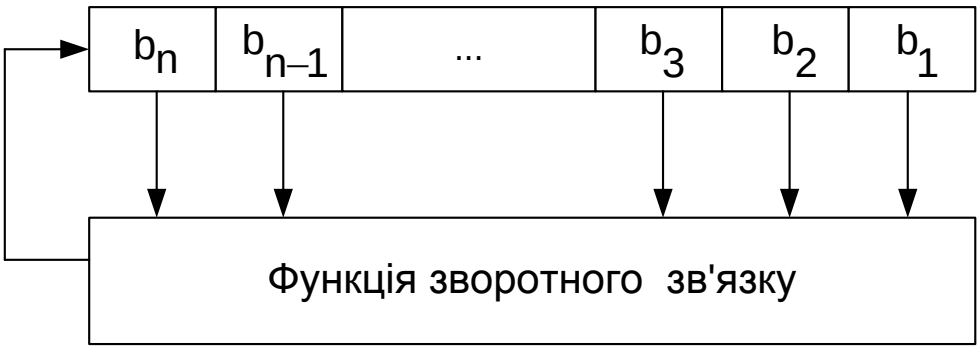


Рис. 4.30. **Регістр зсуву зі зворотним зв'язком**

Регістр зсуву становить послідовність бітів, кількість яких визначається довжиною зсуву регістру. Якщо довжина рівна *n*-бітам, то регістр називається *n*-бітовим регістром зсуву. Щоразу, коли потрібно витягти біт, всі біти регістру зсуву зсуваються вправо на 1 позицію. Новий крайній лівий біт є функцією всіх інших бітів регістру.

На виході регістру виявляється один, зазвичай молодший значущий біт. Періодом регістру називається довжина одержуваної послідовності до початку її повторення. Найпростішим видом регістру зсуву зі зворотним зв'язком є *регістр зсуву з лінійним зворотним зв'язком* (РЗЛЗЗ) (рис. 4.31).

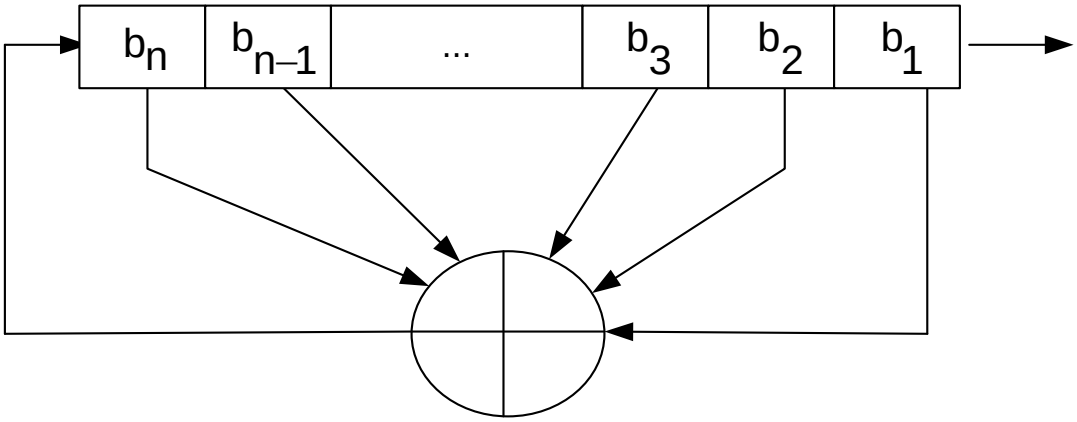


Рис. 4.31. **РЗЛЗЗ**

Зворотний зв'язок становить XOR деяких бітів регістру; ці біти називаються *відповідною послідовністю.*

На рис. 4.32 показаний 4-бітовий РЗЛЗЗ з відводом від першого і четвертого бітів.

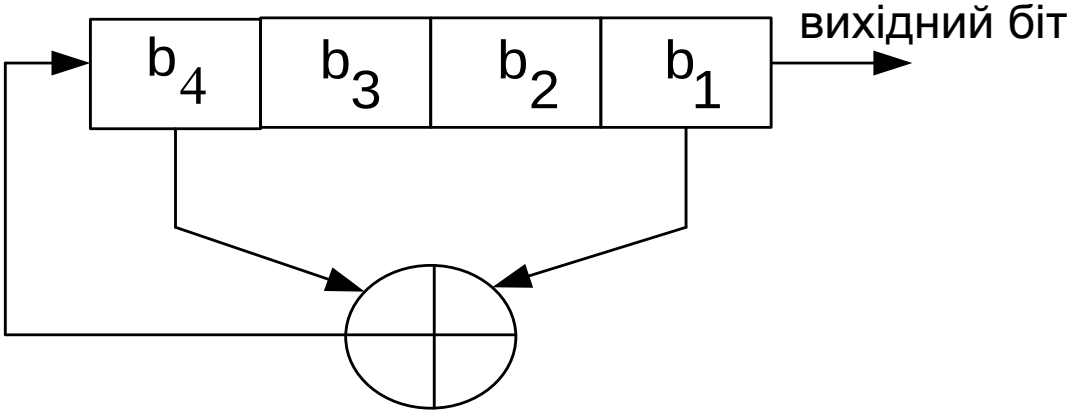


Рис. 4.32. **4-бітний РЗЛЗЗ**

Якщо за його допомогою проаналізувати значення 1111, то до повторення регістр буде приймати такі внутрішні стани:

1111 0111 1011 0101 1010 1101 0110 0011 1001 0100 0010 0001 1000 1100 1110.

Вихідною послідовністю буде рядок молодших значущих бітів: 111101011001000 ...

РЗЛЗЗ (*n*-бітовий) може перебувати в одному з $2^n - 1$ внутрішніх станів. Це означає, що теоретично такий регістр може генерувати псевдовипадкову послідовність із періодом $2^n - 1$ бітів. Кількість внутрішніх станів і періодів рівна $2^n - 1$, тому що заповнення РЗЛЗЗ нулями призведе до того, що регістр зсуву видаватиме нескінченну послідовність нулів, що абсолютно даремно. Тільки при визначених відвідних послідовностях РЗЛЗЗ циклічно пройде через всі $2^n - 1$ внутрішніх станів. Такі РЗЛЗЗ мають максимальний період, а результат, що вийшов, називається *M*-послідовністю. Для того, щоб конкретний РЗЛЗЗ мав максимальний період, многочлен, асоційований *з* відвідною послідовністю, повинен бути примітивним за модулем 2, – тобто не розкладатись на добуток двійкових многочленів меншого степеня. Відповідну математичну теорію можна знайти в [23; 46; 47; 49].

Наприклад, многочлен $x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1$ примітивний за модулем 2. Розглянемо цей многочлен у термінах РЗЛЗЗ із максимальним періодом. Степінь многочлена задає довжину РЗЛЗЗ. Вільний член многочлена завжди рівний 1, і його можна вилучити. Степені формальної змінної многочлена задають довжину РЗЛЗЗ виключенням 0-*ї*, задають відвідну послідовність, яка відлічується від лівого краю регістру зсуву. Тобто члени многочлена з меншим степенем відповідають позиціям, розташованим ближче до правого краю регістру. Тоді для взятого 32-бітового зсувного регістру новий біт генерується за допомогою XOR тридцять другого, сьомого, п'ятого, третього, другого і першого бітів (рис. 4.33).

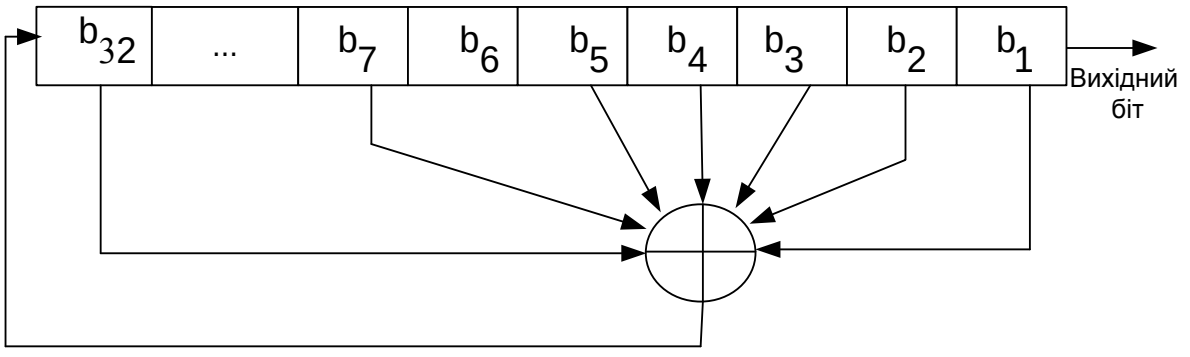


Рис. 4.33. **32-бітний РЗЛЗЗ з максимальним періодом**

Тоді РЗЛЗЗ матиме максимальну довжину, циклічно проходячи до повторення через 2^{32} – 1 різних значень.

Власне РЗЛЗЗ є гарними генераторами псевдовипадкових послідовностей, але вони мають деякі небажані невипадкові властивості. Для РЗЛЗЗ довжини n внутрішній стан становить попередні n вихідних бітів генератора. Навіть якщо схема зворотного зв'язку зберігається в секреті, вона може бути визначена за 2^n вихідними бітами генератора за допомогою алгоритму Берлекемпа – Мессі [23; 46; 47; 49].

Крім того, більші випадкові числа, які генеруються з використанням РЗЛЗЗ, формують підряд біти послідовності з сильнокорельованими властивостями і для деяких типів додатків зовсім не є випадковими.

Незважаючи на це, РЗЛЗЗ часто використовуються при розробці алгоритмів шифрування.

4.8.2. Алгоритм А5

А5 – потоковий шифр, який використовується для шифрування GSM (Group Special Mobile) – європейського стандарту цифрових стільникових мобільних телефонів. А5 складається із трьох РЗЛЗЗ довжиною 19, 22 і 23. Виходом є XOR трьох РЗЛЗЗ. В А5 використовується змінюване керування тактуванням. Кожний регістр тактується залежно від свого біта, потім виконується XOR зі зворотною граничною функцією середніх бітів всіх трьох регістрів. Звичайно на кожному етапі тактується два РЗЛЗЗ. Існує тривіальна атака на відкритий текст, яка заснована на припущенні того, що зміст перших двох РЗЛЗЗ відомий і спробі визначення третього РЗЛЗЗ за ключовою послідовністю. Але ідеї, що лежать в основі А5, дозволяють проектувати надійні потокові шифри. Алгоритм ефективний і задовольняє всім відомим статистичним тестам. Єдина його слабкість – короткі регістри. Варіанти А5 з більш довгими регістрами зсуву і більш щільними многочленами зворотнього зв'язку дозволяють протистояти силовій атаці.

4.8.3. RC4

RC 4 – потоковий шифр зі змінним розміром ключа, розроблений в 1987 році. Рівестом для RSA Data Security, Inc. Алгоритм працює в режимі OFB. Для формування ключів використовується S-блок розміром 8 × 8: S_0, S_1,..., S_{255}. Елементи становлять перестановку чисел від 0 до 255,

а перестановка є функцією ключа змінної довжини. В алгоритмі застосовуються два лічильники, i та j з нульовими початковими j. Для генерації випадкового байта виконуються такі обчислення:

$$i = (i + 1) \bmod 256;$$
$$j = (j + S_i) \bmod 256.$$

Поміняти місцями S_i та S_j:

$$t = (S_i + S_j) \bmod 256;$$
$$K = S_t.$$

K використовується в операції XOR з відкритим текстом для одержання шифротексту або в XOR із шифротекстом для одержання відкритого тексту. Шифрування виконується приблизно в 10 разів швидше, ніж у DES. Також нескладна й ініціалізація *S*-блоку. Спочатку *S*-блок заповнюється за правилом: $S_0 = 0, S_1 = 1, ..., S_{255} = 255$. Після цього ключ записується в масив: $K_0, K_1, ..., K_{255}$. Потім при початковому значенні j = 0 у циклі виконуються афінне обчислення:

$$\text{for I = 0 to 255:}$$
$$j = (j + S_i + K_i) \bmod 256.$$

Поміняти місцями S_i та S_j.

Компанія RSA Data Security, Inc. затверджує, що алгоритм стійкий до диференціального і лінійного криптоаналізу, та що він у високому степені нелінійний. *S*-блок повільно змінюється при використанні: *i* і *j* забезпечується випадкова зміна кожного елементу. C.4 входить у десятки комерційних продуктів, включаючи Lotus Notes, AOCE компанії Apple Computer і Oracle Secure SQL. Цей алгоритм також є частиною специфікації стандарту Стільникової цифрової пакетної передачі даних CDPD (Cellular Digital Packet Data) [23; 40; 44; 61].

4.8.4. SEAL

SEAL – це ефективний потоковий шифр, розроблений в IBM Рогевеєм (P. Rogaway) і Копперсмітом (D. Coppersmith). Алгоритм оптимізований для 32-бітових процесорів. Для нормальної роботи йому потрібно вісім 32-бітових регістрів і пам'ять на декілька кілобайтів. У SEAL передбачений ряд попередніх дій із ключем зі збереженням результатів у декількох таблицях. Таблиці використовуються для приско-

рення процедур шифрування і розшифрування. Особливістю SEAL є те, що він дійсно не є традиційним потоковим шифром, а становить сімейство псевдовипадкових функцій. При 160-бітовому ключі k і 32-бітовому регістрі *n*-SEAL розтягує L в *L*-бітовий рядок k(n). L може приймати будь-яке значення, менше 64 Кбайтів. SEAL використовує таке правило: якщо k вибирається випадковим чином, то k(n) повинне відрізняться від випадкової *L*-бітової функції n. Практичний ефект того, що SEAL є сімейством псевдовипадкових функцій, полягає в тому, що він зручний у ряді додатків, де незастосовані традиційні потокові шифри. При використанні більшості потокових шифрів створюється односпрямована послідовність бітів: єдиним способом визначити *i*-й біт (знаючи ключ і позицію i) є генерування всіх бітів до *i*-го. Відмінність сімейства псевдовипадкових функцій полягає в тому, що можна легко одержати доступ до будь-якої позиції ключової послідовності.

Сімейство псевдовипадкових функцій також спрощує проблему синхронізації, що зустрічається в стандартних потокових шифрах – можна зашифрувати на ключі k *n*-е передане повідомлення x_n, виконавши XOR x_n і k(n). Одержувачу не потрібно зберігати стан шифру для відновлення x_n, йому не доводиться турбуватись і про загублені повідомлення, що впливають на процес розшифрування.

Контрольні запитання

1. Основні операції шифрування у DES (DataEncryptionStandard) – стандарті шифрування даних США.

2. Основні модифікації шифру DES (3DES, DESX). Переваги та недоліки.

3. Основні операції шифрування алгоритму криптографічного перетворення ГОСТ 28147-89.

4. Основні відміності операцій шифрування у алгоритмі Rijndael.

5. Основні специфікації операцій шифрування алгоритмів українського конкурсу на сертифікований симетричний криптоалгоритм.

6. Основні режими роботи блоково-симетричних шифрів на основі використання алгоритму DES.

7. Сучасні потокові шифри, їх переваги та недоліки.

Розділ 5. Алгоритми з відкритим ключем

5.1. Алгоритм RSA, його криптостійкість та швидкість роботи

Не дивлячись на досить велику кількість різних асиметричних криптосистем, широке практичне застосування отримала RSA, назва якої походить від перших літер прізвищ її авторів: Рональда Рівеста (Ron Rivest), Аді Шаміра (Adi Shamir) та Леонарда Аделмана (Leonard Addleman).

Криптосистема RSA використовує односторонню функцію утворення добутку двох великих простих цілих чисел, що значно простіше, ніж розкладання такого великого цілого числа на прості множники. Зрозуміло, що це принципово можна зробити, однак, витрати ресурсів (часу або обчислювальної потужності комп'ютерів) будуть не меншими, ніж просте суцільне перебирання усього ключового масиву.

Звичайно, що наявність гарантованої оцінки криптостійкості алгоритму створила сприятливі умови для розповсюдження криптосистеми RSA на фоні інших алгоритмів. Це й стало однією з причин її популярності у банківських системах для роботи з віддаленими клієнтами.

Варто розглянути алгоритм шифрування за схемою RSA. Звичайно, з навчальною метою автори використовують приклад з малими числами, однак для реальної роботи ці числа не підходять, оскільки криптостійкість такої системи практично дорівнює нулю.

Алгоритм RSA є блоковим алгоритмом, де даними є цілі числа з відрізку $[0,n-1]$ для деякого n.

Отже, для обміну інформацією, зашифрованою за допомогою криптосистеми RSA, необхідно виконати такі кроки.

Крок 1. *Підготовчі обчислення*. Отримувач генерує два великих простих числа p і q (мінімум 128-бітних). Для цього прикладу візьмемо $p = 7$, $q = 11$. Обчислимо добуток, ***модуль*** криптосистеми, $n = p \times q = 77$. Далі необхідно обчислити функцію Ейлера для цього модуля. Відомо, що для простих чисел $\varphi(n) = (p - 1)(q - 1)$. Отже, $\varphi(77) = 6 \times 10 = 60$. Тепер необхідно згенерувати ціле число, взаємно просте як з n, так і з $\varphi(n)$, наприклад, $e = 13$.

Пара чисел (*e*,*n*) буде служити ***публічним ключем*** криптосистеми. У нашому випадку це буде пара (13, 77).

Тепер необхідно обчислити ***приватний ключ*** *d*, парний до обраного публічного. Для цього треба розв'язати рівняння: $(d \times e) \bmod \varphi(n) = 1$. Відповідно до обчислень мультиплікативного оберненого, будемо мати: $d = (1 + k \times \varphi(n))/e$, де k – ціле число. Оскільки приклад дуже простий, то простим перебиранням для $k = 8$ отримуємо $d = 37$. Отже, приватним ключем, парним до нашого публічного (13, 77) буде служити пара чисел (d, n) – (37, 77).

Крок 2. *Розповсюдження ключів*. Для шифрування інформації використовують публічний ключ (хоча можна використовувати і приватний, деякі асиметричні криптосистеми, в тому числі RSA, це дозволяють). Для використання його розміщують на ресурсі, до якого мають доступ усі учасники інформаційного обміну. Зауважимо, що одним публічним ключем можуть користуватися усі, хто бажає обмінюватися з отримувачем зашифрованою інформацією. На відміну від симетричних криптосистем, тут немає необхідності для кожної пари учасників генерувати окрему пару ключів, адже розшифрувати інформацію за допомогою публічного ключа неможливо (в усякому разі, обчислювально складно). Для простого прикладу необхідно в подальшому продемонструвати атаку перешифруванням, коли багаторазове шифрування перехопленого повідомлення призводить до відкритого тексту. Тут йдеться лише про те, щоб трудомісткість такої атаки була не меншою, ніж трудомісткість атаки безпосереднього перебирання ключів. Таким чином, конфіденційність обміну інформації гарантується самим принципом обробки інформації. Єдине, що необхідно зробити, це захистити публічний ключ від підміни (про атаки на асиметричні криптосистеми наведено далі). Найпростіше, що можна зробити, це захистити каталог, де знаходяться ключі, від запису, однак найнадійнішим способом вважається сертифікація публічних ключів. Кожен, хто бажає захистити свій публічний ключ від підміни, повинен отримати сертифікат довірчого центру інфраструктури відкритих ключів, який прив'язує ключ до його власника.

Приватний ключ не розповсюджується. Він використовується для розшифрування інформації та створення ***електронного цифрового підпису***, і повинен бути відомим лише його власникові.

На цьому підготовчі операції закінчено, і можна починати обмін захищеною інформацією.

Крок 3. *Шифрування інформації*. Криптосистемою RSA можна зашифрувати числа (десяткові коди літер) у діапазоні від 0 до n. Відправник повідомлення, використовуючи публічний ключ (e,n), у нашому випадку – (13, 77), за допомогою формули $C_i = (M_i)^e \bmod n$ зашифровує своє повідомлення, де M_i – числове подання чергової літери повідомлення, C_i – черговий символ криптограми. Наприклад, слово "БАНК" (яке має числове представлення "02 01 17 14" за таблицею заміни українського алфавіту, яка починається з 01) зашифрується таким чином:

$$C_1 = 2^{13} \bmod 77 = 30;$$
$$C_2 = 1^{13} \bmod 77 = 1;$$
$$C_3 = 17^{13} \bmod 77 = 73;$$
$$C_4 = 14^{13} \bmod 77 = 49.$$

Отже, криптограма буде мати вигляд: "30 01 73 49". Очевидно, що шифр в нашому прикладі є шифром простої заміни. Як бачимо, літера "А", яка має код "01", не змінилася. Таку ж властивість мають "0" та n – 1. Отже, не всі числа доцільно вибирати в якості кодів літер. Вважається правильним надавати для шифрування числа з діапазону [2, n – 2]. Це дещо ускладнює розкриття шифру.

Зашифрований текст пересилається отримувачу відкритими каналами зв'язку.

З наведеного способу шифрування може скластися враження, що піднесення великого цілого числа у великий степінь та взяття залишку за модулем третього великого числа є надзвичайно складною математичною задачею. Однак насправді це зовсім не так. Для ілюстрації цього наведемо алгоритм обчислення виразу $432^{678} \bmod 987$.

Число 678 можна подати так: 678 = 512+128+32+4+2. Тоді $432^{678} \bmod 987 = (432^{512} \times 432^{128} \times 432^{32} \times 432^{4} \times 432^{2}) \bmod 987$. Використовуючи основні властивості, які вже наведені, можна записати: $432^{678} \bmod 987 = ((432^{2} \bmod 987) \times (432^{4} \bmod 987) \times (432^{32} \bmod 987) \times (432^{128} \bmod 987) \times (432^{512} \bmod 987)) \bmod 987$.

Тепер необхідно обчислити ступені числа 432: $432^{2} \bmod 987 = 81$; $432^{4} \bmod 987 = 81^{2} \bmod 987 = 639$; $432^{8} \bmod 987 = 639^{2} \bmod 987 = 690$; $432^{16} \bmod 987 = 690^{2} \bmod 987 = 366$; $432^{32} \bmod 987 = 366^{2} \bmod 987 = 711$; $432^{64} \bmod 987 = 711^{2} \bmod 987 = 177$; $432^{128} \bmod 987 = 177^{2} \bmod 987 = 732$; $432^{256} \bmod 987 = 732^{2} \bmod 987 = 870$; $432^{512} \bmod 987 = 870^{2} \bmod 987 = 858$.

Підставляючи у вираз ці значення, отримаємо:

$$(81 \times 639 \times 711 \times 732 \times 858) \bmod 987 = 204.$$

Крок 4. *Розшифрування інформації*. Отримувач розшифровує зашифроване повідомлення "30 01 73 49", використавши тільки йому відомий приватний ключ (d, n) та формулу $M_i = (C_i)^d \bmod n$. Доведемо принципову можливість розшифрування зашифрованої на публічному ключі інформації: $(C)^d \bmod n = (M^e \bmod n)^d \bmod n = (M^e)^d \bmod n = (M^{ed}) \bmod n = (M^{1+k\varphi(n)}) \bmod n = M(M^{\varphi(n)}) \bmod n = M$. Таким чином, операція зашифрування на публічному та розшифрування на приватному ключі – взаємно зворотні.

У цьому випадку приватним ключем служить пара (37,77). Тоді отримаємо: $M_1 = 30^{37} \bmod 77 = 2$; $M_2 = 1^{37} \bmod 77 = 1$; $M_3 = 73^{37} \bmod 77 = 17$; $M_4 = 49^{37} \bmod 77 = 14$.

Маючи таблицю заміни, за кодами літер отримаємо "БАНК". Отже, визначено методи зашифрування та розшифрування інформації у криптосистемі RSA. Залишилося вияснити один цікавий факт, характерний для цієї криптосистеми: чи дозволить вона шифрувати інформацію приватним ключем, а розшифровувати – публічним?

Нехай повідомлення для зашифрування те ж саме: "БАНК" або "02 01 17 14". Зашифруємо її на приватному ключі (37,77). Отримаємо таке: $C_1 = 2^{37} \bmod 77 = 51$; $C_2 = 1^{37} \bmod 77 = 1$; $C_3 = 17^{37} \bmod 77 = 52$; $C_4 = 14^{13} \bmod 77 = 42$.

Тепер розшифруємо зашифроване повідомлення "51 01 52 42" на публічному ключі (13,77): $M_1 = 51^{13} \bmod 77 = 2$; $M_2 = 1^{13} \bmod 77 = 1$; $M_3 = 52^{13} \bmod 77 = 17$; $M_4 = 42^{13} \bmod 77 = 14$.

Як бачимо, отримане відкрите повідомлення "БАНК" і таким методом. Отже, криптосистема RSA симетрична відносно застосування парних ключів: можна зашифровувати інформацію на публічному ключі та розшифровувати на приватному і навпаки, зашифровувати на приватному, а розшифровувати – на парному до нього публічному ключі.

5.1.1. Безпека та швидкодія RSA

Існує багато публікацій, що розглядають тему апаратних реалізацій RSA. Швидкодія апаратної реалізації RSA приблизно в 1000 разів нижча, ніж реалізації DES [33]. Наприклад, швидкодія реалізації RSA з 512-бітовим модулем – 64 Кбіт/с. На сьогодні розробляються мікросхеми з 512-бітовим модулем, швидкодія яких прямує до 1Мб/с. Існують також мікросхеми RSA, які оперують із 1024-бітовими числами, але вони вкрай

повільні. Виробники також реалізують RSA в інтелектуальних картках, однак продуктивність цих реалізацій невисока.

Що стосується програмних реалізацій, то наприклад, програмна реалізація DES приблизно в 100 разів швидша програмної реалізації RSA на мові С.

У табл. 5.1 наведені приклади продуктивності програмної реалізації RSA для 8-бітової експоненти шифрування і різної розрядності модуля.

Таблиця 5.1

Ефективність програмної реалізації для 8-бітової експоненти шифрування і різної розрядності модуля (в секундах)

Операція	512-бітовий модуль	768-бітовий модуль	1024-бітовий модуль
Шифрування	0,03	0,05	0,08
Розшифрування	0,16	0,48	0,93
Обчислення підпису	0,16	0,52	0,97
Перевірка підпису	0,02	0,07	0,08

Розглянемо проблеми, які виникають при створенні ключів. Ця процедура включає такі завдання:

1. Визначити два прості числа *p* та *q*.
2. Вибрати *e* та обчислити *d*.

Насамперед, розглянемо проблеми, пов’язані з вибором *p* і *q*. Через те, що значення $n = p \times q$ буде відомо потенціальному зловмиснику, для запобігання розкриття *p* і *q* ці прості числа повинні бути обрані з більшої множини, тобто *p* і *q* повинні бути якомога більшими числами. З другого боку, метод, який використовується для пошуку великого простого числа, повинен бути достатньо ефектив ним.

На сьогодні невідомі ефективні алгоритми, які швидко генерують великі прості числа. Процедура, що використовується для цієї задачі, обирає випадкове непарне число з необхідного діапазону і перевіряє, чи є воно простим. Якщо число не є простим, то обирається інше випадкове число, поки не буде знайдено просте.

Існують різні тести для визначення того, чи є число простим. Ці тести імовірнісні, тобто тест показує, що дане число ймовірно є простим. Незважаючи на це, вони можуть виконуватись таким чином, що дадуть

імовірність цього як завгодно близькою до 1. Якщо *x* не проходить тест, то воно точно складене. Якщо *x* проходить тест, то воно може бути простим. Якщо *x* проходить багато таких тестів, то можна з високою імовірністю сказати, що воно просте. Це досить довга процедура, але вона виконується відносно рідко: тільки при створенні нової пари клю чів.

На складність обчислень також впливає те, яка кількість чисел буде відкинута перед тим, як буде знайдено просте число. Результат з теорії чисел, відомий як теорема про просте число, дає зрозуміти, що простих чисел, розташованих близько до *x*, у середньому, по одному на кожні ln (*x*) чисел.

Таким чином, у середньому потрібно перевірити послідовність із ln (*x*) цілих, перш ніж буде знайдено просте число. Через те, що всі парні числа можуть бути відкинуті без перевірки, то потрібно виконати приблизно ln (*x*)/2 перевірок.

Наприклад, якщо просте число шукається в діапазоні величин 2^{200}, то необхідно виконати близько ln (2^{200}) / 2 = 70 перевірок.

Вибравши прості числа *p* і *q*, далі слід обрати значення *e* так, щоб НСД(Ф(*n*), *e*) = 1 та обчислити значення $d = e^{-1} \bmod \Phi(n)$. Для обчислення *d* можна використати модифікований алгоритм Евкліда, який дозволяє за фіксований час обчислити найбільший спільний дільник двох чисел, і якщо він дорівнює одиниці, обчислює інверсне значення одного за модулем іншого.

Припустимо тепер, що інформацію, яку перехопив зловмисник, зашифровано на публічному ключі (13,77). Якою інформацією володіє в такому разі зловмисник? По-перше, він має криптограму "30 01 73 49"; по-друге, має публічний ключ (13,77). Які зусилля треба йому прикласти для обчислення відкритого тексту? У цьому випадку криптостійкість такого повідомлення наближається до нуля, оскільки розрядність ключа дуже мала. Легко здогадатися, що число 77 єдиним способом розкладається на прості множники, 77 = 7 × 11. Можна спостерігати, що компрометація криптосистеми еквівалентна складності розкладання модуля на множники. Це саме, згідно з теоремою Рабіна, справедливо і для великих модулів. У монографії [33] виконано розрахунок MIPS років (1 MIPS рік = 1 млн інструкцій за секунду протягом 1 року = $3{,}1 \times 10^{13}$

інструкцій), необхідних для розкладання великих чисел на прості множники. Ці дані можна побачити у табл. 5.2.

Таблиця 5.2

Обчислювальна складність розкладання чисел на прості множники

Кількість розрядів *n*	Значення функції *L*(*n*)	Кількість MIPS років
512	$6{,}7\times10^{19}$	$2{,}1\times10^{6}$
576	$1{,}7\times10^{21}$	$5{,}5\times10^{7}$
960	$3{,}7\times10^{28}$	$1{,}2\times10^{15}$
1024	$4{,}4\times10^{29}$	$1{,}4\times10^{16}$

Тут в якості функції *L*(*n*), яка задає апроксимацію швидкості найкращого на сьогодні алгоритму розкладання чисел на прості множники, методу решета числового поля, взято:

$$L(n) = \exp(1+\varepsilon)\sqrt{(\ln n)(\ln\ln n)^2},$$

де n – кількість двійкових розрядів у числі;

ε – мала величина.

У кінці 1995 р. лише єдиний раз вдалося практично реалізувати розкриття шифру RSA для 500-бітного ключа. Для цього за допомогою Інтернет було задіяно 1600 комп’ютерів на протязі 5 місяців неперервної роботи. Тому автори RSA рекомендують використовувати таку довжину модуля *n* [32]:

768 біт – для приватних осіб;

1024 біти – для комерційної інформації;

2048 біт – для особливо таємної інформації.

Постає питання, чи не занадто великі значення модулів? І взагалі, чим повинна визначатися стійкість тієї чи іншої криптосистеми? Відповідь можна отримати, знову таки, у книзі [33]. Для цього спочатку наведемо таблицю типів інформації (див. табл. 5.3) та час її життя, а також рекомендовану довжину ключа симетричної криптосистеми, яка забезпечує необхідну її стійкість [33]. Наведемо також таблицю порівняння криптостійкості симетричних та асиметричних криптосистем (табл. 5.4). У таблиці вказано, за яких довжин ключів досягається приблизно однакова стійкість симетричних та асиметричних систем до методу суцільного перебору ключів (метод "грубої сили").

Таблиця 5.3

Типи інформації та час її життя (в бітах)

Тип інформації	Час життя	Довжина ключа,
Тактична військова інформація	хв./год.	56-64
Оголошення про нову продукцію, злиття компаній	дні/тижні	64
Довготривалі бізнес-плани	роки	64
Торговельні секрети (наприклад, рецептура)	10-річчя	112
Секрети водневої бомби	> 40 років	128
Особи шпигунів	> 50 років	128
Дипломатичні конфлікти	> 60 років	128
Дані перепису населення	> 100 років	>128

Таблиця 5.4

Порівняння довжин ключів симетричних та асиметричних криптосистем еквівалентної стійкості (в бітах)

Довжина ключа симетричної криптосистеми	Довжина відкритого ключа асиметричної криптосистеми
56	384
64	512
80	768
112	1792
128	2304

Як бачимо, для досягнення однакової стійкості асиметричні криптосистеми використовують значно довший ключ (від 7 до 18 разів). З порівняння наданих таблиць видно, що з огляду на терміни зберігання інформації різних типів таємності, довжини ключів асиметричних криптосистем у 2048 біт не виглядають занадто параноїдальними.

5.2. Алгоритм Ель-Гамаля, його безпека та криптостійкість

Стійкість криптосистеми Ель-Гамаля, розробленої у 1985 р., ґрунтується на складності задачі дискретного логарифмування у скінченому

полі. Для встановлення зашифрованого інформаційного обміну необхідно виконати наступні кроки.

Крок 1. *Попередні обчислення*. За допомогою криптографічно стійкого генератора випадкових чисел генерують просте число n таке, що обчислення логарифму за mod n практично важко реалізувати.

Також випадково обирають числа g та a з діапазону [1, n – 1] та обчислюють $h = g^a \bmod n$.

Тепер існує публічний ключ: (n, g, h) та приватний – (n, a).

Крок 2. *Шифрування інформації*. Зашифровують числа m від 0 до n. Для шифрування виконують таке:

Обирають випадкове число r, яке належить відрізку [1, n – 1] та взаємно просте з n – 1.

Обчислюють пару чисел C_1 та C_2 за формулами: $C_1 = g^r \bmod n$; $C_2 = mh^r \bmod n$.

Пара чисел C_1 та C_2 утворює шифрограму для числа m.

Крок 3. *Розшифрування інформації*. Розшифрування виконується за формулою: $m = C_2(C_1^a)^{-1} \bmod n$. Доведемо це. Підставимо значення C_1 та C_2 сюди: $m = mh^r(g^a)^{-1} \bmod n$. Оскільки $h = g^a \bmod n$, то:

$$m = mh^r(g^a)^{-1} \bmod n = mh^r(h^r)^{-1} \bmod n = m.$$

Таким чином, доведено еквівалентність прямого та оберненого перетворення. Криптосистема Ель-Гамаля (з модифікаціями Шнорра) використовується в системах електронного цифрового підпису стандартів США та Росії.

5.2.1. Криптостійкість системи Ель-Гамаля

У реальних застосуваннях, як правило, використовують модуль n криптосистеми довжиною 1024 біти, g – порядку 160 біт.

Безпосередня атака на систему Ель-Гамаля, атака обчислення приватного ключа за публічним, потребує обчислення дискретного логарифму, що для таких великих чисел, n та g перетворюється у математичну задачу надзвичайної обчислювальної складності.

Однак імовірна вразливість криптосистеми Ель-Гамаля полягає в тому, що саме повідомлення міститься лише у C_2. Тому теоретично

можливою видається атака, коли помноживши C_2 на g^u ($u \neq 0$), отримаємо шифротекст для повідомлення $m' = mg^u$.

Що стосується швидкодії криптосистеми Ель-Гамаля, то як показано у роботі [22], швидкість її роботи (на SPARC-M) при програмній реалізації у режимах шифрування та розшифрування зі 160-бітовим показником g для різних довжин модуля n оцінюється величинами, поданими в табл. 5.5.

Таблиця 5.5

Швидкість роботи криптосистеми Ель-Гамаля на SPARC-M

Режим роботи	Довжина модуля, бітів		
	512	768	1024
Шифрування	0,33 с	0,80 с	1.09 с
Розшифрування	0,24 с	0,58 с	0,77 с

Як видно з табл. 5.5, швидкість шифрування та розшифрування значно залежить від довжини модуля: збільшення довжини модуля криптосистеми вдвічі призводить до потрійного зростання часу обробки.

Контрольні запитання

1. Алгоритм асиметричного шифрування даних RSA, його криптостійкість та швидкість роботи.
2. Основні операції алгоритму Ель-Гамаля, його безпека та криптостійкість.
3. Протокол забезпечення автентичності даних за допомогою асиметричного алгоритму RSA.
4. Протокол забезпечення конфіденційності даних за допомогою асиметричного алгоритму RSA.
5. Основні вимоги щодо криптостійкості асиметричних криптосистем.
6. Формування одноосібної криптографічної функції у алгоритмах RSA та Ель-Гамаля.
7. Спеціальні механізми безпеки на основі використання асиметричних алгоритмів шифрування даних в інформаційних системах.

Розділ 6. Протоколи автентифікації

6.1. Поняття про гешувальні алгоритми, їх призначення, вимоги до них

Особливе місце серед механізмів забезпечення цілісності і автентичності займають функції гешування: безключові та ключові, що дозволяють забезпечити широкий спектр послуг безпеки інформації згідно з ISO 7498 [23; 43; 46; 47; 60]. Односторонні геш-функції визначені в окремому міжнародному стандарті ISO/IEC 10118 [23; 43].

Вибір та реалізація механізмів забезпечення цілісності та справжності інформаційних ресурсів у сучасних автоматизованих системах є одними з важливих етапів проектування та розробки підсистем захисту інформації. Це пов'язано з постійним зростанням послуг, які надаються різними мережними службами. Більшість послуг надаються при відсутності фіксованих мережених адрес клієнтів та їх особливостей. В зв'язку з чим, ризик порушення цілісності та автентичності інформації збільшується. Для захисту від таких загроз безпеки інформації, як правило, використовують механізми гешування даних – ключові та безключові геш-функції. Геш-функції також можуть використовуватись у складі електронного цифрового підпису, який є потужним механізмом забезпечення автентифікації в сучасних автоматизованих системах.

До обговорення аспектів безпеки визначимо більш точні визначення геш-функції та її криптографічних властивостей.

Геш-функція – це функція $h : D \to R$, де область визначення $D = \{0,1\}^*$ і область значень $R = \{0,1\}^n$ для деякого $n \geq 1$.

Компресійна функція – це функція $y_1 = h(x_1)$, де $D = \{0,1\}^a \times \{0,1\}^b$ і $R = \{0,1\}^n$ для деяких a, b і $n \geq 1$, з $a + b \geq n$.

Ітеративний геш компресійної функції $f : \left(\{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n\right)$ – це геш-функція $h : \left(\{0,1\}^b\right) \to \{0,1\}^n$, визначена як:

$h(X_1 \ldots X_t) = H_t = H_i = f(H_{i-1}, X_i)$ для $1 \leq i \leq t$ (множина $H_0 \in IV$).

Далі наведені визначення для стійкості за (другим) прообразом і стійкістю до колізій.

Стійкість за прообразом. Геш-функція $h : \{0,1\} \longrightarrow R$ є стійкою за прообразом ступеня $(t,\in)$, якщо не існує імовірнісного алгоритму I_h, який приймає вхід $Y \in_R R$ і виводить значення $X \in \{0,1\}^*$ під час виконання не більше t, де $h(X) = Y$ з імовірністю щонайменше ε, отриманою випадковими виборами I_h і Y.

Стійкість за другим прообразом. Нехай S буде кінцевою підмножиною з $\{0,1\}$. Геш-функція $h : \{0,1\}^* \longrightarrow R$ є стійкою за другим прообразом ступеня $(t,\in,S)$, якщо не існує імовірнісного алгоритму S_h, який приймає вхід $X \in_R S$ і виводить значення $X' \in \{0,1\}^*$ під час виконання не більше t, де $X' \neq X\, h(X') = h(X)$ ймовірністю щонайменше ε, отриманою випадковими виборами S_h і X.

Геш-функції використовуються як будівельний блок у різних криптографічних додатках. Найбільш важливе їх використання для захисту автентифікації інформації і як інструмент для схем цифрових підписів. Геш-функція – це функція, яка відображає вхід довільної довжини в фіксоване число вихідних біт – геш-значення. Для того щоб бути корисною в криптографічних додатках, геш-функція повинна задовольняти деякі вимоги. Геш-функції можуть поділятися на односторонні геш-функції та стійкі до колізій геш-функції.

Одностороння функція повинна бути стійкою за прообразом і другим прообразом, тобто повинно бути "важко" знайти повідомлення із заданим гешем (прообразом) або яке гешується в одне і те ж значення, що і задане повідомлення (другий прообраз).

Стійка до колізій геш-функція – це одностороння геш-функція, для якої "важко" знайти два різні повідомлення, для яких геш-значення однакове.

Для деяких програм можуть знадобитися додаткові властивості до геш-функцій, наприклад, псевдовипадковість виходу, що генерується. У контраст до інших криптографічних примітивів, обчислення геш-функції не залежить від будь-якої секретної інформації.

Одностороння геш-функція – це функція h, яка задовольняє такі умови:

аргумент X може бути довільної довжини, а результат $h(X)$ має фіксовану довжину n біт;

геш-функція повинна бути односторонньою в тому сенсі, що за заданим Y в образі h складно знайти повідомлення X таке, що

$h(X) = Y$ (стійкі за прообразом) і за заданим повідомленням X і значенням $h(X)$ важко знайти повідомлення $X' \neq X$ таке, що $h(X') = h(X)$ (стійкі за другим прообразом).

Стійка до колізій геш-функція – це функція h, яка задовольняє такі умови:

аргумент x може бути довільної довжини, а результат $h(X)$ має фіксовану довжину n біт;

геш-функція повинна бути односторонньою, тобто стійкою за прообразом і стійкою за другим прообразом.

Для того, щоб геш-функція H вважалася *криптографічно стійкою*, вона повинна задовольняти три основні вимоги, на яких заснована більшість застосувань геш-функцій в криптографії:

незворотність або стійкість до відновлення прообразу: для заданого значення геш-функції m має бути обчислювально неможливо знайти блок даних x, для якого $h(x) = m$;

стійкість до колізій першого роду або відновлення другий прообразів: для заданого повідомлення m повинно бути обчислювально неможливо підібрати інше повідомлення n, для якого $h(n) = h(m)$;

стійкість до колізій другого роду: має бути обчислювально неможливо підібрати пару повідомлень, що мають однаковий геш.

Ці вимоги не є незалежними:

оборотна функція нестійка до колізій першого і другого роду;

функція, нестійка до колізій першого роду, нестійка до колізій другого роду; зворотне неправильне.

Односторонні геш-функції можуть застосовуватися для вирішення інших завдань, наприклад, вироблення ключів і псевдовипадкових чисел. Для застосування в таких завданнях геш-функція повинна задовольняти такі вимоги:

відсутність кореляції – вхідні і вихідні біти не повинні корелювати, тобто зміна будь-якого вхідного біта призводить до великих непередбачуваним змін вихідних бітів;

стійкість до близьких колізій – для заданої односторонньої функції обчислювально неможливо знайти два прообрази x і x', для яких геш-значення $h(x)$ і $h(x')$ відрізнялися б на малу кількість біт;

стійкість до часткової односторонності – обчислювально неможливо відновити будь-яку частину вхідного повідомлення так само, як і всі повідомлення. Більш того, за будь-якої відомої частини вхідного повідомлення обчислювально неможливо відновити частину (відновлення t невідомих біт вимагає не менш ніж 2^{t-1} операцій);

можливість роботи в режимі розтягування – можливість обчислення геш-функції при довжині вхідного повідомлення менше ніж довжина геш-значення.

Вимога до застосовуваних у криптографії геш-функцій з секретним ключем:

обчислювальна стійкість – неможливість знаходження геш-значення для заданого повідомлення без відомого секретного ключа, тобто для заданої ключовою геш-функції і однієї або більше коректних пар прообразів і геш-значень $(x_i, h(x_i,k))$ і невідомому секретному ключі k обчислювально неможливо знайти іншу коректну пару $(x, h(x,k))$ для будь-кого $x \neq x_i$.

Вимога обчислювальної стійкості передбачає виконання вимоги стійкості ключа (за однією або більше коректних пар прообразів і геш-значення $(x_i, h(x_i,k))$ обчислювально неможливо відновити секретний ключ), однак вимога стійкості ключа k не передбачає виконання вимоги обчислювальної стійкості.

Більшість геш-функцій мають ітеративні конструкції в тому сенсі, що вони базуються на функції компресії з фіксованими входами, вони обробляють кожен блок повідомлення подібним чином. Введення X доповнюється за однозначним правилом доповнення до кратності розміру блоку. Зазвичай це також включає додавання загальної довжини входу в бітах. Доповнений вхід потім ділиться на t блоків, які охоплюють від X_1 до X_T. Геш-функція включає компресійну функцію f і зв'язує змінну H_i між стадією $i-1$ і стадією i.

Класифікацію геш-функцій нведено на рис. 6.1.

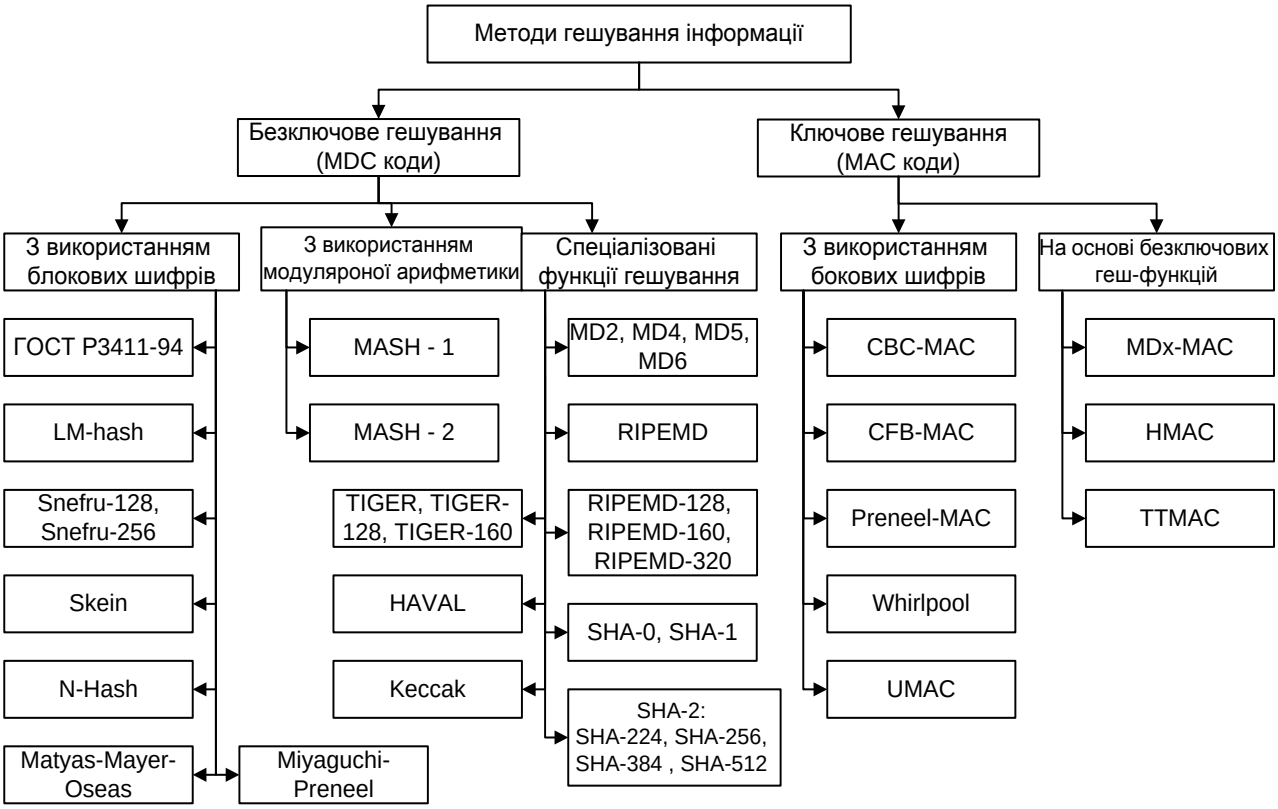


Рис. 6.1. **Класифікація геш-функцій**

До безключових геш-функцій відносяться коди виявлення змін повідомлення (MDC-код, modification detection code), також відомі як коди виявлення маніпуляцій над повідомленнями або коди цілісності повідомлень.

Суттєвим недоліком безключових геш-функцій є те, що вони не захищені від можливості по підбору такого ж самого повідомлення з однаковим гешем, та мають відсутність властивості обчислювальної стійкості. Зрештою MDC-коди забезпечують, спільно з іншими механізмами, цілісність даних.

До ключових геш-функцій відносяться *MAC*-коди.

Визначення автентифікуючих кодів повідомлення згідно з Преніле м (Preneel):

MAC – функція h, що задовольняє такі умови:

1. Аргумент X може бути довільної довжини й результат $h(K;X)$, має фіксовану довжину n біт, де вторинний вхід K позначає секретний ключ.

2. При наявності даних h і X (але з невідомим K), повинне бути складно визначити $h(K;X)$ з імовірністю успіху значно більшою $1/2^n$.

Навіть при великій кількості відомих пар $\{X_i; h(K; X_i)\}$ складно визначити ключ K або обчислити $h(K; X')$ для будь-якого $X' \neq X_i$.

Більшість MAC є повторюваними конструкціями, у тому розумінні, що вони засновані на функції стиску з фіксованим розміром вхідних значень; вони обробляють кожний блок повідомлень аналогічниим способом. Вхід X є однозначним заповненням, кратним розміру блоку. Звичайно це також включає збільшення загальної довжини на бітах вхідних значень. Заповнений вхід потім розділяється на t блоків, що позначають X_1 через X_t. MAC включає функцію стиску f і єднальну змінну H_i між етапом $i-1$ і етапом i:

$$H_0 = IV_K$$
$$H_i = f_K(H_{i-1}, X_i), 1 \leq i \leq t,$$
$$h(K; X) = g_K(H_t).$$

Тут IV позначає початкове значення й g – вихідне перетворення. Секретний ключ K може бути застосований в IV, у функції стиску, і/або у вихідному перетворенні.

Існує кілька алгоритмів, розроблених для специфічної мети автентифікації повідомлення. У більшості випадків код автентифиікації повідомлення створюється із блокового шифру або з геш-функції. Різні методи є сімействами універсальних геш-функцій. Також поєднують процедури, початі декількома організаціями для стандартизації кодів автентифікації повідомлення.

MAC на основі блокового шифру

Найбільш загальним шляхом використання MAC у блоковому шифрі є використання шифру в режимі CBC (блокове з'єднання шифру): ключ MAC використовується як код на кожному кроці ітерації, і блок повідомлення, що обробляється на поточній ітерації, виступає у якості перекладу відкритого тексту в шифр, після побітового додавання з вихідним значенням шифротекста з попереднього кроку:

$$H_1 = E_k(X_1), H_i = E_K(X_i \oplus H_{i-1})(2 \leq i \leq t).$$

Тут допускається, що повідомлення X (після заповнення), ділиться на $X_1 ..., X_t$ блоків з довжиною, що відповідає блокового шифру, який використовується. EK означає шифрування із секретним ключем K і H_t формує вихідне значення алгоритму MAC.

Окремо можна виділити три підходи побудови MAC-Кодів:

1. MAC-Коди, побудовані на основі БСШ (CBC-MAC).

2. MAC-Коди, побудовані на основі безключових геш-функцій (HMAC, MDX-MAC).

3. MAC-Коди, побудовані на основі сімейства універсальних геш-функцій.

Основна конструкція CBC підходить для атаки exor-підробки, отже, може бути використана в додатках, де повідомлення мають фіксовану довжину. Трохи більш безпечні зміни в схемі, проте, існують.

Прикладом є алгоритм EMAC – це використання додаткового шифрування як вихідного перетворення, де ключ шифрувальної операції може бути похідний від ключа MAC. Інший приклад, відомий як перерозподіл MAC, заміняє останнє шифрування двоключовим потрійним шифруванням. Безпека цих конструкцій може бути доведена на основі припущення, що основний блоковий шифр – псевдовипадковий [23].

Усі ці схеми включені в ISO/IEC 9797-1 [23; 36], стандарт для MAC, що використовує (невизначений) блоковий шифр.

Розробку нових методів виробітку CBC-MAC обумовило появу нових атак на CBC-MAC, що докладно описуються в додатку A ISO/IEC 9797-1. В оновлений стандарт увійшли новий метод доповнення повідомлення і новий алгоритм формування MAC-коду зі спеціальною обробкою першого блоку (такий же, як і обробка останнього блоку). Це робить більш трудомістким вичерпний пошук ключа. Крім того, стандартом вводяться два нових "рівнобіжних" варіанти формування CBC-MAC.

Сутність нового методу доповнення полягає в такому. Рядок даних x доповнюється праворуч необхідною кількістю нулів (можливе доповнення і не нулів) до одержання рядка, довжина якого буде кратною n бітам. Потім отриманий рядок доповнюється ліворуч одним n-розрядним блоком L, що містить двійкове представлення довжини, вираженої в бітах, нерозширеної рядка даних x. При необхідності додатковий блок заповнюється ліворуч нулями до довжини n. У новій версії ISO/IEC 9797-1 пропонується шість алгоритмів виробітку MAC-кодів. Перші три алгоритми вже були описані стандартом 1994 року.

Алгоритм 1. Перший алгоритм є просто виробітком CBC-MAC без якої-небудь додаткової обробки.

Алгоритм 2 є виробітком CBC-MAC з додатковою обробкою другого типу (з додатковим шифруванням останнього блоку на ключі k').

Алгоритм 3 є виробітком CBC-MAC з додатковою обробкою першого типу (з додатковим дешифруванням на ключі k' і шифруванням останнього блоку на ключі *k*). Таким чином, останній блок піддається потрійному шифруванню.

Алгоритм 4. У цьому алгоритмі перший і останній блоки повідомлення піддаються подвійному шифруванню.

Алгоритм 5. Алгоритм будується на принципі рівнобіжного застосування двох алгоритмів 1 з різними ключами. Два вихідних значення потім складаються за модулем 2, утворити результуючий MAC-код.

Алгоритм 6. Алгоритм також будується на принципі рівнобіжного застосування двох алгоритмів з різними ключами. Тільки як основа береться алгоритм 4. Два вихідних значення складаються за модулем 2, утворити результуючий MAC-код.

Стандарт функції гешування ГОСТ Р 34.11-94

Стандарт ГОСТ Р 34.11-94 визначає алгоритм і процедуру обчислення геш-функції для будь-яких послідовностей двійкових символів, що застосовуються в криптографічних методах обробки і захисту інформації. Цей стандарт базується на блоковому алгоритмі шифрування ГОСТ 28147-89, хоча, в принципі, для бізнес-структур можна використовувати й інший блоковий алгоритм шифрування з 64-бітовим блоком і 256-бітовим ключем, який має меншу обчислювальну складність.

Геш-функція ГОСТ Р 34.11-94 формує 256-бітове геш-значення.

Функція стиску $H_i = f(M_i, H_{i-1})$ (обидві операнди M_i і H_{i-1} є 256-бітовими величинами) визначається таким чином:

1. Генеруються 4 ключі шифрування K_j, $j = 1...4$, шляхом лінійного змішування M_i, H_{i-1} і деяких констант C_j.

2. Кожен ключ K_j, використовують для шифрування 64-бітових підслів h_i слова H_{i-1} у режимі простій заміни: $S_j = E_{k_1}(h_j)$. Результуюча послідовність S_4, S_3, S_2, S_1 завдовжки 256 біт запам'ятовується в тимчасовій змінній S.

3. Значення H_i є складною лінійною функцією змішування S, M_i і H_{i-1}.

При обчисленні остаточного геш-значення повідомлення M враховуються значення трьох зв'язаних між собою змінних:

H_n – геш-значення останнього блоку повідомлення;

Z – значення контрольної суми, що отримується при складанні за модулем 2 всіх блоків повідомлення;

L – довжина повідомлення.

Ці три змінні і доповнений останній блок M_n повідомлення об'єднуються в остаточне геш-значення таким чином:

$$H = f(Z \oplus M', f(L, f(M' H_n))).$$

Ця геш-функція (рис. 6.2) визначена стандартом ГОСТ Р 34.11-95 для використання спільно із стандартом електронного цифрового підпису ДСТУ 4145 або ГОСТ Р 34.310-95.

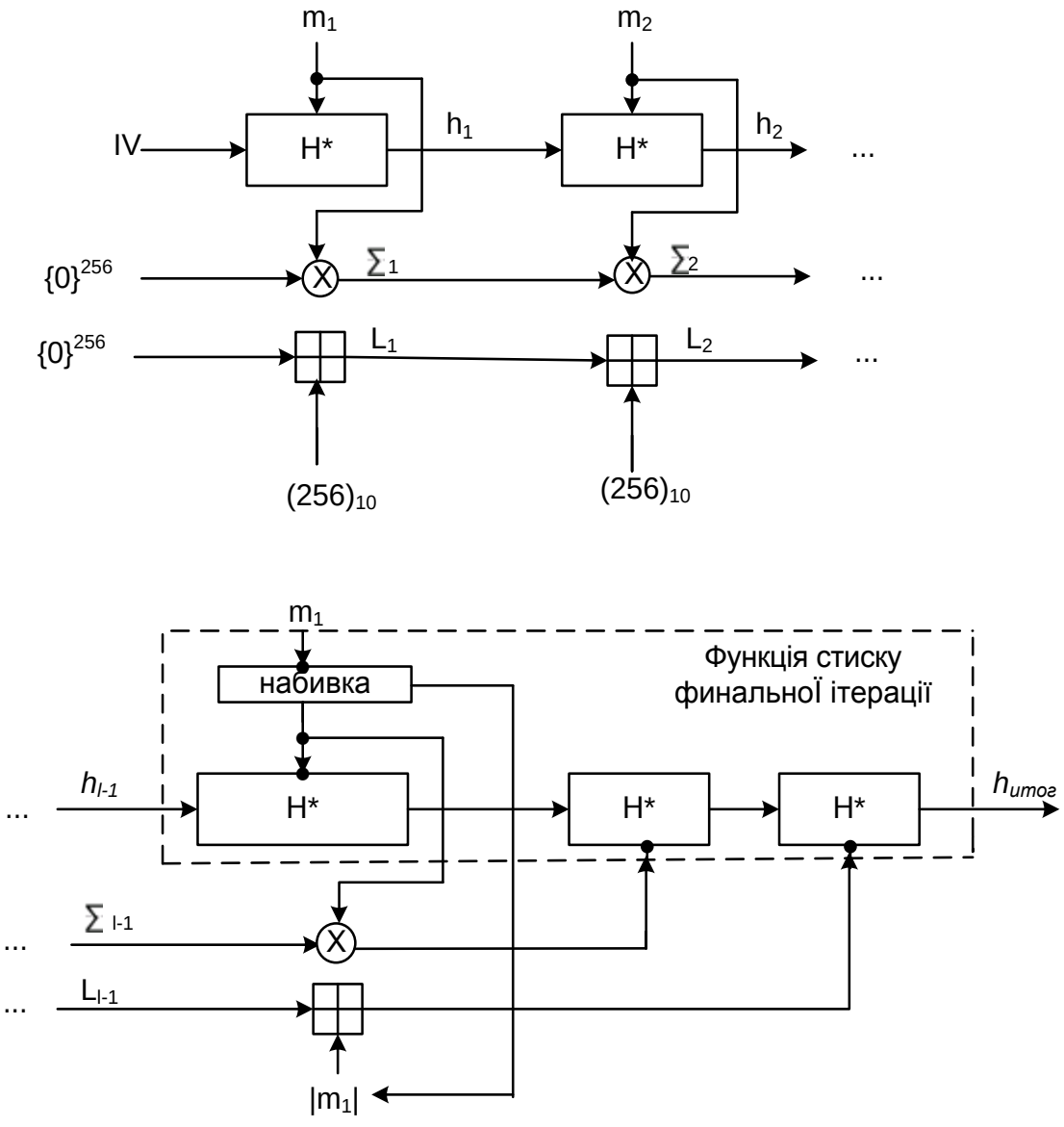


Рис. 6.2. **Геш-функція ГОСТ Р 34.11-94**

За оцінками провідних спеціалістів практична реалізація алгоритму гешування даних ГОСТ Р 34.11-94, ГОСТ 28147-89 (у 4-му режимі) є низько швидкісною. На сучасній ПЕОМ досягає приблизно 100 Кбіт/с,

та має розмір MAC-коду 64 біти, що також вже є недостатнім для багатьох практичних задач. Тому в арсенал сучасного криптографа повинні входити більш гнучкі алгоритми гешування, які володіють теоретичними границями стійкості та дозволяють отримувати високі показники швидкості гешування даних.

Особливості ГОСТ Р 34.11-94.

При обробці блоків використовуються перетворення за алгоритмом ГОСТ 28147-89; обробляється блок довжиною 256 біт, і вихідне значення теж має довжину 256 біт; застосовані заходи боротьби проти пошуку колізій, заснованому на неповноту останнього блоку; обробка блоків відбувається за алгоритмом шифрування ГОСТ28147-89, який містить перетворення на S-блоках, що істотно ускладнює застосування методу диференціального криптоаналізу до пошуку колізій.

Формат виводу.

Згідно зі стандартом, результатом геш-функції є 256-бітове число. Стандарт не вказує, як воно повинно виводитися. Різні реалізації використовують різні формати виводу, що вкупі з двома поширеними S-блоками посилює плутанину. ГОСТР 34.11-94 оперує з Little-endian числами. Багато реалізації (зокрема rhash, mhash library, консольна утиліта openssl) виводять 32 байта результуючого геша в шістнадцятковому поданні, в порядку, в якому вони розташовуються в пам’яті – молодші байти першими. Це подання виправдовується тим, що воно ж використовується при виведенні геш сум широко поширених західних алгоритмів MD5, SHA1, Tiger, Whirlpool та ін.

GOST("Thisismessage,length=32bytes")=B1C466D37519B82E8319819FF32595E047A28CB6F83EFF1CЯ6916A815A637FFFA.

У наведених в стандарті прикладах, результуючий геш записується як шістнадцяткове подання 256-бітного Little-endian числа. Тим самим, виходить зворотний порядок байт (старші розряди першими). Такий же порядок використовує, зокрема, програма gostsum, що постачається з оригіналами бібліотеки OpenSSL.

H = FAFF37A6 15A81669 1CFF3EF8 B68CA247 E09525F3 9F811983 2EB81975 D366C4B1.

Оцінка криптостійкості.

У 2008 році командою експертів з Австрії та Польщі була виявлена технічна вразливість, що скорочує пошук колізій в 2^{23} разів. Кількість операцій, необхідне для знаходження колізії, таким чином, становить 2105, що, однак, на даний момент практично не реалізовується. Проведення колізійної атаки на практиці має сенс тільки у випадку цифрового підпису документів, причому якщо зловмисник може змінювати непідписаний оригінал.

Використання.

Функція використовується при реалізації систем цифрового підпису на базі асиметричного крипто алгоритму за стандартом ГОСТ Р 34.10-2001. Спільнота російських розробників СКЗІ погодило використовувані в Інтернет параметри ГОСТР 34.11-94, см. RFC 4357.

Використання в сертифікатах відкритих ключів;

використання для захисту повідомлень в S/MIME (Cryptographic Message Syntax, PKCS # 7);

використання для захисту з'єднань в TLS (SSL, HTTPS, WEB); використання для захисту повідомлень в XML Signature (XML Encryption);

захист цілісності Інтернет адрес та імен (DNSSEC).

MAC на основі геш-функцій

Код автенфікації повідомлення може бути також отриманий з геш-функції, включаючи секретний ключ в обчисленні. Це найбільш загальний метод, оскільки такі MAC звичайно швидше, ніж MAC на основі блокового шифру. Проте просто додаючи або додаючи ключовий фрагмент на вхід повідомлення геш-функції не дає безпечний MAC.

HMAC – вкладена конструкція, яка обчислює MAC для основної функції геша h, повідомлення X і секретний ключ K таким способом:

$$HMAC(K, X) = h((K \oplus opad) \| h((K \oplus ipad) \| X))).$$

Ключ K спочатку заповнюється нульовими бітами, і $opad$ і $ipad$ – постійні величини. Беллейр (Bellare) довів безпеку цієї конструкції такими припущеннями: основна функція геша є стійкої до помилок для секретної початкової величини; функціональної по ключу стиску початкової величини – безпечний алгоритм MAC (для повідомлень одного блоку); функція стиску є слабкою псевдодовільною функцією.

Альтернативою для HMAC є конструкції MDx-MAC [23], які можуть бути засновані на MD5, SHA, RIPEMD або аналогічних функціях геша.

Тут, основна геш-функція перетворена в MAC невеликими модифікаціями, включаючи секретний ключ на початку, в остаточному підсумку й у кожній ітерації функції геша. Це досягається ключовими залежними модифікаціями початкової величини й константи значення, що додається, яке використовується геш-функцією, і вихідним перетворенням, залежним від ключа. Безпека MDx-MACможе бути доведена на основі припущення, що основна функція стиску псевдодовільна.

Як HMAC, так і MDx-MAC включені в ISO/IEC 9797-2 [23; 36], стандарт для MAC, що використовував власну геш-функцию.

MAC, засновані на універсальному гешуванні

Сімейство геш-функцій геша $H = \{h : D \to R\}$ – це кінцева множина функцій із загальною областю визначення D і (кінцевим) діапазоном R. Його можна також позначити через $H : K \times D \to R$ де $H_k : D \to R$ – функція множини для кожного $K \in K$. В останньому випадку можна вибрати довільну функцію h з множини $K \in K$ й $h = HK$.

Множина універсальних геш-функцій є сімейством геш-функцій з деякою комбінаторною властивістю. Наприклад, сімейство геш-функцій $H = \{h : D \to R\}$ є майже універсальним, якщо для будь-якого різного $X, X' \in D$, імовірність, що $h(X) = h(X')$ – не більш, ніж коли $h \in H$ вибирається довільно.

Це може використовуватися для автентифікації повідомлення, наприклад при гешуванні повідомлення з функцією із сімейства універсальних геш-функцій функції, що кодує вихід геша, й потім здійснює закодований вихід геша як величину MAC. Це підтверджується тим, що безпека результуючої схеми MAC залежить від безпеки шифру, використаного для кодування вихідного геша. Комбінаторні властивості сімейства універсальних геш-функцій часто нескладно довести і схеми, що отримуються на виході є найшвидшими серед MAC. Різновид NESSIE UMAC – приклад MAC, заснований на універсальній геш.

Сучасні стандарти.

Кілька організацій здійснюють ініціативи стандартизації автентифікації кодів повідомлення. ISO/IEC розробив стандарт 9797 для MAC у двох окремих частинах. Частина 9797-1 [23; 36] описує MAC, заснований на блоковому шифрі, а саме CBC-MAC для невизначеного блокового шифру (з деякими додатковими розширеннями, включаючи EMAC і перерозподіл-MAC). Розділ 9797-2 [23; 36] деталі MAC засновані

у відданій функції геша, а саме HMAC і конструкції Mdx-mac для невизначеної геш-функції (і варіант Mdx-mac для коротких вхідних значень).

ANSI прийняв базований DES CBC-MAC (включаючи перерозподіл-MAC) у своєму стандарті X9.19 [23; 36] і HMAC (з невизначеною геш-функцією) у стандарті X9.71 [17]. NIST розробив FIPS 113 [23; 36] для DES CBCMAC і FIPS 198 [23; 36] для SHA-1 HMAC.

Рівень безпеки залежить від розміру внутрішнього блоку (розмір блоку функції гешування), від довжини ключа і від довжини значення MAC-коду. Серед основних недоліків алгоритмів, які вже розглянуті, є погане розсіювання та використання для генерації ключа алгоритму DES (AES). Щодо використання алгоритмів DES та AES для отримання стійкої ключової послідовності, то це накладає обмеження на швидкість гешування самого алгоритму *MAC*-коду. У табл. 6.1 наведено аналіз тестування швидкості роботи алгоритмів гешування.

Таблиця 6.1

Аналіз тестування швидкості роботи алгоритмів гешування

Функція гешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
Whirlpool	10	C	28,013 Мбіт/с	46,961 Мбіт/с
SHA-2 (512)	80	C	41,159 Мбіт/с	68,701 Мбіт/с
SHA-2 (256)	64	C	81,308 Мбіт/с	135,557 Мбіт/с
ГОСТ 34311-95	-	C+Assembler	49,408 Мбіт/с	83,056 Мбіт/с
HAVAL	96(128, 160)	C	337,842 Мбіт/с	564,809 Мбіт/с
SHA-1	80	C, Assembler	206,285 Мбіт/с 361,581 Мбіт/с	344,433 Мбіт/с 605,558 Мбіт/с
RIPEMD-160	160	C	147,465 Мбіт/с	246,568 Мбіт/с
MD5	64	C	278,715 Мбіт/с	574,635 Мбіт/с
MD4	48	C	344,086 Мбіт/с	467,793 Мбіт/с
UMAC	-	C C+Assembler	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с
ГОСТ 28147-89	16	C+Assembler	189,559 Мбіт/с	315,270 Мбіт/с

У табл. 6.2 наведено можливі значення довжин ключа і геш-коду для різних алгоритмів сімейства MAC.

Таблиця 6.2

Довжина ключа k і довжина геш-коду n для MAC-кодів

Алгоритм	k (ключ)	n (геш-код)
UMAC	128	64
TTMAC	160	≤ 160
EMAC-AES	128,192,256	≤ 128
RMAC-AES	128,192,256	≤ 128
HMAC-SHA-1	≤ 512	≤ 160

У табл. 6.3 наведено основні результати оцінки швидкодії MAC-алгоритмів для різних операційних платформ. Швидкість обчислень визначається кількістю циклів процесора, які затрачуються на один байт оброблюваного повідомлення.

Таблиця 6.3

Швидкодія MAC-алгоритмів

Алгоритм	Довжина MAC-коду (біти)	Довжина ключа (біти)	Тип ПК				
			Pentium2	PIII/Linux	Pentium4	Xeon	AMD
TTMAC	160	160	21	21	40	37	21
UMAC-16	64	128	6.1	6.0	6.2	6.1	6.2
UMAC-32	64	128	2.5	2.9	6.7	6.6	1.9
HMAC-Whirlpool	512	512	86	72	98	103	100
HMAC-MD4	128	512	4.7	4.7	6.4	6.4	4.7
HMAC-MD5	128	512	7.2	7.3	9.4	9.4	7.4
HMAC-RIPE-MD	160	512	23	18	27	26	21
HMAC-SHA-0	160	512	16	15	23	23	13
HMAC-SHA-1	160	512	16	15	25	24	12
HMAC-SHA-2	256 384 512	512	40 84 84	39 84 84	40 124 124	39 132 132	33 72 72
HMAC-Tiger	192	512	24	21	28	26	20
CBC MAC-Rijndael (EMAC)	128	128	24	26	26	27	31
CBC MAC-DES	64	56	62	61	72	69	54
CBC MAC-Shacal	512	160	31	31	67	74	29

Аналіз табл. 6.3 показує, що схеми ключового гешування UMAC на основі поліноміальних функцій дозволяють одержати найвищу швидкість гешування.

Геш-функції RIPEMD-160, RIPEMD-128

У 1995 році були виявлені деякі недоліки в першій версії геш-функції RIPEMD, що формує 128-розрядний геш-код. Зокрема, були виявлені колізії для останніх двох із трьох циклів RIPEMD. Розроблена методика пошуку колізій була успішно застосована і для пошуку колізій у MD4. На основі отриманих результатів були зроблені припущення, що ця методика може бути використана для пошуку колізій у MD5 і в повній версії RIPEMD. Версія RIPEMD (разом з SHA-1) широко використовується в банківських технологіях. Однак ситуація з криптоаналізом геш-функцій сімейства MD*x* і перспективи розвитку обчислювальних потужностей призвели до висновку про необхідність відновлення RIPEMD. У результаті для стандартизації були запропоновані більш могутні версії геш-функції – RIPEMD-160 і RIPEMD-128. За оцінками розроблювачів геш-функція RIPEMD-160 повинна залишатися безпечною протягом десяти найближчих років. Удосконалена версія RIPEMD-128 замінила геш-функцію RIPEMD.

Розглянемо ці функції більш докладно.

Розмір геш-кода і перемінних зчеплень RIPEMD-160 дорівнює 160 бітам (п’ять 32-х розрядних слів). Кількість циклів збільшена з трьох (як у RIPEMD) до п’яти. Крім того, внесені додаткові зміни в паралельні лінії алгоритму. Тепер права і ліва лінії відрізняються не тільки застосовуваними константами, але і порядком проходження бульових функцій. Для алгоритму RIPEMD-160 визначені такі бульові функції:

$$f(u,v,w) = u \oplus v \oplus w;$$
$$g(u,v,w) = (u \wedge v) \vee (\bar{u} \wedge w);$$
$$h(u,v,w) = (u \vee \bar{v}) \oplus w;$$
$$k(u,v,w) = (u \wedge w) \vee (v \wedge \bar{w});$$
$$l(u,v,w) = u \oplus (v \vee \bar{w}).$$

Усі позначення аналогічні раніше використаним позначенням при описі алгоритму SHA-1.

На рис. 3.27 наведена загальна структура геш-функції RIPEMD-160.

Алгоритм RIPEMD-160.

Вхід. Двійковий рядок *x* довжиною $b \geq 0$

Вихід. 160-бітний геш-код за рядком **x.**

1. *Ініціалізація.*

а) ініціалізувати п'ять векторів ініціалізації:

$$h_1 = 67452301_x;\ h_2 = efcdab89_x;\ h_3 = 98badcfe_x;$$

$$h_4 = 10325476_x;\ h_5 = c3d2e1f0_x;$$

б) задати для лівої і правої ліній алгоритму додаткові константи:

- ліва лінія:

$y_L[j] = 00000000_x,\ 0 \leq j \leq 15;$

$y_L[j] = 5a827999_x = \lfloor 2^{30} \times \sqrt{2} \rfloor,\ 16 \leq j \leq 31;$

$y_L[j] = 6ed9eba1_x = \lfloor 2^{30} \times \sqrt{3} \rfloor,\ 32 \leq j \leq 47;$

$y_L[j] = 8f1bbcdc_x = \lfloor 2^{30} \times \sqrt{5} \rfloor,\ 48 \leq j \leq 63;$

$y_L[j] = a953fd4e_x = \lfloor 2^{30} \times \sqrt{7} \rfloor,\ 64 \leq j \leq 79;$

- права лінія:

$y_R[j] = 0 \times 50a28be6 = \lfloor 2^{30} \times \sqrt[3]{2} \rfloor,\ 0 \leq j \leq 15;$

$y_R[j] = 0 \times 5c4dd124 = \lfloor 2^{30} \times \sqrt[3]{3} \rfloor,\ 16 \leq j \leq 31;$

$y_R[j] = 0 \times 6d703ef3 = \lfloor 2^{30} \times \sqrt[3]{5} \rfloor,\ 32 \leq j \leq 47;$

$y_R[j] = 7ad76e9_x = \lfloor 2^{30} \times \sqrt[3]{7} \rfloor,\ 48 \leq j \leq 63;$

$y_R[j] = 00000000_x,\ 64 \leq j \leq 79;$

в) задати порядок подачі слів повідомлення на ліву $z[j]$ і праву $z[j]$ лінії. Порядок подачі слів визначається на основі таких перестановок:

для лівої лінії визначається перестановка ρ, а для циклів порядок слів визначається в такий спосіб:

i	z_L[0..15]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	z_L[16..31]	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
$\rho^2(i)$	z_L[32..47]	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
$\rho^3(i)$	z_L[48..63]	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
$\rho^4(i)$	z_L[64..79]	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13

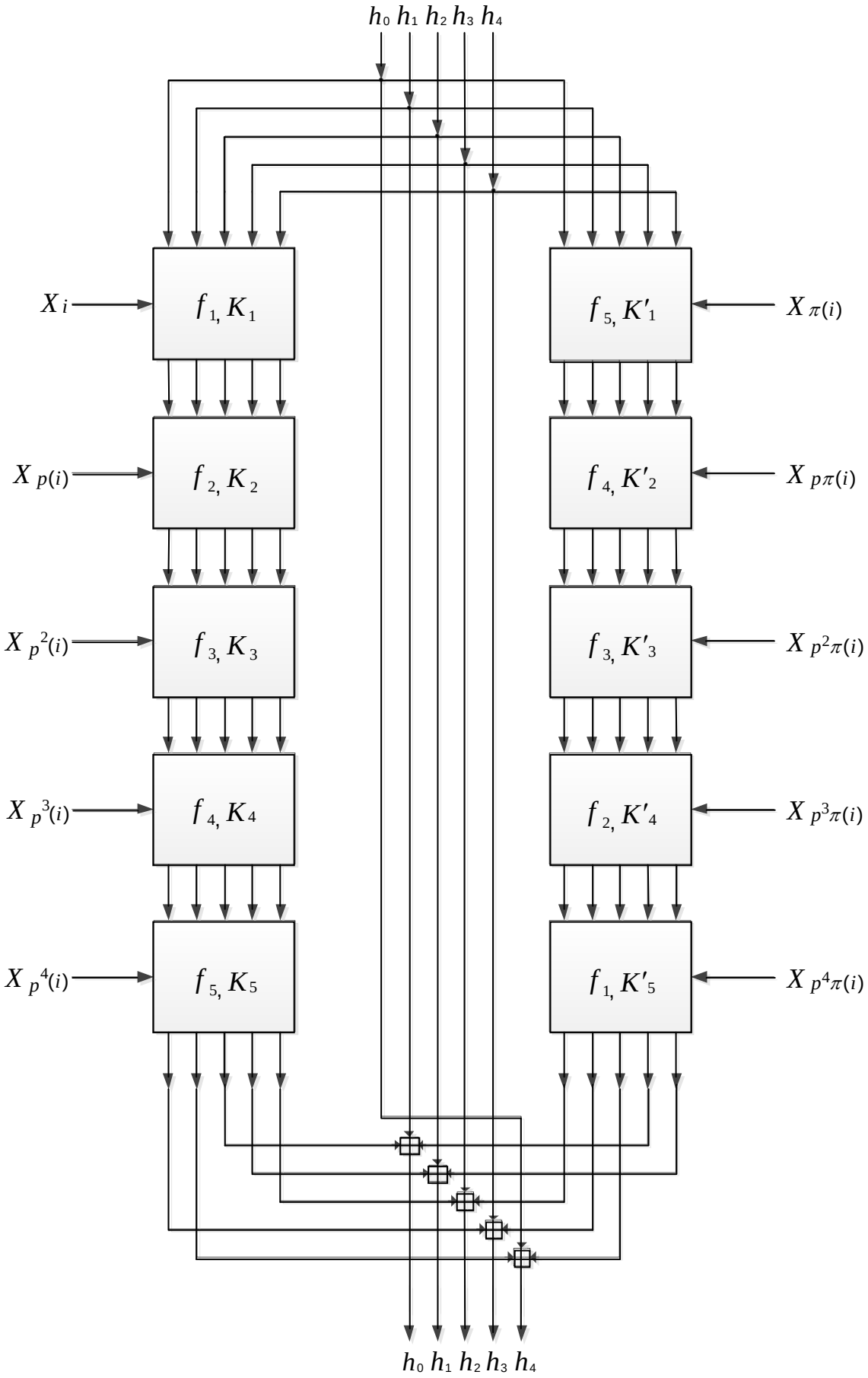


Рис. 6.3. **Схема геш-функції RIPEMD-160**

Для правої лінії визначається перестановка $\pi(i) = 9i + 5(\lfloor 16 \rfloor)$ і порядок подачі слів визначається з використанням перестановки ρ у такий спосіб:

$\pi(i)$	z_R[0..15]	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
$\rho\pi(i)$	z_R[16..31]	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
$\rho^2\pi(i)$	z_R[32..47]	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
$\rho^3\pi(i)$	z_R[48..63]	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
$\rho^4\pi(i)$	z_R[64..79]	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11

г) визначити для лівої $s[j]$ і правої $s[j]$ ліній аргументи для операторів циклічного зсуву відповідно до таблиці:

s_L[0..15]	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
s_L[16..31]	7	6	8	13	11	9	7	15	7	12	15	9	11	7	13	12
s_L[32..47]	11	13	6	7	14	9	13	15	14	8	13	6	5	12	7	5
s_L[48..63]	11	12	14	15	14	15	9	8	9	14	5	6	8	6	5	12
s_L[64..79]	9	15	5	11	6	8	13	12	5	12	13	14	11	8	5	6
s_R[0..15]	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6
s_R[16..31]	9	13	15	7	12	8	9	11	7	7	12	7	6	15	13	11
s_R[32..47]	9	7	15	11	8	6	6	14	12	13	5	14	13	13	7	5
s_R[48..63]	15	5	8	11	14	14	6	14	6	9	12	9	12	5	15	8
s_R[64..79]	8	5	12	9	12	5	14	6	8	13	6	5	15	13	11	11

2. Передобробка. Доповнити рядок *x* так, щоб його довжина була кратна числу 512. Для цього додати в кінець рядка одиницю, а потім стільки нульових біт, скільки необхідно для одержання рядка довжиною на 64 біта коротше довжини кратної 512. Нарешті, додати останні два 32-х розрядні слова, що містять двійкове подання числа b. Нехай m – кількість 512-бітних блоків в отриманому доповненому рядку. У такий спосіб форматований вхід складається з 16^m 32-х розрядних слів.

$$X_0 X_1 \text{ к } X_{16m-1}.$$

Ініціалізувати вектор змінних зчеплення:

$$(H_1, H_2, H_3, H_4 . H_5) = (h_1, h_2, h_3, h_4, h_5).$$

3. Обробка. Для всіх *i* від 0 до *m*-1 кожен *i*-й блок з шістнадцяти 32-х бітних слів записати в тимчасовий масив:

$$X[j] = x_{16i+j}, 0 \le j \le 15.$$

Потім:

а) виконати п'ять 16-ти крокових циклів лівої лінії.

Ініціалізувати робочі змінні:

$$(A_L, B_L, C_L, D_L, E_L) = (H_1, H_2, H_3, H_4, H_5).$$

Виконати:

Для $j = 0$ до 15

$$t = (A_L + f(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для $j = 16$ до 31 X_i

$$t = (A_L + g(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для $j = 32$ до 47

$$t = (A_L + h(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для $j = 48$ до 63

$$t = (A_L + k(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для $j = 64$ до 79

$$t = (A_L + l(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

б) виконати паралельно з циклами лівої лінії п'ять 16-ти крокових циклів правої лінії. Ініціалізувати робочі змінні:

$$(A_R, B_R, C_R, D_R, E_R) = (H_1, H_2, H_3, H_4, H_5).$$

Виконати:

Для $j = 0$ до 15

$$t = (A_R + I(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для $j = 16$ до 31

$$t = (A_R + k(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для $j = 32$ до 47

$$t = (A_R + h(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для $j = 48$ до 63

$$t = (A_R + g(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для $j = 64$ до 79

$$t = (A_R + I(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

в) оновити змінні зчеплення:

$$t = H_1;$$

$$H_1 = H_2 + C_L + D_R;$$

$$H_2 = H_3 + D_L + E_R;$$

$$H_3 = H_4 + E_L + A_R;$$

$$H_4 = H_5 + A_L + B_R;$$

$$H_5 = t + B_L + C_R.$$

4. *Завершення*.

Як геш-код прийняти величину: $H_1 \parallel H_2 \parallel H_3 \parallel H_4 \parallel H_5$.

Основні відмінності алгоритму RIPEMD-128 від RIPEMD-160 полягають в такому:

довжина геш-кода складає 128 біт (чотири 32-х бітні слова);

функція складається з чотирьох циклів і, відповідно, використовується чотири 32-х розрядні змінні зчеплення;

основна операція циклу має вигляд:

$$t = (A + f(B,C,D)) + X(z[j] + y[j]);$$
$$(A,B,C,D) = (D,D + (t <<< s[j]),A,B,C);$$

визначені тільки чотири бульові функції f, g, h і k;

з переліку констант виключаються значення $[2^{30}\sqrt{7}]$ і $[2^{30}\sqrt[3]{7}]$, а з переліку векторів ініціалізації виключають h_5.

Деякі додатки, у яких використовують геш-функції, можуть вимагати виробітки більш довгих геш-кодів. Прямим шляхом розв'язання цієї задачі є паралельне виконання однієї і тієї ж геш-функції з різними векторами ініціалізації. Однак у цьому випадку можлива поява залежностей між двома результатами гешування, що і було виявлено при використанні MD4. Геш-функції RIPEMD-128 і RIPEMD-160 уже мають паралельні лінії і для одержання геш-кодів довжиною, відповідно, 256 і 320 біт досить виключити кінцеву комбінацію результатів гешування по кожній лінії.

6.1.1. Колізійно-стійкі функції гешування Whirpool та SHA-256, SHA-384, SHA-512

Whirlpool

Whirlpool – це стійка до збігів (колізій) геш-функція, представлена в NESSIE Пауло Баррето й Вінсентом Райменом. Конструкція Whirlpool базується на основі блокового шифрі, який використовується в так званому "Миагучі – Прінель" – режимі, що гешує. Цей блоковий шифр подібний за структурою із шифром AES, але працює з більшими блоками 512 біт.

Короткий опис примітива.

Whirlpool відображає повідомлення M, що складається з довільної кількості біт, в 512-бітне геш-значення Whirlpool(M). Алгоритм базується на функції компресії, яка використовується ітеративно на блоках повідомлення в 512 біт. Ця функція базується на лежачому в основі спеціально призначеному 512-бітному блоковому шифрі, що використовує 512-бітний ключ.

Спочатку повідомлення розширюється до необхідної довжини й ділиться на блоки повідомлення $M_0 \ldots M_{t-1}$, де кожний блок $M_i (0 \le i \le t-1)$ складається з 512 біт. Початкове значення визначене як рядок, що складається з 512 біт-нулів: $M_0 = 0$

Компресійна функція Whirlpool перетворює вхідне значення H_i і блок повідомлення Mi у вихідне значення H_{i+1}. Ця функція залежить від зашифрування із внутрішнім блоковим шифром W, де M_i формує відкритий текст, і H_i служить як ключ. Далі отриманий шифртекст складається за модулем 2 з M_i і H_i (усі ці рядки по 512 біт). Тобто $H_{i+1} = W_{H_i}(M_i) \oplus M_i \oplus H_i$. У наступному використанні компресійна функція H_{i+1} формує вхід спільно з наступним блоком повідомлення M_{i+1}. Результат цього – в наступному описі алгоритму Whirlpool.

Розширення повідомлення M і розподіл його на блоки повідомлення $M_0 \ldots M_{t-1}$.

Визначення початкового значення $M_0 = 0$.

Для / використання функції компресії для обчислення /.

Встановлення геш-значення Whirlpool (M) = H_t.

Надійність і продуктивність.

Надійність Whirlpool може бути доведена, базуючись на припущенні, що певні ідеальні (бажані) властивості справедливі для лежачого в основі блокового шифру W. Цей блоковий шифр дуже схожий з AES. Не існує відомих більш швидких, ніж перебір, атак на Whirlpool, і алгоритм має високий рівень надійності завдяки довгому геш-значенню (512 біт).

Продуктивність Whirlpool залежить від продуктивності лежачого в основі блокового шифру, який досить ефективний. Однак він потребує перерахування процедури формування ключів для цього шифру для кожного використання функції компресії (це для кожного блоку повідомлення з 512 біт). Структура алгоритму не орієнтована на якусь певну платформу, але можуть бути зроблені різні оптимізації для різних платформ. Також спеціальна структура компонентів, особливо S-блоку, що використовується, дозволяє виконувати ефективні апаратні реалізації.

SHA-256, SHA-384 and SHA-512

У 1993 році НІСТ видав FIPS-180 [23; 36], який представив стандартну геш-функцію, названу SHA. В FIPS-180-1 [23; 36] від 1995 року SHA був поліпшений (додана ротація вліво на 1 бітову позицію) і поліп-

шена версія була названа SHA-1. Обидві версії мають розмір дайджесту 160 біт. Тільки у роботі Саарінєна [61] була наведена слайд-атака на SHA-1.

Оскільки індустрія захисту інформації вимагала більших запасів надійності, у серпні 2002 року НИСТ опублікував ще три геш-функції: SHA-256, SHA-384 і SHA-512. FIPS-180-2 [23; 36] визначає ці три нові геш-функції разом з SHA-1 як стандартні геш-функції для використання урядом США. Три нові функції мають більший дайджест-розмір, – 256 біт для SHA-256, 384 біт для SHA-384 і 512 біт для SHA-512.

Опис примітива.

SHA-256 базується на діленні повідомлення на блоки по 512 біт кожний. Кожний блок потім уводиться в компресійну функцію, яка розширює 512-бітний блок до 2048 біт, які використовуються для перетворення поточного значення стану в 256 біт у нове значення стану.

Початковий 256-бітний стан завантажується у вісім 32-бітних слів. Потім протягом 64 раундів ці слова впливають один на одного, використовуючи розширений блок повідомлення й різні функції. Ці 64 раунди можуть також бути використані в режимі блокового шифру (див. "Шакал-2", розд. 4 у ч. В (блокові шифри) NESSIE [61].

SHA-384 і SHA-512 базуються на діленні повідомлення на блоки по 1024 біт кожний. Кожний блок вводиться у функцію компресії й використовується для зміни стану в 512 біт протягом 80 раундів функції компресії.

І SHA-384, і SHA-512 мають подібну структуру до використаної в SHA-256, але розмір слова подвоєний і функції трохи відрізняються. Єдиною відмінністю між SHA-384 і SHA-512 є початковий стан і той факт, що в SHA-384 вихід – це стан, усічений до 384 біт.

Стійкість і продуктивність.

Не оприлюднено ніяких атак проти геш-функцій SHA-256, SHA-384 і SHA-512.

Дані щодо продуктивності для цих трьох геш-функцій показують, що вони дуже ефективні. Програмна реалізація для NESSIE досяглася швидкостей в 30 – 70 циклів (тактів) на байт для SHA-256, і 16-190 циклів (тактів) для SHA-384 і SHA-512. Простота операцій, що використовуються у всіх трьох представлених алгоритмах, вселяє впевненість, що апаратна реалізація буде мати невелику кількість вентилів. Більше того, SHA-256 може бути використана разом з алгоритмом зашифрування Шакал-2 для збереження (зменшення кількості) вентилів або розміру реалізації.

SHA-384 і SHA-512 підігнані для оточення, де існують 64-бітні регістри (або більше), і вони оперують із 64-бітними словами.

Опис. Усі три геш-функції мають ту саму базову конструкцію. Кожна функція додає біти до повідомлення, щоб його довжина була кратною розміру блоку, з якими працює компресійна функція (1024 біт для SHA-384 і SHA-512 і 512 біт для SHA-256).

Нехай *l* буде довжина повідомлення *M*, яке необхідно гешувати. Додаємо 1, за якою іде *k* нульових біт, де *k* – це найменше ненегативне значення, для якого l+1+k ≡ 448 mod 512 для SHA-256 і l + 1 + k ≡ 896 mod 1024 для SHA-384 і SHA-512. Для досягнення кратності довжини блоку, з яким працює геш-функція, потім додаємо довжину повідомлення (mod2^{64} для SHA-256 або mod2128 для SHA-384 і SHA-512).

Як тільки цей крок виконаний, ділимо повідомлення M на блоки однакового розміру (1024 біт для SHA-384 і SHA-512 і 512 біт для SHA-256). Позначимо цю декомпозицію як $M = m_1 m_2 ... m_n$.

Тепер той самий процес виконується для якої-небудь із трьох геш-функцій (константи й точний опис функцій можуть відрізнятися):

1. Початкове значення восьми слів завантажується в 8 слів: A, B, C, D, E, F, G і H.

2. Для i від 1 до n виконати:

-Set AA = A, BB = B, CC = C, DD = D, EE = E, FF = F, GG = G and HH = H.

-for t = 0 to 63 (79 for SHA-512 and SHA-384) do:

$$T_1 = H + \sum_1(E) + C_h(E,F,G) + K_t + W_t$$

$$T_2 = \sum_0(A) + M_{aj}(A,B,C)$$

H = G, G = F, F = E, E = D + T_1, D = C, C = B, B = A, A = T_1+T_2

-set A = A+AA, B = B+BB, C = C+CC, D = D+DD, E = E+EE, F = F+FF, G = G+GG, and H = H+HH.

3. Продукувати геш-значення A|B|C|D|E|F|G|H для SHA-256 і SHA-512, і A|B|C|D|E|F для SHA-384.

Додавання виконано за модулем 232 для SHA-256 і 264 для SHA-384 і SHA-512.

K_t – це раундові константи, що додаються на кожному раунді компресійної функції. W_t – це множини 64 або 80 слів, зроблених із блоку повідомлення, який обробляється. $C_h(x,y,z)$ і $M_{aj}(x,y,z)$ – це нелінійні

функції, використовувані в компресійній функції. $\sum_0$ і $\sum_1$ – це дві лінійні функції, які різні для SHA-256 і для SHA-384 і SHA-512.

Кожний блок Mi ділиться на 16 слів $m_i^0, m_i^1, \ldots m_i^{15}$, потім розраховуються W_t:

$$W_t = \begin{cases} m_i^t, 0 \le t \le 15 \\ s_1(W_{t-2}) + (W_{t-7}) + s_0(W_{t-15}), 16 \le t \le 63 (or 79) \end{cases}.$$

Функції s_0 і s_1 є лінійні функції, але вони різні для SHA-256 і для SHA-384 і SHA-512.

Функції, використані в SHA-256.

SHA-256 використовує 6 функцій: $C_h(x, y, z)$ і $M_{aj}(x, y, z)$ – нелінійні функції з 96-бітним входом і 32-бітним виходом, і 4 лінійні функції з 32-бітним входом і 32-бітним виходом: $\sum_0, \sum_1, s_0, s_1$.

Спочатку визначимо 4 лінійні операції:

$$\sum_0(X)=X>>>_2 \oplus X>>>_{13} \oplus X>>>_{22};$$
$$\sum_1(X)=X>>>_6 \oplus X>>>_{11} \oplus X>>>_{25};$$
$$\sigma_0(X)=X>>>_7 \oplus X>>>_{18} \oplus X>>>_3;$$
$$\sigma_1(X)=X>>>_{17} \oplus X>>>_{19} \oplus X>>>_{10}.$$

Операція C_h приймає три 32-бітних слова X, Y, Z. Операція є функцією побітового вибору, тобто якщо і-тий біт X установлений, то і-тий біт виходу є і-тий біт Y. З іншого боку, і-тий біт виходу є і-тий біт Z. Ця операція може бути легко реалізована як:

$$C_h(X, Y, Z) = (X \wedge Y) \oplus (\overline{X} \wedge Z) \quad C_h(X,Y,Z)=(X \wedge Y) \oplus (\overline{X} \wedge Z).$$

Операція M_{aj} також приймає три 32-бітних слова X, Y, Z. Операція є функцією побітової мажоритарності, тобто якщо більшість і-тих бітів X, Y і Z установлені, тоді і-тий біт виходу встановлений, і навпаки. Ця операція може бути легко реалізована як:

$$M_{aj}(X,Y,Z)=(X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z).$$

Функції, використані в SHA-384 і SHA-512.

SHA-384 і SHA-512 використовують 6 функцій $C_h(x,y,z)$ і $M_{aj}(x,y,z)$ нелінійні функції з 192-бітним входом і 64-бітним виходом, і 4 лінійні функції з 64-бітним входом і 64-бітним виходом: $\sum_0$, $\sum_1$, s_0, s_1.

Спочатку визначимо 4 лінійні операції:

$$\sum_0(X)=X>>>_{28}\oplus X>>>_{34}\oplus X>>>_{39};$$
$$\sum_1(X)=X>>>_{14}\oplus X>>>_{18}\oplus X>>>_{41};$$
$$\sigma_0(X)=X>>>_{1}\oplus X>>>_{8}\oplus X>>>_{7};$$
$$\sigma_1(X)=X>>>_{19}\oplus X>>>_{19}\oplus X>>>_{6}.$$

Операція C_h приймає три 64-бітних слова X, Y, Z. Операція є функцією побітового вибору, тобто якщо і-тий біт X установлений, те і-тий біт виходу є і-тий біт Y. З іншого боку, і-тий біт виходу є і-тий біт Z. Ця операція може бути легко реалізована як $C_h(X,Y,Z)=(X\wedge Y)\oplus(\overline{X}\wedge Z)$.

Операція M_{aj} також приймає три 64-бітних слова X, Y, Z. Операція є функцією побітової мажоритарності, тобто якщо більшість і-тих біт X, Y і Z установлені, тоді і-тий біт виходу встановлений, і навпаки. Ця операція може бути легко реалізована як $M_{aj}(X,Y,Z)=(X\wedge Y)\oplus(X\wedge Z)\oplus(Y\wedge Z)$.

MDC, на основі модулярної арифметики

У ISO/IEC 10118-4 визначаються дві геш-функції MASH-1 і MASH-2, основані на ітеративному зведенні в степінь за модулем. Ці функції є покращеними варіантами функцій, визначених у стандартах CCITT X.509:1988 і ISO/IEC 9594-8:1989, що були визнані недостатньо безпечними. Внаслідок цього вони були виключені з обновлених версій стандартів ITU-T X.509:1993 і ISO/IEC 9594-8:1995.

Основною ідеєю геш-функцій є використання ітеративної функції як циклової функції, яка використовує модулярну арифметику. Причинами стандартизації і застосування таких геш-функцій є, по-перше, можливість використання існуючих програмних і апаратних засобів модулярної арифметики, застосовуваних у несиметричних криптосистемах, і по-друге, забезпечення необхідного рівня стійкості. Основним недоліком функцій є низька швидкість формування геш-коду.

Геш-функція MASH-1 (Modular Arithmetic Secure Hash-1) використовує RSA подібні модулі N, довжина яких забезпечує необхідну стійкість.

Кількість N повинна важко розкладатись, на чому і ґрунтується стійкість алгоритму. Розмір модуля визначає довжину блоків оброблюваного повідомлення, а також розмір геш-коду (наприклад 1025-бітний модуль забезпечує формування 1024-бітного геш-коду). У якості вхідної послідовності використовується двійковий рядок *x* довжиною $0 \le b \le 2^{n/2}$, де *n* – розрядність геш-коду.

Алгоритм MASH-1 складається з таких етапів:

1. Системні установки та визначення констант. Установити RSA-подібний модуль $N = pq$ довжиною m біт, де p і q випадково обрані більші прості числа, що зберігаються в секреті. Визначити двійкову довжину n геш-коду як найбільший добуток числа 16 і довжини геш-коду, що задовольняє нерівності $16 \times n < m$.

Як вектор ініціалізації вибрати $H_0 = 0$. Визначити n-бітне ціле кількістю як константа $A = 0xf00...00$.

2. Предобробка. Доповнити, якщо необхідно, рядок x нульовими бітами, для того, щоб одержати двійковий рядок довжиною $t \times n/2$ для найменшого $t \ge 1$. Розділити доповнений текст на $n/2$-розрядні блоки $x_1,...,x_t$ і додати останній блок x_{t+1}, що містить $n/2$-розрядне подання числа b. Цей етап алгоритму наведено на рис. 6.4.

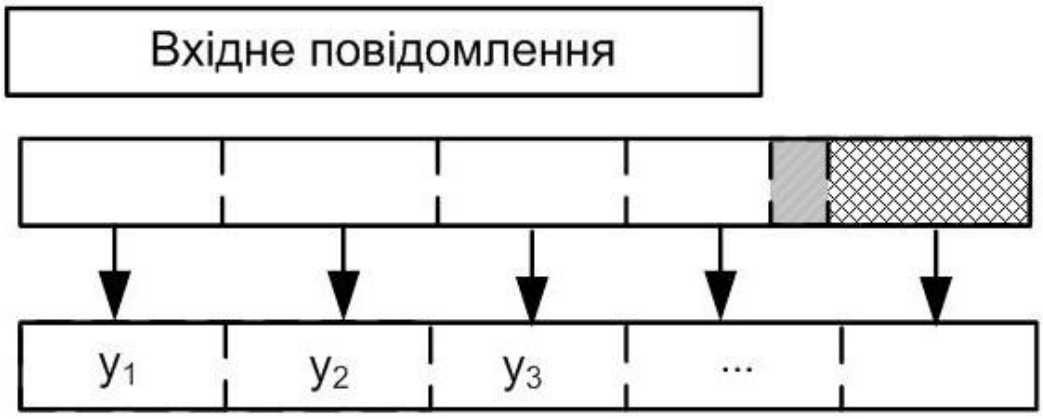


Рис. 6.4. **Етап алгоритму-предобробки повідомлення**

3. Розширення. Розширити кожний x_i блок в n-розрядний блок y_i шляхом вставки між 4-розрядними напівбайтами блоку x_i комбінації із чотирьох одиниць (1111). Останній блок y_{t+1} формується аналогічним образом за винятком того, що вставляється комбінація 1010.

4. Циклова функція. Для всіх $1 < i \le t+1$ відобразити два n-розрядних вхідних блоки (H_{i-1}, y_i) в один n-розрядний блок відповідно до виразу:

$$H_i = \left(\left(\left(\left(H_{i-1} \oplus y_i\right) \vee A\right)^2 \bmod N\right) \perp n\right) \oplus H_{i-1},$$

де $\vee$ – операція побітового логічного АБО;

$\oplus$ – додавання по модулі два (XOR);

$\perp$ – збереження молодших n -розрядів m -розрядного результату.

Загальний вид циклової функції наведено на рис. 6.5.

5. Закінчення. У якості геш-коду приймається n -розрядний блок H_{t+1}..

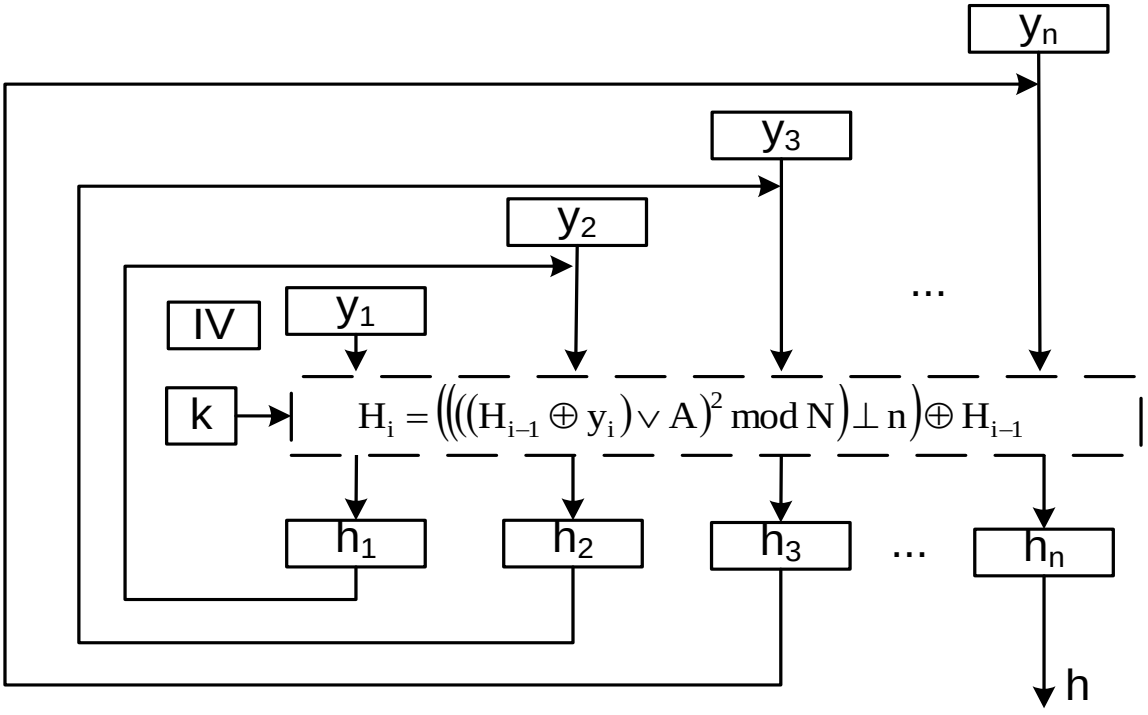


Рис. 6.5. **Загальний вид циклової функції MASH-1**

Алгоритм MASH-2 відрізняється від алгоритму MASH-1 тільки показником ступеня в цикловій функції, що має такий вигляд:

$$H_i = \left(\left(\left((H_{i-1} \oplus y_i) \vee A\right)^{2^8+1} \bmod N\right) \perp n\right) \oplus H_{i-1}.$$

Загальний вигляд циклової функції наведено на рис. 6.6.

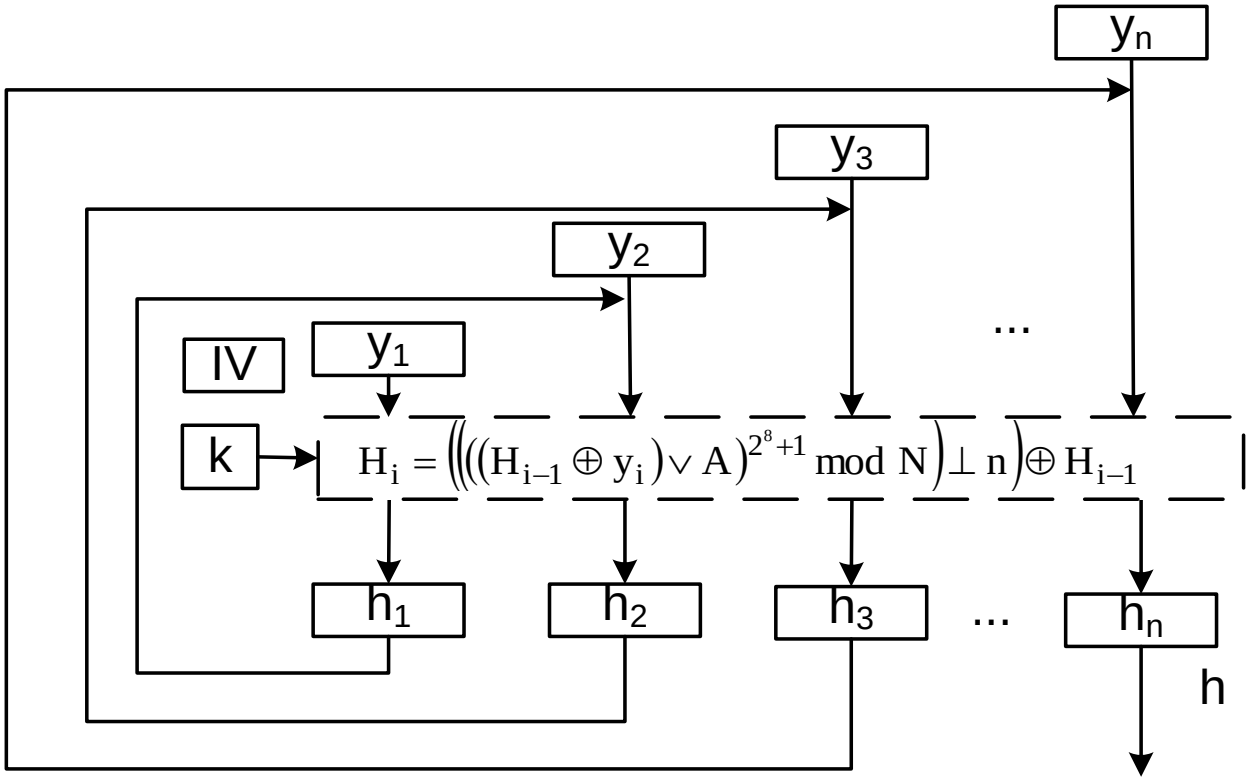


Рис. 6.6. **Загальний вид циклової функції MASH-2**

Пізніша версія стандарту MASH-2 відрізняється від MASH-1 показником ступеня, що забезпечує велику нелінійність циклової функції, і зниження вірогідності колізій з одного боку, а з іншою викликає зниження швидкості гешування MASH-2. Таким чином, використання модулярної арифметики в алгоритмах ключового гешування дозволяє отримати стійкі схеми автентичності, та це призводить до підвищення розрахункової складності криптоперетворень і часу формування геш-коду.

Порівняно з 128-розрядними геш-функціями 160-розрядний геш-код, який виробляється SHA-1, забезпечує більшу стійкість до силових атак. Геш-функції SHA-1 і RIPEMD-160 за стійкістю приблизно рівні та обидва перевершують MD5. Не менш важливим параметром є швидкість роботи алгоритму. SHA-1, RIPEMD-160 містить більшу кількість кроків і виконується на 160-бітовому буфері порівняно зі 128-бітовим буфером MD5. Таким чином, SHA-1 виконується на 25 % повільніше, ніж MD5 на тих же апаратурах. RIPEMD-160 має 160 кроків, і виконується приблизно на 15 % повільніше, ніж SHA-1. Порівняльний аналіз геш-функцій наведено у табл. 6.4.

Таблиця 6.4

Порівняльний аналіз геш-функцій

Геш-функція	Клас функції	Базові перетворення	Довжина гешу, біти
Whirlpool	однонаправлена	в кінцевих полях Галуа	512
SHA-2	однонаправлена	логічні та арифметичні	256, 384, 512
ГОСТ 34311-95	однонаправлена	БСШ	256
HAVAL	однонаправлена	логічні та арифметичні	128, 160, 192, 256
SHA-1	однонаправлена	логічні та арифметичні	160
RIPEMD-160	однонаправлена	логічні та арифметичні	160
MD5	однонаправлена	логічні та арифметичні	128
MD4	однонаправлена	логічні та арифметичні	128
UMAC	однонаправлена, МАС	в кінцевих полях Галуа	128, 64
Rijndael,CBC–MAC	виробка МАС	БСШ	128
ГОСТ28147	виробка МАС	БСШ	64

Проведений аналіз табл. 6.4 показує, що використання MDC-коду за даним вхідним повідомленним геш-коду може обчислити будь-який суб'єкт, а при використанні МАС-коду обчислити геш-код за даним вхідним повідомленням може тільки суб'єкт, що володіє секретним ключем. Таким чином, використання МАС-кодів дозволяє інтегровано вирішувати завдання гешування і забезпечення стійкості отриманих геш-кодів [23, 35; 45].

Коди автентифікації повідомлень UMAC, TTMAC, EMAC, HMAC

Основними задачами проекту NESSIE є відбір кращих десяти криптографічних примітивів. У цей набір входять алгоритми блокового і потокового шифрування, генератори випадкових чисел, схеми швидкої автентифікації даних (МАС), геш-функції і алгоритми цифрового підпису. В якості основних критеріїв відбору претендентів обрані реальна безпека, продуктивність, гнучкість і вимоги ринку.

Перевагою МАС є те, що він дозволяє одночасно отримати і перевірити інформацію за допомогою того ж секретного ключа. Це означає, що відправник і одержувач повідомлення повинні домовитися про ключ до початку повідомлення, як це має місце у випадку з симетричним шифруванням.

Код автентифікації повідомлення (MAC) є коротким фрагментом інформації, який використовується для перевірки достовірності повідомлення. MAC – код перевірки повідомлення, який використовує функцію відображення і надає дані у вигляді значень фіксованого розміру, а потім – гешує саме повідомлення [23; 45].

Алгоритм MAC приймає як введення секретний ключ і повідомлення достовірності довільної довжини, і видає MAC. MAC-код захищає цілісність повідомлення, а також його автентичність, дозволяючи контролерам (які також володіють секретним ключем) виявляти які-небудь зміни в первинному змісті повідомлення, яке передається.

Алгоритм MAC-коду працює таким чином, що перші блоки відкритих даних, які беруть участь у виробленні MAC-коду, можуть містити службову інформацію (наприклад, адресну частину, час, синхропосилку) і не зашифровуватися.

Головною перевагою цього механізму при порівнянні з електронно-цифровим підписом є більш простий алгоритм генерації та верифікації, який дозволяє забезпечити високу швидкодію алгоритмів автентифікації повідомлень у локальних мережах та комп'ютерних системах [5; 8 – 10; 13 – 15; 17].

Загальний принцип роботи MAC-кодів приведено на рис. 6.7.

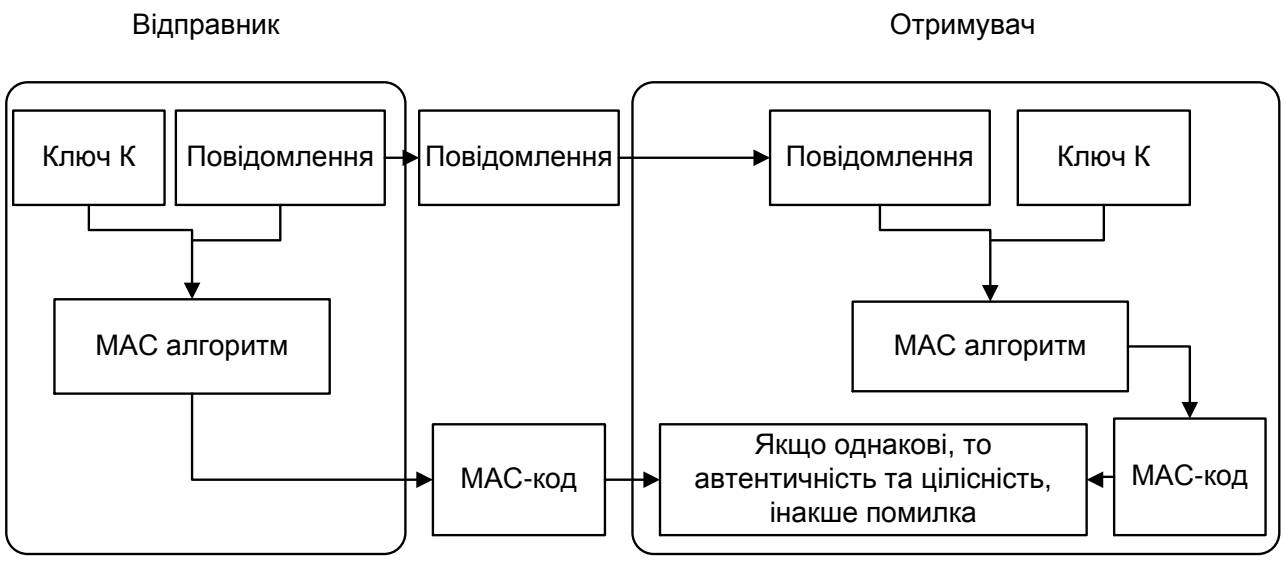


Рис. 6.7. **Принцип роботи MAC алгоритмів**

MAC алгоритми можуть бути виготовлені з інших криптографічних примітивів, таких, як криптографічні геш-функції (як у випадку з UMAC), або для блокових алгоритмів шифрування (OMAC, CBC-MAC і PMAC).

Для вивчення основних характеристик MAC кодів і їх порівняльної оцінки в науково-дослідному проекті NESSIE (IST-1999-12324) були розглянуті основні практичні алгоритми MAC кодів і відібрані чотири кращі.

Two-Track-MAC

TTMAC (також відомий як Two-Track-MAC) має найвищий рівень безпеки MAC примітивів, розглянутих у NESSIE. Дизайн TTMAC заснований на геш-функції RIPEMD-160 (з невеликими змінами). Безпека може бути доведена в припущенні, що основною функцією стиснення є псевдовипадковою. TTMAC має конкретне виконання переваги: вона особливо ефективна в разі коротких повідомлень, а також має оптимальну спритність ключів.

Особливість Two-Track-MAC полягає в шифруванні повідомлення по двох незалежних шляхах (на рис. 6.8 позначені як L і R). Такий підхід дозволяє збільшити розмір внутрішнього стану. У результаті чого отримуємо більше можливих значень внутрішньої функції шифрування. Це дозволяє ускладнити атаки, засновані на переборі всіляких значень.

У порівнянні з MDx-MAC, який так само заснований на RIPEMD-160 Two-Track-MAC набагато ефективніше для коротких повідомлень (512 або 1024 біт), і також ефективніше на довгих повідомленнях.

Інша важлива перевага TTMAC є можливість швидкої зміни ключа шифрування. Це дозволяє збільшити стійкість системи, без зниження швидкості. При досить частій зміні ключа зловмиснику не вдасться зібрати великої кількості пар повідомлення – MAC-код, що сильно знижує ймовірність підбору ключа або MAC-коду.

Проект двухтрекового MAC засновано на геш-функції RIPEMD-160. Спочатку повідомлення, що автентифікується, заповнюється 1-бітами, і потім 0-бітами, поки довжина не досягнеться 448 mod 512. Потім двійкове представлення довжини вихідного повідомлення (mod 264) додається так, що довжина повідомлення стає кратною 512. Кожний 512-бітовий блок розділено на набори шістнадцяти 32-бітових слів, W_0, …, W_{15}.

Секретний ключ є набором з 5-ти 32-бітових слів, K_0, …, K_4. Алгоритм працює на основі повторення функції стиску таким способом.

Два набори з п’яти 32-бітових слів X_0, …, X_4 і Y_0, …, Y_4 і набір 16 слів повідомлення є входом для двох різних функцій f_L і f_R, що генерують на виході п’ять 32-бітових слова кожна:

$$A_0, \ldots, A_4 = f_L(X_0, \ldots, X_4, W_0, \ldots, W_{15});$$
$$B_0, \ldots, B_4 = f_R(Y_0, \ldots, Y_4, W_0, \ldots, W_{15}).$$

Ці дві функції ідентичні використаним в RIPEMD-160 (деталі наведені далі). Потім необхідно обчислити два нові набори з п’яти слів кожний, віднімаючи вхідні повідомлення з результатів попереднього кроку:

$$C_i = A_i - X_i \bmod 2^{32},\ 0 \le i \le 4;$$
$$D_i = B_i - Y_i \bmod 2^{32},\ 0 \le i \le 4.$$

Для завершення функції стиску, десять слів C_i і D_i змішані у двох лінійних перетвореннях g_L і g_R:

$$E_0, \ldots, E_4 = g_L(C_0, \ldots, C_4, D_0, \ldots, D_4);$$
$$F_0, \ldots, F_4 = g_R(C_0, \ldots, C_4, D_0, \ldots, D_4).$$

E_i і F_i разом з наступним набором слів повідомлення, формують вхідні значення для наступного проходу функції стиску. У першій ітерації п’ять секретних ключових слова K_i є входом як X_i так і Y_i, $0 \le i \le 4$

В останній ітерації, де останні слово повідомлення є входом, f_L і f_R здійснюють свопінг простору. Після віднімання вхідних слів, замість виконання g_L і g_R наприкінці цієї ітерації обчислюємо:

$$E_i = C_i - D_i \bmod 2^{32},\ 0 \le i \le 4.$$

Результати цього вирахування формують вихід двотрекового MAC. Додаткове вихідне перетворення визначене для обчислення більш коротких значень MAC.

Функції f_L і f_R.

Функції f_L і f_R, які відомі як лівий і правий слід функції стиску, ідентичні функціям, використаним у функції стиску RIPEMD-160. Вони складаються з 80 послідовних кроків. Спершу визначаються константи й функції.

Аддитивні константи:

$k_i = 00000000_x$, $k_i' = 50a28be6_x$, $0 \le i \le 15$,
$k_i = 5a827999_x$, $k_i' = 5c4dd124_x$, $16 \le i \le 31$,
$k_i = 6ed9eba1_x$, $k_i' = 6d703ef3_x$, $32 \le i \le 47$,
$k_i = 8f1bbcdc_x$, $k_i' = 7a6d76e9_x$, $48 \le i \le 63$,
$k_i = a953fd4e_x$, $k_i' = 00000000_x$, $64 \le i \le 79$.

Нелінійні функції на бітовому рівні:

$$f_i(x,y,z) = x \oplus y \oplus z, \quad 0 \le i \le 15,$$
$$f_i(x,y,z) = (x\&y) \mid (\bar{x}\&z), \quad 16 \le i \le 31,$$
$$f_i(x,y,z) = (x\&\bar{y}) \oplus z, \quad 32 \le i \le 47,$$
$$f_i(x,y,z) = (x\&z) \mid (y\&\bar{z}), \quad 48 \le i \le 63,$$
$$f_i(x,y,z) = x \oplus (y|\bar{z})\ z, \quad 64 \le i \le 79.$$

Вибір слова повідомлення:

r[i] = i 0 ≤ i ≤ 15
r[i] = 7,4,13,1,10,6,15,3,12,0,9,5,2,14,11,8 16 ≤ i ≤ 31
r[i] = 3,10,14,4,9,15,8,1,2,7,0,6,13,11,5,12 32 ≤ i ≤ 47
r[i] = 1,9,11,10,0,8,12,4,13,3,7,15,14,5,6,2 48 ≤ i ≤ 63
r[i] = 4,0,5,9,7,12,2,10,14,1,3,8,11,6,15,13 64 ≤ i ≤ 79
r'[i] = 5,14,7,0,9,2,11,4,13,6,15,8,1,10,3,12 0 ≤ i ≤ 15
r'[i] = 6,11,3,7,0,13,5,10,14,15,8,12,4,9,1,2 16 ≤ i ≤ 31
r'[i] = 15,5,1,3,7,14,6,9,11,8,12,2,10,0,4,13 32 ≤ i ≤ 47
r'[i] = 8,6,4,1,3,11,15,0,5,12,2,13,9,7,10,14 48 ≤ i ≤ 63
r'[i] = 12,15,10,4,1,5,8,7,6,2,13,14,0,3,9,11 64 ≤ i ≤ 79

Константи обертання:

s[i] = 11,14,15,12,5,8,7,9,11,13,14,15,6,7,9,8, 0 ≤ i ≤ 15
s[i] = 7,6,8,13,11,9,7,15,7,12,15,9,11,7,13,12, 16 ≤ i ≤ 31
s[i] = 11,13,6,7,14,9,13,15,14,8,13,6,5,12,7,5, 32 ≤ i ≤ 47
s[i] = 11,12,14,15,14,15,9,8,9,14,5,6,8,6,5,12, 48 ≤ i ≤ 63
s[i] = 9,15,5,11,6,8,13,12,5,12,13,14,11,8,5,6, 64 ≤ i ≤ 79
s'[i] = 8,9,9,11,13,15,15,5,7,7,8,11,14,14,12,6, 0 ≤ i ≤ 15
s'[i] = 9,13,15,7,12,8,9,11,7,7,12,7,6,15,13,11, 16 ≤ i ≤ 31
s'[i] = 9,7,15,11,8,6,6,14,12,13,5,14,13,13,7,5, 32 ≤ i ≤ 47
s'[i] = 15,5,8,11,14,14,6,14,6,9,12,9,12,5,15,8, 48 ≤ i ≤ 63
s'[i] = 8,5,12,9,12,5,14,6,8,13,6,5,15,13,11,11, 64 ≤ i ≤ 79

Нехай дані вихідні значення a_0, b_0, c_0, d_0, e_0. Функція f_L (лівий слід функції стиску), складається з таких кроків для $0 \le l \le 79$ (додавання по mod 2^{32}):

$$a_{i+1} = e_i$$
$$b_{i+1} = (a_i + f_i(b_i,c_i,d_i) + W_{r[i]} + k_i)^{<<s[i]} + e_i\ ,$$
$$c_{i+1} = b_i$$
$$d_{i+1} = c_i^{<<10}$$
$$e_{i+1} = d_i$$

Аналогічно, коли дані вихідні значення a_0, b_0, c_0, d_0, e_0, функція f_R (правий слід функції стиску), складається з таких кроків для для $0 \le l \le 79$ (операції є mod 2^{32}):

$$a_{i+1} = e_i$$
$$b_{i+1} = (a_i + f_{79-i}(b_i,c_i,d_i) + W_{r'[i]} + k_i')^{<<s'[i]} + e_i ,$$
$$c_{i+1} = b_i$$
$$d_{i+1} = c_i^{<<10}$$
$$e_{i+1} = d_i$$

Функції g_L і g_R.

Лінійні перетворення g_L і g_R використовуються для змішування вихідних значень двох слідів функції стиску. Для входів C_0, …, C_4 і D_0, …, D_4, функція g_L обчислює п'ять слів E_0, …, E_4 таким способом (операції є mod 2^{32}):

$$E_0 = (C_1 + C_4) – D_3,$$
$$E_1 = C_2 - D_4,$$
$$E_2 = C_3 – D_0,$$
$$E_3 = C_4 – D_1,$$
$$E_4 = C_0 – D_2$$

Для входів C_0, …, C_4 і D_0, …, D_4, функція gr обчислює слово F_0, …, F_4 у таким способом (операції є модулем 2^{32}):

$$F_0 = C_3 – D_4,$$
$$F_1 = (C_4 + C_2) – D_0,$$
$$F_2 = C_0 – D_1,$$
$$F_3 = C_1 – D_2,$$
$$F_4 = C_2 – D_3,$$

Аналіз безпеки.

Безпека двухтрекового MAC може бути доведена на основі припущення, про те, що основна функція стиску псевдо-довільна. Ця функція дуже схожа на функцію стиску, використану RIPEMD-160 [16], що і є добре вивченим примітивом, у якому не виявлені ніякі вразливості.

Функції f_L і f_R складаються з 80 ітерацій кроків і використовують різні бітові операції И, АБО, XOR і побітове додавання. Кроки функції також включають бітові обертання й додавання mod 2^{32}, при 80-кратному повторенні цього, здається досить складним простежити невідомі ключові біти. Ніякі вразливості у двотрековому MAC не виявлені.

Необхідно зазначити, що єдиною ділянкою, де використовується ключовий матеріал, є початкове значення. 160-бітовий вихід алгоритму – різниця між двома 160-бітовими значеннями, і знання цієї різниці усе ще

дає 160 бітів невизначеності щодо двох оброблених величин. Інакше кажучи, міркування на базі здогадів про значення цих величин і зворотні розрахунки в пошуках вхідного значення, тобто ключа, ненабагато ефективніше, ніж спроба безпосереднього вгадування ключа.

Великий розмір внутрішнього стану (320 біт) у двотрековому MAC надає алгоритму високий рівень безпеки проти атаки, заснованої на внутрішніх помилках. Якщо позначити вихідну довжину m (значення m перебувають між 32 і 160 бітами із кроком 32), складність загальної атаки в цьому примітиві визначається таким способом:

близько 2^{159} обчислень MAC і 160/m відомих пар текст-MAC необхідні для повного ключового пошуку.

Імовірність вгадування величини MAC становить 2^{-m}.

Атака, заснована на внутрішній помилці вимагає близько 2^{160} відомих пар текст-MAC і близько 2^{320-m} обраних текстів.

Схема роботи алгоритму Two-Track-MAC наведена на рис. 6.8.

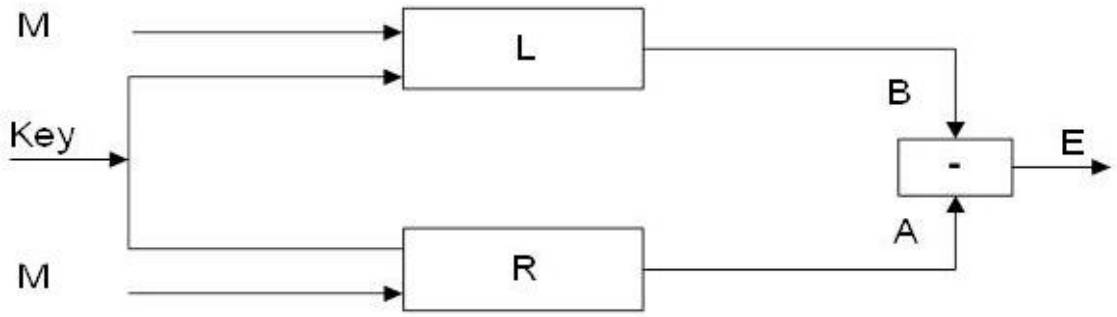


Рис. 6.8. **Схема роботи алгоритму Two-Track-MAC**

CBC-MAC (ISO/IEC 9797-1), до яких відносяться EMAC та RMAC;

Блоковий шифр E є функцією $E : K_E \times \{0,1\}^n \to \{0,1\}^n$, де кожна E (K, •) = = Ек (•) є перестановкою на $\{0,1\}^n$, K_E – це безліч можливих ключів, а n довжина блоку.

CBC MAC найбільш простий і добре відомий алгоритм, щоб зробити MAC з блочного шифру E. Нехай M = M [1] ◦ M [2] ◦ … ◦ M [m] буде рядком повідомлення, де | M [1] | = | M [2] | = … = | M [m] | = n. Тоді CBC_K (M), CBC MAC з M з ключами K, визначається як Y [m], де Y [i] = E_K (M [i] $\oplus$ Y[i -1]).

для i = 1,..., m і Y [0] = 0^n.

В існуючих реалізаціях MAC-алгоритмів CBC MAC Bellare, Kilian, і Rogaway запропонували EMAC.

Алгоритм EMAC

EMAC (також відомий як DMAC) має таку перевагу, що дозволяє повторне використання існуючих блокових шифрів реалізації (у CBC-режимі з додатковим шифруванням у вигляді перетворення на виході). Безпека може бути доведена в припущенні, що основний блоковий шифр є псевдовипадковим. Продуктивність і спритність ключів є розумними (EMAC кращий для коротких повідомлень, оскільки довжина блоку менше порівняно зі схемами, заснованими на геш-функції). NESSIE рекомендує використовувати цю конструкцію з 128-бітним блоковим шифром, включеним в NESSIE. На рис. 6.9 наведена схема роботи алгоритму CBC-MAC.

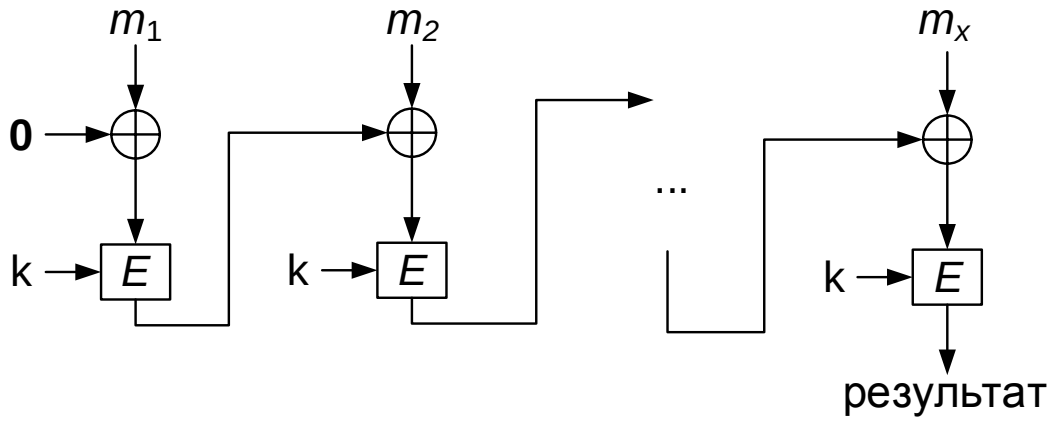


Рис. 6.9. **Схема роботи алгоритму CBC-MAC**

Обчислення EMAC для секретного ключа K і повідомлення X – поділеного (після заповнення) на 128-бітові блоки X_1, …, X_t – оброблених таким способом (тут EK позначає шифрування з використанням 128-бітового блокового шифру AES на ключі K):

1. Обчисліть $H_1 = EK_1(X_1)$.

2. Для i = 2, …, t: обчисліть $H_1 = E_{K1}(X_i \oplus H_{i-1})$.

3. Обчисліть вихідне перетворення: $H_{out} = EK_2(H_t)$. Ключ K_2 може бути похідним від K_1, отриманим в ході такої процедури:

$$K_1 = K_1 \oplus (f_0 f_0 \ldots f_{0r}).$$

4. Для того, щоб одержати величину m-біт MAC, необхідно вибрати тільки ліві m біти H_{out}.

RMAC є випадковим варіантом цієї схеми, що забезпечує кращу протидію атаці на основі внутрішніх помилок. Єдина відмінність –

у вихідному перетворенні, де можна кодувати на ключі, отриманому порозрядовим доповненням K_2 і R:

$$H_{out} = E_{K2} \oplus_R (H_t).$$

Для RMAC, ключі K_1 і K_2 можуть бути незалежними або похідними від одного головного ключа стандартним чином. R – значення, довжиною r біт, повинне бути заповнене 0-бітами, якщо воно менше, ніж K_2. П'ять різних наборів параметрів були визначені для розмірів m і r.

Аналіз безпеки. Безпека EMAC може бути доведена на базі припущення, що основний блоковий шифр – псевдовипадковий [25; 27; 50; 61], у такому випадку Rijndael, який було проаналізовано, протягом тривалого часу протягом і після процесу AES реалізує вимоги безпеки.

Необхідно також зазначити, що без додаткового шифрування в остаточному підсумку (або у випадку, якщо K_2 повинен бути обраний рівним K_1), проста (адаптивно обраний текст) підробка буде можлива для схеми CBC-MAC (через атаку підробки exor). Якщо K_1 і K_2 повинні бути обрані незалежно, рівень безпеки проти атаки відновлення ключа буде менш імовірним, ніж передбачено за розміром алгоритмів формування ключа (через атаку "розділяй і пануй"). Внутрішня помилка утворює підроблене значення, що вимагає близько 2^{64} відомих пар текст-MAC і 1 обраний текст при вихідній довжині 128 біт. Більша кількість обраних текстів була б потрібна при виключенні виходу MAC, наприклад, при виборі тільки лівих 64 біт, атаці потрібно близько 2^{64} відомих пар і 2^{64} обраних текстів. З іншого боку, це збільшує ймовірність успішної атаки вгадування величини MAC.

Основна перевага випадкового варіанта RMAC – те, що надається краща протидія атаці, заснованій на внутрішніх помилках. Наприклад, якщо обрані параметри m = 128 і r = 128, то така атака вимагає близько 2^{128} відомих пар і 1 обраний текст. З іншого боку, RMAC необхідні міцні основи для доказу безпеки шифру; наприклад, основний блоковий шифр повинен бути безпечним проти атаки зчепленого ключа. Відомо, що для RMAC із двома незалежними ключами K_1 і K_2 очікується, що повний пошук ключів зажадає генерації 22^{k-1} MAC, де k – розмір одного ключа. Проте, це можна зробити значно швидше для параметрів m = 128 і r = 128: при атаці за обраним повідомленням з одним відомим повідомленням і одним обраним повідомленням K_2 може бути виявлено в 2^k операціях розшифрування, згодом K_1 може бути виявлено протягом

приблизно того ж часу. Альтернативна атака на RMAC (m = 128, r = 128) вимагає 2^{123} обраних текстів (при часі виконання 2^{124}) є серйозною атакою на RMAC, що використовує трехключовий Triple-des в основі блокового шифру замість AES (у певних випадках проведення цих атак має складність близько 2^{56} операцій і ймовірність успіху 2^{-16}).

Алгоритм HMAC (ISO/IEC 9797-1)

HMAC має таку перевагу, що дозволяє повторне використання існуючих реалізацій геш-функції. Безпека може бути доведена на таких припущеннях: основна геш-функція стійка до колізій з невідомим початковим значенням; функція стиснення забезпечена ключами на початкове значення безпечного MAC примітиву (для повідомлень з одного блоку); функція стиснення є слабкою псевдовипадковою функцією. Ці припущення слабкіші, ніж припущення, необхідні для TTMAC і EMAC. Продуктивність і спритність ключів є розумними. NESSIE рекомендує використовувати цю конструкцію з колізіонно-стійкою геш-функцією, включеною в NESSIE.

Формування MAC-кодів за допомогою HMAC:

$$HMAC(K,M) = h((K \oplus opad) \,||\, h(K \oplus ipad) \,||\, M),$$

де h – геш-функція;

K – секретний ключ, доповнений нулями до розміру блоку;

M – повідомлення для ідентифікації;

II – конкатенація;

opad – 0x5c5c..5c (довжина дорівнює розміру блока);

ipad – 0x3636..36 (довжина дорівнює розміру блока).

Схему роботи алгоритму HMAC наведено на рис. 6.10.

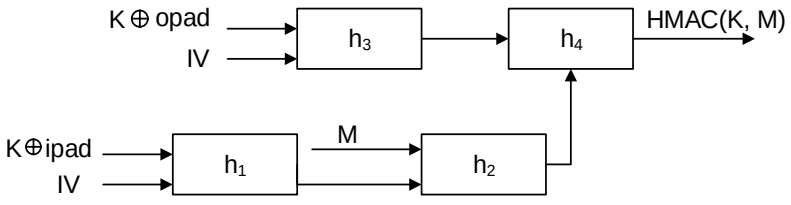


Рис. 6.10. **Схема роботи алгоритму HMAC**

Обчислення MAC для секретного ключа K і повідомлення X відбувається таким чином (тут h позначає геш із геш-функцією SHA-1):

1. Обчислити ключову величину K’ довжиною 512 біт. Припустимо, що K має довжину l біт. Якщо l = 512, то установимо K’ = K; якщо l < 512,

то одержимо K', додаючи 512 – l нульових біт до K; якщо l > 512, то одержимо K', обчислюючи геш h(K) (160 біт довжиною) і додаючи 352 нульових біт до цієї величини геша.

2. Exor K' З 512-бітовою константою ipad і додаємо до повідомлення X: (K'⊕ipad) || X.

3. Обчислити геш рядок із кроку 2: h((K'⊕ipad) || X).

4. Exor K' з 512-бітовою константою opad і додайте 160-бітовий результат із кроку 3: (K'⊕opad) || h((K'⊕ipad) || X).

5. Обчислити геш рядка із кроку 4: h((K'⊕opad) || h((K'⊕ipad) || X)).

6. Щоб одержати величину m-біт MAC, необхідно виберати тільки ліві m біт результату кроку 5.

Рядок ipad визначений як 64-кратна конкатенація шістнадцяткової величини "36", і рядок opad – як 64-кратна конкатенація шістнадцяткової величини "5c".

Аналіз безпеки.

Беллейр (Bellare) [36] дає теоретичну основу HMAC, що зв'язує безпеку схеми MAC з безпекою основної геш-функції, і в цьому випадку SHA-1 виступає як досконально вивчений примітив, у якому не знайдені вразливості.

А саме, доведено, що HMAC безпечний, якщо враховувати такі припущення (тут f – функція стиску, яка повторюється геш-функцією для кожного 512-бітового блоку):

геш-функція є постійною помилкою при секретній початковій величині;

функція стиску f за ключем з початковою величиною є досить міцним алгоритмом MAC (означає, що вихід важко передбачити);

величини f(K'⊕ipad) і f(K'⊕opad) не відрізняються від дійсно довільних величин. Це означає, що функція стиску f є "слабкою" псевдовипадковою функцією ("слабкою" оскільки криптоаналітик не має прямого доступу до K').

Зазначимо також, що якщо конструкція HMAC використовується двома незалежними ключами (замість використання K_1 = K'⊕ipad і K_2 = K'⊕opad), рівень безпеки проти атаки відновлення ключа буде менше, ніж дає розмір ключа алгоритму (через атаку "розділяй і пануй"). Внутрішня помилка утворює підроблене значення для HMAC, засноване на SHA-1, якому необхідно близько 2^{80} відомих пар текст-MAC і 1

обраний текст при вихідній довжині 160 біт. Більша кількість обраних текстів була б потрібна при урізаному виході MAC, наприклад, при виборі самих лівих 80 біт, атака потрібує близько 2^{80} відомих пар і 2^{80} обраних текстів. З іншого боку, це збільшує ймовірність успіху для атаки вгадування величини MAC.

UMAC: розробка корпорації Intel (США), Університету з штату Невада в Рено (США), Науково-Дослідної лабораторії IBM (США), Technion (Ізраїль) і Університету з Каліфорнії в Девісі (США);

Код автентифікації повідомлення UMAC заснований на сімействах універсальних геш-функцій і забезпечує те, що перевіряється безпека в тому розумінні, що є доказові межі помилки геш частини обчислення MAC, так що безпека в підсумку залежить від шифрувального примітива, використаного для кодування залишку. Примітив, установлений у специфікаціях AES (Rijndael) блокового шифру, який аналізувався довгий час протязгом і після процесу AES, реалізує вимоги до безпеки. Існує також додаткова перевага, яка полягає в тому, що шифрування виконується на невеликому залишку.

Ніяких вразливостей не було виявлено в ході доведення безпеки UMAC. Для прошарку, що використовує універсальне сімейство геш-функцій NH, перший доказ безпеки говорить про те, що NH є майже 2^{-w} – універсальним (це означає, що ймовірність зіткнення – не більш 2^{-w}) для рядків рівної довжини, для слів довжиною w біт. Це відповідає використанню NH на одному блоці повідомлення встановленої довжини. Другий доказ безпеки дозволяє поширювати цей результат на NH, що працює з будь-якою парою рядків подібно UMAC. Причина того, що ймовірність помилки після прошарку геша NH у схемі Uhash16 становить 2^{-15}, а не 2^{-16}, у тому, що використовується знакова версія NH. Перший варіант доведення безпеки показує, що знакова версія NH 2^{-w+1} – майже універсальна.

Uhash16 і Uhash32 мають два додаткових прошарки, що використовують RP і універсальні сімейства геш-функцій IP; вони повторюють тришарову схему, відповідно, два й чотири рази. Доведено, що сімейство геша Uhash16 4 має 4 рівні ($2^{-15} + 2^{-18} + 2^{-28}$) і майже універсально, і, що сімейство геша Uhash32 має 2 рівні ($2^{-31} + 2^{-33}$) і також майже універсально.

На рис. 6.11 наведено схему роботи алгоритмів UMAC16/32.

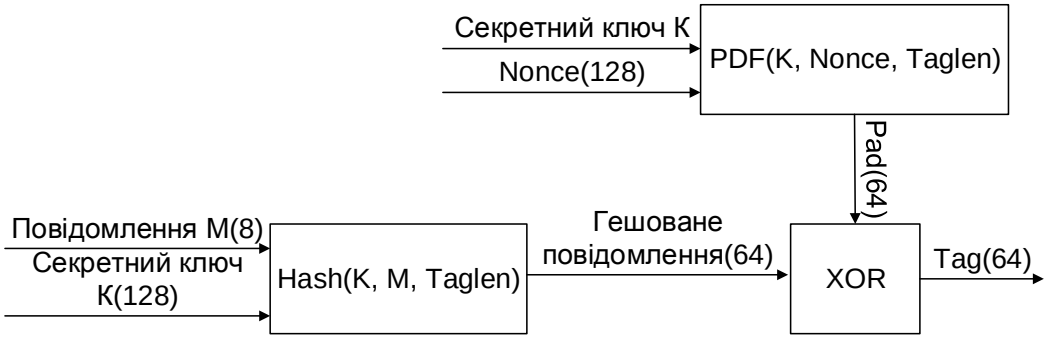


Рис. 6.11. **Схема роботи алгоритму UMAC16/32**

Алгоритм дозволяє забезпечити більш високу швидкість (особливо для довгих повідомлень) і підтверджувану безпеку, ціною більшої складності.

Для перевірки автентичності довгого потоку повідомлень UMAC на сьогоднішній день є найшвидшим з MAC примітивів розглянутими NESSIE (за рахунок більшої складності і гіршої спритності ключів порівняно з іншими примітивами). UMAC заснований на універсальному сімействі геш-функцій й доказовою безпекою: розрив примітиву означатиме і розрив блокового шифру, який використовується в схемі як псевдовипадкова функція (поточна специфікація в якості блокового шифру вибирає AES).

Основними характеристиками алгоритмів формування MAC кодів, за якими виконується їх порівняльна оцінка відповідно до рекомендацій проекту NESSIE є:

рівень захищеності MAC кодів від загальних атак;

швидкодія алгоритмів формування MAC кодів;

статистичні властивості розподілів MAC кодів.

У табл. 6.5 наведено аналіз тестування швидкості роботи алгоритмів гешування.

Таблиця 6.5

Аналіз тестування швидкості роботи алгоритмів гешування

Функція гешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
ГОСТ 34311-95	-	C + ASM	49,408 Мбіт/с	83,056 Мбіт/с
UMAC	-	C C + ASM	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с
ГОСТ 28147-89 (OFB)	16	C+ ASM	189,559 Мбіт/с	315,270 Мбіт/с

Проведений аналіз показав, що найбільш перспективним алгоритмом обробки інформації є UMAC, який забезпечує обробку величезних розмірів даних за секунди (10^9 біт/с). При додатковому використанні алгоритму AES вірогідність зламу значно зменшується.

Для вирішення завдань забезпечення цілісності спостереження і достовірності інформації за допомогою протоколів автентифікації застосовуються криптографічні контрольні суми. Методи формування криптографічних контрольних сум можна розділити на два класи: на базі симетричних криптографічних перетворень (коди автентифікації повідом-лень (MAC-коди)) і несиметричні перетворення (ЕЦП або електронні цифрові підписи) із застосуванням секретних ключів. Такі функції можуть застосовуватися безпосередньо як криптографічна контрольна сума, так і в інших перетвореннях.

Відомі методи забезпечення автентичності і цілісності даних засновані на внесенні надмірності до оброблюваної послідовності.

При розгляді стандартів ISO 7498-2 і ISO/IEC 10181 визначено, що одним з найнадійніших способів вирішення завдань, пов’язаних з автентифікацією (процедура встановлення достовірності твердження про те, що об’єкт (або суб’єкт) володіють заявленими властивостями) даних і джерел повідомлень, є процедури формування цифрового підпису, побудовані на основі асиметричних криптографічних алгоритмів [50; 52; 60].

Забезпечення цілісності і автентичності інформації полягає у налагодженні процесу автентифікації відправника і контролю цілісності повідомлень за допомогою цифрового підпису.

Зазначимо, що обчислювальна стійкість схем ключового гешування і вірогідність колізії впливає на стійкість цифрового підпису, MAC-кодів і інших механізмів забезпечення безпеки інформації.

За використовуваними внутрішніми перетвореннями функції гешування можна поділити на: функції, що використовують бітові логічні перетворення. Ці функції застосовують до вхідного повідомленням побітові нелінійні операції "І", "АБО", "НЕ", "ВИКЛЮЧАЮЧЕ АБО", різні зрушення і, як правило, є багатоцикловими; функції, що використовують блокові симетричні шифри. Використовуються в основному для реалізації функцій вироблення MAC; функції, що використовують перетворення в групах, полях та кільцях з цілочисленним або поліноміальним базисом; функції, що використовують матричні перетворення.

Аналіз умов застосування функцій гешування та їх практичного використання дозволив сформулювати вимоги, які пред'являються до застосовуваних у криптографії безключовим геш-функціям. Вони полягають у такому: стійкість до обчислення; стійкість до обчислення другого; стійкість до колізій.

Вимога стійкості до колізій є більш жорсткою, ніж вимога стійкості до обчислення другого прообразу, оскільки припускає довільний вибір двох прообразів.

На практиці зазвичай використовуються геш-функції, що є одночасно безколізійними і односпрямованими.

Усі атаки на геш-функції можна розділити на дві групи: атаки, що базуються на вразливості алгоритму перетворень (аналітичні) і атаки, не залежні від алгоритму.

Атаки, не залежні від алгоритму: атака "грубою силою", атака методом "дня народження", повний перебір ключів. До таких атак уразливі всі алгоритми, єдина можливість їх уникнути – збільшити довжину геш-значення, що виробляється однонаправленою або безколізійною геш-функцією, і секретного ключа у функції вироблення MAC-коду.

Аналітичні атаки: атака "зустріч посередині", атака з корекцією блоку, атака з фіксованою точкою, атака на базовий алгоритм шифрування, диференціальний аналіз. Ці атаки ґрунтуються на недоліках внутрішньої структури геш-функцій.

Згідно з проведеними дослідженями найбільш стійкою геш-функцією до різних типів атак є функція вироблення MAC-коду, яка використовується для автентифікації повідомлення.

Слід розрізняти функції вироблення MAC-коду і однонаправлені геш-функції з секретним ключем, що є частиною повідомлення, оскільки вони володіють різними властивостями. У функціях вироблення MAC-коду секретний ключ застосовується до кожного блоку повідомлення, а в однонаправлених геш-функціях ключ використовується префіксним (на початку повідомлення), постфіксним (в кінці повідомлення) або комбінованим методом, що знижує стійкість функції. Повідомлення цілісності коду (MDC) відрізняється від MAC в тому, що секретний ключ не використовується в їх діяльності. Хоча ці терміни іноді використовуються як взаємозамінні, MDC завжди повинен бути закодований в ході передачі, якщо він використовуватиметься як надійний гарант цілісності повідомлення. З іншого боку, MAC, який використовує секретний ключ,

не обов'язково повинен бути зашифрований щоб забезпечити такий же рівень надійності.

Результати досліджень у галузі захисту інформації, а також аналіз літератури показали [18; 21; 23; 32; 33; 36; 42 – 46], що одним з найбільш перспективних методів забезпечення цілісності та автентичності інформації є введення в інформацію надмірного коду на основі використання кодів автентифікації повідомлень (MAC кодів), що дозволяє із заданою імовірністю встановлювати дійсність переданого повідомлення [23; 43].

Для проведення статистичного дослідження колізійних властивостей формованих елементів введемо методику, в основі якої лежить емпірична оцінка максимумів числа ключів (правил гешування) при яких:

1. Для довільних $x_1, x_2 \in A$, $x_1 \neq x_2$ виконується рівність:

$$h(x_1) = h(x_2); \tag{6.1}$$

2. Для довільних $x_1 \in A$ та $y_1 \in B$ виконується рівність:

$$h(x_1) = y_1; \tag{6.2}$$

3. Для довільних $x_1, x_2 \in A$, $x_1 \neq x_2$ та $y_1, y_2 \in B$ виконується рівність:

$$h(x_1) = y_1, h(x_2) = y_2. \tag{6.3}$$

Оцінка за першим критерієм відповідає перевірці здійсненності умови для універсального класу геш-функцій, оцінка за другим і третім критеріями – умов для строго універсального класу геш-функцій.

Введемо такі позначення:

$$n_1(x_1, x_2) = \left|\{h \in H : h(x_1) = h(x_2)\}\right|,\ x_1, x_2 \in A,\ x_1 \neq x_2;$$
$$n_2(x_1, y_1) = \left|\{h \in H : h(x_1) = y_1\}\right|,\ x_1 \in A,\ y_1 \in B;$$
$$n_3(x_1, x_2, y_1, y_2) = \left|\{h \in H : h(x_1) = y_1, h(x_2) = y_2\}\right|,\ x_1, x_2 \in A,\ x_1 \neq x_2,\ y_1, y_2 \in B.$$

Перший показник $n_1(x_1, x_2)$ характеризує кількість правил гешування, при яких для заданих $x_1, x_2 \in A$, $x_1 \neq x_2$, виконується рівність (6.1), тобто кількість ключів, при яких існує колізія (збіг геш-кодів) для двох вхідних послідовностей x_1 і x_2.

Другий показник $n_2(x_1, y_1)$ характеризує кількість правил гешування, при яких для заданих $x_1 \in A$, $y_1 \in B$, виконується рівність (6.2), тобто скількість ключів, при яких для вхідної послідовності x_1 значення геш-коду y_1 не змінюється.

Третій показник $n_3(x_1, x_2, y_1, y_2)$ характеризує кількість правил гешування, при яких для заданих $x_1, x_2 \in A$, $x_1 \neq x_2$, $y_1, y_2 \in B$ виконується рівність (6.3), тобто кількість ключів, при яких для двох вхідних послідовностей x_1 та x_2 відповідні їм значення геш-кодів y_1 і y_2 не змінюються.

Оскільки кількість ключів, при яких можуть виконуватися рівності (6.1 – 6.3), не повинне перевершувати відповідних їм $P_{кол} \cdot |H|$, $|H|/|B|$ та $P_{кол}|H|/|B|$, то необхідно знати максимальну кількість таких ключів для кожного з розглянутого набору елементів.

Треба обмежитися вивченням статистичних характеристик максимумів цих величин, а потім порівняти отримані результати з числом $P_{кол} \cdot H$ (для першого критерію), з числом $|H|/|B|$ (для другого критерію) і числом $P_{кол}|H|/|B|$ (для третього критерію).

Таким чином, в якості статистичних показників оцінки колізійних властивостей, за якими будемо проводити експериментальні дослідження, пропонується використовувати:

математичні очікування $m(n_1)$ $m(n_2)$ та $m(n_3)$ максимумів кількості правил гешування, при яких виконуються рівності (6.1 – 6.3), відповідно;

дисперсії $D(n_1)$, $D(n_2)$ та $D(n_3)$, що характеризують розсіювання значень кількості правил гешування, при яких виконуються рівності (6.1 – 6.3), щодо їх математичних сподівань, $m(n_1)$, $m(n_2)$ та $m(n_3)$ відповідно.

Оцінку колізійних властивостей за наведеними критеріями необхідно виробляти в середньостатистичному сенсі. Іншими словами, при постановці експерименту треба використовувати обмежений набір елементів $x_1, x_2 \in A$, $x_1 \neq x_2$, і відповідних їм геш-образів $y_1, y_2 \in B$, розглядаючи відповідні результати як вибірку з генеральної сукупності.

Природною оцінкою для математичного сподівання m випадкової величини X є середнє арифметичне її значень X_i (або статистичне середнє).

$$\tilde{m} = \frac{1}{N}\sum_{i=1}^{N} X_i,$$

де N – кількість реалізацій випадкової величини X.

Оцінка дисперсії випадкової величини *X* визначається виразом:

$$\tilde{D} = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \tilde{m})^2.$$

У силу центральної граничної теореми теорії ймовірностей при великих значеннях кількості реалізацій N середнє арифметичне буде мати розподіл, близький до нормального з математичним очікуванням:

$$m[\tilde{m}] \approx \tilde{m}$$

і середнім квадратичним відхиленням:

$$\sigma[\tilde{m}] \approx \frac{\sigma}{\sqrt{N}}, \qquad (6.4)$$

де σ – середнє квадратичне відхилення оцінюваного параметру.

При цьому вірогідність того, що оцінка відхилиться від свого математичного сподівання менше, ніж на (довірча ймовірність), дорівнює:

$$P(|\tilde{m} - m| < \varepsilon) \approx 2\Phi\left(\frac{\varepsilon}{\sigma[\tilde{m}]}\right), \qquad (6.5)$$

де $\Phi(x)$ – функція Лапласа, визначається виразом:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-\frac{t^2}{2}} dt.$$

Таким чином, при проведенні експериментальних досліджень колізійних властивостей треба використовувати методи статистичної перевірки гіпотез і математичної статистики.

1. З генеральної сукупності випадкової величини *X* сформуємо вибірку обсягу N: $X_1, X_2, ..., X_N$:

для середньостатистичної оцінки математичного сподівання $m(n_1)$ і дисперсії $D(n_1)$ в якості випадкової величини виступає максимум $n_1(x_1, x_2)$ за всіма $h(x_1) = h(x_2)$ для заданих x_1 і x_2, а отже, вибірку сформуємо відбором з *N* пар елементів, $x_1, x_2 \in A$, $x_1 \neq x_2$;

для середньостатистичної оцінки математичного сподівання $m(n_2)$ і дисперсії $D(n_2)$ в якості випадкової величини виступає максимум за всіма $y_1 = h(x_1)$, а отже, вибірку сформуємо відбором з N пар елементів $x_1 \in A$, $y_1 \in B$;

для середньостатистичної оцінки математичного сподівання $m(n_3)$ і дисперсії $D(n_3)$ в якості випадкової величини виступає максимум $n_3(x_1, x_2, y_1, y_2)$ за всіма парами $y_1 = h(x_1)$ та $y_2 = h(x_2)$, а отже, вибірку сформуємо відбором з *N* четвірок елементів $x_1, x_2 \in A$, $x_1 \neq x_2$, $y_1, y_2 \in B$.

2. При експериментальних дослідженнях колізійних властивостей гешування:

за першим критерієм будемо оцінювати середнє арифметичне $\tilde{m}(n_1)$ спостережуваних значень максимумів $n_1(x_1, x_2)$ і дисперсію $\tilde{D}(n_1)$;

за другим критерієм будемо оцінювати середнє арифметичне $\tilde{m}(n_2)$ спостережуваних значень максимумів $n_2(x_1, y_1)$ і дисперсію $\tilde{D}(n_2)$;

за третім критерієм будемо оцінювати середнє арифметичне $\tilde{m}(n_3)$ спостережуваних значень максимумів $n_3(x_1, x_2, y_1, y_2)$ і дисперсію $\tilde{D}(n_3)$.

3. Обґрунтуємо достовірність отриманих середньостатистичних оцінок. Для цього зафіксуємо точність ε і розрахуємо значення функції Лапласа, яка, відповідно до виразу (6.5), дає відповідні довірчі ймовірності:

$$P(|\tilde{m}(n_1) - m(n_1)| < \varepsilon) \approx 2\Phi\left(\frac{\varepsilon}{\sigma[\tilde{m}(n_1)]}\right),\ \sigma[\tilde{m}(n_1)] \approx \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}};$$

$$P(|\tilde{m}(n_2) - m(n_2)| < \varepsilon) \approx 2\Phi\left(\frac{\varepsilon}{\sigma[\tilde{m}(n_2)]}\right),\ \sigma[\tilde{m}(n_2)] \approx \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}};$$

$$P(|\tilde{m}(n_3) - m(n_3)| < \varepsilon) \approx 2\Phi\left(\frac{\varepsilon}{\sigma[\tilde{m}(n_3)]}\right),\ \sigma[\tilde{m}(n_3)] \approx \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}}.$$

При зворотній постановці завдання, тобто для фіксованої довірчої ймовірності $P_д$ при обсязі вибірки N довірчий інтервал визначимо таким чином:

$$\tilde{m} - t_\rho \cdot \sigma[\tilde{m}] < m < \tilde{m} + t_\rho \cdot \sigma[\tilde{m}],$$

де t_ρ – корінь рівняння $2\Phi(t_\rho) = P_д$.

Іншими словами, в цьому випадку межі довірчого інтервалу будуть відповідати заданій довірчій ймовірності $P_д$, а точність оцінок визначається як $\varepsilon = t_\rho \cdot \sigma[\tilde{m}]$.

Для випадку, який розглядається, при заданій імовірності $P_д$ маємо:

$$\tilde{m}(n_1) - t_\rho \cdot \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}} < m(n_1) < \tilde{m}(n_1) + t_\rho \cdot \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}},\ \varepsilon = t_\rho \cdot \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}};$$

$$\tilde{m}(n_2) - t_\rho \cdot \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}} < m(n_2) < \tilde{m}(n_2) + t_\rho \cdot \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}},\ \varepsilon = t_\rho \cdot \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}};$$

$$\tilde{m}(n_3) - t_\rho \cdot \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}} < m(n_3) < \tilde{m}(n_3) + t_\rho \cdot \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}},\ \varepsilon = t_\rho \cdot \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}}.$$

На основі описаної методики будуть проводитися дослідження статистичної безпеки ключових геш-функцій. Визначені рекомендації вибору параметрів геш-кодів використовуються в процедурі гешування повідомлення в ЕЦП для забезпечення необхідного рівня цілісності й автентичності даних.

Проведені дослідження дозволяють зробити висновок, що для автентичності і цілісності інформації слід використовувати MAC-коди, які не тільки формують геш-код повідомлення, але й забезпечують належний рівень криптостійкості для протистояння атакам зловмисника. Перспективним напрямом розвитку сучасних механізмів та протоколів є використання MAC-кодів, які дозволяють (при рівних умовах з MDC-кодами) формувати геш-код з достатньою швидкістю для забезпечення автентичності та цілісності інформації.

6.2. Алгоритми сімейства MD

Спеціалізовані геш-функції (Customised hash functions або dedicated hash functions) спеціально розроблені тільки для цілей гешування й оптимізовані для виконання цієї задачі. Третя частина стандарту ISO/IEC 10118-3 визначає три спеціалізовані геш-функції, а саме функції RIPEMD-128, RIPEMD-160 і SHA-1. Дані геш-функції засновані на використанні принципів побудови, закладених у геш-функції сімейства MD*x* (MD2, MD4, MD5), які спеціально розроблялися для реалізації на 32-розрядних ЕОМ. Алгоритм MD4 був запропонований Р. Райвестом у 1990 році, а в 1991 той же автор запропонував модифіковану версію алгоритму – MD5. У даний час геш-функції MD4, MD5 є найбільш розповсюдженими в практичних додатках геш-функціями, однак Р. Райвест у 2010 році визнав її небезпечною і запропонував не використовувати при розроблені програмних продуків. Крім цього деякі їх недоліки не дозволили стандартизувати їх на міжнародному рівні.

Європейський консорціум RIPE, спираючись на свої дослідження властивостей цих алгоритмів, запропонував посилену версію MD4, що одержала назву RIPEMD. Геш-функція RIPEMD по суті складається з двох паралельно працюючих і модифікованих функцій MD4, тобто функція має дві лінії. Іншою альтернативою алгоритмам MD*x* є алгоритм SHA-1, розроблений спільно Агентством національної безпеки США і NIST і прийнятий як американський національний стандарт (FIPS 180-1).

MD4

MD4(Message Digest 4) – геш-функція, розроблена професором Массачусетського університету Рональдом Рівестом в 1990 році,

а вперше описана в RFC 1186. Для довільного вхідного повідомлення функція генерує 128-розрядне геш-значення, зване дайджестом повідомлення. Цей алгоритм використовується в протоколі автентифікації MS-CHAP, розробленому корпорацією Майкрософт для виконання процедур перевірки достовірності віддалених робочих станцій Windows.

Гешування з MD4 складається з 48 операцій, згрупованих в 3 раунди по 16 операцій. F-нелінійна функція; в кожному раунді функція змінюється. Mi означає 32-бітний блок вхідного повідомлення, а K_i – 32-бітова константа, різна для кожної операції. На рис. 6.6 наведено структуру раунда MD4.

Алгоритм MD4.

Передбачається, що на вхід подано повідомлення, що складається з біт, геш якого необхідно обчислити. Тут – довільне невід'ємне ціле число, воно може бути нулем, не повинно бути кратним восьми, і може бути як завгодно великим. Запишемо повідомлення побітова, а саме у вигляді: $m_0 m_1 \ldots m_{b-1}$.

Далі наведено 5 кроків, які використовуються для обчислення гешу повідомлення.

Крок 1. Додавання відсутніх бітів.

Повідомлення розширюється так, щоб його довжина в бітах за модулем 512 дорівнювала 448. Таким чином, у результаті розширення, повідомленням бракує 64 біти до довжини, кратної 512 бітам. Розширення проводиться завжди, навіть якщо повідомлення спочатку має потрібну довжину.

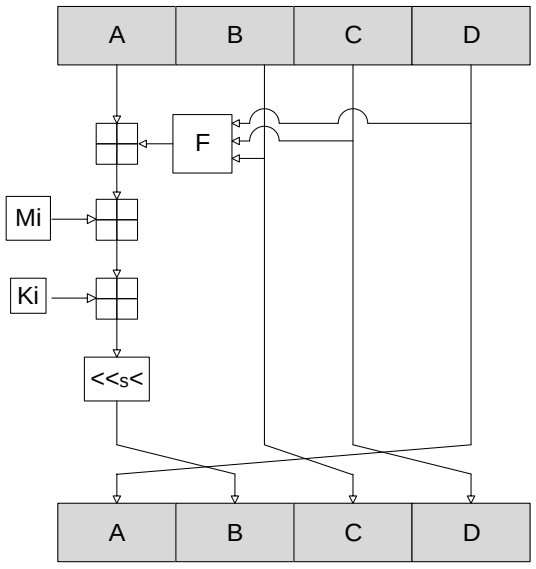


Рис. 6.6. **Структура раунда MD4**

Розширення проводиться таким чином: один біт, що дорівнює 1, додається до повідомлення, а потім додаються біти, рівні 0, до тих пір, поки довжина повідомлення не стане рівною 448 за модулем 512. У результаті до повідомлення додається як мінімум 1 біт, і як максимум 512.

Крок 2. Додавання довжини повідомлення.

64-бітове уявлення b (довжини повідомлення перед додаванням набивальних бітів) додається до результату попереднього кроку. У малоймовірному випадку, коли b більше, ніж 2^{64}, використовуються тільки 64 молодших біти. Ці біти додаються у вигляді двох 32-бітових слів, і першим додається слово, що містить молодші розряди.

На цьому етапі (після додавання бітів і довжини повідомлення) отримуємо повідомлення довжиною, кратною 512 бітам. Це еквівалентно тому, що це повідомлення має довжину, кратну 16-ти 32-бітовим словами. Кожне 32-бітове слово містить чотири 8-бітних, але йдуть вони не підряд, а навпаки (наприклад, з восьми 8-бітових слів (abcdefgh) отримуємо два 32-бітових слова (dcba hgfe)). Нехай $M[0...N-1]$ означає масив слів отриманого повідомлення (тут N кратно 16).

Крок 3. Ініціалізація MD-буфера.

Для обчислення гешу повідомлення використовується буфер, що складається з 4 слів (32-бітних регістрів): (A,B,C,D). Ці регістри ініціалізувалися такими шістнадцятковим числами (молодші байти спочатку):

word A : 67 45 23 01;

word B : ef cd ab 89;

word C: 98 ba dc fe;

word D : 10 32 54 76.

Крок 4. Обробка повідомлення блоками по 16 слів.

Для початку необхідно визначити три допоміжні функції, кожна з яких отримує на вхід три 32-бітових слова, і за ними обчислює одне 32-бітове слово.

$$F(X,Y,Z) = XY;$$
$$F(X,Y,Z) = XY \vee XZ;$$
$$G(X,Y,Z) = XY \vee XZ \vee YZ;$$
$$H(X,Y,Z) = XYZ.$$

На кожну бітову позицію F діє як умовний вираз: якщо X, то Y; інакше Z. Функція F могла б бути визначена з використанням $\oplus$ замість

V, оскільки XY і $\overline{XZ}$ не можуть рівнятися 1 одночасно. G діє на кожну бітову позицію як функція максимального значення: якщо, щонайменше, в двох словах з X, Y, Z відповідні біти рівні 1, то G видасть 1 в цьому біті, а інакше G видасть біт, що дорівнює 0. Зазначимо, що якщо біти X, Y і Z статистично незалежні, то біти $F(X, Y, Z)$ і $G(X, Y, Z)$ будуть також статистично незалежні. Функція H реалізує побітовий XOR, вона володіє такою же властивістю, як F і G.

Крок 5. Формування гешу.

Результат (геш-функція) виходить як $ABCD$. Тобто треба виписати 128 біт, починаючи з молодшого біта A, а закінчуючи старшим бітом D.

Реалізація алгоритму мовою C міститься в RFC 1320.

Порівняння з MD5.

MD4 використовує три цикли з 16 кроків кожен, в той час, як MD5 використовує чотири цикли з 16 кроків кожен.

У MD4 додаткова константа в першому циклі не застосовується. Аналогічна додаткова константа використовується для кожного з кроків у другому циклі. Інша додаткова константа використовується для кожного з кроків в третьому циклі. У MD5 різні додаткові константи, $T[i]$, застосовуються для кожного з 64 кроків.

MD5 використовує чотири елементарні логічні функції, по одній на кожному циклі, порівняно з трьома в MD4, по одній на кожному циклі.

У MD5 на кожному кроці поточний результат складається з результатом попереднього кроку. Наприклад, результатом першого кроку є зміненим словом A. Результат другого кроку зберігається в D і утворюється додаванням A до циклічно зрушеному вліво на певне число біт результату елементарної функції. Аналогічно, результат третього кроку зберігається в C і утворюється додаванням D до циклічно зрушеному вліво результату елементарної функції. MD4 це останнє додавання не включає.

Безпека.

Рівень безпеки, який закладався в MD4, був розрахований на створення досить стійких гібридних систем електронного цифрового підпису, заснованих на MD4 і криптосистеми з відкритим ключем. Рональд Рівест вважав, що алгоритм гешування MD4 можна використовувати і для систем, які потребують сильної криптостійкості. Але в той

же час він зазначав, що MD4 створювався передусім як дуже швидкий алгоритм гешування, тому він може бути поганий в плані криптостійкості. Як показали дослідження, він був правим, і для додатків, де важлива насамперед крипостійкість, став використовуватися алгоритм MD5.

Вразливості.

Вразливості в MD4 були продемонстровані у статті Берта ден Боєра і Антона Босселаріса в 1991 році. Перша колізія була знайдена Гансом Доббертіном в 1996 році.

MD2

MD2 (The MD2 Message Digest Algorithm) – геш-функція, розроблена Рональдом Рівестом (RSA Laboratories) в 1989 році, і описана в RFC 1319. Розмір геша – 128 біт. Розмір блоку вхідних даних – 512 біт.

Алгоритм MD2.

Передбачається, що на вхід подано повідомлення, що складається з b байт, геш якого нам належить обчислити. Тут b – довільне невід'ємне ціле число, воно може бути нулем або яким завгодно великим. Запишемо повідомлення побайтово, у вигляді: $m_0 m_1 ... m_{b-1}$.

Далі наведено 5 кроків, які використовуються для обчислення гешу повідомлення.

Крок 1. Додавання відсутніх біт.

Повідомлення розширюється так, щоб його довжина в байтах за модулем 16 дорівнювала 0. Таким чином, у результаті розширення довжина повідомлення стає кратною 16 байтам. Розширення проводиться завжди, навіть якщо повідомлення спочатку має потрібну довжину.

Розширення проводиться таким чином: і байт, рівних i, додається до повідомлення, так щоб його довжина в байтах стала рівною 0 по модулю 16. У результаті до повідомлення додається як мінімум 1 байт, і як максимум 16.

На цьому етапі (після додавання байт) повідомлення має довжину в точності кратну 16 байтам. Нехай $M[0...N-1]$ означає байти отриманого повідомлення (N кратно 16).

Крок 2. Додавання контрольної суми.

16-байтним контрольна сума повідомлення додається до результату попереднього кроку.

Цей крок використовує 256-байтову "випадкову" перестановчу матрицю, що складається з цифр числа пі:

PI_SUBST [256] = {
41, 46, 67, 201, 162, 216, 124, 1, 61, 54, 84, 161, 236, 240, 6, 19,
98, 167, 5, 243, 192, 199, 115, 140, 152, 147, 43, 217, 188, 76, 130, 202,
30, 155, 87, 60, 253, 212, 224, 22, 103, 66, 111, 24, 138, 23, 229, 18,
190, 78, 196, 214, 218, 158, 222, 73, 160, 251, 245, 142, 187, 47, 238, 122,
169, 104, 121, 145, 21, 178, 7, 63, 148, 194, 16, 137, 11, 34, 95, 33,
128, 127, 93, 154, 90, 144, 50, 39, 53, 62, 204, 231, 191, 247, 151, 3,
255, 25, 48, 179, 72, 165, 181, 209, 215, 94, 146, 42, 172, 86, 170, 198,
79, 184, 56, 210, 150, 164, 125, 182, 118, 252, 107, 226, 156, 116, 4, 241,
69, 157, 112, 89, 100, 113, 135, 32, 134, 91, 207, 101, 230, 45, 168, 2,
27, 96, 37, 173, 174, 176, 185, 246, 28, 70, 97, 105, 52, 64, 126, 15,
85, 71, 163, 35, 221, 81, 175, 58, 195, 92, 249, 206, 186, 197, 234, 38,
44, 83, 13, 110, 133, 40, 132, 9, 211, 223, 205, 244, 65, 129, 77, 82,
106, 220, 55, 200, 108, 193, 171, 250, 36, 225, 123, 8, 12, 189, 177, 74,
120, 136, 149, 139, 227, 99, 232, 109, 233, 203, 213, 254, 59, 0, 29, 57,
242, 239, 183, 14, 102, 88, 208, 228, 166, 119, 114, 248, 235, 117, 75, 10,
49, 68, 80, 180, 143, 237, 31, 26, 219, 153, 141, 51, 159, 17, 131, 20};

Крок 3. Формування гешу.

Геш обчислюється як результат $X[0...15]$, на початку йде байт $X[0]$, а наприкінці $X[15]$.

На цьому завершується опис алгоритму MD2. В RFC 1319 можна знайти реалізацію алгоритму мовою С.

Безпека.

Роже і Шаво в 1997 році опублікували приклад колізій для MD2, хоча і не змогли представити алгоритм знаходження інших колізій.

У 2004 році було показано, що MD2 схильний атаці на знаходження колізій зі складністю, еквівалентній 2104 операцій гешування (Міллер, 2004). Міллер заявив: "MD2 не може більше розглядатися як криптостійкий алгоритм гешування".

MD5

MD5 – це поліпшена версія MD4 [16]. Хоча вона складніше MD4, їх схеми схожі, і результатом MD5 також є 128-бітове значення. На рис. 6.7 наведено структуру раунда MD5.

Розглянемо алгоритм отримання дайджеста повідомлення MD5 (RFC 1321), розроблений Роном Рівестом з MIT.

Логіка виконання MD5.

Алгоритм отримує на вході повідомлення довільної довжини і створює в якості виходу дайджест повідомлення довжиною 128 біт. Логіку виконання наведено на рис. 6.8.

Алгоритм складається з таких кроків:

Крок 1: додавання відсутніх бітів.

Повідомлення доповнюється таким чином, щоб його довжина стала дорівнювати 448 за модулем 512 (довжина 448 mod 512). Це означає, що довжина доданого повідомлення на 64 біта менше, ніж число, кратне 512. Додавання виробляється завжди, навіть якщо повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 бітів, воно доповнюється 512 бітами до 960 бітів. Таким чином, число біт, що додаються, знаходиться в діапазоні від 1 до 512.

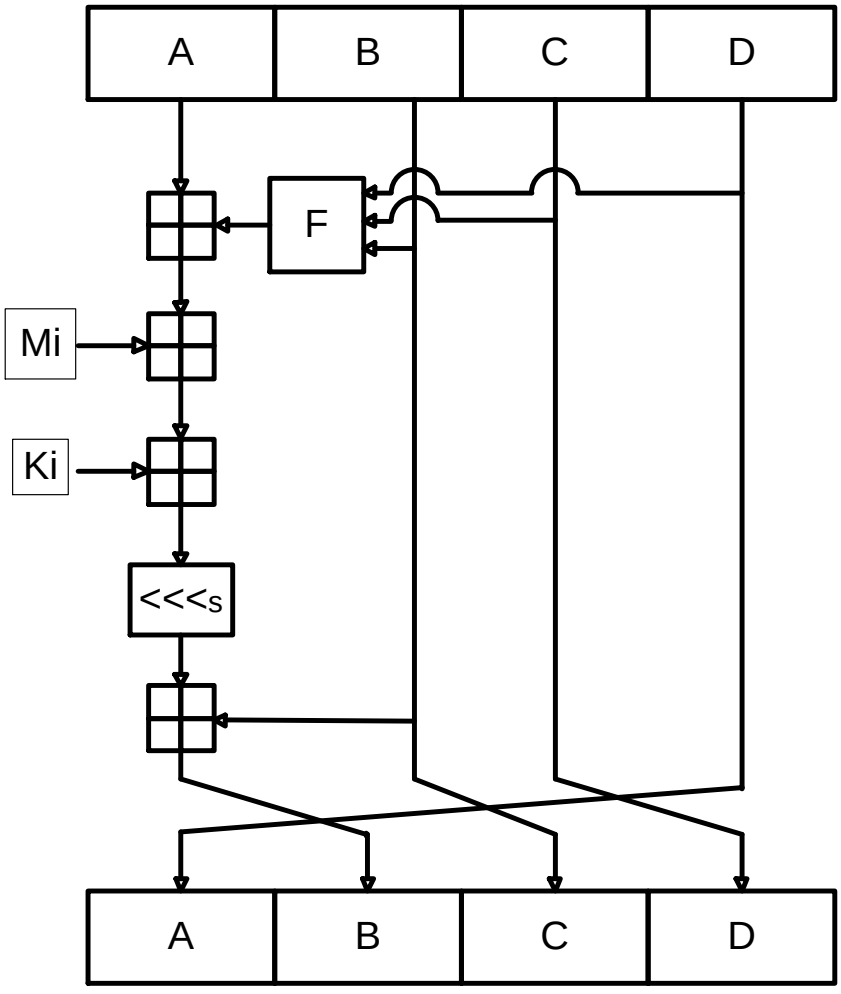


Рис. 6.7. **Структура раунда MD5**

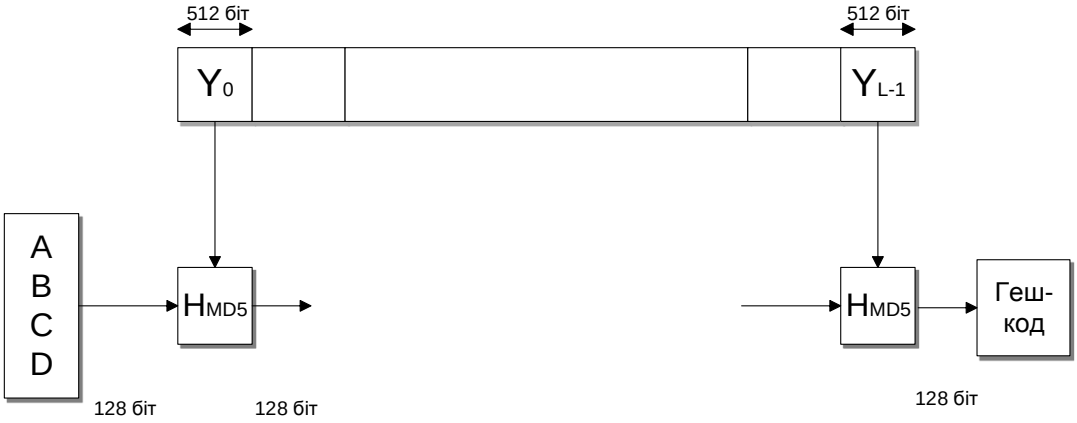


Рис. 6.8. **Логіка виконання MD5**

Додавання складається з одиниці, за якою слід необхідну кількість нулів.

Крок 2: додавання довжини.

64-бітове уявлення довжини вихідного (до додавання) повідомлення в бітах приєднується до результату першого кроку. Якщо первісна довжина більше, ніж 2^{64}, то використовуються тільки останні 64 біти. Таким чином, поле містить довжину вихідного повідомлення за модулем 2^{64}.

У результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам. Це розширене повідомлення, що наведено на рис. 6.9, подається як послідовність 512-бітних блоків $Y_0, Y_1, ..., YL-1$, при цьому загальна довжина розширеного повідомлення дорівнює $L*512$ бітам. Таким чином, довжина отриманого розширеного повідомлення кратна шістнадцяти 32-бітовим словами.

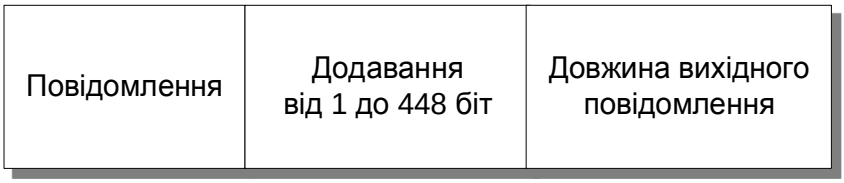


Рис. 6.9. **Структура розширеного повідомлення**

Крок 3: ініціалізація MD-буфера.

Використовується 128-бітний буфер для зберігання проміжних і остаточних результатів геш-функції. Буфер може бути представлений як чотири 32-бітних регістри (A, B, C, D). Ці регістри ініціалізувалися такими шістнадцятковим числами:

A = 01234567; B = 89ABCDEFC; C = FEDCBA98; D = 76543210.

Крок 4: обробка послідовності 512-бітних (16-слівних) блоків.

Основою алгоритму є модуль, що складається з чотирьох циклічних обробок, позначений як HMD5. Чотири цикли мають схожу структуру, але кожен цикл використовує свою елементарну логічну функцію, що позначається f_F, f_G, f_H і f_I відповідно. На рис. 6.10 наведено послідовність обробки 512-бітного блоку.

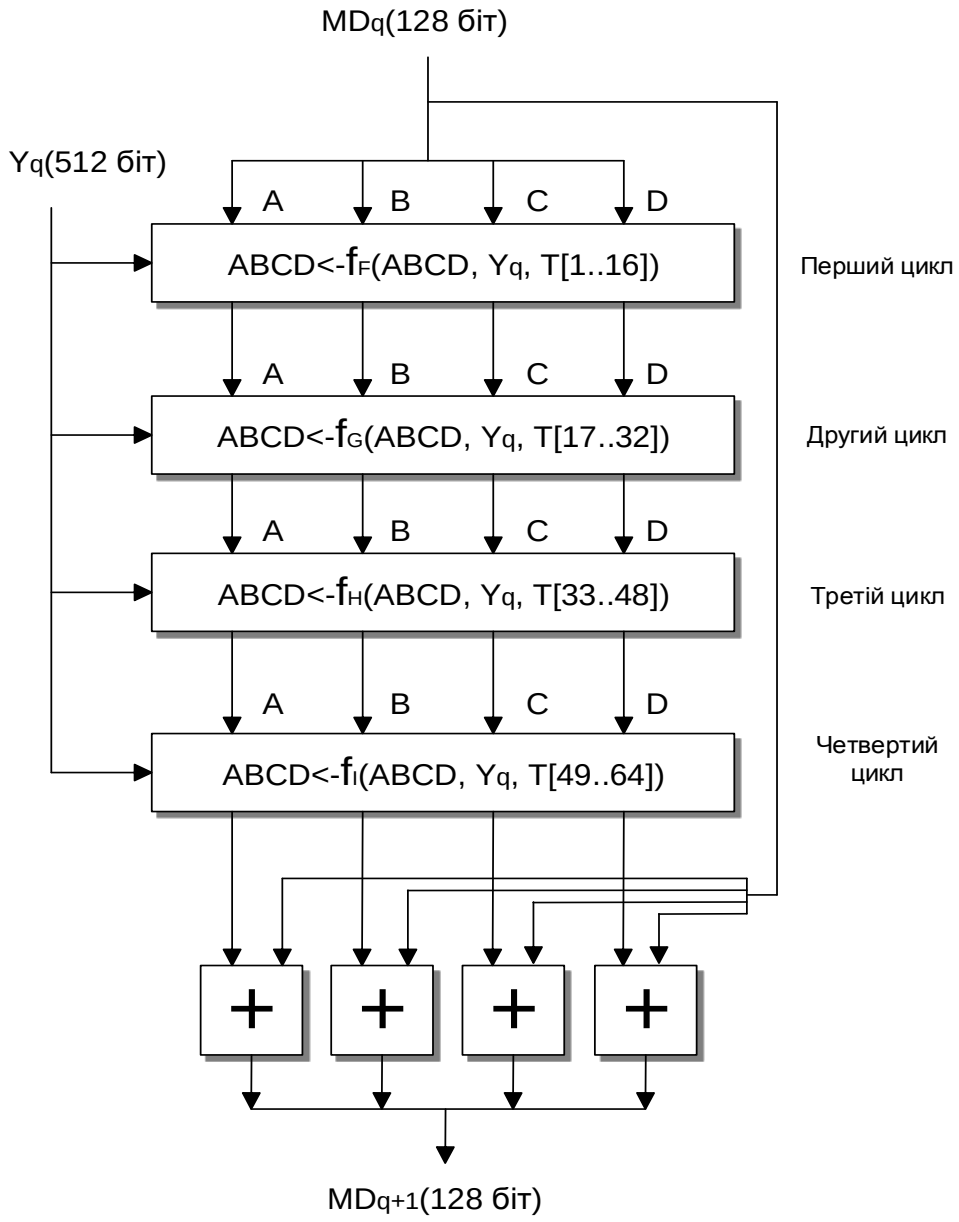


Рис. 6.10. **Обробка чергового 512-бітного блоку**

Кожен цикл приймає на вході поточний 512-бітний блок Y_q, що обробляється в даний момент, і 128-бітове значення буферу ABCD, яке є проміжним значенням дайджесту, і змінює вміст цього буфера. Кожен

цикл також використовує четверту частину 64-елементної таблиці $T[1...64]$, побудованої на основі функції sin. i-ий елемент T, що позначається $T[i]$, має значення, рівне цілій частині від $2^{32} * abs(sin(i))$, і заданий в радіанах. Оскільки $abs(sin(i))$ є числом між 0 і 1, кожен елемент T є цілим, яке може бути представлено 32 бітами. Таблиця забезпечує "випадковий" набір 32-бітних значень, які повинні ліквідувати будь-яку регулярність у вхідних даних. Для отримання MD_{q+1} вихід чотирьох циклів додається за модулем 2^{32} с MD_q. Додавання відбувається незалежно для кожного з чотирьох слів у буфері.

Крок 5: вихід.

Після обробки всіх L 512-бітних блоків виходом L-ої стадії є 128-бітний дайджест повідомлення.

Розглянемо більш детально логіку кожного з чотирьох циклів виконання одного 512-бітного блоку. Кожен цикл складається з 16 кроків, що оперують з буфером ABCD.

Логіку виконання окремого кроку наведено на рис. 6.11. Кожен крок можна подати у вигляді:

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

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 біт 32-бітного аргументу;

$X[k] - M[q*16+k] - k - e$ 32-бітове слово в q-му 512 блоці повідомлення;

$T[i]$ – i-те 32-бітове слово в матриці T;

$+$ – додавання за модулем 2^{32}.

На кожному з чотирьох циклів алгоритму використовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-бітових слова на вході і на виході створює одне 32-бітове слово. Кожна функція є множиною побітових логічних операцій, тобто n-ий біт виходу є функцією від n-ого біта трьох входів. Елементарні функції такі: $f_F = (B \wedge C) \vee (\overline{B} \wedge D)$, $f_G = (B \wedge D) \vee (C \wedge \overline{D})$, $f_H = B \oplus C \oplus D$, $f_I = C \oplus (B \wedge \overline{D})$.

Масив з 32-бітових слів $X[0...15]$ містить значення поточного 512-бітного вхідного блоку, який обробляється в даний момент. Кожен цикл виконується 16 разів, а оскільки кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожен блок вхідного повідомлення обробляється за схемою, показаною на рис. 6.10, 64 рази. Якщо уявити вхідний 512-бітний блок у вигляді шістнадцяти 32-бітових слів, то кожне вхідне 32-бітове слово використовується чотири рази, по одному разу в кожному циклі, і кожен елемент таблиці T, що складається з 64 32-бітних слів, використовується тільки один разів.

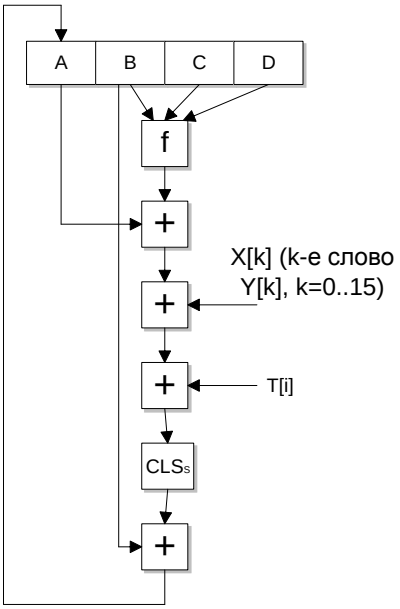


Рис. 6.11. **Логіка виконання окремого кроку**

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 $ABCD$ Отже, кожне слово буфера змінюється 16 разів, і потім 17-ий раз у кінці для отримання остаточного виходу даного блоку. Додавання алгоритмом можна виконувати таким чином:

$$MD_0 = IV$$

$$MD_{q+1} = MD_q + f_I\left[Y_q, f_H\left[Y_q, f_G\left[Y_q, f_F\left[Y_q, MD_q\right]\right]\right]\right]$$

$$MD = MD_{L-1},$$

де IV – початкове значення буфера ABCD, визначене на кроці 3;

Y_q – g-ий 512-бітний блок повідомлення;

L – кількість блоків у повідомленні (включаючи поля доповнення та довжини).

6.3. Алгоритми SHA-1, SHA-2

Геш-функція SHA-1

Алгоритм SHA-1 розроблений у 1992 році і формує за вхідним двійковим рядком довільної довжини 160-бітний геш-код. Алгоритм носить циклічний характер і у своїх циклах використовує тикий набір нелінійних функцій:

$$f(u,v,w) = (u \wedge v) \vee (\bar{u} \wedge w);$$
$$g(u,v,w) = (u \wedge v) \vee (u \wedge w) \vee (v \wedge w);$$
$$h(u,v,w) = u \oplus v \oplus w;$$

де u, v, w – 32-бітні змінні (слова);

$u \wedge v$ – логічне "І" за розрядами;

$u \vee v$ – логічне "АБО" за розрядами;

$\bar{u}$ – логічне "доповнення" за розрядами;

$\oplus$ – додавання за модулем 2.

Далі при описі роботи алгоритму символ "$+$" позначає додавання за модулем 2^{32}, "<<<*s*" – циклічний зсув уліво на *s* розрядів.

SHA-1 реалізує геш-функцію, побудовану на ідеї функції стиснення. Входами функції стиснення є блок повідомлення довжиною 512 біт і вихід попереднього блоку повідомлення. Вихід є значення всіх геш-блоків до цього моменту. Іншими словами геш-блоку M_i дорівнює $h_i = f(M_i, h_{i-1})$ Геш-значенням всього повідомлення є вихід останнього блоку.

Алгоритм SHA-1.

Вхід. Двійковий рядок *x* довжиною $b \geq 0$.

Вихід. 160-бітний геш-код за рядком *x*.

1. *Ініціалізація.* Ініціалізувати п’ять векторів ініціалізації:

$h_1 = 67452301_x$; $h_2 = efcdab89_x$; $h_3 = 98badcfe_x$;

$h_4 = 10325476_x$; $h_5 = c3d2e1f0_x$.

Задати чотири додаткові константи.

$y_1 = 5a827999_x$; $y_2 = 6ed9eba1_x$;

$y_3 = 8f1bbcdc_x$; $y_4 = ca62c1d6_x$.

2. *Передобробка.* Доповнити рядок *x* так, щоб його довжина була кратна числу 512. Для цього додати в кінець рядка одиницю, а потім стільки нульових біт, скільки необхідно для одержання рядка довжиною на 64 біти коротше довжини, кратної 512. Додати останні два 32-х розрядні слова, що містять двійкове представлення числа b. Нехай m – кількість 512-бітних блоків в отриманому доповненому рядку. Таким чином, форматований вхід буде складатися з 16^m 32-х розрядних слів $x_0x_1...x_{16m-1}$. Ініціалізувати вектор змінних зчеплення:

$$(H_1,H_2,H_3,H_4,H_5) = (h_1,h_2,h_3,h_4,h_5).$$

3. *Обробка.* Для всіх i від 0 до m-1 кожний i-й блок із шістнадцяти 32-х бітних слів перетворити у вісімдесят 32-х бітних слів і записати в тимчасовий масив за таким алгоритмом:

$$X_i = x_{16i+j}, \quad 0 \le j \le 15;$$

$$X_i = ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{J-16}) <<< 1).$$

Зазначимо, що у початковій версії, що називалася SHA, алгоритм не містив лівого зрушення на один біт. Ця зміна введена для посилення геш-функції. У результаті отриманий алгоритм SHA-1.

Ініціалізувати робочі змінні:

$$(A,B,C,D,E) = (H_1,H_2,H_3,H_4,H_5).$$

Обчислити для всіх i від 0 до m-1:

Для $j = 0$ до 19

$$t = ((A <<< 5) + f(B,C,D) + E + X_j + y_1);$$
$$(A,B,C,D,E) = (t, A, B <<< 30, C, D).$$

Для $j = 20$ до 39

$$t = ((A <<< 5) + h(B,C,D) + E + X_j + y_2);$$
$$(A,B,C,D,E) = (t, A, B <<< 30, C, D).$$

Для $j = 40$ до 59

$$t = ((A <<< 5) + g(B,C,D) + E + X_j + y_3);$$
$$(A,B,C,D,E) = (t, A, B <<< 30, C, D).$$

Для $j = 60$ до 79

$$t = ((A <<< 5) + h(B,C,D) + E + X_j + y_4);$$
$$(A,B,C,D,E) = (t, A, B <<< 30, C, D).$$

Обновити вектор змінних зчеплення:

$$(H_1,H_2,H_3,H_4,H_5) = (H_1 + A,H_2 + B,H_3 + C,H_4 + D,H_5 + E).$$

4. *Завершення*. Як геш-код прийняти величину:

$$H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4 \,||\, H_5 .$$

Порівняно з 128-розрядними геш-функціями, 160-розрядний геш-код, який виробляється SHA-1, забезпечує більшу стійкість до силових атак. Геш-функції SHA-1 і RIPEMD-160 за стійкістю приблизно рівні й обидві перевершують MD5. Розширення блоку вхідного повідомлення введено з метою забезпечення більшої відмінності між вхідними блоками. Надмірність, що вводиться, сприяє підвищенню стійкості геш-функції. На рис. 6.12 наведена схема однієї ітерації алгоритму SHA-1.

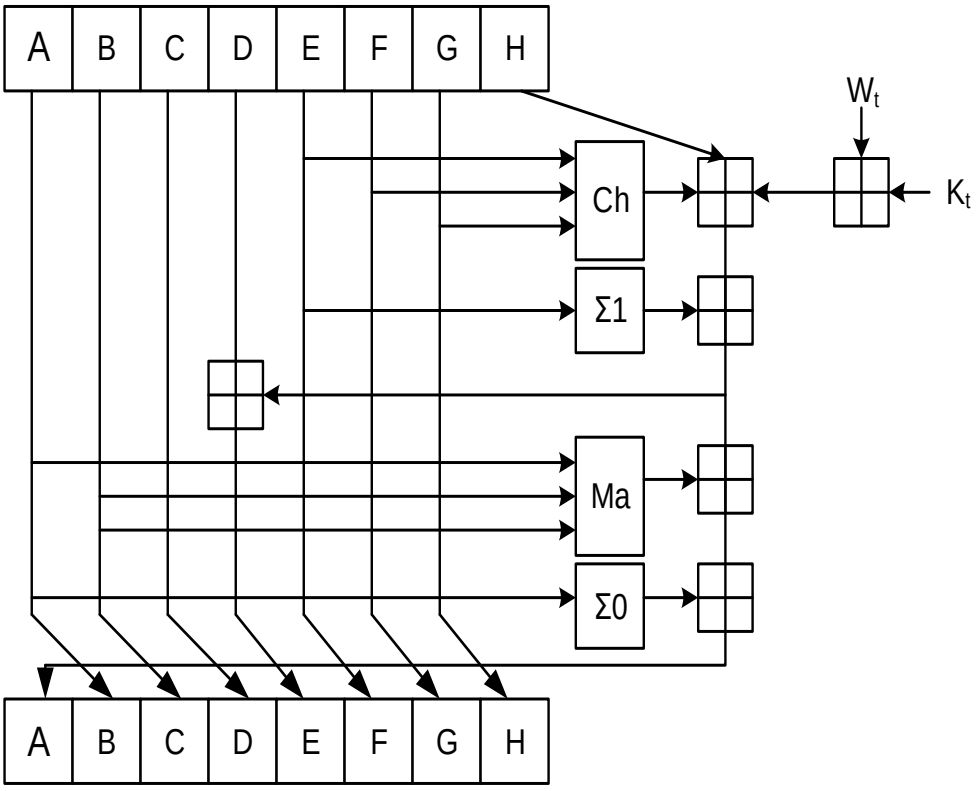


Рис. 6.12. **Одна ітерація алгоритму SHA1**

SHA-2

Secure Hash версія Алгоритм 2 – безпечний алгоритм гешування, версія 2 – збірна назва односпрямованих геш-функцій SHA-224, SHA-256, SHA-384 і SHA-512. Геш-функції призначені для створення

"відбитків" або "дайджестів" повідомлень довільної бітової довжини. Застосовуються в різних додатках або компонентах, пов’язаних із захистом інформації.

Геш-функції SHA-2 розроблені Агентством національної безпеки США і опубліковані Національним інститутом стандартів і технологій у федеральному стандарті обробки інформації FIPS PUB 180-2 в серпні 2002 року. У цей стандарт також увійшла геш-функція SHA-1, розроблена в 1995 році. У лютому 2004 року в FIPS PUB 180-2 була додана SHA-224. У жовтні 2008 року вийшла нова редакція стандарту – FIPS PUB 180-3.

У липні 2006 року з’явився стандарт RFC 4634 "Безпечні геш-алгоритми США (SHA і HMAC-SHA)", що описує SHA-1 і сімейство SHA-2. Агентство національної безпеки від імені держави випустило патент на SHA-2 під ліцензією Royalty Free.

Загальний опис.

Геш-функції сімейства SHA-2 побудовані на основі структури Меркле – Дамгарда.

Початкове повідомлення після доповнення розбивається на блоки, кожен блок – на 8 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64-ма чи 80-ма ітераціями (раундами). На кожній ітерації 2 слова з восьми перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням геш-функції.

Алгоритм використовує такі бітові операції:

‖ – конкатенація;

+ – додавання;

and – побітове "І";

or – побітове "АБО";

xor – виключає "АБО";

shr – логічний зсув вправо;

rotr – циклічний зсув вправо.

На рис. 6.13 наведена схема однієї ітерації алгоритму SHA-2.

У табл. 6.12 наведено деякі технічні характеристики різних варіантів SHA-2. "Внутрішній стан" означає проміжну геш-суму після обробки чергового блоку даних.

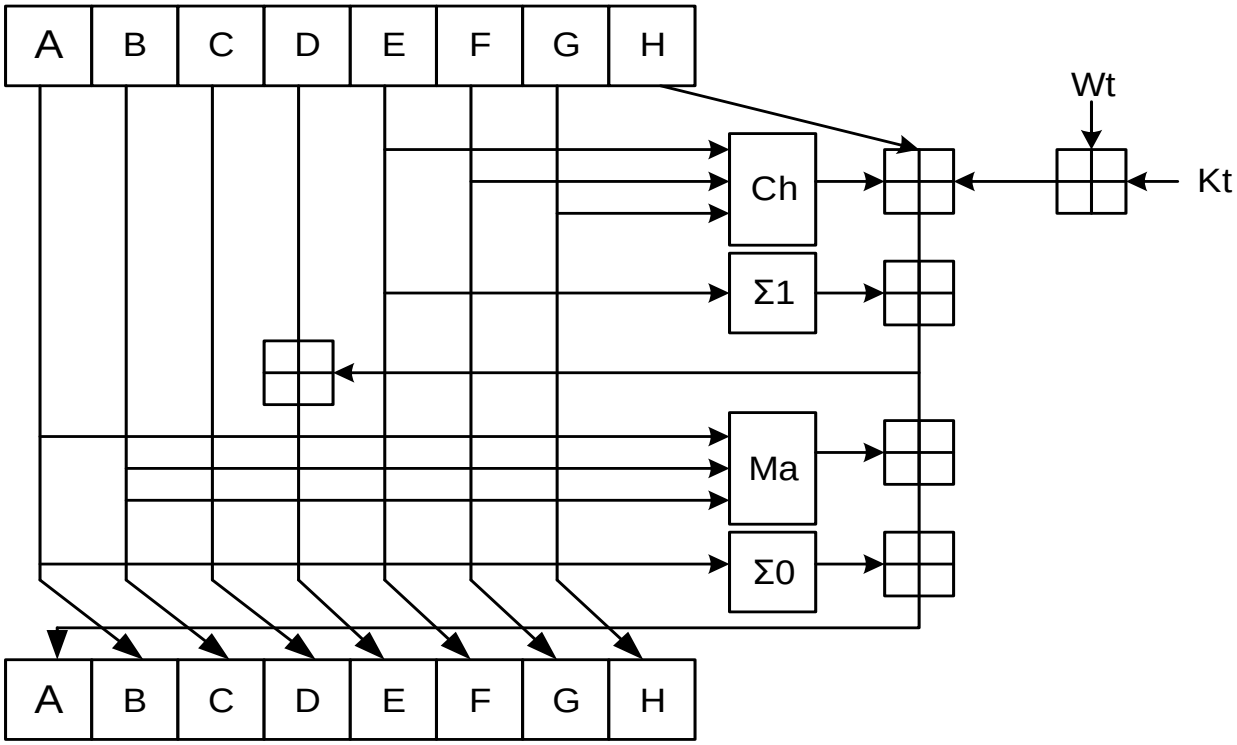


Рис. 6.13. **Схема однієї ітерації алгоритмів SHA-2**

Таблиця 6.12

Характеристики SHA-2

Геш-функція	Довжина дайджесту повідомлення (біт)	Довжина внутрішнього стану (біт)	Довжина блоку (біт)	Максимальна довжина повідомлення (біт)	Довжина слова (біт)	Кількість ітерацій в циклі
SHA-256/224	256/224	256	512	$2^{64} - 1$	32	64
SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80

Геш-функції SHA-2 використовуються для перевірки цілісності даних і в різних криптографічних схемах. На 2008 рік сімейство геш-функцій SHA-2 не має такого широкого розповсюдження, як MD5 і SHA-1 незважаючи на виявлені в останніх недоліки.

6.4. Огляд алгоритмів гешування конкурсу SHA-3

Основні вимоги, висунуті Національним інститутом стандартів і технологій США (NIST) до алгоритмів-конкурсантів, припускають

створення класу геш-функцій потенційно стійких до атак, націлених на SHA-2, а також збереження або збільшення ефективності гешування порівняно з SHA-2 [23; 57].

Алгоритм-переможець конкурсу SHA-3 повинен підтримувати розмір вихідного блоку 224, 256, 384 і 512 бітів. Використання дайджестів геш-кодів довжиною 160-біт не допускаються через можливість знаходження колізій атаками грубої сили (повного перебору всіх варіантів). При проведенні конкурсу зберігаються ті ж вимоги, що і до попередніх геш-функцій: максимальний розмір вхідного значення, розмір вихідного значення, колізійна стійкість, стійкість до знаходження прообразу і другого прообразу, потоковий режим обчислення "за один прохід" [23].

Алгоритми функцій обчислення для різного розміру блоків повинні бути ідентичні і мати мінімум відмінностей в реалізації. Використання абсолютно різних наборів алгоритмів для отримання чотирьох фіксованих значень довжини виходу не допускається [23].

Крім цього висуваються вимоги щодо удосконалення існуючих алгоритмів роботи геш-функцій: можливо включення опції рандомізованого гешування, покращене розпаралелювання, оптимальна робота на безлічі сучасних платформ (включаючи як 64-бітові процесори, так і 8-бітові процесори, а також смарт-карти) [3].

Керівники конкурсу допускають використання параметризації в алгоритмах-конкурсантів – зміна параметрів числа раундів з урахуванням допустимих меж "ефективності в обмін на безпеку". Хоча в стандарт така можливість не буде включена.

У геш-функції класу SHA-3 можуть бути вбудовані процедури погодження для обчислення кодів автентифікації повідомлень (HMAC): наприклад, передача в якості одного з вхідних параметрів довжини блоку вхідного повідомлення, якщо вона заздалегідь точно відома. Значення всіх вбудованих параметрів і констант повинні бути сконструйовані таким чином, щоб не дати можливості вбудовування лазівок і відмичок з боку розробників, для чого мають бути приведені відповідні докази та процедури перевірки [3].

Проведений аналіз специфічних вимог до стійкості показав, що основною вимогою є забезпечення теоретичних меж стійкості (в реаль-

ності вони можуть бути трохи менше) на розмір вихідної блоку в n біт і становитиме:

в конструкціях кодів автентифікації повідомлень (HMAC) і псевдовипадкових функцій (PRF) стійкість за кількістю запитів повинна бути не менше 2 (n / 2) біт проти атак-розрізнювачів;

стійкість рандомізованого гешування проти атаки знаходження H (M_1, r_1) = H (M_2, r_2) – не менше n біт, за умови, що значення рандомізатора r_1 не контролюється атакуючим.

Стандартні та додаткові параметри стійкості:

стійкість до знаходження колізій – не менше n / 2 біт;

стійкість до знаходження прообразу – n біт;

стійкість до знаходження другого прообразу – n-k бітів для будь-якого повідомлення, коротше 2^k бітів;

стійкість до атак на зміну довжини повідомлення;

підмножина геш-функцій (наприклад, отримане усіканням числа бітів виходу) розміром m бітів повинно зберігати властивості n-бітного множини в перерахунку на m, з урахуванням статистичних (не відрізняються від випадкових) відхилень. Не повинно існувати атаки на швидке знаходження малостійких підмножин геш-функції з вихідної множини n;

стійкість до нових типів атак, наприклад, на основі мультиколізій.

Національним інститутом стандартів і технологій США 29 березня 2012 року був проведений третій раунд конкурсу SHA-3, у табл. 6.13 наведені основні характеристики алгоритмів-претендентів.

Таблиця 6.13

Характеристики алгоритмів гешування кандидатів 3 раунду

Назва алгоритму	Платформа	Стійкість алгоритму	Швидкодія (Мбіт/с)
BLAKE	8, 32,64	є достатньо стійким	44,37
Gröstl		нестійкий до атак "напіввільний початок"	7,98
JH	8, 32, 64	низький алгебраїчний ступінь у висновках функції стиснення	10,6
Keccak	32, 64	кількість раундів, необхідна для захисту від атак – 18	8,05
Skein	8, 32, 64	Захищений від атак – підбір подовжених повідомлень і псевдо коллізій	39,23

Аналіз табл. 6.13 показує, що основну увагу розробників алгоритмів-конкурсантів було направлено на виконання основних вимог щодо продуктивності і можливості оптимальної роботи алгоритму при його реалізації на безлічі сучасних платформ. Разом з тим керівники конкурсу при відборі алгоритмів-кандидатів у другий тур звернули увагу на стійкість даних алгоритмів до різних видів атак.

У 3 раунд були відібрані алгоритми гешування, в яких не була порушена будь-яким чином безпека. У деяких випадках, представлений варіант був занадто повільним чи вимагав занадто багато пам'яті, але NIST вважає, що в деяких випадках, настроюються параметри можуть бути відкоректовані, щоб дати прийнятний рівень продуктивності без шкоди для безпеки [33].

Попереднє вивчення алгоритмів-учасників показало, що кожен з них запозичив деякі принципи побудови у алгоритму AES для забезпечення стійкості геш-кодів.

Таким чином, головною вимогою до алгоритмів-конкурсантів керівники NIST висунули безпеку, що і стало основним критерієм відбору кандидатів у 3 раунд. Тільки 5 алгоритмів з 14 відібраних після другого раунду (всього на конкурс було подано 51 алгоритм) змогли продемонструвати необхідний рівень захисту від криптографічних атак. До основних критеріїв відбору також відноситься продуктивність і можливість роботи на великому числі платформ, що також дозволило керівникам конкурсу "відсіяти" кілька кандидатів (алгоритми Vortex, LUX і т. д.).

BLAKE (Jean-PhilippeAumasson)

BLAKE Алгоритм стиснення, функція якого заснована на використанні ключової підстановки в конструкції Davies-Meyer. Ключова підстановка заснована на внутрішній організації потокового шифру ChaCha. Отримав свою не лінійність з накладенням модульного складання і операцій XOR. Сама інноваційна частина BLAKE – своя ключова перестановка.

Структурна схема алгоритму наведено на рис. 6.14.

Геш-функція BLAKE-32 працює з 32-бітними словами й повертає 32-байтовое значення геша. Цей відрізок визначає BLAKE-32, виходячи з постійних параметрів власної функції стиску, а потім переходить до його ітеративного режиму.

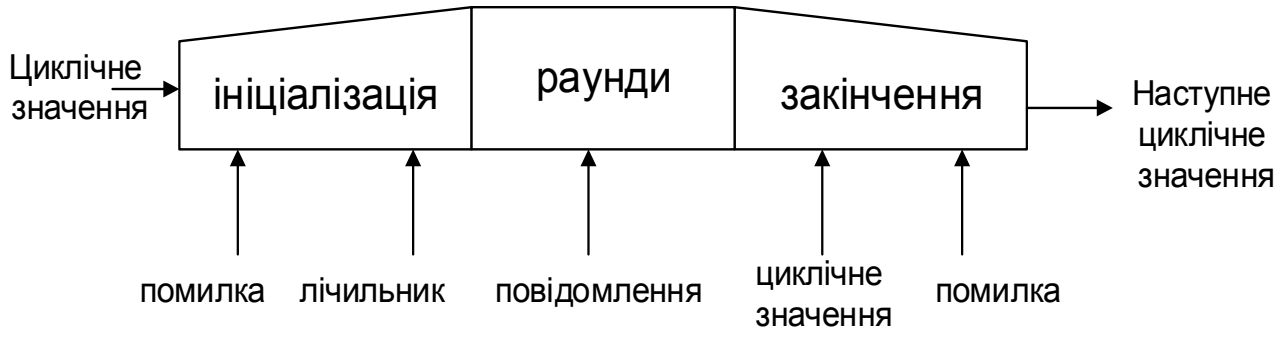


Рис. 6.14. **Структурна схема алгоритму BLAKE**

Константи.

BLAKE-32 запускає гешування від того ж початкового значення, що й SHA-256:

IV_0 = 6A09E667 IV_1 = BB67AE85

IV_2 = 3C6EF372 IV_3 = A54FF53A

IV_4 = 510E527F IV_5 = 9B05688C

IV_6 = 1F83D9AB IV_7 = 5BE0CD19

BLAKE-32 використовує 16 постійних значень:

c_0 = 243F6A88 c_1 = 85A308D3

c_2 = 13198A2E c_3 = 03707344

c_4 = A4093822 c_5 = 299F31D0

c_6 = 082EFA98 c_7 = EC4E6C89

c_8 = 452821E6 c_9 = 38D01377

c_{10} = BE5466CF c_{11} = 34E90C6C

c_{12} = C0AC29B7 c_{13} = C97C50DD

c_{14} = 3F84D5B5 c_{15} = B5470917

Десять перестановок {0, . . ., 15}, що використовуються всіма функціями BLAKE, наведено в табл. 6.14.

Функція стиску.

Функція стиску BLAKE-32 приймає на вході чотири значення:

• ланцюгове значення h = h_0, . . ., h_7;

• блок повідомлення = m_0, . . .,m_{15};

• сіль s = s_0, . . ., s_3;

• лічильник t = t_0, t_1.

Таблиця 6.14

Перестановки {0, . . ., 15} використовувані функціями BLAKE

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

Ці чотири входи є в підсумку 30 словами (тобто, 120 байт = 960 біт). На виході функції – нове ланцюгове значення h’ = $h'_0, \ldots, h'_7$ з восьми слів (тобто, 32 байта = 256 біт). Записуємо стиск h, m, s, t в h’ як h′ = compress (h, m, s, t).

Ініціалізація.

Стан 16-слова $v_0, \ldots, v_{15}$ ініціалізовано так, що різні входи виробляють різні початкові стани. Стан представляється як матриця розміром 4×4, і заповнюється так, як зазначено далі:

$$\begin{pmatrix} \nu_0 & \nu_1 & \nu_2 & \nu_3 \\ \nu_4 & \nu_5 & \nu_6 & \nu_7 \\ \nu_8 & \nu_9 & \nu_{10} & \nu_{11} \\ \nu_{12} & \nu_{13} & \nu_{14} & \nu_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}.$$

Циклічна функція.

Як тільки стан ν проініціалізовано, функція стиску виконує ітерації серіями по 10 раундів. Раунд – це перетворення значення ν, яке обчислюється:

$G_0(v_0, v_4, v_8, v_{12})$ $G_1(v_1, v_5, v_9, v_{13})$ $G_2(v_2, v_6, v_{10}, v_{14})$ $G_3(v_3, v_7, v_{11}, v_{15})$
$G_4(v_0, v_5, v_{10}, v_{15})$ $G_5(v_1, v_6, v_{11}, v_{12})$ $G_6(v_2, v_7, v_8, v_{13})$ $G_7(v_3, v_4, v_9, v_{14})$

де, на раунді r, $G_i(a, b, c, d)$ множини.

$$a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)})$$
$$d \leftarrow (d \oplus a) <<< 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) <<< 12$$
$$a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)})$$
$$d \leftarrow (d \oplus a) <<< 8$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) <<< 7$$

Перші чотири виклики $G_0,\ldots,G_3$ можуть бути розраховані паралельно, тому що кожний з них обновляє окремий стовпець матриці. Необхідно звернутися до процедури обчислення $G_0,\ldots,G_3$ крок стовпець.

Так само останні чотири виклики $G_4,\ldots,G_7$ обновлюють певні діагоналі й таким способом можуть бути синхронізовані так, що треба звертатися до діагонального кроку.

На рис. 6.15 і 6.16 наведено G_i, крок-стовпець і діагональний крок.

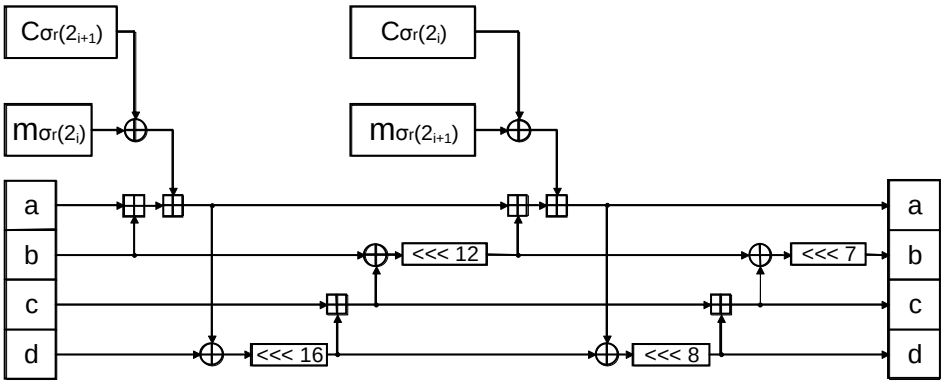


Рис. 6.15. **Функція Gi**

Завершення.

Після послідовності циклів нове ланцюгове значення $h'_0, \ldots, h'_7$ отримується із установленого $v_0,\ldots,v_{15}$ із входом початкового ланцюгового значення $h_0,\ldots,h_7$ і сіллю $s_0,\ldots,s_3$:

$$h'_0 \leftarrow h_0 \oplus s_0 \oplus \nu_0 \oplus \nu_8$$
$$h'_1 \leftarrow h_1 \oplus s_1 \oplus \nu_1 \oplus \nu_9$$
$$h'_2 \leftarrow h_2 \oplus s_2 \oplus \nu_2 \oplus \nu_{10}$$
$$h'_3 \leftarrow h_3 \oplus s_3 \oplus \nu_3 \oplus \nu_{11}$$
$$h'_4 \leftarrow h_4 \oplus s_0 \oplus \nu_4 \oplus \nu_{12}$$
$$h'_5 \leftarrow h_5 \oplus s_1 \oplus \nu_5 \oplus \nu_{13}$$
$$h'_6 \leftarrow h_6 \oplus s_2 \oplus \nu_6 \oplus \nu_{14}$$
$$h'_7 \leftarrow h_7 \oplus s_3 \oplus \nu_7 \oplus \nu_{15}$$

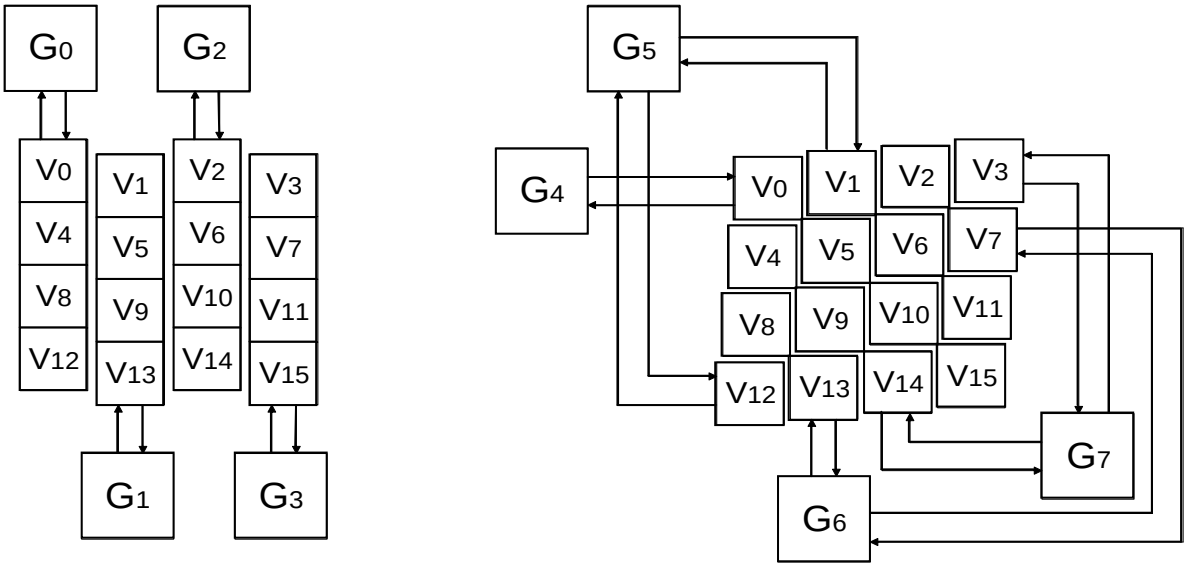


Рис. 6.16. **Крок-стовпець і діагональний крок**

Гешування повідомлення.

Процедура для гешування повідомлення *m* розрядної довжини $\ell < 2^{64}$ притаманна циклічним геш-функціям, повідомлення спочатку *доповнюється:* при додаванні BLAKE використовує правило, подібне до правила HAIFA, потім оброблений блок приймається функцією стиску.

Доповнення.

Спочатку повідомлення розширюють таким чином, що його довжина збігається з 447 за модулем 512. Довжина збільшується, додаючи в кінець достатню кількість 0 біт. Збільшення як мінімум на один біт і як максимум на 512. Потім приєднується одиничний біт, за ним іде 64-бітне невизначене багатобайтово представлення ℓ. Доповнення може бути представлено, як:

$$m \leftarrow m \,||\, 1000...0001\langle l \rangle_{64}.$$

Ця процедура гарантує, що бітова довжина повідомлення, що доповнюється, кратна 512.

Гешування.

Щоб приступити до гешування, доповнене повідомлення розбивається на блоки по 16-слів $m_0,\ldots,m_{n-1}$. Нехай ℓ_i кількість біт у повідомленні $m_0,\ldots,m_i$, тобто крім бітів, доданих при доповненні. Наприклад, якщо оригінальне (не доповнене) повідомлення 600-біт у довжину, то доповнене повідомлення буде мати два блоки, і $\ell^0 = 512$, $\ell^1 = 600$. Часний випадок відбувається, коли останній блок не містить жодного біта оригінального повідомлення; наприклад 1020-розрядне повідомлення приводить до доповненого повідомлення із трьома блоками (які містять

відповідно 512, 508, і 0 біт повідомлення), і треба встановити $\ell^0 = 512$, $\ell^1 = 1020$, $\ell^2 = 0$. Загальне правило: якщо останній блок не містить жодного біта оригінального повідомлення, тоді лічильник зводиться до нуля; це гарантує, що, якщо $i \neq j$, то $\ell_i \neq \ell_j$.

Сіль *s* вибирається користувачем, і встановлюється нульовою, якщо ніякої солі не потрібно (тобто, $s_0 = s_1 = s_2 = s_3 = 0$). Гешування доповненого повідомлення m проводиться, як зазначено далі:

$$h^0 \leftarrow IV$$
$$for i = 0,..., N-1$$
$$h^{i+1} \leftarrow compress(h^i, m^i, s, l^i)$$
$$return h^N .$$

Процедура гешування m з BLAKE-32 записується BLAKE-32(m, s)= h_n, де m (не доповнене) повідомлення, і s – соль. Запис BLAKE-32(m) означає гешування m, коли ніяка сіль не використовується (тобто, s = 0).

Регульований параметр.

При його виклику для нової геш-функції [36] NIST сприяє опису параметра, який допускає швидкість/довіру вибору оптимального розв'язку. Для BLAKE цей параметр – *кількість циклів*. 5 раундів – мінімум для BLAKE-32 (і BLAKE-28). Для BLAKE-64 (і BLAKE-48), 7 раундів – мінімум.

Grostl (LarsRamkildKnudsen)

Grostl – колекція геш-функцій, здатних повертати короткі виклади повідомлення будь-якої кількості байтів від 1 до 64, тобто від 8 до 512 біт у 8 бітних кроках. Різновиди, що повертають n біт називаються Grostl-n. Це включає розміри короткого викладу повідомлення 224, 256, 384, і 512 біт.

Конструкція геш-функції.

Геш-функції Grostl повторюють функцію стиску f таким чином. Повідомлення M доповнене й розділене на блоки повідомлення по ℓ біт $m_1, \ldots, mt$, і кожний блок повідомлення обробляється послідовно.

Початкове ℓ-бітне значення h_0 = iv визначене, і згодом блоки повідомлень mi обробляються як: $h_i \leftarrow (h_{i-1}, m_i)$, for i = 1,…,t.

Отже, в картах два входи по ℓ біт кожний і вихід з ℓ біт. Перший вхід називається послідовним входом, а другий вхід названий блоком

повідомлення. Різновиди Grostl повертають аж до 256 біт, ℓ визначено як 512. Для більших різновидів, ℓ – 1024.

Після того, як останній блок повідомлення був оброблений, вихід геш-функції H(M) обчислюється, як:

$$H(M)=\Omega(h_t),$$

де Ω – вихідне перетворення. Вихідний розмір Ω – n біт, тому варто зазначити, що n< ℓ . Струтктуру геш-функції наведено на рис. 6.17.

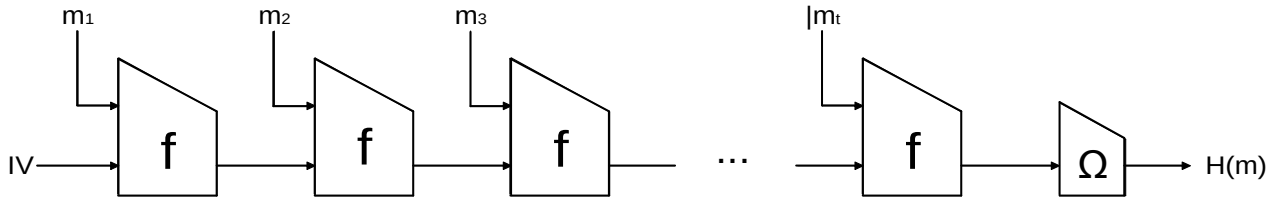


Рис. 6.17. **Геш-функція Grostl**

Конструкція функції стиску.

Функція стиску заснована на двох основні ℓ-бітних перестановках P і Q, які визначені в такий спосіб:

$$f(h,m)= P(h\oplus m)\oplus Q(m)\oplus h.$$

На рис. 6.18 наведено конструкцію функції стиску f.

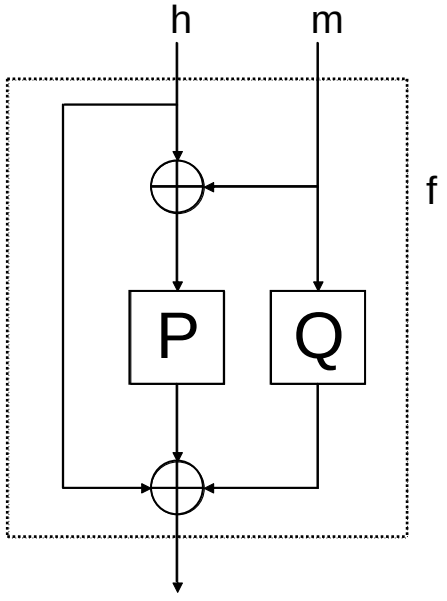


Рис. 6.18. **Функція стиску *f*. *P* і *Q* – ℓ-бітні перестановки**

Вихідне перетворення.

Нехай truncn(x) є операцією, яка відкидає всі майже кінцеві n біти x. Потім вихідне перетворення (рис. 6.19) визначається, як:

$$\Omega(x) = \text{trunc}n(P(x) \oplus x).$$

Тобто вихідне перетворення Ω розраховується як P(x) ⊕x, а потім виключає вихід, повертаючи тільки останні n біт.

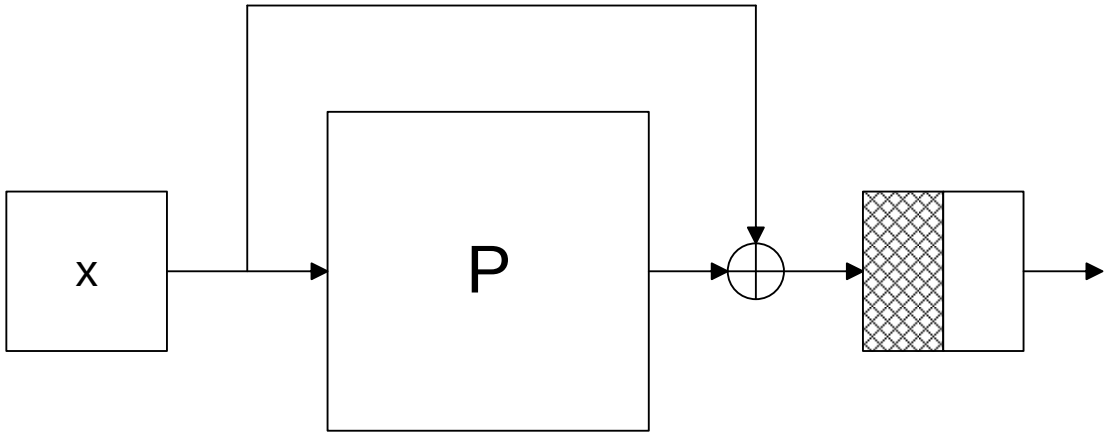


Рис. 6.19. **Вихідне перетворення Ω**

Проектування від байтової послідовності до встановленої матриці й навпаки. Оскільки Grostl оперує байтами, це в основному невизначені байти. Однак, необхідно конкретизувати, як саме байтова послідовність проектується на матрицю *A* і навпаки. Цей розподіл зроблений аналогічним способом як в Rijndael. Отже, 64-байтова послідовність 00 01 02...3f відображається як матриця 8 × 8.

$$\begin{bmatrix} 00 & 08 & 10 & 18 & 20 & 28 & 30 & 38 \\ 01 & 09 & 11 & 19 & 21 & 29 & 31 & 39 \\ 02 & 0a & 12 & 1a & 22 & 2a & 32 & 3a \\ 03 & 0b & 13 & 1b & 23 & 2b & 33 & 3b \\ 04 & 0c & 14 & 1c & 24 & 2c & 34 & 3c \\ 05 & 0d & 15 & 1d & 25 & 2d & 35 & 3d \\ 06 & 0e & 16 & 1e & 26 & 2e & 36 & 3e \\ 07 & 0f & 17 & 1f & 27 & 2f & 37 & 3f \end{bmatrix}$$

Для матриці 8×16, цей метод розширюється природним шляхом. Розподіл від матриці до байтової послідовності – просто зворотна операція.

Кількість раундів.

Кількість раундів r є регульованим параметром безпки. Рекомендуються такі величини r для чотирьох перестановок.

Перестановки	Розмір викладу	Рекомендоване значення *r*
P_{512} і Q_{512}	8-256	10
P_{1024} і Q_{1024}	264-512	14

Початкові значення.

Початкове значення ivngrostl-n – ℓ-бітне представлення n. У табл. 6.15 наведено початкові значення необхідні для вихідних розмірів 224, 256, 384, і 512 біт.

Таблиця 6.15

Початкові значення

n	ivn
224	00...00 00 e0
256	00...00 01 00
384	00...00 01 80
512	00...00 02 00

Доповнення. Довжина кожного блоку повідомлення – ℓ. Щоб бути здатним подіяти на входи змінної довжини, функція доповнення pad визначена. Ця функція доповнення приймає рядок x довжиною в N біт і повертає доповнений рядок x* = pad(x) довжиною, яка є кратною ℓ.

Функція, що доповнює, робить такі дії. Спочатку вона додає "1" біт до x. Потім вона додає w = – N – 65 modℓ "0" біт, і нарешті вона додає 64-бітне представлення (N + w + 65)/ℓ. Це число є цілим через вибір w і представляє кількість блоків повідомлення в кінцевому, доповненому повідомленні.

Оскільки це повинне бути доступним для кодування кількості блоків повідомлення, в доповненому повідомленні в межах 64 біт, максимальна довжина повідомлення – 65 біт до 264-1 блоків повідомлення. Для коротких варіантів, максимальна довжина повідомлення в бітах – 512·$(2^{64} – 1) – 65 = 2^{73} – 577$, і для довших віріантів – це $1024·(2^{64} – 1) – 65 = = 2^{74} – 1089$. Початкове значення Grostl-n – ℓ-бітне представлення n.

В остаточному підсумку вихід останнього виклику f обробляється вихідним перетворенням Ω, яке скорочує розмір висновку з ℓ до n бітів.

JH (HongjunWu)

JH – використовує нову конструкцію, що дещо нагадує конструкцію "sponge" для вбудовування алгоритму геш у блок-підстановку. Блок-підстановка – це комбінація з двох 4-бітових S-боксу з низкою лінійних

операцій змішування і розрядних підстановок. Уся нелінійність у цьому дизайні отримана з S-боксів.

Структурну схему алгоритму наведено на рис. 6.20.

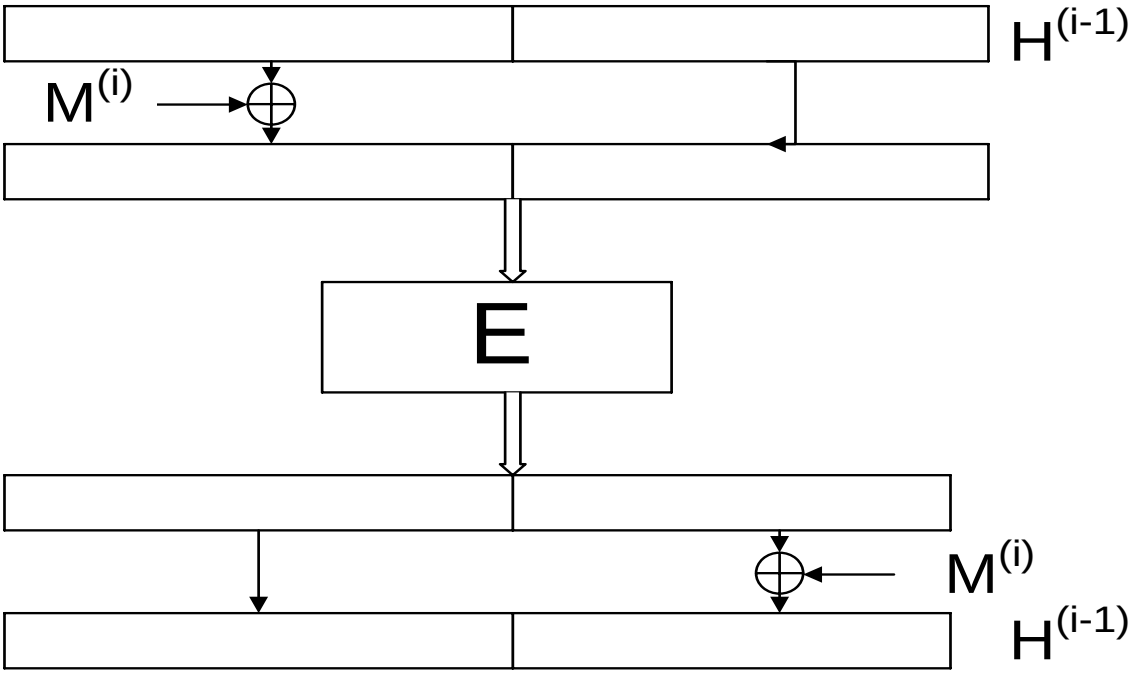


Рис. 6.20. **Структурна схема алгоритму JH**

Keccak (JoanDaemen)

Keccak – використовує конструкцію "sponge" і блок-підстановку. Підстановка може бути реалізована на основі 5-бітових S-блоків або на комбінації лінійної і нелінійної операціях змішування.

Структурну схему алгоритму наведено на рис. 6.21.

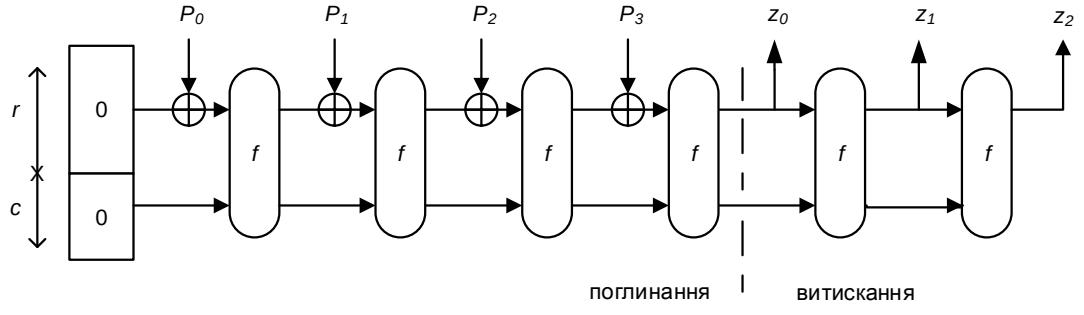


Рис. 6.21. **Структурна схема алгоритму Keccak**

Skein

Skein – це сімейство геш-функцій із трьома різними внутрішньо встановленими розмірами: 256, 512, і 1024 біти.

Skein-512. Воно може безпечно використовуватися для всіх поточних додатків, що гешують, і повинно залишитися безпечним для найближчого майбутнього.

Skein 1024. Має двічі внутрішньо встановлений розмір – Skein-512, він невдало сприятливий, навіть якщо якій-небудь майбутній атаці вдалося зламати Skein-512, то зовсім імовірно, що Skein-1024 залишився б безпечним. Skein-1024 може також працювати близько двох раз швидше, ніж Skein-512 у призначених апаратних реалізаціях.

Skein-256. Він може реалізовуватися, використовуючи близько 100 байтів RAM.

Нова ідея Skein у тому, щоб побудувати геш-функцію після використання блокового шифру. Використання стандартного блокового шифру дозволяє Skein гешувати дані конфігурації разом із вхідним текстом у кожному блоці, і зробити будь-яке середовище функції стиску унікальним. Ця властивість безпосередньо адресує множину атак геш-функцій, а також суттєво поліпшує гнучкість Skein.

Точніше кажучи, Skein створений із трьох нових компонентів:

Threefish. Threefish – це модифікований блоковий шифр у ядрі Skein, певний 256-, 512-, і 1024-бітним розміром блоку.

Unique Block Iteration (унікальний блок ітерацій, UBI). UBI – режим, що підключається, який використовує Threefish, щоб побудувати функцію стиску, яка перетворює довільний вхідний розмір у фіксований вихідний.

Optional argument system (додаткова система аргументу). Вона дозволяє Skein підтримувати ряд додаткових характеристик, не вимагаючи ніяких перевантажень при виконанні й додатків, які не використовують характеристики.

Основний Threefish алгоритм описує багаторічні знання проектування блокового шифру й аналізу. UBI доведено безпечний і може використовуватися з будь-якими налагодженими шифрами. Додаткова система аргументу дозволяє Skein пристосовуватися для різних цілей. Ці три компоненти незалежні й можуть використовуватися окремо, але їх комбінація забезпечує реальні переваги.

Гешування Skein

Skein побудований на числених викликах UBI. На рис. 6.22 наведено Skein як просту геш-функцію. Починаючи з ланцюгового значення 0, існує три виклики UBI: за кожним на конфігураційний блок, повідомлення (аж до 296 – 1 байт у довжину), і вихідне перетворення.

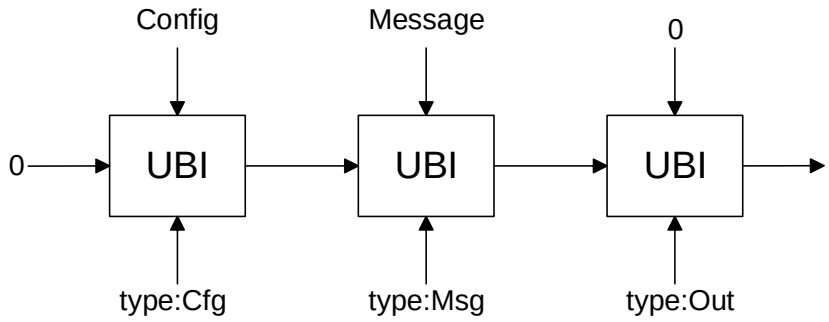


Рис. 6.22. **Режим нормального гешування Skein**

32-байтовий рядок конфігурації шифрує бажану вихідну довжину й деякі параметри, щоб підтримувати дерево гешування. Якщо Skein використовується як стандартна геш-функція – з фіксованим вихідним розміром і без дерева гешування або MAC ключа – результатом обчислення конфігураційного блоку UBI є константа для всіх повідомлень і може бути попередньо розрахована як IV.

Вихідні перетворення потрібні для досягнення гешування необхідної довільності. Це також дозволяє Skein породжувати виходи будь-якого розміру аж до 264 біт. Якщо одного вихідного блоку недостатньо, вихідні перетворення запускаються кілька раз, як наведено на рис. 6.23. З'єднані входи з усіма вихідними перетвореннями однакові, поле даних складається з 8-байтового лічильника. По суті, використовується Threefish у режиму лічильника. Створення більших виходів часто зручно, але звичайно безпека Skein обмежена внутрішньо встановленим розміром.

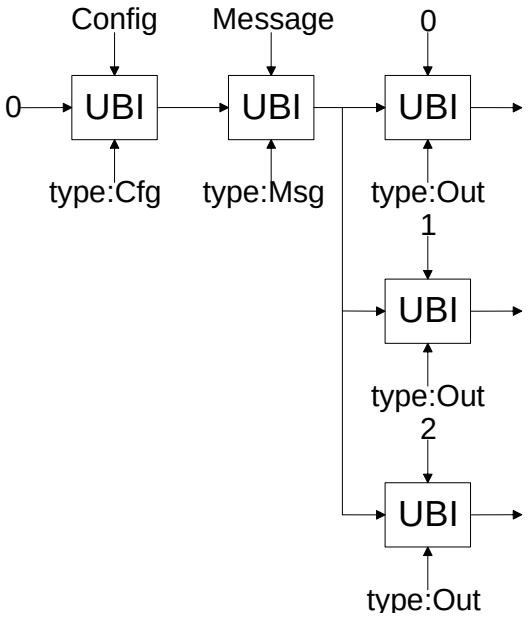


Рис. 6.23. **Skein з більшим вихідним розміром**

Додаткові аргументи.

Для того, щоб збільшити гнучкість Skein, кілька додаткових входів можуть бути дозволені, у якості необхідних. Усі ці опції управляються додатками реального часу.

Ключ (додатковий). Ключ, який перетворює Skein у MAC або KDF функцію. Ключ завжди обробляється в першу чергу, щоб підтримувати деякі з доказів безпеки.

Конфігурація (необхідна).

Персоналізація (додаткова). Рядок, який можуть використовувати додатки для створення різних функцій для різного використання.

Відкритий ключ (додатковий). Відкритий ключ використовується при гешуванні повідомлення для його підпису. Це зв'язує геш підпису й відкритий ключ. Таким способом ця характеристика перевіряє те, що одне й теж повідомлення генерує різний геш для різних відкритих ключів.

Ключ ідентифікатор висновку (додатковий). Використовується для добутку ключів. Для того, щоб одержати ключ, заготовлюють головний ключ як ключ для входу, і ідентифікатор запитуваного похідного ключа.

Поточний час (додатковий). Поточне значення часу використовується в потокових шифрах і рандомізованому гешуванні.

Повідомлення (додаткове). Нормальне вхідне повідомлення для геш-функції.

Вихідне перетворення.

Обчислення Skein складаються з обробки цих опцій у порядку, що використовується UBI. Кожний вхід має різний "тип" значення для модифікації, який гарантує, що входи не взаємозамінні.

Жодне з них не впливає на виконання й складність основної геш-функції жодною мірою; інші реалізації можуть вибирати яку опцію реалізувати, а яку проігнорувати.

Очевидно, Skein може бути розширений з іншими необов'язковими аргументами. Вони можуть бути додані в будь-який час, навіть, коли функція вже була стандартизована, тому що додавання нових необов'язкових аргументів є оборотно-сумісним.

Skein – MAC

Стандартний спосіб використовувати геш-функцію для автентифікації – використовуючи конструкцію HMAC [23]. Skein звичайно ж може бути використаний з HMAC, але це вимагає як мінімум двох геш-обчислень для кожної автентифікації, що є неефективним для коротких повідомлень. Перетворення Skein у MAC наведено на рис. 6.24.

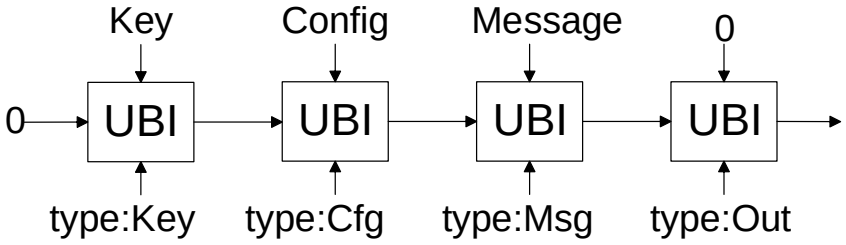


Рис. 6.24. **Skein – MAC**

Замість обробки блоку конфігурації, яка починається з нуля, починається обробка ключа з нуля, а потім вже блоку конфігурації. Або дивлячись із іншого боку, Skein гешування просто Skein-MAC з нульовим ключем. І подібно тому, як вихід Skein конфігураційного блоку попередньо обчислюється для наданого ключа. З того часу, як найбільш загальні шляхи були використані MAC для автентифікації численних повідомлень із єдиним ключем, це значно збільшує виконання для коротких повідомлень.

Дерево гешування.

Замість гешування даних як одного великого рядка, вони розділяються на частини. Кожна частина гешується й результуючий геш розглядається як нове повідомлення. Ця процедура може застосовуватися рекурсивно, поки результат не стане єдиним значенням геша.

Skein включає режим гешування деревом для підтримки додатків цього типу. Оскільки різні додатки мають різні вимоги, то ці три параметри для додатка можуть обиратися для конкретного використання додатка: вузловий розмір листа, treefan-out, і максимальна висота дерева.

Контрольні запитання

1. Поняття про гешувальні алгоритми, їх призначення, вимоги до них.
2. Алгоритми формування кодів-гешування за допомогою безключових геш-функцій.
3. Коди цілісності даних (MDC-коди). Способи використання у сучасних інформаційних системах.
4. Коди автентичності даних (MAC-коди). Способи їх побудови.
5. Основні вимоги щодо алгоритмів гешування на міжнародних криптографічних конкурсах NESSIE та SHA-3.
6. Основні переваги алгоритмів-переможців міжнародних криптографічних конкурсів NESSIE та SHA-3.
7. Основні поняття універсальних класів гешування даних. Побудова каскадних схем гешування на основі використання геш-функцій на універсальних класах.

Розділ 7. Цифрові підписи

7.1. Поняття про цифровий підпис (на прикладі RSA), вимоги до нього

Електронний цифровий підпис (ЕЦП) – реквізит електронного документа, призначений для захисту даного електронного документа від підробки, отриманий у результаті криптографічного перетворення інформації з використанням закритого ключа електронного цифрового підпису, що дозволяє ідентифікувати власника сертифіката ключа підпису, а також установити відсутність перекручування інформації в електронному документі.

У кінці звичайного листа або документа його автор або відповідальна особа звичайно ставлять свій підпис. Ця дія переслідує таке:

по-перше, отримувач має можливість впевнитися у істинності документа, порівнявши підпис зі зразком;

по-друге, особистий підпис є юридичною гарантією авторства документа.

Останній аспект особливо важливий під час укладання різного роду комерційних угод, створення доручень та інших документів, коли сторони принципово не можуть довіряти одна одній.

Підробити підпис людини на папері досить непросто, і зловмисник повинен мати дуже серйозні аргументи для виконання такої операції, бо встановити авторство підпису сучасними криміналістичними методами – справа техніки.

А як діяти у випадку, коли вигідніше використати електронний документообіг, і не витрачати величезні кошти та час на неодноразові відрядження.

Для цього було розроблено зовсім новий криптографічний механізм, який став можливим лише після винайдення асиметричної криптографії. Цей механізм В. Діффі та М. Хеллман назвали *цифровим підписом.* Його суть пояснимо на прикладі системи RSA. До повідомлення *M* застосуємо перетворення за допомогою приватного ключа *d* і назвемо його *цифровим підписом*, тобто:

$$S = M^d \bmod n.$$

Повідомлення *M* та його електронний підпис *S* відправляють за призначенням.

Отримувач, маючи *(M, S)* та публічний ключ відправника повідомлення *e,* може перевірити виконання співвідношення:

$$S^e \bmod n = M.$$

Якщо обчислене *M* співпадає з отриманим повідомленням, то підпис справжній.

Така схема призводить до такого:

отримувач, перевіривши справжність підпису, впевнений у тому, що це повідомлення *M* сформував саме власник приватного ключа *d* (оскільки більше ніхто не має до нього доступу);

відправник не зможе відмовитися від цього листа з тієї ж самої причини. Отже, створюється можливість утворення юридично чинних документів на основі такого механізму електронного підписування.

Ця схема має суттєвий недолік: цифровий підпис має ту ж довжину, як і документ, що ним підписаний. Отже, каналом зв'язку пересилається вдвічі більше інформації, ніж це потрібно для самого документа.

Таким чином, підписання довгих повідомлень потребує видозміни схеми. Для досягнення цієї мети було запропоновано підписувати не саме повідомлення, а його хеш-образ, що значно зменшить навантаження на канали зв'язку.

В Україні всі стосунки електронних документів та підписів визначаються Законами України "Про електронні документи та електронний документообіг" та "Про електронний цифровий підпис".

Цифровий підпис повинен мати такі властивості [6; 7; 11; 23; 34; 35; 46]:

1. Повинна бути можливість перевірити автора, дату й час створення підпису.
2. Повинна бути можливість автентифікувати повідомлення під час створення підпису.
3. Необхідно передбачити можливість перевірки підпису третьою стороною для вирішення суперечок.

На підставі цих властивостей можна сформулювати такі вимоги до цифровогопідпису:

1. Підпис повинен бути бітовим відбитком повідомлення, що підписується.
2. Підпис повинен використовувати деяку унікальну інформацію про відправника для запобігання підробки або відмови.

3. Створювати цифровий підпис повинно бути відносно легко.

4. Повинно бути розрахунково неможливо підробити цифровий підпис як створенням нового повідомлення для існуючого *цифрового* підпису, так і створенням підробленого *цифрового* підпису для деякого повідомлення.

5. Цифровий підпис повинен бути компактним аби не перевантажувати канали зв'язку.

Ці умови повністю задовольняє сильна геш-функція, зашифрована приватним ключем відправника.

Розглянемо основні алгоритми електронного цифрового підпису.

7.2. Основні алгоритми електронного цифрового підпису

Основними стандартами ЕЦП є:

міжнародний стандарт ISO/IEC 9796, який визначає ЕЦП з відновленням повідомлення (digital signature with message recovery);

міжнародний стандарт ISO/ IEC 14888, який визначає ЕЦП з додаванням (digital signature with appendix);

російський стандарт цифрового підпису на еліптичній кривій ГОСТ Р34.10-2001 [6];

американський національний стандарт цифрового підпису (FIPS 186);

американський фінансовий стандарт цифрового підпису з додаванням еліптичною кривої (ANSI X9.62);

стандарт на ЕЦП PKCS #1, який визначає ЕЦП на основі алгоритму RSA;

стандарт цифрового підпису з додаванням і відновленням повідомлення IEEE 1363;

стандарт цифрового підпису з додаванням еліптичної кривої IEEE P1363;

міжнародний стандарт ISO/IEC CD 15946-2 стандартизується ЕЦП еліптичною кривою з додаванням;

Державний стандарт України ДСТУ-4145 – 2002 [11].

На основі існуючих стандартів ЕЦП в [6; 7; 11; 23; 34; 35; 36; 46] запропонована класифікація ЕЦП.

За способом побудови схеми ЕЦП діляться на два класи:

схема ЕЦП із відновленням повідомлення;

схема ЕЦП із додаванням.

За кількістю учасників ЕЦП підрозділяється на:

одиночну схему ЕЦП;

групову схему ЕЦП.

У процесі виконання алгоритму формування цифрового підпису в одиночних схемах ЕЦП досить одного учасника, у групових схемах їх два або більше.

За способом перевірки ЕЦП поділяються на два класи:

інтерактивні схеми ЕЦП, що вимагають протокольної взаємодії;

не інтерактивні схеми ЕЦП, які не потребують протокольної взаємодії.

Існуючі алгоритми ЕЦП можна розділити також за типами використовуваних односпрямованих функцій із секретом:

схеми ЕЦП, засновані на стійкості факторизації великого числа;

схеми ЕЦП, засновані на стійкості дискретного логарифма;

схеми ЕЦП, засновані на стійкості дискретного логарифма в групі точок ЕК.

Кожна з цих схем може бути детермінована або рандомована. Застосування детермінованих схем характеризується тим, що цифровий підпис одним і тим же вхідним рядком даних приводить до формування однакових цифрових підписів. У рандомованій схемі при генерації підпису використовується деякий випадковий параметр, що приводить до формування різних підписів, навіть для однакових вхідних рядків. У рандомізованих схемах необхідно забезпечити непередбачуваність випадкових чисел [23; 36].

У свою чергу детерміновані схеми діляться на схеми ЕЦП одноразового застосування і схеми ЕЦП багаторазового застосування. Розглянемо основні стандарти ЕЦП, застосованих у комплексних системах захисту банківської інформації, проведемо зіставлення основних характеристик ЕЦП (довжину ключів, довжину цифрового підпису, складність (час) обчислення і складність (час) перевірки дійсності цифрового підпису) з метою виявлення їх переваг і недоліків, за умови, що рівень стійкості підпису стосовно будь-яких методів фальсифікації не нижче, ніж 10^{21} (або 30 років безперервної роботи мережі з 1 000 суперкомп'ютерів).

У якості "базової" довжини ключів і довжини самого цифрового підпису будемо розглядати довжину в 64 байти.

Цифрові підписи з відновленням повідомлення є об'єктом розгляду двох стандартів ISO/IEC 9796 (1991 року) і ISO/IEC 9796-2 (1997 року). У стадії розробки перебуває четверта частина стандарту ISO/IEC 9796-4. Механізми ЕЦП певні в ISO/IEC 9796 застосовуються тільки до коротких

повідомлень, тоді як механізми ISO/IEC 9796-2 застосовуються до повідомлень довільної довжини.

Стандарт ISO/IEC 14888 визначає механізми ЕЦП другого класу. Дані механізми застосовані до повідомлень довільної довжини. При обчисленні цифрових підписів з додаванням особливу роль відіграють однобічні геш-функції. Геш-функції також є об'єктом міжнародної стандартизації. Зокрема основним нормативним документом у даній області є міжнародний стандарт ISO/IEC 10118. Він складається з декількох частин і вводить модель геш-функції (ISO/IEC 10118-1(1994 року), розглядає два методи для побудови геш-функцій на основі блокових шифрів (ISO/IEC 10118-2 (1994 року), визначає три спеціалізованих (dedicated) геш-функції, тобто геш-функції, які розроблені спеціально для обчислення контрольних сум (ISO/IEC 10118-3 (1998 року).

Одна з них є стандартом NIST "Secure Hash Standard" (SHS). Дві інші RIPEMD-128 і RIPEMD-160 є розробкою Європейської організації зі стандартизації в рамках проекту "RIPE". Нарешті ISO/IEC 10118-4 (1998 року) визначає дві геш-функції на основі модульного зведення у квадрат для використання з механізмами ЕЦП, що базуються на модульній арифметиці.

Відомі методи забезпечення автентичності та цілісності даних засновані на внесенні надмірності (імітовставки, коду автентифікації, цифрового підпису) в оброблювану послідовність. В останні роки ця галузь знань бурхливо розвивається, запропоновано велику кількість різних криптографічних методів і алгоритмів. Найбільшого поширення набули протоколи, засновані на використанні односторонніх геш-функцій.

Для опису процесів обробки інформації з використанням механізмів ЕЦП скористаємося такою термінологією.

1. *Алгоритм генерації ЕЦП* – це метод формування ЕЦП.

2. *Алгоритм перевірки (верифікації) ЕЦП* – метод перевірки того, що підпис є автентичним, тобто дійсно створений конкретним об'єктом і не модифікований при передачі.

3. *Схема ЕЦП* (або *механізм ЕЦП*) – сукупність взаємозалежних алгоритмів генерації і верифікації цифрового підпису.

4. *Процес* (*процедура*) *накладання ЕЦП* – це сукупність математичного алгоритму генерації ЕЦП і методів представлення (форматування) даних, що підписуються.

5. *Процес* (*процедура*) *зняття ЕЦП* – сукупність алгоритму верифікації ЕЦП і методів відновлення даних.

Для побудови схеми ЕЦП необхідно визначити два алгоритми: алгоритм генерації ЕЦП і алгоритм верифікації ЕЦП. Алгоритм верифікації доступний для всіх потенційних одержувачів підписаних повідомлень, у той час, як алгоритм генерації ЕЦП відомий тільки особі, яка підписує, що для деякого повідомлення $m \in M$ визначає відповідний підпис $s \in S$. Верифікатор, одержавши пари (m, s) і деяку відкриту інформацію про особу, що підписує, застосовує відповідний алгоритм верифікації ЕЦП. Цей алгоритм видає двійковий результат: "так", якщо підпис правильний (автентична) і "ні" – у протилежному випадку.

Існуючі на сьогоднішній день схеми ЕЦП діляться на два класи (рис. 7.1): схеми ЕЦП із відновленням повідомлення; схеми ЕЦП із додаванням.

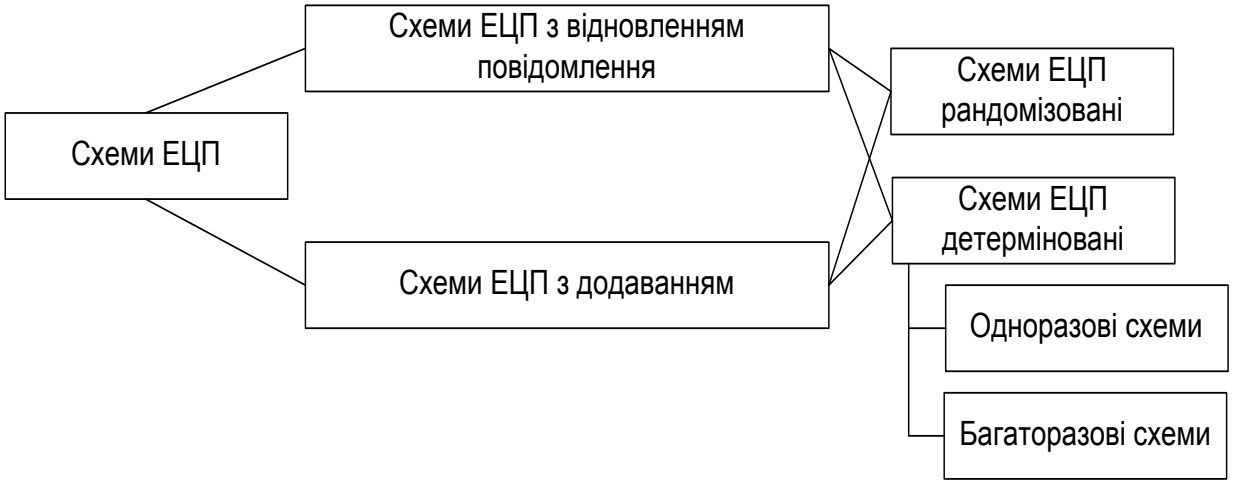


Рис. 7.1. **Класифікація схем ЕЦП**

У *схемах ЕЦП із відновленням повідомлення* всі або частина підписаного повідомлення може бути відновлена безпосередньо з цифрового підпису. Таким чином, на вхід алгоритму верифікації надходить лише цифровий підпис *s*.

У *схемах ЕЦП із додаванням* цифровий підпис приєднується до повідомлення й у такому вигляді відправляється адресату. Для верифікації такого ЕЦП необхідно мати і підпис *s,* і відповідне повідомлення *m*.

Кожна з цих схем може бути *детермінованою* або *рандомізованою*. Застосування детермінованих схем характеризується тим, що цифровий підпис одного і того ж вхідного рядка даних призводить до формування однакових цифрових підписів. У рандомізованій схемі при генерації підпису використовується деякий випадковий параметр (число), що

призводить до формування різних підписів для однакових вхідних рядків (при використанні тих самих ключів). У рандомізованих схемах необхідно забезпечити непередбачуваність випадкових чисел.

У свою чергу детерміновані схеми діляться на схеми ЕЦП одноразового застосування (one-time) і схеми ЕЦП багаторазового застосування (multiple-use).

Нарешті, у ISO/IEC 14888-3 описується два типи схем. По-перше, це схеми на основі використання проблеми дискретного логарифма, а саме: схема DSA, схема Pointcheval-Vaudenay і схема ECDSA (схема DSA на еліптичних кривих). По-друге, це схеми, стійкість яких заснована на проблемі факторизації. У стандарті подана: схема підпису за ISO/IEC 9796-1 з використанням гешування і схема ESIGN.

Схеми підпису на основі використання ідентифікаційної інформації. Схема підпису Гіллоу – Куіскуотера (GQ-схема)

У схемах формування підпису на основі ідентифікаційної інформації відкритий ключ перевірки підпису формується з використанням ідентифікаційної інформації сторони, яка підписує. Таким чином, відпадає необхідність використання сертифікатів відкритих ключів. Особисті (секретні) ключі генерації підпису повинна генерувати довірча третя сторона. У таких схемах ДТС матиме доступ до всіх особистих ключів. У зв’язку з цим схема формування підпису на основі використання ідентифікаційної інформації може використовуватися не для всіх додатків. Найчастіше вона реалізується в закритих доменах безпеки, наприклад, усередині великої компанії або корпоративних закритих мереж, де існує "природна" довірча сторона.

ISO/IEC 14888-2 визначає три схеми підпису, що відносяться до класу рандомізованих схем і є різними варіантами схеми підпису Гіллоу – Куіскуотера (Guillou – Quisquater). В основу *GQ*-схеми підпису покладено використання ідентифікаційного протоколу GQ. Схема підпису, яка визначена в ISO/IEC 14888-2, допускає обов’язкове залучення ДТС.

ДТС генерує особисті ключі підпису, у той час, як ключ верифікації може генеруватися верифікатором.

Генерація ключів для GQ-схеми підпису.

У результаті генерації ключів утворюється відкритий ключ (n, e ,I_A) і відповідний особистий ключ *a*.

Довірча третя сторона виконує такі дії:

1. Обрати випадкові різні прості числа p і q, які зберігаються в секреті, і формує модуль n = p × q.

2. Обрати ціле $e \in \{1,2,\ldots,n-1\}$ таке, що НСД $(e,(p-1)(q-1))=1$.

3. Будь-яка сторона, що бажає використовувати GQ-схему підпису, надає ДТС унікальну ідентифікаційну інформацію у вигляді двійкового рядка I.

Ціле число I_A, $1 < I_A < n$ є ідентифікатором сторони A (тобто містить інформацію про ім'я, адресу, номер паспорта, PIN і т. д.). ДТС на основі інформації I_A формує величину:

$$Y = f(I_A),$$

де f – функція уведення надмірності (ISO/IEC 9796-1) така, що НСД (Y, n)= 1.

4. ДТС визначає ціле число $a \in Z_n$ таке, що $Ya^e \equiv 1 \pmod n$ відповідно до такого алгоритму.

4.1. Обчислення $Y^{-1} \bmod n$.

4.2. Обчислення $d_1 = e^{-1} \bmod (p-1)$ і $d_2 = e^{-1} \bmod (q-1)$.

4.3. Обчислення $a_1 = \left(Y^{-1}\right)^{d1} \bmod p$ і $a_2 = \left(Y^{-1}\right)^{d2} \bmod q$.

4.4. Пошук значення a, що задовольняє рівнянням:

$a = a_1 \pmod p$;

$a = a_2 \pmod q$.

За відкритий ключ сторони A приймається вектор (n, e, I_A). За особистий ключ сторони A приймається число a.

Процедури накладання і зняття *GQ*-підпису.

Процедура накладання *GQ*-підпису містить виконання 3 кроків. Сторона A підписує двійкове повідомлення m довільної довжини. Для цього сторона A повинна:

1. Вибрати випадкове число k, $1 \le k \le n-1$, і обчислити $r = k^e \bmod n$.

2. Обчислити геш-код $v = h(m||r)$.

Значення геш-коду повинно задовольняти нерівності $0 < v < e-1$. Якщо v не задовольняє нерівності, то значення v приводять за модулем e.

3. Обчислити $s = r \cdot a^v \bmod n$.

Процедура зняття *GQ*-підпису. Для того, щоб верифікувати цифровий підпис (s, v) за повідомленням m, сторона B повинна одержати

повідомлення m, підпис (s, v) і автентичний відкритий ключ сторони A (n, e, I_A). Далі сторона B повинна обчислити:

$$u = s^e I_A^v \bmod n;$$
$$v' = h(m||n).$$

Якщо v = v', підпис вважається дійсним. Справедливість цього твердження випливає з того, що:

$$u \equiv s^e I_A^v \equiv (ka^v)^e I_A^v \equiv k^e (a^e I_A)^v \equiv k^e \equiv r (\text{mod } n).$$

Отже, u = r × i, звідси, v = v'.

Стійкість схеми GQ-підпису опирається на складність задачі факторизації складового модуля. Сучасні методи розкладання числа на складові множники і будуть визначати вимоги до параметрів схеми. Останні досягнення в області факторизації говорять про те, що модуль n повинен бути, принаймні, 768-бітним. З метою забезпечення стійкості схеми ЕЦП до атак, що опираються на парадокс дня народження, довжина експоненти e повинна бути не менше 128 бітів. Розміри геш-кодів, що можуть застосовуватися в *GQ*-схемі підпису, рівні 128 або 160 бітам. Таким чином, при 768-бітному модулі n і 128-бітній експоненті e розмір відкритого ключа складе 896+u бітів, де u – кількість бітів, необхідна для представлення ідентифікатора I_A. Довжина особистого ключа a складе 768 бітів.

З погляду продуктивності *GQ*-схема вимагає виконання двох операцій зведення в степінь за модулем і однієї операції модульного множення. При використанні 768-бітного модуля n 128-бітної величини e і 128-бітного геш-коду v генерація підпису потребуватиме виконання приблизно такої ж кількості операцій. При схожих початкових умовах схема підпису RSA вимагає виконання 1152 модульних множень, а схема Фейга – Фіата – Шаміра (Feige – Fiat – Shamir) або схема FFS – 64 модульні множення. Але порівняно з останньою *GQ*-схема вимагає значно меншого місця для збереження ключів. У схемі FFS довжина особистого ключа складе 98304 біти, а довжина відкритого ключа – 99072 біти.

Схема цифрового підпису Клауса Шнорра

Проблема схеми цифрового підпису Ель-Гамаля полягає в тому, що *p* повинно бути дуже великим, щоб зробити важкою проблему дискретного логарифма z_p^*. Рекомендується довжина *p* принаймні 1024 біт.

Можна зробити підпис розміром 2048 біт. Щоб зменшити розмір підпису, Шнорр запропонував нову схему, засновану на схемі Ель-Гамаля, але із зменшеним розміром підпису.

Стійкість схеми Шнорра ґрунтується на важкій задачі обчислення дискретних логарифмів. Для генерації пари ключів спочатку вибираються два прості числа p і q так, щоб q було співмножником p – 1. Потім вибирається, a не рівне 1, таке, що $aq = 1(\mathrm{mod}\ p)$. Всі ці числа можуть бути вільно опубліковані та використовуватися групою користувачів. Для генерації конкретної пари ключів вибирається випадкове число, менше q. Воно служить закритим ключем s. Потім обчислюється відкритий ключ $v = a - s(\mathrm{mod}\, p)$. Для підпису повідомлень використовується геш-функція H(M).

Попередні обчислення.

Користувач вибирає випадкове число r, менше q, і обчислює $x = ar(\mathrm{mod}\, p)$. Підпис повідомлення M. Для того, щоб підписати повідомлення M користувачеві необхідно виконати такі дії:

1) об'єднати M і x і гешувати результат: $e = H(M, x)$;

2) обчислити $y = (r + se)(\mathrm{mod}\, q)$. Підписом є значення e та y, їх потрібно вислати одержувачу B.

Перевірка підпису для повідомлення M.

Одержувач B обчислює $x' = a * y * v * e(\mathrm{mod}\ p)$. Потім він перевіряє, що геш-значення для об'єднання M і x' одне e, $e = H(M, x')$. Якщо це так, то він вважає підпис правильним.

Цифровий підпис Шнорра

ЕЦП Шнорра заснована на складності завдання дискретного логарифмування. Тому всі параметри схеми Шнорра повинні задовольняти умовам існування дискретного логарифма.

Параметри схеми:

p – просте число, для реальних задач 160 £ p £ 256;

q – просте число, $q \mid p - 1$, тобто q ділить p-1;

g – число g, $g \mid Z_p$, де Z_p – клас відрахувань за модулем p;

H – геш-функція;

x – випадкове число з інтервалу $[1, q - 1]$;

x – секретний ключ схеми;

y – відкритий ключ схеми;

Y = g-x.

Припустимо, що існують два учасники А і В. Учасник повинен підписати повідомлення М для учасника В.

Алгоритм схеми підпису Шнорра.

1. Учасник А вибирає випадкове число k і обчислюється $r = g * k \bmod p$.

2. Учасник А формує підпис, для цього обчислюються e та s за формулою:

$$e = h(r, A), s = k + xe.$$

3. Підпис (e,s) і підписується текст m пересилається учаснику В.

4. Учасник В робить перевірку підпису, обчислюючи значення r' і e'

$$r' = g * s * y * e \bmod p;$$
$$e' = h(r', m).$$

1. Якщо e = e', то підпис приймається, в іншому випадку відкидається. На рис. 7.2. наведено схему цифрового підпису Клауса – Шнорра.

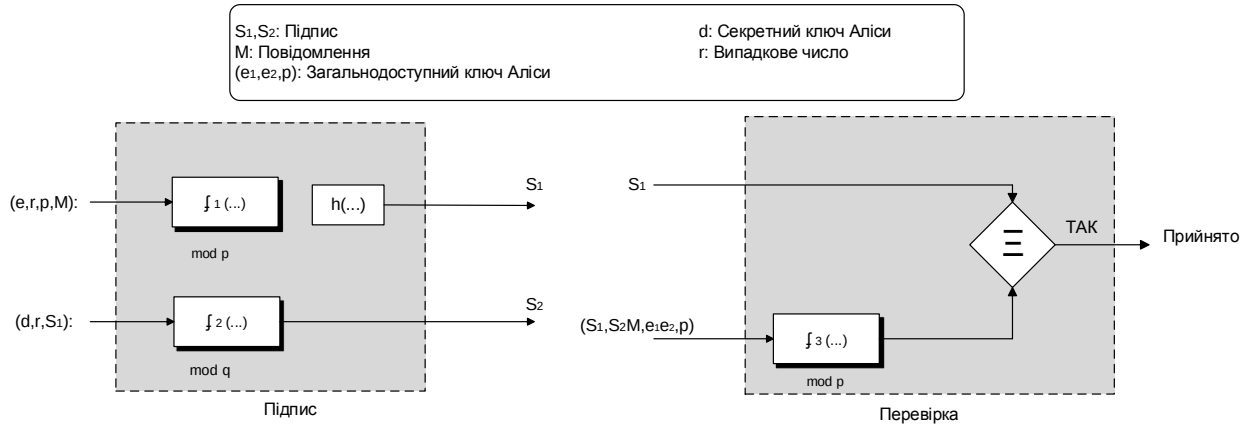


Рис. 7.2. **Схема цифрового підпису Клауса – Шнорра**

Схема цифрового підпису ESIGN

Схема ESIGN (Efficient digital SIGNature) запропонована корпорацією Nippon (Японія) і є схемою, чия стійкість заснована на складності розв'язання задачі факторизації великого цілого числа.

Генерація ключів. Кожен об'єкт створює відкритий ключ і відповідний закритий ключ. Для цього об'єкт А повинен виконати такі операції.

1. Обрати два великих простих числа p і q таких, що $p \geq q$ і p, q приблизно рівних за довжиною.

2. Обчислити модуль $n = p^2q$.

3. Обрати позитивне ціле число $k \geq 4$.

4. Відкритим ключем сторони А є пара чисел (n, k), закритим ключем – (p, q).

Процедура накладання підпису.

Алгоритм генерації підпису обчислює ціле число s таке, що $s^k \bmod n$ лежить у визначеному інтервалі, обумовленому повідомленням m. Для того, щоб підписати повідомлення m, що є двійковим рядком довільної довжини, об'єкт A повинен виконати такі дії.

1. Обчислити геш-код v = h(m).
2. Обрати випадкове секретне ціле число x, $0 \le x \le pq$.
3. Обчислити:

$$w = \lceil ((v - x^k) \bmod n)/(p \cdot q) \rceil;$$
$$y = w \cdot (kx^{k-1})^{-1} \bmod p;$$
$$s = x + ypq \bmod n.$$

4. Цифровим підписом повідомлення *m* є число *s*.

Процедура верифікації підпису.

У ході верифікації демонструється, що величина $s^k \bmod n$ належить визначеному інтервалу. Для цього об'єкт B повинен виконати такі дії:

Одержати автентичний відкритий ключ (n, k) сторони A.

Обчислити:

$$u = s^k \bmod n;$$
$$z = h(m).$$

Прийняти підпис, якщо: $z \le u \le z + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil}$.

У протилежному випадку підпис вважається недійсним.

Коректність підпису випливає з доказу.

Зазначимо, що:

$$s^k \equiv (x+ypq)^k \equiv \sum_{i=0}^{k} \binom{k}{i} x^{k-i} (ypq)^i \equiv x^k + kypqx^{k-1} (\bmod n).$$

Але $kx^{k-1}y \equiv w \bmod p$ i, таким чином $kx^{k-1}y \equiv w + lp$ для деяких $l \in Z$.

Отже,

$$s^k \equiv x^k + pq(w + lp) \equiv x^k + pqw \equiv x^k + pq \left\lceil \frac{(h(m) - x^k) \bmod n}{pq} \right\rceil \equiv$$

$$\equiv x^k + pq \left(\frac{h(m) - x^k + jn + \varepsilon}{pq} \right) (\bmod n), \text{ де } \varepsilon = (x^k - h(m)) \bmod pq.$$

Звідси $s^k \equiv x^k + h(m) - x^k + \varepsilon \equiv h(m) + \varepsilon \ (\bmod n)$.

Оскільки $0 \le \varepsilon < pq$, то:

$h(m) \le s^k \bmod n \le h(m) + pq \le h(m) + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil}$, що і потрібно було довести.

У схемі підпису ESIGN використовується модуль $n = p^2q$, що відрізняється від RSA-модулів. Розроблювачі ESIGN стверджують, що за стійкістю ця схема не поступається схемам RSA і DSA. Однак не відомо, наскільки простіше або складніше розкладання модуля $n = p^2q$ на складові співмножники порівняно зі звичайним модулем $n = pq$.

Специфічний доказ коректності підпису ESIGN дозволяє реалізувати наступні атаки.

Для цього правильного підпису s за повідомленням m зловмисник може підробити підпис для іншого повідомлення м', якщо геш-код h(м') приймає значення таке, що:

$$h(m') \le u \le h(m') + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil},$$

де $u = s^k \bmod n$.

Якщо буде знайдене повідомлення м', що володіє такими властивостями, то підпис s буде дійсним і для нього. Така ситуація можлива, якщо h(m) і h(м') збігаються в (lg n)/3 старших значущих бітах. Припускаючи, що h має властивості випадкової функції, противнику для пошуку такого повідомлення необхідно випробувати $2^{(\lg n)/3}$ варіантів.

Інший можливий варіант підробки підпису полягає в перебуванні пари повідомлень m і м' таких, що h(m) і h(м') збігатимуться в старших (lg n)/3 бітах. Відповідно до парадокса дня народження цього необхідно випробувати $O(2^{(\lg n)/6})$ варіантів. Якщо противник здатний сформувати коректний підпис для повідомлення m, то цей же підпис буде дійсним і для повідомлення м'.

Розглянуті приклади атак і визначають вимоги до величини модуля.

Схема підпису ESIGN з погляду продуктивності перевершує схему підпису RSA. Крім того, схема ESIGN дозволяє виконувати передобчислення, що сприяють збільшенню обчислювальної ефективності алгоритму. Після вибору випадкового числа *x* сторона A може здійснити такі передобчислення:

$$u_1 = x^k \bmod n;$$
$$u_2 = 1/(kx^{k-1}) \bmod p.$$

Потім визначити:

$$w = \lceil w\text{-}u_1/(pq) \rceil;$$
$$s = x + (wu_2 \bmod p) \cdot p \cdot q.$$

Використання передобчислювань дозволяє збільшити швидкість формування підпису до 10 разів. На ефективність обчислень підпису впливає значення параметра k. Розроблювачі рекомендують вибирати його з множини k = {4, 8, 16, 32, 64, 128, 256, 512, 1024}.

Так, для k = 4 і 768-бітний модулі n генерації підпису ESIGN перевершує за швидкістю генерацію підпису RSA від 10 до 100 разів. Процедура верифікації також досить ефективна і порівняна зі схемою RSA з малими відкритими ключами. Цей алгоритм може бути реалізований і на еліптичних кривих.

Цифрові підписи з відновленням повідомлення

Схеми ЕЦП із відновленням повідомлення є об'єктом стандартизації міжнародного стандарту ISO/IEC 9796. Цей міжнародний стандарт складається з трьох частин:

ISO/IEC 9796-1 стандартизує схему ЕЦП на основі RSA-перетворення;

ISO/IEC 9796-2 стандартизує схему ЕЦП аналогічну схемі, визначеній в першій частині, але використовуючи для введення надмірності геш-функцію. У цьому варіанті схеми ЕЦП забезпечується часткове відновлення повідомлення і схема може бути використана для підпису повідомлень довільної довжини;

ISO/IEC 9796-3 описує схему, в якій як алгоритм генерації підпису використовується дискретний логарифм (схема ElGamal).

Схема підпису за ISO/IEC 9796-1

Загальна ідея застосування цифрового підпису з відновленням повідомлення полягає в тому, що вихідне коротке повідомлення подовжується, потім у нього вводиться надмірність, після чого воно підписується. Схема ЕЦП, визначена в ISO/IEC 9796-1, забезпечує необхідні показники стійкості тільки при дотриманні всіх обмежень, визначених у стандарті. Схема ЕЦП за ISO/IEC 9796-1 заснована на використанні криптографії з відкритими ключами з використанням схеми RSA або Rabin. Алгоритм підпису повинен відображати *k*-бітний вхідний рядок у *k*-бітний рядок підпису. Схема не вимагає застосування геш-функції і може використовуватися для підпису коротких повідомлень, що потім відновлюються з підпису.

При описі схеми ЕЦП використовуються такі позначення:

s – двійковий рядок підпису;

$d \le 8\lfloor (s+3)/16 \rfloor$ – двійковий рядок повідомлення m, яке підписується, обирається так, що:

$z + \lceil d/8 \rceil$– кількість байтів у доповненому повідомленні;

$r = 8z - d + 1$– число, більше на одиницю кількості доповнених бітів;

$t = \lceil (s-1)/16 \rceil$– найменше ціле число таке, що рядок з 2t байтів включатиме принаймні s – 1 біт.

Приклад. Нехай необхідно підписати повідомлення m довжиною 150 бітів. При цьому довжина підпису повинна дорівнювати 1024 бітам. Тоді наведені параметри приймуть такі значення: s = 1024 біти, d = 150 бітів, z = 19 байтів, r = 3 біти, t = 64 байти.

Процес накладання ЕЦП за ISO/IEC 9796-1.

Процес накладання ЕЦП містить у собі виконання п'яти кроків (рис. 7.3).

1. *Доповнення.* Нехай m – повідомлення, які необхідно підписати. Доповнення полягає у формуванні доповненого повідомлення MP (massage padding) шляхом конкатенації ліворуч вихідного повідомлення m з r нульовими бітами:

$$MP = 0^{r-1} \| m, z,$$

де $1 \le r \le 8$ обирається таким, що кількість бітів у MP буде кратним восьми.

Кількість байтів у MP позначатимемо символом:

$$MP = m_z \| m_{z-1} \| \ldots \| m_2 \| m_1,$$

де m_i – байт.

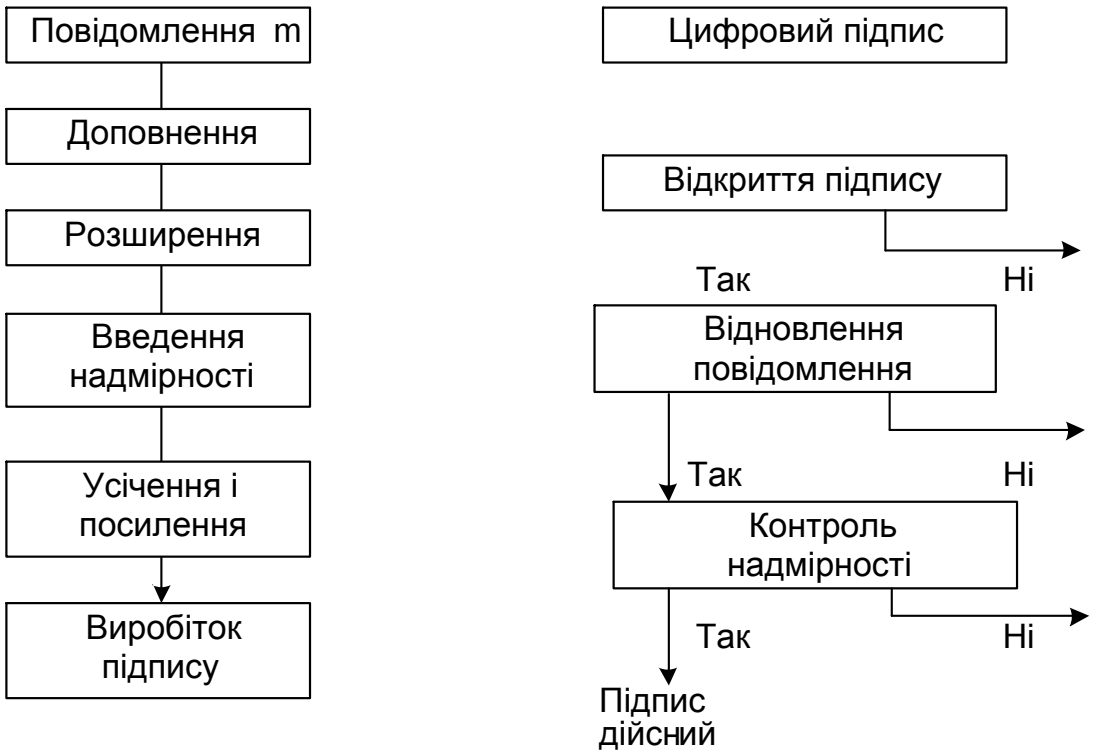
2. *Розширення повідомлення.* З доповненого повідомлення формується розширене повідомлення ME (message extension) шляхом ітеративної конкатенації ліворуч доповненого повідомлення із самим собою.

Повторення конкатенації здійснюється доти, доки не буде сформований рядок:

$$ME = ME_t \| ME_{t-1} \| \ldots \| ME_2 \| ME_1,$$

який містить рівно t байтів. Якщо число t не кратне числу z, то останні приєднані байти будуть частиною множини байтів з MP, що є послідовними праворуч доповненого повідомлення, тобто:

$$ME_i = m_{i \bmod z+1}, \forall i = \overline{0, t-1}$$



Процедура накладення підпису Процедура верифікації підпису

Рис. 7.3. **Процедури накладання і верифікації підпису за ISO/IES 9796-1**

Уведення надмірності. Надмірність вводиться в ME з метою одержання надлишкового повідомлення MR (message redundancy) вигляду $ME = ME2_t \,||\, ME_{2t-1} \,||\, ... \,||\, ME_2 \,||\, ME_1$, Уведення надмірності здійснюється в такий спосіб. Надлишкове повідомлення формується шляхом перемноження t байтів розширеного повідомлення з t надлишковими байтами, з наступним узгодженням M_{2z}-го байта отриманого рядка. Усі байти надлишкового повідомлення формуються згідно з такими виразами:

$$MR_{2i-1} = ME_i, MR_{2i} = F(M_i) \forall i = \overline{1,t},$$

де F(u) – функція перетворення байта u.

Функція F(u) визначена в такий спосіб. Якщо байт $u = u_1 \,||\, u_2$, де u_1 і u_2 – напівбайти, то:

$$F(u) = \pi(u_2) \,||\, \pi(u_1),$$

де π – перестановка вигляду:

$$\pi = \begin{pmatrix} 0 & 12 & 34 & 56 & 78 & 9A & BC & DE & F \\ E & 35 & 89 & 42 & F0 & DB & 67 & AC & 1 \end{pmatrix}.$$

Нарешті останній байт MR формується шляхом заміни байта M_{2z} на байт вигляду $r \oplus MR_{2z}$. Таке перетворення дозволяє верифікатору підпису відновити довжину повідомлення. Оскільки довжина повідомлення дорівнює $d = 8z - r + 1$, то для визначення значення d досить визначити значення z і r. Ці значення можуть бути відновлені з MR.

4. Усічення і посилення. На основі надлишкового повідомлення формується s-бітне проміжне ціле число IR. Формування IR здійснюється в такий спосіб:

а) до s – 1 молодшим значущим бітам надлишкового повідомлення ліворуч додається одиничний біт. Інші старші біти (якщо вони є) просто відкидаються;

б) здійснюється модифікація молодшого значущого байта $u_2 \| u_1$ шляхом заміни його на байт вигляду $u_1 \| 0110$.

5. Генерація підпису. На цьому кроці формується цифровий підпис шляхом застосування алгоритму генерації ЕЦП до рядка *IR:*

$$s = SIG_k(IR)$$

Схема ЕЦП ISO/IEC 9796-1 орієнтована на генерацію ЕЦП на основі алгоритмів RSA або Rabin. Розглянемо застосування даних алгоритмів.

Алгоритм генерації ЕЦП RSA за ISO/IEC 9796-1

Спочатку сторона, яка підписується, формує необхідні ключі відповідно до наступного алгоритму.

Сформувати два великих відмінних один від одного простих числа p і q, приблизно однієї довжини. При цьому $p \neq q \bmod 8$.

Обчислити $n = pq$ і $\varphi(n) = (p-1)(q-1)$.

Вибрати випадкове ціле число v, при цьому $1 < v < \varphi(n)$:

$$(e, p-1) = 1 \text{ і } (e, q-1) = 1 \text{ при e непарному;}$$

$$\left(e, \frac{p-1}{2}\right) = 1 \text{ і } \left(e, \frac{q-1}{2}\right) = 1 \text{ при e парному.}$$

Використовуючи розширений алгоритм Евкліда, обчислити число v з рівнянь:

$$ve \equiv 1 \bmod \varphi(n) \text{ при e парному;}$$

$$ve \equiv 1 \bmod \frac{\varphi(n)}{2} \text{ при e непарному.}$$

Відкритим ключем сторони, що підписує, є пара чисел (e,n). Число *e* ще називають перевірочним компонентом або ключем верифікації. Особистим ключем є число d.

Обчислення підпису здійснюється в два етапи. Спочатку з рядка IR формується репрезентативний елемент RR. Формування RR здійснюється в такий спосіб:

якщо e непарне, то RR = IR;

якщо e парне і символ Якобі $J\left(\frac{IR}{n}\right) = 1$, то RR=IR;

якщо e парне і символ Якобі $J\left(\frac{IR}{n}\right) = -1$,то RR=IR/2;

Зазначимо, що: $J\left(\frac{a}{n}\right) = (a^{(p-1)/2} \bmod p)(a^{(q-1)/2} \bmod q)$, де n = pq.

Цифровий підпис s обчислюється як:

$$s = \min(RR^d \bmod n, n - (RR^d \bmod n)).$$

Алгоритм генерації ЕЦП Rabin

Схема Рабіна аналогічна схемі RSA, але як відкритий ключ e використовуються тільки парні числа і найчастіше e = 2.

Процес зняття ЕЦП за ISO/IEC 9796-1.

Зняття цифрового підпису здійснюється в три етапи:

1) відкриття підпису;

2) відновлення повідомлення;

3) контроль надмірності.

На кожному з цих трьох етапів підпис може бути відкинутий як недійсний, якщо він не пройде відповідної перевірки. У таких випадках обробка підпису припиняється з відповідним повідомленням при цьому зацікавлених сторін.

Розглянемо більш докладно кожний з цих етапів.

1. Відкриття підпису. На даному етапі здійснюється перетворення цифрового підпису s у рядок IR' – відновлене проміжне ціле:

$$IR' = VER_{k_v}(s).$$

Для схеми RSA k_v = e; для схеми Rabin k_v = 2. Підпис відкидається, якщо рядок IR' не є *s*-бітним рядком зі старшим значущим бітом, рівним

"1" і молодшими значущими бітами, рівними "0110". У випадку правильного ЕЦП справедлива рівність IR = IR'.

2. *Відновлення повідомлення.* У ході відновлення повідомлення з рядка IR' формується рядок надлишкового повідомлення MR', що складається з *2t* байтів. Відновлення здійснюється в такий спосіб:

а) нехай X буде рядком з s – 1 молодших значущих бітів рядка IR';

б) молодший значущий байт рядка X, що представляється як $u_1 \| u_3 \| u_1 \| 0110$, де u_i – напівбайт, замінюється відповідно до виразу $\pi^{-1}(u_4) \| u_1$;

в) відновлене повідомлення MR' формується шляхом доповнення ліворуч рядка X нульовими бітами (від 0 до 15 бітів), у результаті чого загальна довжина MR' складе 2t байтів.

Якщо цифровий підпис правильний, то значення MR' буде збігатися із значенням рядка MR за винятком 16t – s +1 старших значущих бітів.

Далі здійснюється обчислення значення r і z;

а) здійснюється порівняння парних і непарних байтів відновленого надлишкового повідомлення MR'. Для усіх $i = \overline{1,t}$ перевіряють виконання умови:

$$MR'_{2i} \oplus F(MR'_{2i-1}) = 0.$$

Якщо для всіх $i = \overline{1,t}$ ця умова справедлива, то цифровий підпис вважається недійсним;

б) нехай *z* – найменше позитивне ціле число *i*, при якому значення:

$$f = MR'_{2i} \oplus F(MR'_{2i-1}) \neq 0;$$

в) нехай r буде найменшим значущим напівбайтом числа f. Підпис вважається недійсним, якщо шістнадцятиричне значення r лежить поза інтервалом $1 \le r \le 8$.

Відновлене доповнене повідомлення MP' довжиною z байтів формується з MR' відповідно до виразів:

а) $MR'_{2i} = MR'_{2i-1}$ для усіх $i = \overline{1,z}$;

б) якщо r – 1 старших значущих бітів рядка MP' не дорівнюють нулю, то цифровий підпис вважається недійсним.

Оригінальне повідомлення M' є 8z – r + 1 молодших значущих бітів рядка MP'.

3. *Контроль надмірності.* Цифровий підпис верифікується в такий спосіб:

а) з отриманого на попередньому етапі повідомлення M’ формується рядок MR’’ шляхом застосування операцій доповнення, розширення повідомлення і введення надмірності (аналогічно до процесу накладання цифрового підпису);

б) цифровий підпис вважається дійсним тільки тоді, коли s – 1 молодших значущих бітів рядка MR’’ збігаються з s – 1 молодшими значущими бітами рядка MR’.

Розглянута схема цифрового підпису може бути модифікована в схему ЕЦП з додаванням і, отже, буде застосовна для підпису повідомлення довільної довжини. У цьому випадку можливе застосування однобічної вільної від колізій геш-функції.

Процедура накладання цифрового підпису полягає в тому, що повідомлення довільної довжини m спочатку гешується, а потім геш-код h(m) є входом у процедуру накладання підпису за ISO/IEC 9796-1.

Цифровий підпис з частковим відновленням повідомлення

Схема ЕЦП із частковим відновленням повідомлення визначена в ISO/IEC 9796-2 і має такі властивості.

По-перше, схема забезпечує часткове відновлення повідомлення і застосована для повідомлень довільної довжини. Тобто якщо повідомлення коротке, то воно може бути цілком відновлене з підпису. Якщо повідомлення занадто довге, тобто коли довжина повідомлення d не погодиться з довжиною підпису s, то частина повідомлення може бути відновлена з підпису, а інша частина повинна бути передана одержувачу будь-яким іншим способом.

По-друге, в раніше розглянутій схемі підпису для введення надмірності необхідно відводити не менше половини доступного простору блоку підпису. У ISO/IEC 9796-2 метод уведення надмірності допускає використання геш-функції, що дозволяє збільшити обсяг даних у блоці підпису. Так, якщо за алгоритм генерації підпису взяти 768-бітне перетворення RSA то, відповідно до процедури формування підпису за ISO/IEC 9796-1, безпосередньо для переданих даних буде доступно не більше 384 бітів. При застосуванні схеми, визначеної в другій частині стандарту, дозволяється використовувати для даних до 600 бітів.

Основна ідея схеми ЕЦП із частковим відновленням повідомлення полягає в тому, що:

1. Повідомлення m гешується з використанням геш-функції, у результаті чого формується геш-код h(m).

2. Якщо повідомлення занадто довге, щоб бути цілком включеним у підпис, то виділяється деяка його частина, що надалі буде відновлюватися з підпису.

3. До відновлюваної частини повідомлення додається прапорний біт (more date bit).

4. Відновлювана частина повідомлення, прапорний біт і геш-код разом з іншими бітами формату (наприклад, ідентифікатор геш-функції) конкатуються і є вхідними параметрами в алгоритм генерації підпису. Як алгоритм генерації підпису рекомендується використовувати схему RSA.

Схема цифрового підпису Нуберга-Рюппела

Четверта частина стандарту ISO/IEC 9796-4 визначає схему Нуберга – Рюппела (Nyberg-Rueppel, NR-схема), у якій за алгоритм генерації підпису використовується варіант схеми підпису Ель – Гамаля. В даний момент запропоновані дві версії NR-схеми підпису: на основі обчислення дискретного логарифма в Z_p^* і на основі обчислення логарифма на еліптичних кривих.

NR-схема є рандомізованою схемою з відновленням повідомлення. Для цієї схеми простір, що підписується, $M_s = Z_p^*$, де p – просте число. Простір підписів $S = Z_p \times Z_q$, де q – просте число і q ділить (p – 1). Функція введення надмірності відображає множину повідомлень M у множину M_s. Алгоритм генерації ключів аналогічний алгоритму генерації ключів для схеми DSA, за винятком того, що на довжину чисел p і q не накладається ніяких обмежень.

Генерація ключів для NR-схеми підпису.

Для формування відкритого і відповідного закритого ключа сторона *A* повинна виконати такі дії:

1. Вибрати прості числа p і q такі, що q ділить (p – 1).

2. Визначити генератор Θ циклічної групи порядку q в Z_p^*. Для цього:

2.1. Вибрати елемент $g \in Z_p^*$ і обчислити:

$$\Theta = g^{(p-1)/q} \bmod p$$

2.2. Якщо $\Theta = 1$, повернутися в 2.1.

3. Вибрати випадкове ціле число a таке, що $1 \le a \le q - 1$.

4. Обчислити число $y = \Theta^a \bmod p$.

5. Відкритим ключем сторони A є вектор (p, q,Θ, y), закритим ключем є число a.

Процедура накладання цифрового підпису.

Нехай сторона *A* бажає підписати повідомлення $m \in M$. Тоді необхідно:

1. Обчислити $\bar{m} = R(m)$ (уведення надмірності).

2. Вибрати випадкове секретне ціле число k, $1 \le k \le q - 1$ і обчислити параметр:

$$r = \Theta^{-k} \bmod p.$$

3. Обчислити:

$$e = \tilde{m} r \bmod p;$$
$$s = ae + k \bmod q.$$

4. Цифровим підписом повідомлення m є пари чисел (e, s).

Процедура зняття цифрового підпису.

Для того, щоб верифікувати ЕЦП і відновити повідомлення, об'єкт B повинен:

1. Одержати автентичну копію відкритого ключа (p, q,Θ, y) сторони A.

2. Перевірити, що 0 < e < p. Якщо умова не виконується, то підпис відкидається.

3. Перевірити, що 0 < s < q. Якщо умова не виконується, то підпис відкидається.

4. Обчислити:

$$v = \Theta^s y^{-e} \bmod p;$$
$$\tilde{m} = v^e \bmod p.$$

5. Перевірити, що $\tilde{m} \in M_R$. Якщо $\tilde{m} \notin M_R$, то підпис відкидається.

6. Відновити повідомлення $m = R^{-1}(\tilde{m})$.

Коректність ЕЦП випливає зі справедливості такого ствердження. Якщо абонент A сформував підпис, то:

$$v \equiv \Theta^s y^{-e} \equiv \Theta^{s-ae} \equiv \Theta^k (\bmod p).$$

У такий спосіб $v^e \equiv \Theta^k \tilde{m} \Theta^{-k} \equiv \tilde{m} (\bmod p)$, що і потрібно було довести.

Стійкість *NR*-схеми ґрунтується на складності розв'язання задачі дискретного логарифма. Противник може спробувати підробити підпис шляхом вибору числа k і обчислення параметрів r. Така атака рівноцінна

тривіальному випадковому підбору підпису s. Імовірність успіху такої атаки складає 1/p, що при великих значеннях p складає досить малу величину.

Для запобігання визначення особистого ключа необхідно на кожне повідомлення обирати різне k. Ці числа повинні формуватися з урахуванням спеціальних вимог, мати рівномірний закон розподілу і бути непередбачуваними. Для формування числа k можна використовувати генератор, визначений у ANSI X 9.17. Оскільки *NR*-схема підпису забезпечує відновлення повідомлення, то пред'являються тверді вимоги до функції введення надмірності R(m). При виборі функції надмірності R необхідно враховувати мультиплікативну природу схеми підпису. Припустимо, що $m \in M$, $\tilde{m} = R(m)$ і (e, s) – цифровий підпис для m. Тоді $e = \tilde{m}\Theta^{-k} \bmod p$ для деякого цілого k і $s = ae + k \bmod q$. Нехай $\tilde{m}^* = \tilde{m}\Theta^e \bmod p$ для деякого e. Якщо $s^* = s + e \bmod q$ і $\tilde{m} \in M_R$, то (e, s^*) є коректним підписом для нового повідомлення $m^* = R^{-1}(\tilde{m}^*)$. Це випливає з того, що:

$$v \equiv \Theta^s y^{-e} \equiv \Theta^{s+e}\Theta^{-ae} \equiv \Theta^{k+e} (\bmod p);$$
$$ve \equiv \Theta^{k+e}\tilde{m}\Theta^{-k} \equiv \tilde{m}\Theta^e \equiv \tilde{m}^* (\bmod p).$$

Оскільки $\tilde{m} \in M_R$, то підроблений підпис (e, s^*) буде прийнятий як коректний для повідомлення m^*.

Нарешті перевірка умови 0 < e < p є обов'язковою. Припустимо (e, s) є підписом сторони A для повідомлення m. Тоді $e = \tilde{m}r \bmod p$ і $s = ae + k \bmod q$. Зловмисник може використовувати цей підпис для формування підпису під обраним ним самим повідомленням m^*. Він визначає e^* таке, що $e^* \equiv \tilde{m}^* r (\bmod p)$ і $e^* \equiv e (\bmod q)$. Якщо не перевірити умову, що $0 < e^* < p$, то пара (e, s^*) буде прийнята як справжній підпис.

Розглянемо основні стандарти ЕЦП, застосовані у комплексних системах захисту банківської інформації, проведемо зіставлення основних характеристик ЕЦП (довжину ключів, довжину цифрового підпису, складність (час) обчислення і складність (час) перевірки дійсності цифрового підпису) з метою виявлення їх переваг і недоліків за умови, що рівень стійкості підпису стосовно будь-яких методів фальсифікації не нижче, ніж 10^{21} (або 30 років безперервної роботи мережі із 1 000 супер-комп'ютерів).

У якості "базової" довжини ключів і довжини самого цифрового підпису будемо розглядати довжину в 64 байти.

Алгоритм RSA ґрунтується на NP-повній задачі факторизації числа (розкладання цілого параметра n у вигляді добутку двох різних простих чисел приблизно рівних один по одному величини, тобто n = p × q на ці прості множники). За сучасними оцінками складність задачі розкладання на прості множники при цілих числах n з 64 байтів становить порядку 10^{17} – 10^{18} операцій, тобто перебуває десь на грані досяжності для серйозного "противника". Тому звичайно в системах цифрового підпису на основі алгоритму RSA застосовують більш довгі цілі числа n (звичайно від 75 до 128 байтів). Це відповідно приводить до збільшення довжини самого цифрового підпису відносно 64-байтного варіанта приблизно на 20 – 100 % (у цьому випадку її довжина збігається з довжиною запису числа n), а також від 70 до 800 % збільшує час обчислень при підписуванні і перевірці.

Крім того, при генерації і обчисленні ключів у системі RSA необхідно перевіряти велику кількість досить складних додаткових умов на прості числа p і q, а невиконання кожного з них може уможливити фальсифікацію підпису з боку того, хто виявить невиконання хоча б однієї із цих умов (при підписуванні важливих документів допускати, навіть теоретично, таку можливість небажано). Схему створення і перевірки підпису за алгоритмом RSA наведено на рис. 7.4.

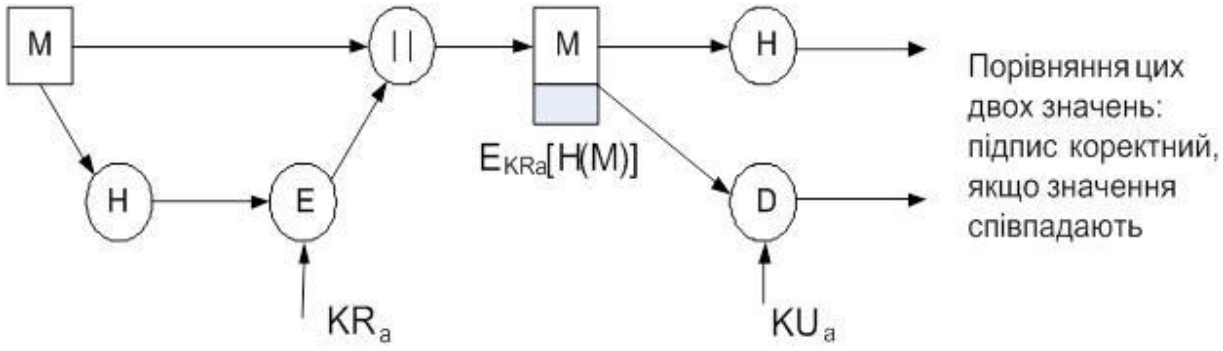


Рис. 7.4. **Створення і перевірка підпису за алгоритмом RSA**

Алгоритм НОТАРІУС

Алгоритм НОТАРІУС-1 [46] – аналог алгоритму Ель – Гамаля. Основна відмінність алгоритму полягає в тому, що замість звичайної операції множення цілих чисел за модулем великого простого p використовується множення за модулем простого числа, ця операція набагато ефективніше обчислюється на розповсюджених процесорах.

Процедури підписування електронних документів і перевірки цифрових підписів за алгоритмом НОТАРІУС-1 виглядають аналогічно відповідним процедурам алгоритму Ель – Гамаля та забезпечують той же рівень стійкості підпису, але виконуються швидше. Алгоритм НОТАРІУС-S дозволив при збереженні стійкості підпису скоротити її довжину ще на 32,5 %.

Для базового варіанта з ключами з 64 байтів довжина підпису скоротилася відносно DSA і НОТАРІУСА-D з 40 байтів до 27 байтів. Відповідно зменшився час обчислення і перевірки підпису. Стійкість залишилася на тому ж рівні – 10^{21}[23].

Алгоритм EGSA

Алгоритм EGSA [23; 45]. Істотним кроком уперед у розробці сучасних алгоритмів цифрового підпису був новий алгоритм Ель – Гамаля. У цьому алгоритмі ціле число n покладається рівним спеціально обраному великому простому числу p, за модулем якого і виробляються всі обчислення.

Такий вибір дозволяє підвищити стійкість підпису при ключах з 64 байтів приблизно в 1000 разів, тобто при такій довжині ключів забезпечується необхідний нам рівень стійкості порядку 10^{21}.

При цьому довжина самого цифрового підпису збільшується у два рази і становить 128 байтів. Головна "заслуга" алгоритму Ель – Гамаля полягала в тому, що надалі він послужив основою для прийняття декількох стандартів цифрового підпису, у тому числі національного стандарту США DSS і державного стандарту РФ ГОСТ Р-34.10-2001.

Алгоритм цифрового підпису ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) – алгоритм із відкритим ключем для створення цифрового підпису, аналогічний по своїй побудові до DSA, але визначений, на відміну від нього, не над полем цілих чисел, а в групі точок еліптичної кривої.

Особливості.

Стійкість алгоритму шифрування ґрунтується на проблемі дискретного логарифма в групі точок еліптичної кривої. На відміну від проблеми простого дискретного логарифма й проблеми факторизації цілого числа, не існує субекспонеціального алгоритму для проблеми дискретного логарифма в групі точок еліптичної кривої. Із цієї причини "сила на один біт ключа" істотніше вище в алгоритмі, який використовує еліптичні криві. Д. Брауном (Daniel R. L. Brown) було доведено, що алгоритм ECDSA не є більш безпечним, ніж DSA. Їм було сформульоване обмеження безпеки

для ECDSA, яке привело до такого висновку: "Якщо група еліптичної кривої може бути змодельована основною групою і її геш-функція задовольняє певному обґрунтованому припущенню, то ECDSA стійка до chosen-message атаки з існуючою фальсифікацією". Алгоритм ECDSA в 1999 році був прийнятий як стандарт ANSI, в 2000 році – як стандарт IEEE і NIST. Також в 1998 році алгоритм був прийнятий стандартом ISO. Незважаючи на те, що стандарти ЕЦП створені зовсім недавно й перебувають на етапі вдосконалювання, одним найбільш перспективних з них на сьогоднішній день є ANSI X9.62 ECDSA від 1999 – DSA для еліптичних кривих.

Вибір параметрів. Для підписування повідомлень необхідна пара ключів – відкритий і закритий. При цьому закритий ключ повинен бути відомий тільки тому, хто підписує повідомлення, а відкритий – будь-якому бажаючому перевірити дійсність повідомлення. Також загальнодоступними є параметри самого алгоритму.

Параметри алгоритму.

1. Вибір геш-функції H(x). Для використання алгоритму необхідно, щоб повідомлення, що підписується, було числом. Геш-функція повинна перетворити будь-яке повідомлення в послідовність біт, яке можна потім перетворити в число.

2. Вибір великого простого числа q – порядок однієї із циклічних підгруп групи точок еліптичної кривої. Якщо розмірність цього числа в бітах менше розмірності в бітах значень геш-функції $H(x)$ то використовуються тільки ліві біти значення геш-функції.

3. Простим числом p позначається характеристика поля координат F_p.

Генерування ключів ECDSA.

Для простоти необхідно розглянути еліптичні криві над полем Fp, де Fp – кінцеве просте поле. Причому, якщо необхідно, конструкцію можна легко адаптувати для еліптичних кривих над іншим полем. Нехай E – еліптична крива, певна над F_p, і P – точка простого порядку q кривої $E(F_p)$. Крива E і точка P є системними параметрами. Число p – просте. Кожен користувач конструює свій ключ за допомогою таких дій:

4. Вибирає випадкове або псевдовипадкове ціле число x з інтервалу [1,q − 1].

5. Обчислює добуток Q = x*P.

Відкритим ключем користувача A є число Q, а закритим – x. Замість використання E і P у якості глобальних системних параметрів, можна

фіксувати тільки поле F_p для всіх користувачів і дозволити кожному користувачеві вибирати свою власну еліптичну криву E і точку $P \in E(F_p)$. У цьому випадку певне рівняння кривої E, координати точки P, а також порядок q цієї точки P повинні бути включені у відкритий ключ користувача. Якщо поле F_p фіксоване, то апаратна й програмна складові можуть бути побудовані так, щоб оптимізувати обчислення в тому полі. У той же час є величезна кількість варіантів вибору еліптичної кривої над полем F_p.

Обчислення цифрового підпису.

Для того, щоб підписати яке-небудь повідомлення, для якого підрахований значення h геш-функції H, користувач A повинен зробити таке:

1. Вибрати випадкове ціле число k в інтервалі [1,q − 1].
2. Обчислити $k \cdot P = (x_1, y_1)$ й припустити в r = x_1(mod q), де r отримується із цілого числа x_1 між 0 і (p − 1) зведенням за модулем q.

Зауваження: якщо r = 0, то рівняння підпису s = k^{-1}(h + xr)(modq) не залежить від секретного ключа a, і отже, (r,s) не підходить у якості цифрового підпису. Виходить, у випадку r = 0 необхідно повернутися до кроку 1.

3. Обчислити k^{-1}(mod q) й припустити s = k^{-1}(h + xr)(mod q).

Зауваження: якщо s = 0, то значення s^{-1}(mod q), потрібне для перевірки, не існує.

Виходить, у випадку s = 0 необхідно повернутися до кроку 1. Підписом для повідомлення є пара цілих чисел (r,s).

Перевірка цифрового підпису.

Для того, щоб перевірити підпис користувача A (r,s) на повідомлення, користувач B повинен зробити таке:

1. Одержати підтверджену копію відкритого ключа Q користувача A.
2. Перевірити, що числа r і s є цілими числами з інтервалу [1,q − 1], і обчислити значення геш-функції h від повідомлення.
3. Обчислити $u_1 = s^{-1}h(\text{mod } q) i u_2 = s^{-1}r(\text{mod } q)$.
4. Обчислити $u_1P + u_2Q = (x_0, y_0)$, відносно x_0 як цілого числа між 0 і (p − 1), покласти v = x_0(mod q).
5. Прийняти підпис, якщо і лише якщо v = r.
6. Прийняти підпис, якщо й тільки якщо v = r.

Зазначимо, що якщо користувач A обчислив свій підпис правильно, то $u_1P + u_2Q = (u_1 + xu_2)\ P = kP$, оскільки $k = s^{-1}(h + xr)(\text{mod } q)$, і тому v = r.

ECDSA відповідно до стандарту ANSI X9.62

Для практичного застосування алгоритму ECDSA накладають обмеження на поля, у яких визначені еліптичні криві. Більш того, для запобігання деяких відомих атак, обмеження накладаються й на рівняння, що задають еліптичні криві, і на порядок базових точок. Для простоти в цьому розділі будемо розглядати тільки кінцеві F_p.

Вимоги до еліптичної кривої.

Для того, щоб уникнути відомих атак, заснованих на проблемі дискретного логарифма в групі точок еліптичної кривої, необхідно, щоб число точок еліптичної кривої E ділилося на досить велике просте число n. Стандарт ANSI X9.62 вимагає $n > 2^{160}$. Рівняння еліптичної кривої складається специфічним образом, використовуючи випадкові (псевдовипадкові) коефіцієнти.

Головними параметрами при побудові еліптичної кривої є:

1) розмірність поля p, де p є непарним простим числом;

2) два елементи поля F_p – a і b, визначені рівнянням еліптичної кривої E, де E має вигляд:

$$y^2 = x^3 + ax + b,$$

де $a,b \in F_p$, і $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\ p)$;

3) два елементи поля F_p – xg і yg, які визначають кінцеву точку G = $= (x_g,y_g)$ – генератор $E(F_p)$;

4) порядок q точки G, де $q > 2^{160}$ і $q > 4\sqrt{p}$;

5) співмножник $h = \#E(F_p)/q$, де позначення $\#E(F_p)$ означає порядок групи точок еліптичної кривої $E(F_p)$.

Генерація головних параметрів.

Один зі способів генерування криптографічно надійних параметрів полягає в такому:

1) обираємо коефіцієнти a і b специфічним чином, використовуючи в обчисленнях випадкові/псевдовипадкові числа. Нехай E – еліптична крива – $y^2 = x^3 + ax + b$;

2) обчислюємо $N = \#E(F_p)$;

3) перевіряємо, що N має дільник, який є найбільшим простим числом q($q > 2^{160}$ і $q > 4\sqrt{p}$). Якщо ні, то потрібно повернутися на крок 1;

4) перевіряємо, що q не ділить $p_k - 1$ для кожного k, $1 \le k \le 100$. Якщо ні, то потрібно повернутися на крок 1;

5) перевіряємо, що $q \neq p$. Якщо ні, то потрібно повернути на крок 1;

6) беремо випадкову точку $G` \in E(F_p)$ і вважаємо G = (N / q)G'. Повторюємо доти поки $G \neq 0$.

У 1985 році Скооф (Schoof) представив алгоритм, що працює за поліноміальний час, для підрахунку $\#E(F_p)$ числа точок еліптичної кривої, визначеної над полем F_p (p – непарне просте число). Алгоритм Скоофа є досить неефективним на практиці для значень p, які дійсно становлять інтерес, тобто $p > 2^{160}$. В останні кілька років було зроблено багато роботи з поліпшення й прискорення алгоритму Скоофа, зараз він називається SEA (Schoof-Elkies-Atkin) алгоритм. З таким поліпшенням криптографічно придатні еліптичні криві, визначені над полями, чиї порядки більш, ніж 2^{200} можуть бути згенеровані за кілька годин на робочих станціях.

У табл. 7.1 наведені порівняльні характеристики алгоритмів RSA і ECDSA (непарний випадок) при створенні і перевірці цифрових підписів. Обидва алгоритми тестовано на паралельних процесорах Motorola 56303 DSP (66 Мгц). При цьому функція перевірки підпису RSA використовує e = 65 537.

Таблиця 7.1

Порівняльні характеристики алгоритмів RSA і ECDSA (непарний випадок) при створенні і перевірці цифрового підпису

Алгоритм (довжина ключа, біти)	Час виконання, мс	
	Створення підпису	Перевірка підпису
RSA (1024)	25	< 2
ECDSA (160)	32	33
RSA (2048)	120	5
ECDSA (216)	68	70

Як видно з табл. 7.1, при збільшенні розмірів ключа створення підписів за допомогою ECDSA виробляється значно швидше, ніж у системах RSA. Це розходження ще більшою мірою проявляється для однопроцесорних систем. З іншого боку, перевірка підпису за допомогою ECDSA робиться набагато повільніше, ніж ця ж процедура в системах RSA і знову ж розходження підсилюється для систем з одним процесором. Обробка ECDSA може трішки прискоритися в "парному" випадку.

Потужність процесора, витрачена на перевірку підпису ECDSA, може сповільнити виконання інших додатків у системі. У деяких випадках системи RSA (навіть більші ключі, що використовують), можливо, будуть більш прийнятні, ніж криптосистеми на основі еліптичної кривої. Проте криптосистеми на основі еліптичної кривої одержують все більше поширення скоріше як альтернатива, а не заміна систем RSA, оскіль ки системи ECDLP мають деякі переваги, особливо при використан ні в пристроях з малопотужними процесорами та маленькою пам'яттю [23; 45].

7.3. DSA

Алгоритм DSA було прийнято в якості федерального стандарту цифрового підпису у США в 1991 році. Після кількох модифікацій він був остаточно затверджений як стандарт для несекретних застосувань у 1994 році. Оскільки він уводився безкоштовно, то проти нього активно виступали прихильники міжнародного стандарту цифрового підпису ISO 9796, зокрема Рон Рівест та Аді Шамір, які були фінансово не зацікавлені у введенні нового стандарту, оскільки міжнародний використовував розроблену ними криптосистему RSA. Суттєві зауваження крипто-аналітиків були враховані розробниками алгоритму і він був уведений у дію.

Алгоритм DSA (Digital Signature Algorithm) є варіантом цифрового підпису Шнорра – Ель – Гамаля та використовує такі параметри:

p – просте число довжиною L бітів, де L може набувати значень від 512 до 1024 біти (у першому варіанті алгоритму p мало фіксоване значення у 512 біт, що критикувалося криптографами як мале значення модуля). Значення p має бути кратним 64;

число q, що є дільником p – 1 (щонайменше 160 біт у двійковому представленні);

елемент h з множини {1,p} порядку q;

випадкове число a<q;

число $b = h^a \bmod p$.

Алгоритм також використовує хеш-функцію $H(M)$. Стандарт вимагає використання SHA.

Величини *p, q, h* та *b* є відкритими та утворюють ***публічний ключ***.

Приватним ключем в такому разі буде число *a*.

Для ***формування електронного підпису*** повідомлення *M* власник приватного ключа виконує такі дії:

а) обирає випадкове число $r < q$;

б) обчислює $r` = r^{-1} \bmod q$;

в) обчислює $s_1 = (h^r \bmod p) \bmod q$;

г) обчислює $s_2 = r`(H(M) + as_1) \bmod q$;

д) надає пару чисел (s_1, s_2) в якості підпису для повідомлення *M*.

Перевірка електронного підпису зводиться до такого:

а) отримувач обчислює $s`= s_2^{-1} \bmod q$;

б) обчислює $u_1 = H(M)s` \bmod q$;

в) обчислює $u_2 = s`s_1 \bmod q$;

г) обчислює $t = (h^{u_1} b^{u_2} \bmod p) \bmod q$.

У разі, якщо $t = s_1$ – підпис справжній.

Доведення коректності процедури підписування. Коректність процедури ґрунтується на тому, що $h^r = h^{u_1} b^{u_2} \bmod p$.

Справді, оскільки $b=h^a \bmod p$, то $h^r = h^{u_1} b^{u_2} \bmod p$; $h^r = h^{u_1+au_2} \bmod p$. Отже, для доведення треба показати, що $r=(u_1+au_2) \bmod q$. Це рівнозначне доведенню істинності співвідношення $1 \equiv r`(u_1+au_2) \bmod q$.

Підставивши сюди значення u_1u_2, будемо мати:

$r`(u_1+au_2) \equiv r`(H(M)s`+as`s_1) \bmod q \equiv r`(H(M) +as_1)\, s` \bmod q \equiv s_2s` \bmod q \equiv s_2s_2^{-1} \bmod q \equiv 1$.

Отже, процедура утворення та перевірки підпису коректні.

Безпека DSA

Як уже згадувалося, основним недоліком першої версії DSA була замала довжина модуля *p*, усього 512 біт. Тому в подальших редакціях цього алгоритму було вирішено надати можливість користувачам змінювати довжину модуля від 512 до 1024 біти, що значно підсилило криптостійкість алгоритму до атаки грубою силою.

Як видно з самого алгоритму, випадкові числа, що використовуються для створення цифрового підпису, відкриті. Це створює можливість для аналізу стійкості цих чисел усьому криптографічному загалові. Цей факт свідчить на користь алгоритму DSA, бо найбільш популярний алгоритм RSA зберігає генеровані прості числа у таємниці, і перевірити їх якість можна лише знаючи приватний ключ.

Отже, алгоритм цифрового підпису DSA можна вважати більш стійким, ніж решта алгоритмів, що утворюють 128-бітний геш-образ. Відомості про успішні атаки на DSA досі невідомі.

7.4. Стандарти ЕЦП ГОСТ Р 34.10-94 та ГОСТ Р 34.10-2001

Алгоритм ГОСТ Р 34.10-94 дуже схожий на DSA та використовує такі параметри [6]:

p – просте число, довжина якого може бути між 509 та 512 або 1020 – 1024 біти;

q – просте число, множник p – 1, довжиною 254 – 256 біт;

a – будь-яке число, менше за p – 1, для якого $a^q \bmod p = 1$;

x – довільне число, менше за q;

$y = a^x \bmod p$.

Перші три параметри, p, q, a відкриті та можуть використовуватися усіма користувачами мережі. Величина y складає публічний ключ, а x – приватний.

Щоби підписати повідомлення M, необхідно виконати такі кроки.

1. Відправник генерує випадкове число k, менше за q.

2. Відправник обчислює два числа: $r' = a^k \bmod p$ та $r = r' \bmod q$. Якщо $r = 0$, то генерують інше число k.

3. З використанням приватного ключа користувача обчислюється таке: $s = (xr + kH(M)) \bmod q$. Якщо $s = 0$, то генерують нове число k.

4. Каналом зв'язку відправляють повідомлення M разом з підписом s.

Рівняння перевірки підпису в такому разі буде таким:

$$r \equiv (a^{sH(M)^{-1}} y^{-rH(M)^{-1}} \bmod p) \bmod q.$$

Зазначимо, що повідомлення, яке дає нульове значення геш-образу $H(M)$, не підписується, оскільки в такому разі рівняння перевірки підпису спрощується настільки, що зловмисник може легко обчислити приватний ключ.

Зі збільшенням продуктивності процесорів і розробкою нових типів криптоаналітичних атак з'явилася потреба у зміцненні алгоритму цифрового підпису. Тому стандарт ГОСТ Р 34.10-94 було модифіковано. Новий стандарт, а саме ГОСТ Р 34.10-2001, було уведено в дію 1 липня 2002 року замість старого.

У цьому стандарті використовують алгоритм з операціями над кінцевим полем групи точок еліптичної кривої [6]. Використовують еліптичну криву *E* у формі Вейєрштраса над простим полем, яка задається коефіцієнтами *a* та *b* або інваріантом кривої:

$$J(E) \equiv 1728\frac{4a^3}{4a^3 + 27b^2} \bmod p.$$

Коефіцієнти *a* та *b* визначаються за відомим інваріантом таким чином:

$$a \equiv 3k \bmod p;$$
$$b \equiv 2k \bmod p;$$

де $k \equiv \frac{J(E)}{1728 - J(E)} \bmod p; J(E) \neq 0; 1728.$

Точку Q будемо називати точкою кратності *k*, якщо для деякої точки *P* справджується рівність *Q* = *kP*.

Параметрами такої схеми ЕЦП будуть такі значення:

p – модуль еліптичної кривої, просте число, $p > 2^{255}$;

еліптична крива, яку задано інваріантом *J*(*E*) або коефіцієнтами *a* та *b*;

ціле число *m* – порядок групи точок еліптичної кривої *E*;

просте число *q* – порядок циклічної підгрупи групи точок еліптичної кривої *E*, для якого виконуються такі умови:

$m = nq$, $n \in Z$, $n \geq 1$; $2^{254} < q < 2^{256}$;

базова точка $P \neq O$ на кривій порядку *q* (тобто $qP = O$). Координати цієї точки позначимо через (x_p, y_p);

геш-функція, передбачена стандартом ГОСТ Р 34.11-94, яка перетворює двійкову послідовність довільної довжини у двійкову послідовність довжиною 256 біт (у цьому стандарті в якості геш-функції рекомендовано використання шифру ГОСТ 28147-89).

Кожен учасник інформаційного обміну повинен мати власну пару ключів:

приватний ключ, ціле число *d*, $0 < d < q$;

публічний ключ, точка Q з координатами (x_p, y_p), що задовольняють рівняння *dP* = *Q*.

Для формування цифрового підпису необхідно виконати такі дії:

1. Обчислити геш-образ повідомлення *M*, *h*(*M*).
2. Двійковому вектору $h = (\alpha_{255}, \ldots, \alpha_0)$ ставиться у відповідність число:

$$\alpha = \sum_{i=0}^{255} \alpha_i 2^i$$

та обчислити число $e \equiv \alpha \bmod q$. Якщо *e* = 0, то встановити *e* = 1.

3. Згенерувати випадкове число $0 < k < q$.

4. Обчислити точку еліптичної кривої $C = kP$ та визначити величину $r \equiv x_C \bmod q$, де x_C – x-координата точки C. Якщо $r = 0$, то згенерувати наступне випадкове число r.

5. Обчислити значення $s \equiv (rd + ke) \bmod q$. Якщо $s = 0$, тоді обираємо нове число k.

6. Обчислити двійкові вектори, що відповідають числам r та s. Визначити цифровий підпис, як конкатенацію цих двійкових представлень: $\xi=\{r \| s\}$.

Для перевірки справжності ЕЦП виконують такі дії:

1. За отриманим значенням ЕЦП ξ відновлюють значення r та s. Якщо $0 < r < q$, $0 < s < q$, то переходимо до наступного кроку. Якщо ні, то підпис несправжній.

2. Обчислюють геш-образ отриманого повідомлення $h(M)$.

3. Обчислюють число α, двійковим представленням якого є вектор h та число $e \equiv \alpha \bmod q$. Якщо $e = 0$, то встановити $e = 1$.

4. Обчислюють $v \equiv e^{-1} \bmod q$, $z_1 \equiv sv \bmod q$, $z_2 \equiv -rv \bmod q$.

5. Обчислюють точку еліптичної кривої $C = z_1P + z_2Q$ та визначають $R \equiv x_C \bmod q$, де x_C – x-координата точки C.

6. Якщо виконується рівність $R = r$, то підпис справжній. В іншому випадку підпис не підтверджено.

Схожим чином працюють й інші системи електронного цифрового підпису, які ґрунтуються на еліптичних кривих (наприклад, ECDSA). Вони привертають увагу криптографів, оскільки дозволяють конструювати "елементи" та "правила об’єднання", які формують групи. Властивості цих груп відомі достатньо добре, щоб використати їх у криптографічних застосуваннях. Однак групи точок еліптичних кривих, на відміну від інших, не мають характерних властивостей, що полегшують криптоаналіз. Наприклад, їм не властиве поняття "гладкості".

За допомогою еліптичних кривих можна реалізувати багато криптографічних алгоритмів з відкритими ключами, наприклад, Діффі – Хеллмана, Ель – Гамаля та інші.

Детальніше про криптографічні системи на еліптичних кривих можна почитати у [23; 43; 48].

7.5. Український алгоритм ЕЦП ДСТУ 4145

Алгоритм електроного цифрового підпису стандарту ДСТУ 4145-2002 заснований на еліптичних кривих над полем (несуперсінгулярні криві) [11]. У стандарті використовуються криві в поліноміальному базисі і криві в оптимальному нормальному базисі. Причому останні (ОНБ) двох різних типів: другого і третього типу. Основою для розробки даного стандарту послужив міжнародний стандарт IEEE P1363a. Тому принциповою відмінністю ДСТУ 4145 є тільки використання "своєї" функції гешування. Для гешування використовується алгоритм – міждержавний стандарт ГОСТ 34.311-95 (раніше ГОСТ 28147-89 у режимі імітовставки).

Цей стандарт встановлює механізм цифрового підпису, заснований на властивостях груп точок еліптичних кривих над полями $GF(2^m)$, і правила застосування цього механізму до повідомлень, які передаються каналами зв'язку й/або обробляються в комп'ютеризованих системах загального призначення. Застосування даного стандарту гарантує цілісність підписаного повідомлення, автентичність його автора й незаперечність авторства при дотриманні певних правил використання цифрового підпису, які не є об'єктом даного стандарту. Далі визначені правила реалізації процедур, які необхідні для побудови криптографічних алгоритмів, визначених даним стандартом.

Датчик псевдовипадкових послідовностей.

Датчик псевдовипадкових послідовностей використовується для одержання випадкових даних, необхідних для побудови загальних параметрів цифрового підпису, обчислення секретних і відкритих ключів цифрового підпису й обчислення цифрового підпису.

У якості датчика псевдовипадкових послідовностей повинен використовуватися датчик псевдовипадкових послідовностей, або будь-який інший датчик псевдовипадкових послідовностей, рекомендований уповноваженим органом державного управління.

Функція гешування.

У цьому стандарті функція гешування використовується при обчисленні й перевірці цифрового підпису. Функція гешування H перетворить текст T довжини $L_т$ у двійковий рядок H(T) фіксованої довжини L_H.

У цьому стандарті повинна використовуватися функція гешування, визначена ДСТУ 34.311-95, або будь-яка інша функція гешування, рекомендована вповноваженим органом державного упраління. Значення параметра L_H однозначно визначається ідентифікатором i_h конкретної функції гешування, яка використовується разом з даним стандартом. Параметр i_h входить до числа загальних параметрів цифрового підпису. Таких параметрів у групі користувачів може бути кілька. При цьому $L_H \geq 160, L(IH) \leq 64$. Значення IH = 1, L(IH) = 8 відповідають функції гешування, визначеної ГОСТ 34.311-95 [7; 11].

Дозволяється використання геш-функції за замовчуванням. У цьому випадку параметра $L_т$ може не бути. Якщо використовувана функція гешування накладає обмеження на довжину тексту $L_т$, то ці обмеження мають силу й для цього стандарту.

Обчислення випадкового цілого числа.

У цьому підрозділі визначається алгоритм обчислення випадкового цілого числа a, такого що L(a) < L(n). У якості датчика псевдовипадкових послідовностей повинен використовуватися датчик псевдовипадкових послідовностей. Вихідні дані алгоритму: порядок базової крапки еліптичної кривій n, довжина t псевдовипадкової послідовності, яка створюється датчиком псевдовипадкових послідовностей при одному звертанні до нього. Результат виконання алгоритму – випадкове ціле число a, що задовольняє умову L(a) < L(n).

Алгоритм обчислення випадкового цілого числа:

1) обчислюється довжина L(n) двійкового представлення цілого числа n;

2) обчислюється мінімальне значення k, таке що $kt \geq L(n) - 1$;

3) за k звертань до датчика псевдовипадкових послідовностей формується випадковий двійковий рядок довжини kt; перші l(n) – 1 елементів цієї послідовності утворюють випадковий двійковий рядок $R_{L(n)\ 2}, ... R_0$ довжиною l(n) – 1;

4) вважають, що $a_i = R_i$ для i = 0,…, l(n) – 2;

5) визначається індекс j, який дорівнює найбільшому значенню індексу i, для якого a_i = 1; якщо такого індексу немає, то вважають, що j = 0, a_0 = 0;

6) випадковий рядок $R_{L(n)\ 2},...R_0$ знищується;

7) двійковий рядок $a_j,\ldots,a_0$ задає випадкове ціле число a довжиною j + 1.

Обчислення випадкового елемента базового поля.

У цьому підрозділі визначається алгоритм обчислення випадкового елемента x базового поля $GF(2^m)$. У якості датчика псевдовипадкових послідовностей повинен використовуватися датчик псевдовипадкових послідовностей. Вихідні дані алгоритму: степінь базового поля m, довжина t псевдовипадкової послідовності, яка створюється датчиком псевдовипадкових послідовностей при одному звертанні до нього.

Результат виконання алгоритму – випадковий елемент базового поля m. Алгоритм обчислення випадкового елемента базового поля:

1) обчислюється мінімальне значення k, таке що $kt \geq m-1$;

2) за k звертань до датчика псевдовипадкових послідовностей формується випадковий двійковий рядок довжини kt;

3) перші m елементів цієї послідовності утворюють випадкову двійкову послідовність $R_{m-1},...R_0$;

4) вважають, що $x_i = R_i$, для i = 0,…,m – 1;

5) випадковий рядок $R_{m-1},...R_0$ знищується;

6) двійковий рядок $x_{m-1},...x_0$ задає випадковий елемент базового поля x.

Обчислення сліду елемента базового поля.

У цьому підрозділі визначається алгоритм обчислення сліду елемента x базового поля.

Вихідні дані алгоритму: елемент x базового поля $GF(2^m)$.

Результат виконання алгоритму – слід tr(x) елемента x.

Алгоритм обчислення сліду:

1) вважають, що t = x;

2) для i від 1 до m-1 обчислюється $t \leftarrow t^2 + x$;

3) результат обчислення сліду $tr(x) = t$.

Обчислення напівсліду елемента базового поля.

У цьому підрозділі визначається алгоритм обчислення напівсліду елемента x базового поля, непарного ступеня m. Результат виконання

алгоритму – напівслід htr(x) елемента x. Алгоритм обчислення напівсліду:

1) вважають, що t = x;

2) для і від 1 до $\frac{m-1}{2}$ обчислюється $t \leftarrow t^4 + x$;

3) результат обчислення напівсліду htr(x) = t.

Розв'язок квадратного рівняння в базовому полі.

У цьому підрозділі визначається алгоритм розв'язку квадратного рівняння $z^2 + uz = w$ в базовому полі.

Вихідні дані алгоритму: квадратне рівняння $z^2 + uz = w$, $u, w \in GF(m^2)$. Результат виконання алгоритму – кількість розв'язків k квадратного рівняння й один з розв'язків цього рівняння, якщо k>0.

Алгоритм розв'язку квадратного рівняння:

1. Якщо u = 0, то $z = w^{2m-1} = \sqrt{w, k = 1}$ й виконується перехід до кроку 8.
2. Якщо w = 0, то вважають, що k = 2, z = 0 й виконується перехід до кроку 8.
3. Обчислюється елемент базового поля $v = wu^{-2}$.
4. Обчислюється слід елемента v.
5. Якщо слід елемента tr(v) = 1, то вважають, що k = 2, z = 0 й виконується перехід до кроку 8.
6. Обчислюється напівслід t = htr(v) елемента v.
7. Обчислюється елемент базового поля z = tu і вважають, що k = 2.
8. Результат виконання алгоритму: кількість розв'язків k квадратного рівняння й розв'язок цього рівняння z, якщо k > 0.

Обчислення випадкової точки еліптичної кривої.

У цьому підрозділі визначений алгоритм обчислення випадкової точки еліптичної кривої. Вихідні дані алгоритму: еліптична крива $y^2 + xy = x^3 + Ax^2 + B$ над полем $GF(2^m)$, $B \neq 0, A \in \{0,1\}$.

Результат виконання алгоритму – випадкова точка цієї еліптичної кривої P = (X_p,Y_p). Алгоритм обчислення випадкової точки еліптичної кривої:

1. Обчислюється випадковий елемент і базового поля.
2. Обчислюється елемент базового поля $W = U^3 + Au^2 + B$.

3. Розв'язується квадратне рівняння $z^2 + uz = w$.

4. Якщо число розв'язків квадратного рівняння дорівнює 0, то виконується перехід до кроку 1, а якщо ні, то виконується перехід до кроку 5.

5. Вважають, що $x_p = u, y_p = z$, z – розв'язок квадратного рівняння, отриманий на кроці 3.

6. Результат виконання алгоритму – випадкова точка еліптичної кривої p з координатами (x_p, y_p).

Стиск точки еліптичної кривої.

У цьому підрозділі визначається алгоритм перетворення точки P простого порядку n еліптичної кривої з координатами (X_p, Y_p) в стисле представлення $P \in GF(m^2)$, m – непарне число. Вихідні дані алгоритму: точка еліптичної кривої P простого порядку n з координатами (X_p, Y_p).

Результат виконання алгоритму – стисле представлення $P \in GF(m^2)$ P еліптичної кривої. Алгоритм стиску точки еліптичної кривої:

1. Якщо $X_p = 0$, то вважають, що P = 0 і виконується перехід до кроку 3.

2. Якщо $X_p \neq 0$, то обчислюється елемент базового поля $y = y_p x_p^{-1} = (y_{m-1}, ..., y_0)$, обчислюється слід елемента y, i = tr(y) і вважають, що $P = (P_{m-1}, ..., P_0) = (X_{p,m-1}, ..., X_{p,1}, i)$, тобто крайній правий двійковий розряд координати x_p замінюється значенням сліду елемента в.

3. Результат виконання алгоритму: $P \in GF(m^2)$ – стисле представлення точки p еліптичної кривої.

Обчислення ключів цифрового підпису.

Цей розділ установлює порядок обчислення секретного d і відкритого Q ключів цифрового підпису.

Обчислення секретного ключа цифрового підпису.

Секретний ключ d цифрового підпису обчислюється таким способом:

1. Обчислюється випадкове ціле число.

2. Якщо $d \neq 0$, то d береться в якості секретного ключа цифрового підпису, а якщо ні, то виконується перехід до кроку 1.

Умови обчислення й зберігання секретного ключа цифрового підпису повинні виключати можливість несанкціонованого доступу до секретного ключа або його частини, а також до даних, які використовувалися в процесі обчислення секретного ключа. Умови зберігання секретного ключа повинні виключати можливість модифікації, знищення або підміни секретного ключа.

Обчислення відкритого ключа цифрового підпису.

Відкритий ключ цифрового підпису є точкою еліптичної кривої виду:

$$Q = -dP,$$

где P – базова точка еліптичної кривої;

d – секретний ключ цифрового підпису.

Умови зберігання відкритого ключа повинні виключати можливість модифікації або підміни відкритого ключа цифрового підпису. Допускається зберігання й передача відкритого ключа цифрового підпису в стислому виді.

Перевірка коректності ключів цифрового підпису.

Висока криптографічна стійкість цифрового підпису, обчисленому згідно з даним стандартом, гарантується тільки в тому випадку, якщо секретний і відкритий ключі цифрового підпису обчислені коректно, тобто в строгій відповідності до даного стандарту. Цей розділ установлює правила перевірки коректності відкритого й секретного ключів цифрового підпису.

Перевірка коректності відкритого ключа цифрового підпису.

Відкритий ключ Q цифрового підпису повинен задовольняти такі умови:

1) координати точки еліптичної кривої, що представляє відкритий ключ цифрового підпису, належать базовому полю, тобто є двійковими рядками довжини m;

2) $Q \neq 0$;

3) відкритий ключ $Q = (X_Q, Y_Q)$ лежить на еліптичній кривій, тобто його координати задовольняють рівнянню еліптичної кривої;

4) порядок відкритого ключа $Q = (X_Q, Y_Q)$ рівний n, тобто $nQ = 0$.

Якщо умови 1 – 4 виконані, то відкритий ключ цифрового підпису є коректним.

Визначена в цьому підрозділі перевірка може не виконуватися, якщо використані в конкретній реалізації цифрового підпису способи зберігання відкритого ключа цифрового підпису виключають можливість його підміни, модифікації або знищення.

Перевірка коректності секретного ключа виконується винятково власником секретного ключа таким способом:

1. Обчислюється точка еліптичної кривої:

2.

$$Q` = – dP,$$

де P – базова точка еліптичної кривої;

d – секретний ключ ЕЦП.

2. Якщо Q` = Q, де Q – відкритий ключ цифрового підпису, то секретний ключ відповідає відкритому ключу цифрового підпису і є правильним.

Визначена в цьому підрозділі перевірка може не виконуватися, якщо використані в конкретній реалізації цифрового підпису способи зберігання секретного ключа цифровому підпису виключають можливість його підміни, модифікації або знищення.

Обчислення цифрового предпідпису.

У цьому розділі визначається алгоритм обчислення цифрового предпідпису.

Вихідні дані алгоритму: загальні параметри цифрового підпису. Результат виконання алгоритму – цифровий предпідпис (e, F_e), $e \in GF(n)$ – ціле число, $F_e \in GF(m^2)$.

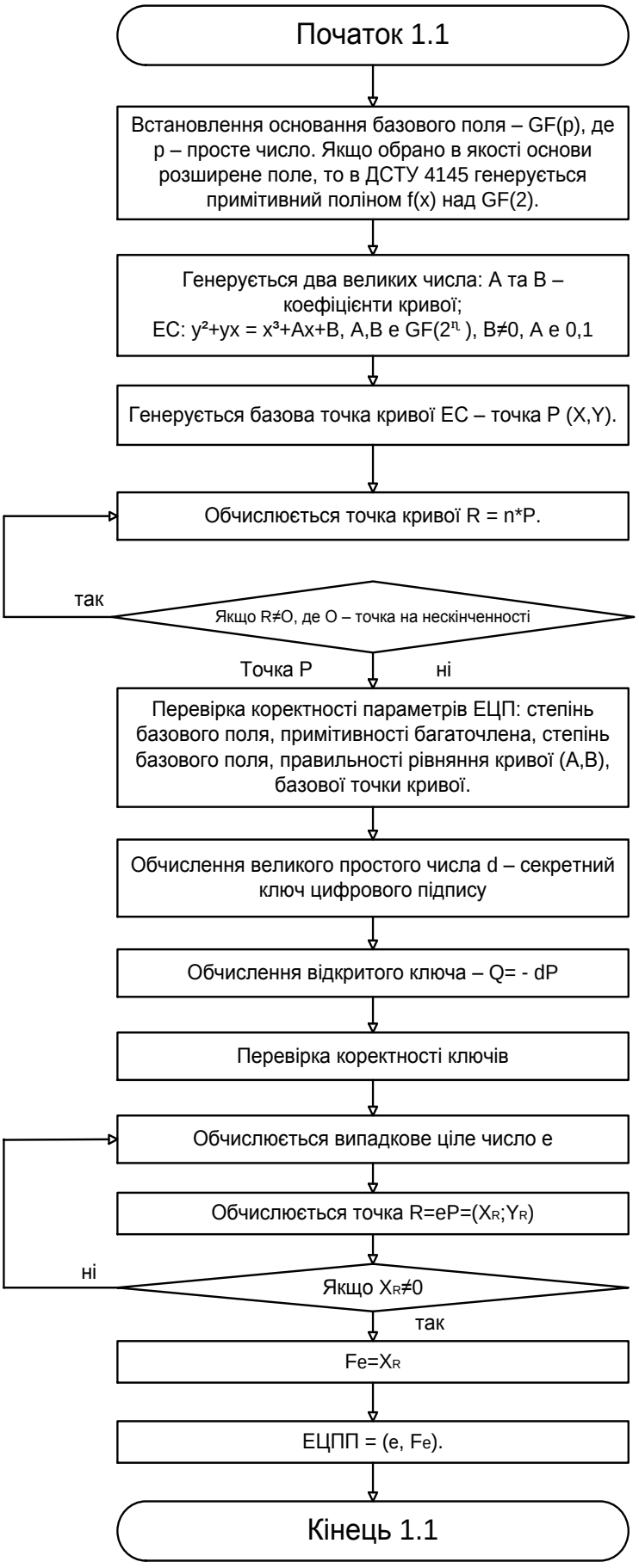
Алгоритм обчислення цифрового предпідпису:

1. Обчислюється випадкове ціле число e.
2. Обчислюється точка еліптичної кривої $R = eP = (X_R, Y_R)$.
3. Якщо координата $X_R = 0$, то виконується перехід до кроку 1, а якщо ні, то вважають, що $F_e = X_R$ й виконується перехід до кроку 4.
4. Результат виконання алгоритму – цифровий предпідпис (e, F_e).

На рис. 7.5 та 7.6 наведено алгоритм електронного цифрового підпису стандарту ДСТУ 4145 [11].

Допускається попереднє обчислення довільного числа цифрових предпідписів. Після використання цифрового предпідпису для обчислення цифрового підпису цифровий предпідпис негайно знищується.

Цифровий предпідпис секретний. Умови обчислення й зберігання цифрового предпідпису повинні виключати можливість несанкціонованого доступу до неї або її частини, а також до даних, які використовувалися в процесі обчислення цифрового предпідпису. Умови зберігання цифрового предпідпису повинні виключати можливість його модифікації або підміни.



Незведений поліном – це поліном, який ділиться тільки на себе та на одиницю. Примітивним поліномом називається поліном f(x), корінь якого є примітивним елементом поля.

Початок генерації базових параметрів ЕЦП

Початок генерації ЕЦП

Обчислення ЕЦП (підрозділ 6.3-6.8)

Цифровий предпідпис

Рис. 7.5. **Частина 1 алгоритму електронного цифрового підпису**

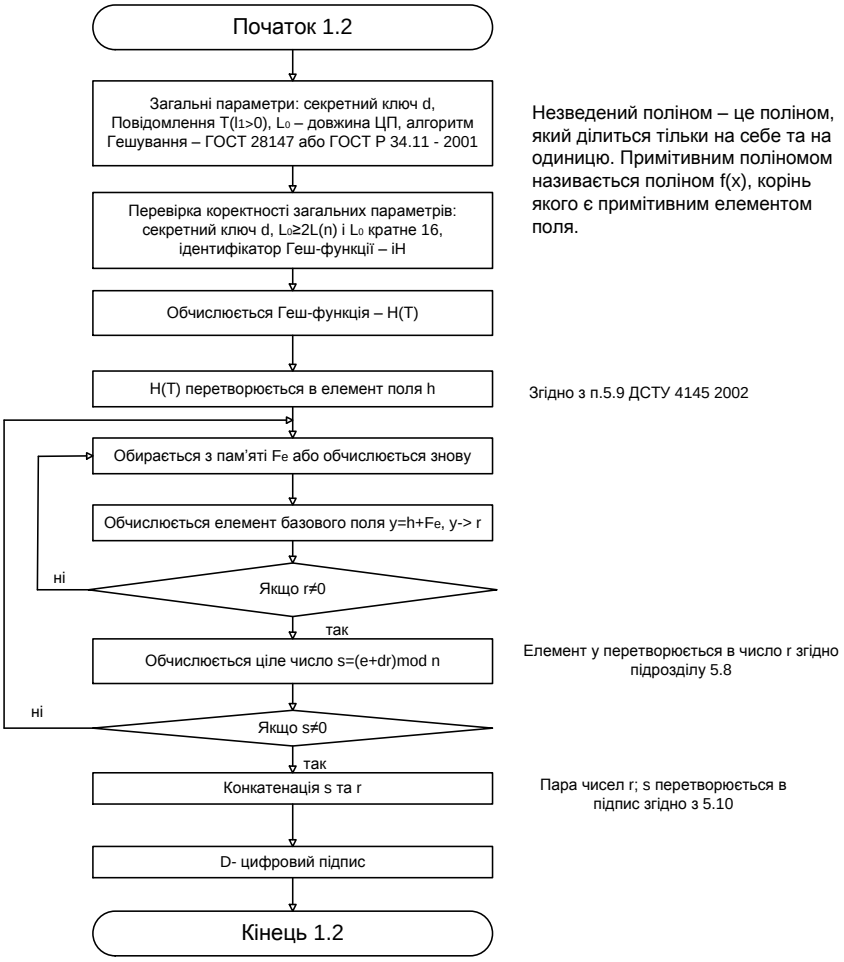


Рис. 7.6. **Частина 2 алгоритму електронного цифрового підпису**

Протоколи колективного цифрового підпису на еліптичних і гіпереліптичних кривих

У якості джерела абелевої групи для протоколу колективного підпису на основі ДСТУ-4145, запропонованого в [7], можна обрати групу дивізорів гіпереліптичної кривої. Основна перевага використання гіпереліптичних кривих полягає в тому, що розмір основного поля, над яким визначена крива, зменшується пропорційно до роду кривої без втрати стійкості, хоча сама формула групового додавання виглядає більш громіздко.

Нехай F – кінцеве поле й нехай $\tilde{F}$ – алгебраїчне замикання поля F. Гіпереліптична крива 3 роду $g \le 1$ над F є [7] набором розв'язків $(x, y) \in F \times F$ рівняння:

$$C : y^2 + h(x)y = f(x), \qquad (1)$$

де $h(x) \in F[x]$ – поліном ступеня не більш $g, f(x) \in F[x]$ – нормований поліном степеня 2g + 1 і не існує розв'язків $(x, y) \in \tilde{F} \times \tilde{F}$, які б одночасно задовольняли рівнянню (1) і рівнянням 2y + h(x) = 0 і h`(x)y – f`(x) = 0.

Вважаємо, що нескінченно віддалена точка ∞ також належить кривій.

Згідно з роботою [7], у якості групової структури у випадку гіпереліптичних кривих розглядається якобіан кривої C. Кожний елемент якобіана – це клас еквівалентності дивізорів, який може бути представлений унікально наведеним дивізором у вигляді пари поліномів у формі Мамфорда. На якобіані визначені групові операції додавання й дублювання дивізорів.

Згідно з роботою [8], порядок якобіана гіпереліптичної кривої обмежений інтервалом Хассе – Вейла:

$$\left|\left(\sqrt{q}-1\right)^{2g}\right| \le \#J/F_q \le \left[\left(\sqrt{q}+1\right)^{2g}\right],$$

де q – характеристика поля, над яким визначена крива;

g – рід кривої.

Будемо вважати, що порядок якобіана $J/F_q \approx q^g$.

Більшість криптографічних додатків базуються на еліптичних або гіпереліптичних кривих з довжиною ключа не менш 160 біт, тобто з порядком групи не менш 2^{160}. Отже, для криптосистем на гіпереліптичних кривих над полем F_q повинне виконуватися як мінімум:

$$g \times \log_2 q \approx 160.$$

Зокрема, для кривої роду 2 необхідно вибрати основне поле F_q з $|F_q| \approx 2^{80}$, з довжиною операндів 80 біт. Для кривої 3 роду потужність основного поля $|F_q| \approx 2^{54}$ для кривої 4 роду $|F_q| \approx 2^{40}$.

Оскільки елементи підпису належать основному полю, над яким визначена крива, у випадку гіпереліптичних кривих розмір підсумкового колективного підпису зменшується пропорційно роду кривої. Так, при використанні гіпереліптичної кривої другого роду розмір колективного підпису виявиться приблизно у два рази менше, ніж при використанні еліптичної кривої, що володіє аналогічним рівнем криптостійкості. Відповідно при використанні кривої третього роду розмір колективного підпису зменшиться приблизно в три рази і т. д.

З криптографічною метою використовуються гіпереліптичні криві роду 2 і 3. Криві більш високого роду не є стійкими.

Протокол колективного підпису на основі стандарту ДСТУ 4145-2002 на гіпереліптичних кривих.

Введемо позначення [11]:

D – базовий дивізор гіпереліптичної кривої;

l – кількість користувачів;

n – порядок циклічної підгрупи якобіана гіпереліптичної кривої;

d_i – секретний ключ і-го користувача;

h – геш-образ повідомлення.

$\Psi(R)$ – функція перетворення дивізора в елемент основного поля.

Авторами пропонується таке перетворення: коефіцієнти першого полінома дивізора R представимо у вигляді числа в системі числення з основою, що дорівнює модулю основного поля, над яким визначена крива (у випадку простого поля). А потім переведемо це представлення в десяткову систему числення.

Генерація відкритого колективного ключа.

Кожний і-й користувач (i = 1…l) формує відкритий ключ виду:

$$Q_i = -d_i D.$$

Колективний відкритий ключ обчислюється як сума відкритих ключів групи з l користувачів:

$$Q = \sum_{i-1}^{l} Q_i = \sum_{i-1}^{l} d_i D.$$

Формування колективного підпису.

1. Кожний користувач розраховує дивізор R_i у такий спосіб:

а) вибирає випадковий параметр k_i, 1 < k < n;

б) обчислює $R_i = k_i D$.

2. За представленими користувачами дивізорами R_i обчислюється загальний дивізор:

$$R = \sum_{i=1}^{l} R_i.$$

3. Обчислюється значення функції $\Psi(R)$.

4. Перша частина підпису визначається за формулою:

$$r = h\Psi(R) \bmod n.$$

5. Кожний користувач обчислює свій параметр s_i:

$$s_i = (k_i + d_i r) \bmod n.$$

6. Друга частина підпису визначається за формулою:

$$s = \sum_{i=1}^{l} s_i \bmod n.$$

Колективним підписом є пара чисел – (r,s).

Перевірка колективного підпису.

Перевіряючий обчислює геш-образ h` загального повідомлення.

Використовуючи відкритий колективний ключ Q, обчислює дивізор R` = sD + rQ, R` і значення $v = h`\Psi(R`)$. Якщо v = r, то підпис визнається справжнім.

Протокол колективного підпису на основі протоколу ECRP на еліптичних кривих. Протокол електронного цифрового підпису з перед обчисленнями ECRP [6] було запропоновано з метою зменшити трудомісткість операції верифікації підпису в корпоративній мережі за рахунок множення тільки на базову точку, яке можна виконати з перед-обчисленнями. Модифікуємо цей протокол для реалізації колективного підпису. Введемо позначення:

P – базова крапка еліптичної кривої;

l – кількість користувачів;

n – порядок циклічної підгрупи точок еліптичної кривої;

d_i – секретний ключ i-го користувача;

h – геш-образ повідомлення;

$\pi(R) = X_R \bmod n$ – виділення x-координати точки $R = (X_R, Y_R)$ еліптичної кривої.

Генерація відкритого колективного ключа.

Кожний i-й користувач (i = 1…l) формує відкритий ключ виду:

$$Q_i = -d_i P.$$

Колективний відкритий ключ обчислюється як сума відкритих ключів групи з l користувачів:

$$Q = \sum_{i=1}^{l} Q_i = \sum_{i=1}^{l} - d_i P.$$

Формування колективного підпису.

Кожний i-й користувач (i = 1…l) розраховує точку R_i таким чином:

а) вибирає випадковий параметр k_i, $1 < k_i < n$;

б) обчислює значення $t_i = \frac{k_i}{n} \bmod n$;

в) і точку $R_i = t_i P$.

За представленими користувачами точками R_i обчислюється загальна точка $R = \sum_{i=1}^{l} R_i = (X_R, Y_R)$ і значення $\omega = \pi(R) = X_R \bmod n$.

Формується точка $\omega R = (x, y)$ і перша частина колективного підпису $r = \pi(x, y) = x \bmod n$.

Кожний користувач обчислює свій параметр s_i:

$$s_i = (\omega k_i + h d_i) \bmod n$$

і надає його для обчислення другої частини колективного підпису $s = \sum_{i=1}^{l} s_i$. Колективним підписом є пара чисел – (r,s).

Перевірка колективного підпису.

Перевіряючий обчислює геш-образ h` h′ загального повідомлення і значення $t = \frac{s}{h`} \bmod n$.

Використовуючи відкритий колективний ключ Q формує точку:

$$tP + Q = (x,y)$$

і обчислює значення $v = \pi(x,y) = x \bmod n$.

Якщо v = r, то підпис визнається справжнім.

Обґрунтування коректності представленого протоколу. Оскільки при формуванні підпису значення r визначається формулою $r = \pi(\omega R) = \pi\left(\omega \sum_{i-1}^{l} \frac{k_i}{h} P\right)$, а при перевірці підпису перевірочний вираз tP + Q дає точку ωR.

$$tP + Q = \frac{s}{h} P - \sum_{i=1}^{l} d_i P = \frac{\sum_{i=1}^{l} (wk_i + hd_i)}{h} P - \sum_{i=1}^{l} d_i P = \omega \sum_{i-1}^{l} \frac{k_i}{h} P + \sum_{i=1}^{l} d_i P - \sum_{i=1}^{l} d_i P =$$

$$= \omega \sum_{i-1}^{l} \frac{k_i}{h} P = \omega R.$$

У підсумку маємо: $v = \pi(tP + Q) = \omega R$, що відповідає r при формуванні підпису.

Протокол колективного підпису на основі протоколу ECPR на гіпереліптичних кривих. У якості джерела абелевої групи для запропонованого протоколу можна також обрати групу дивізорів гіпереліптичної кривої, як це було зроблено для ДСТУ-4145. У цьому випадку базовій точці, відкритим ключам користувачів, проміжній точці R відповідають дивізори гіпереліптичної кривої.

Генерація відкритого колективного ключа.

Кожний і-й користувач (i = 1…l) формує відкритий ключ виду:

$$Q_i = -d_i D.$$

Колективний відкритий ключ обчислюється як сума відкритих ключів групи з l користувачів:

$$Q = \sum_{i-1}^{l} Q_i = \sum_{i-1}^{l} d_i D.$$

Формування колективного підпису.

Кожний і-й користувач (i = 1…l) розраховує дивізор R_i таким чином:

а) вибирає випадковий параметр k_i, 1 < k < n;

б) обчислює значення $t_i = \frac{k_i}{n} \bmod n$;

в) $R_i = t_i P$;

г) за представленими користувачами дивізорами R_i обчислюється загальний дивізор $R = \sum_{i-1}^{l} R_i$ і значення $\omega = \Psi(R)$;

д) формується дивізор ωR і перша частина колективного підпису $r = \Psi(\omega R)$.

Кожний користувач обчислює свій параметр s_i

$$s_i = (\omega k_i + h d_i) \bmod n$$

і надає його для обчислення другої частини колективного підпису $s = \sum_{i=1}^{l} s_i$. Колективним підписом є пара чисел – (r,s) (r, s).

Перевірка колективного підпису.

Перевіряючий обчислює геш-образ h` загального повідомлення і значення $t = \frac{s}{h`} \bmod n$.

Використовуючи відкритий колективний ключ Q, формує дивізор tD + Q = (x,y) і обчислює значення $v = \Psi(tD + Q)$. Якщо v = r, то підпис визнається справжнім.

Контрольні запитання

1. Поняття цифрового підпису, вимоги до нього.
2. Класифікація схем цифрового підпису. Основні алгоритми (стандарти) ЕЦП.
3. Операції схеми цифрого підпису на основі алгоритму Ель-Гамаля (стандарт DSA).
4. Російський стандарт ЕЦП ГОСТ Р-3411. Переваги та недоліки.
5. Український алгоритм ЕЦП ДСТУ 4145. Основні операції схеми ЕЦП з використанням операцій на еліптичних кривих.
6. Протокол колективного підпису на основі протоколу ECPR на гіпереліптичних кривих.
7. Протокол колективного підпису на основі протоколу ECRP на еліптичних кривих.
8. Протокол підпису ECDSA відповідно до стандарту ANSI X9.62.

Розділ 8. Основні види атак, принципи криптоаналізу

Багато років тому, коли сторонами, що обмінювалися конфіденційною інформацією, були переважно дипломати та військові, можна було бути впевненими у надійності учасників обміну, довіряти один одному. Основною проблемою сторін тоді було забезпечення неможливості втручання у конфіденційний зв'язок третіх сторін. Практично не було випадків, коли адресати поводили себе недобросовісно: відмовлялися від отриманих повідомлень або стверджували про якісь зміни у їх тексті. Основне завдання тогочасних криптографів полягало лише у забезпеченні ***конфіденційності*** інформації.

Принципово інакше складається картина на сьогодні, коли інформацією обмінюються суб'єкти комерційної діяльності, які можуть не довіряти один одному. І якщо раніше важливо було забезпечити лише ***конфіденційність*** інформації, то зараз не менш важливо забезпечити її ***цілісність*** та ***юридичну чинність***, а також захиститися від ***нав'язування хибних повідомлень***.

Отже, метою суб'єкта в такому разі є перешкодження здійсненню намірів законних учасників інформаційного обміну. В загальному випадку зловмисник може не лише перехоплювати зашифровані повідомлення, але й модифікувати їх, а також направляти фальсифіковані повідомлення легальним сторонам, що обмінюються інформацією, ніби від імені їх легальних партнерів.

Таким чином, існує чотири можливих типи загроз з боку зловмисників:

порушення конфіденційності інформації – дешифрування, повне чи часткове, переданого повідомлення або отримання додаткової інформації про його зміст;

порушення цілісності інформації – внесення змін у повідомлення, що змінюють його зміст;

забезпечення ***неможливості відмови*** від отриманого чи відправленого повідомлення;

порушення істинності повідомлень – формування хибних повідомлень, які легальні учасники інформаційного обміну можуть класифікувати як істинні.

Дешифрування перехопленого повідомлення можливе у випадку обчислення криптографічного ключа або так званого "безключового читання", тобто за допомогою знаходження еквівалентного алгоритму, що не вимагає знання ключа.

Процес, при якому реалізується спроба отримати відкритий текст, ключ або й те, й інше, називається *криптоаналізом*. Однією з можливих атак на алгоритм шифрування є атака грубою силою (лобова атака), тобто просте перебирання усіх можливих ключів. Якщо кількість ключів досить велика, то підібрати потрібний дуже й дуже складно. При довжині ключа n бітів кількість можливих ключів дорівнює 2^n. Таким чином, чим довший ключ, тим стійкішим вважається алгоритм для лобової атаки.

Існують різні типи атак, які ґрунтуються на тому, що зловмиснику відома певна кількість пар відкритого тексту-шифротексту. При аналізі зашифрованого тексту зловмисник часто застосовує статистичні методи аналізу тексту. При цьому він має загальну уяву про тип тексту, наприклад, це англійський або російський текст, exe-файл конкретної операційної системи, вихідний текст деякою мовою програмування тощо. У деяких випадках криптоаналітик має багато інформації про відкритий текст і може перехоплювати одне або кілька незашифрованих повідомлень разом з їх шифротекстом. Іноді криптоаналітик може знати основний формат або основні характеристики повідомлення.

В усіх випадках вважають, що криптографічна схема безпечна, якщо зашифроване повідомлення не містить ніякої інформації про відкритий текст. Схема буде обчислювально безпечною, якщо:

1. Вартість розшифрування повідомлення більша від вартості інформації у відкритому повідомленні.
2. Час, необхідний для розшифрування повідомлення, більший періоду життя повідомлення.

Розглянемо класифікацію атак на криптосистеми.

8.1. Класифікація атак на симетричні криптоалгоритми

Коротко охарактеризуємо основні типи атак. Далі вони перелічені у порядку зростання небезпеки [1; 4; 15; 16; 19; 24].

1. ***Атака на основі лише шифротексту*** (*ciphertext-only-attack*). У цьому випадку зловмиснику відомі лише шифротексти, зашифровані

на одному ключі. Це найслабший тип криптоаналітичної атаки, успіх якої зовсім неочевидний. Він залежить від багатьох чинників та визначається кваліфікацією аналітика (за умови ручного криптоаналізу) або повністю визначається потужністю комп'ютерів криптоаналітичної системи.

Завжди процес криптоаналізу починається зі збирання інформації про відкритий текст:

якою мовою написаний оригінал;

які лінгвістичні особливості цієї мови;

які слова або фрази може містити оригінал та у якій послідовності;

якою приблизно може бути довжина оригінального тексту;

які методи шифрування могли застосувати для зашифрування цього тексту;

яка службова інформація може міститися у тексті (дата, контрольна сума, адреси тощо);

якою апаратурою було зашифровано текст.

Ці або подібні дані збираються агентурними або аналітичними засобами і значно полегшують роботу криптоаналітиків.

2. ***Атака на основі невибраного (відомого) відкритого тексту*** (*known-plaintext attack*). Супротивник знає або може встановити деякі частини відкритого тексту у криптограмі. Завдання полягає в тому, щоб дешифрувати увесь текст. Це можна виконати шляхом покрокового обчислення ключа.

3. ***Атака на основі обраного відкритого тексту*** (*chosen-plaintext attack*). Зловмисник може обрати будь-який відкритий текст і отримати для нього зашифрований. Завдання полягає у визначенні ключа шифрування. Деякі алгоритми шифрування досить вразливі для атак цього типу. Тому у випадку використання таких систем, серйозної уваги вимагає внутрішня безпека використання системи, щоб зловмисник ні в якому разі не зміг отримати доступ до відкритих текстів.

Така атака буде називатися ***простою*** в разі, коли усі відкриті тексти зловмисник отримує до перехоплення першої криптограми, та ***адаптивною***, коли зловмисник обирає черговий відкритий текст, маючи шифровки усіх попередніх.

4. ***Атака на основі обраного шифротексту*** (*chosen-ciphertext attack*). Зловмисник може обирати потрібну кількість криптограм та отримати для них відкриті тексти. Тут також, аналогічно до попереднього типу атак, існують різновиди простої та адаптивної атак.

5. ***Атака на основі обраного тексту*** (*chosen-text attack*). Це найнебезпечніший тип атак, оскільки зловмисник може передавати спеціально підготовлені відкриті тексти для зашифрування, а потім отримувати відповідні шифровки. Завдання полягає у розкритті ключа. Ця атака також може бути простою та адаптивною.

Якщо зловмисник здійснює атаку на основі лише шифротексту, у нього є дуже обмежений набір можливостей. До них можна віднести метод грубої сили або частотний криптоаналіз.

У разі атаки на основі відомого відкритого тексту розкриття шифру буде примітивним після того, коли у текстах зустрінеться більшість літер абетки.

Якщо зловмисник має можливість атакувати на основі обраного тексту, то криптосистему буде розкрито вже після шифрування відкритого тексту типу: "*АБВГД…ЬЮЯ*".

8.2. Класифікація атак на асиметричні криптоалгоритми

Специфічний тип атаки, ***"посередництво"*** (*Man-in-middle attack*). Ця атака спрямована на злам криптографічних комунікацій та протоколів обміну ключами. Атака здійснюється приблизно таким чином. Припустимо, що зловмисник, назвемо його *З*, має змогу перехоплювати усі повідомлення сторін *А* та *В*. Припустимо також, що сторони хочуть обмінятися криптографічними ключами, тож *А* відправляє свій ключ шифрування $K_{АВ}$ стороні *В*. Зловмисник *З* перехоплює цей ключ, зберігає його, та відправляє стороні *В* вже свій криптографічний ключ, $K_{ЗВ}$, ніби від сторони *А*. Сторона *В*, отримавши цей ключ, у відповідь відправляє стороні *А* свій криптографічний ключ $K_{ВА}$. Зловмисник *З* також підміняє цей ключ своїм, $K_{ЗА}$ і надсилає його стороні *А*. Таким чином, сторони *А* та *В* "вірять", що мають криптографічні ключі один одного, хоча насправді вони мають лише два різні ключі зловмисника *З*. Той, у свою чергу, має обидва ключі сторін та може читати усю їхню зашифровану інформацію (рис. 8.1).

Спроба обміну інформацією між сторонами призводить до такого. *А* шифрує повідомлення ключем $K_{АВ}$ та надсилає його. *З* перехоплює цього листа, розшифровує його перехопленим ключем сторони $A(K_{АВ})$, перешифровує ключем, який він надіслав *В* ($K_{ЗВ}$) та відправляє. Сторона *В*, отримавши зашифрованого листа, розшифровує його і, оскільки сеанс

розшифрування пройшов успішно, "вірить", що обмін інформацією триває нормально. При спробі сторони *B* відповісти на лист від *A*, зловмисник виконує аналогічну процедуру.

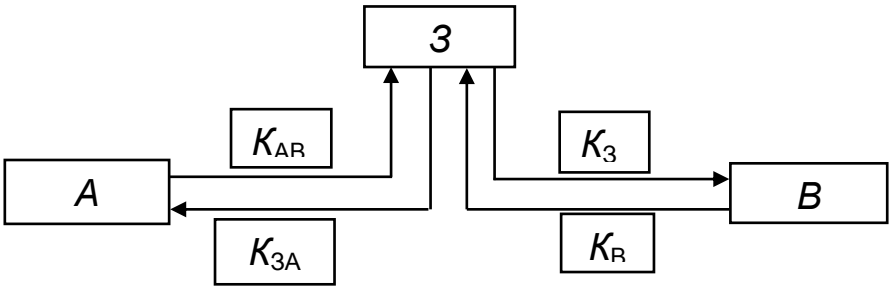


Рис. 8.1. **Схема атаки *Man-in-Middle* при обміні криптографічними ключами**

Таким чином, зловмисник читає усю інформацію, що курсує між сторонами *A* та *B*, оскільки володіє усіма чотирма криптографічними ключами, і продовжуватися це може доти, поки сторони *A* та *B* фізично не зустрінуться та зрозуміють, що оперують незнайомими криптографічними ключами, або *З* не використає проти них перехоплену інформацію.

Посередництво – одна з найнебезпечніших атак проти систем зв'язку. Захист від нього потребував створення потужної системи – інфраструктури відкритих ключів та застосування геш-функцій. Успішний захист від посередництва полягає у обчисленні геш-образу повідомлень та підписуванні цього образу електронним цифровим підписом. У такому разі отримувач має змогу перевірити цілісність інформації порівнянням геш-функцій та власника цифрового підпису за допомогою сертифіката центра PKI.

Розглянемо деякі інші специфічні типи атак на криптосистему RSA.

Атака на близькі значення p і q. Однією з найвідоміших атак на криптосистему RSA є така, що використовує близькі значення *p* та *q*. Припустимо, що $p > q$, хоча це ніяк не обмежує подальші міркування. Тоді можна записати: $x = (p + q)/2$, $y = (p - q)/2$, а для *n* справедливе рівняння:

$$n = x^2 - y^2.$$

Для знаходження множників *p* та *q* досить підібрати числа *x* та *y*, що задовольняють (*). Знайдемо $x_0 = \sqrt{n}$ та оберемо найближче до нього більше ціле x_1. Підставляючи його у (*), знаходимо відповідне

значення y. Повторюємо цю операцію доти, поки не отримаємо цілі значення x та y, що задовольняють (*). У результаті знаходимо $p = x + y$, $q = x - y$.

Розглянемо ***приклад***. Нехай $n = p \times q = 851$.

Скористаємося описаним правилом для знаходження p та q. Оскільки $\sqrt{n} = 29.17$, то припустивши, що $x = 30$, знаходимо $y^2 = 30^2 - 851 = 49$. Отже, $y = 7$, з першої спроби знаходимо розв'язок (*): $x = 30$, $y = 7$. Отже, $p = 30 + 7 = 37$, а $q = 30 - 7 = 23$. Отимавши значення p та q, зловмисник може легко обчислити приватний ключ, зламавши таким чином криптосистему.

Атака з вибраним шифротекстом. Нехай перехоплено повідомлення $C = E_e(M)$, зашифроване на публічному ключі алгоритму RSA.

Обираємо число $r < n$. Тоді можна обчислити таке:

$x = r^e \bmod n;$

$y = xC \bmod n;$

$t = r^{-1} \bmod n.$

Якщо $x = r^e \bmod n$, тоді $r = x^d \bmod n$.

Тепер просимо власника ключа підписати у приватним ключем d. Будемо мати $u = y^d \bmod n$.

Обчислимо $tu \bmod n = r^{-1}y^d \bmod n = r^{-1}x^d C^d \bmod n = C^d \bmod n = M$.

Таким чином, отримано розшифроване повідомлення M.

Захист від такої атаки може здійснюватися двома шляхами. Перший полягає в тому, що пари ключів для шифрування та електронного підпису повинні бути різними. Другий – підписувати завжди потрібно гешобраз повідомлення, а не власне повідомлення.

Атака на електронний підпис. Якщо зловмиснику необхідно отримати електронний підпис повідомлення M, він може діяти таким чином. Створюються повідомлення M_1 та M_2 такі, що $M = M_1M_2 \bmod n$, та підписуються окремо повідомлення M_1 та M_2.

Якщо є ЕЦП повідомлень M_1 та M_2, тоді

$M_3^d = (M_1^d \bmod n)\ (M_2^d \bmod n)$. Отже, повідомлення M підписано.

Атака на спільний модуль. При реалізації RSA іноді буває вигідним роздати усім користувачам спільний модуль n та різні ключі e та d. Однак така схема працює доти, доки два користувача не зашифрують на різних ключах однакове повідомлення. Якщо e_1 та e_2 – взаємно прості числа, тоді можлива така атака.

Нехай m – повідомлення, e_1 та e_2 – два публічних ключа, n – спільний модуль. Процес зашифрування можна зобразити рівняннями:

$$C_1 = m^{e_1} \bmod n; C_2 = m^{e_2} \bmod n.$$

Отже, криптоаналітик знає n, e_1, e_2, C_1, C_2. За допомогою алгоритму Евкліда він знаходить такі r та s, які задовольняють рівняння:

$se_1 + re_2 = 1$ ($r < 0$; $s > 0$) та обчислює $C_1^{-1} \bmod n$.

Тоді $(C_1^{-1})^{-r} C_2^s \bmod n = m$.

Доведемо це, підставивши замість C_1 та C_2 їх значення:

$$((m^{e_1})^{-1})^{r} m^{e_2 s} \bmod n = m^{e_1 r} m^{e_2 s} \bmod n = m^{e_1 r + e_2 s} \bmod n = m.$$

Зважаючи на таку атаку, задавати спільний модуль для групи користувачів дуже небезпечно.

Атака дешифрування ітераціями. Суть методу полягає в тому, що перехоплену криптограму повторно шифрують на публічному ключі і при деякій кількості ітерацій отримують вихідне повідомлення. Кількість ітерацій, звичайно, залежить від довжини модуля та ключа. За невеликої довжини модуля цього можна досягти вже при кількох ітераціях. Розглянемо ***приклад***. Нехай p = 3, q = 11; n = 33; e = 7, d = 3; повідомлення m = "02". Обчислення криптограми дає: $C = 2^7 \bmod 33 = 29$. Спробуємо перешифрувати криптограму на публічному ключі (7,33):

перша ітерація – $C_1 = 29^7 \bmod 33 = 17$;

друга ітерація – $C_2 = 17^7 \bmod 33 = 8$;

третя ітерація – $C_3 = 8^7 \bmod 33 = 2 = m$.

Отже, як бачимо, вже третя ітерація дозволила отримати відкритий текст. На практиці це повинно було б виглядати таким чином. Перехоплена шифровка перешифровується на публічному ключі доти, поки на виході не отримується зв'язний текст. Зрозуміло, що це обов'язково станеться, однак час, потрібний для цього, не повинен бути меншим за час для повного перебору ключів чи розкладання модуля на множники.

8.3. Диференціальний криптоаналіз

Поняття диференціального криптоаналізу було введено Елі Біхамом (A. Biham) і Аді Шаміром (A. Shamir) в 1990 році. Диференціальний криптоаналіз використовують для атак на симетричні криптосистеми. Кінцеве завдання диференціального криптоаналізу – використовуючи властивості алгоритму, в основному властивості S-блоків, визначити

підключ раунду. Конкретний спосіб диференціального криптоаналізу залежить від алгоритму шифрування.

Якщо в основі алгоритму лежить сітка Фейстеля, то можна вважати, що блок m складається з двох половинок – m_0 і m_1. Диференціальний криптоаналіз розглядає відмінності, які відбуваються в кожній половині при шифруванні. Наприклад, для алгоритму DES "відмінності" визначаються за допомогою операції XOR, для інших алгоритмів можливий інший спосіб. Обирається пара незашифрованих текстів з фіксованою відмінністю. Потім аналізуються відмінності, що вийшли після шифрування одним раундом алгоритму, і визначаються ймовірності різних ключів. Якщо для багатьох пар вхідних значень, що мають ту саму відмінність X, при використанні того самого підключа (Y) однаковими виявляються і відмінності відповідних вихідних значень, то можна припустити, що в X використано підключ Y з певною ймовірністю. Якщо ця ймовірність близька до одиниці, то можна вважати, що підключ раунду знайдений з даною ймовірністю. Імовірність знаходження загального ключа визначається добутком ймовірностей підключів кожного раунду, оскільки раунди незалежні.

Результати диференціального криптоаналізу використовуються як при розробці конкретних S-блоків, так і при визначенні оптимальної кількості раундів.

Диференціальний криптоаналіз на основі відмов пристрою. У вересні 1996 року група фахівців з компанії Bellcore оголосила про новий метод криптоаналізу, що дозволяє ефективно розкривати секретний ключ, який зберігається в пам’яті портативного шифрувального обладнання, наприклад, Smart Card або PCMCIA. Збереження ключа в таких пристроях забезпечується за рахунок унікальних характеристик технології TEMPEST. Вперше метод був успішно застосований для розкриття ключа криптосистеми RSA. Подальші дослідження нового методу, що одержав назву диференціального криптоаналізу на основі відмов обладнання (Differential Fault Analysis, DFA), продемонстрували його ефективність при атаці на DES й інші блокові шифри. Так, для розкриття ключа DES знадобилося проаналізувати двісті шифротекстів, отриманих шифруванням відкритих текстів, що зберігаються в пам’яті обладнання. Було доведено, що навіть застосування потрійного DES не впливає на складність атаки.

Відомо, що деякі види випромінювання (наприклад, радіоактивне) призводять до відмов електронного устаткування. Саме ця ідея і лягла в основу крипоаналітичної атаки, розробленої криптоаналітиками компанії Bellcore. При цьому передбачається, що криптоаналітик має необмежений доступ до шифрувального обладнання. Відкритий текст і секретний ключ зберігаються в пам'яті обладнання і недоступні. Криптоаналітик штучно викликає відмови в роботі обладнання, піддаючи його опроміненню. Опромінення призводить до інверсії біта (або бітів) в одному з регістрів на деякому проміжному етапі криптографічного перетворення. Наприклад, для блокових шифрів конструкції Фейстеля інверсія виникає на одному із циклів перетворення. Відмова призводить до спотворення шифротексту на виході обладнання. Криптоаналітик намагається розкрити секретний ключ, аналізуючи спотворені і неспотворені шифротексти.

Розглянемо описаний метод на прикладі криптоаналізу DES. Припустимо, що є два різні шифротексти, отримані при шифруванні того самого відкритого тексту на фіксованому ключі. Відомо, який шифротекст отриманий у результаті інверсії одиночного біта в процесі шифрування. На першому етапі атаки необхідно встановити номер циклу перетворення, на якому відбулась інверсія. Припустимо, що інверсія відбулася на останньому, шістнадцятому циклі DES-перетворення в правій половині блоку до заключної перестановки. Звідси зрозуміло, що відмінність лівих половин блоків шифротексту визначається виходами тих S-блоків (одного або двох), на вході яких з'явився інвертований біт. Застосування методу диференціального криптоаналізу дозволяє розкрити шість бітів ключа для кожного такого S-блоку.

Для розкриття підключа останнього циклу перетворення досить проаналізувати менше 200 шифротекстів. Подальший розвиток атаки можливий у двох напрямах. Перший варіант – пошук секретного ключа шляхом вичерпної перевірки $2^8 = 2^{56}$ кандидатів при заданому підключі (нагадаємо, що розрядність підключа – 48 бітів). Альтернативний варіант полягає у використанні знання про підключ, отриманий на останньому циклі для аналізу попередніх циклів. Останній варіант дозволяє успішно атакувати DES у режимі EDE-шифрування на трьох різних ключах. Описаний метод працює й у тих випадках, коли інверсія виникає всередині F-функції або процедури генерації підключів.

Метод диференціального криптоаналізу на основі відмов обладнання можна застосовувати для атаки на такі блокові шифри, як IDEA, RC5 і Feal. Деякі блокові шифри, наприклад, Khufu, Khafre і Blowfish використовують заданий секретний ключ для генерації S-блоків. У цьому випадку описаний метод дозволяє розкривати не тільки ключі, але й власне S-блоки.

Розглянутий криптоаналітичний метод дозволяє ефективно розкривати ключ шифрування навіть тоді, коли сам алгоритм криптографічного перетворення невідомий. Так, наприклад, деталі криптоалгоритму Fortezza становлять державну таємницю і засекречені Агентством національної безпеки США. Однак апаратна реалізація криптоалгоритму у вигляді мікросхеми Clipper входить до складу багатьох комерційних обладнань шифрування, наприклад, Fortezza PCMCIA. Більше того, можливе відновлення деталей невідомого алгоритму.

8.4. Лінійний криптоаналіз

Метод лінійного криптоаналізу вперше був застосований для атаки блокового шифру FEAL, а пізніше DES. Метод використовує лінійні наближення. Це означає, що якщо виконати операцію XOR над деякими бітами відкритого тексту, потім над деякими бітами шифротексту, а потім виконати XOR результатів попереднього сумування, можна одержати біт, який буде сумою кількох бітів ключа. Це і є лінійним наближенням, яке може бути правильним з деякою ймовірністю p. Якщо $p \neq 1/2$, то цей факт можна використовувати для розкриття бітів ключа. Розглянемо цей метод.

У загальному вигляді лінійний криптоаналіз матиме вигляд:

$$\sum_{i=1}^{a} P_i \oplus \sum_{j=1}^{b} C_j = \sum_{l=1}^{c} K_l ,$$

де $i_1,i_2,...,i_a j_1$, ..., j_b та $l_1,l_2,....,l_c$ – позиції деяких бітів відкритого тексту P_i, шифротексту C_j ключа K_l.

Успіх лінійного криптоаналізу сильно залежить від структури S-блоків і виявилося, що S-блоки DES не оптимізовано проти такого

способу розкриття. Стверджують, що стійкість до лінійного криптоаналізу не входила у число критеріїв при проектуванні DES. Причина цього невідома: або розробники не знали про можливості лінійного криптоаналізу, або віддали перевагу стійкості проти диференціального, найбільш небезпечного з їхньої точки зору методу.

Підраховано, що для найкращого лінійного наближення необхідно 2^{47} відомих відкритих блоків, а результатом буде 1 біт ключа. Це дуже мало. Якщо змінити напрям і використовувати для аналізу шифротекст, а не відкритий текст, а також розшифрування замість шифрування, то в результаті лінійного криптоаналізу можна отримати 2 біти ключа. Це все ще недостатньо.

Однак існують деякі тонкощі. Якщо використати лінійний криптоаналіз паралельно 2^{12} разів та обрати правильний варіант, ґрунтуючись на ймовірностях, це дасть 12 бітів ключа. 13 бітів ключа можна отримати, змінивши шифрування на розшифрування, а інших 30 біт – "грубою силою", тобто повним перебиранням.

Розкриття повного 16-раундового DES за такою схемою вимагає 2^{43} відомих відкритих текстів. Програмна реалізація цього методу на 12 робочих станціях HP9735, розкрила ключ DES за 50 днів. Лінійний криптоаналіз молодший за диференціальний, тому дуже імовірний подальший розвиток цих ідей.

Контрольні запитання

1. Основні поняття теорії криптоаналізу.
2. Класифікація атак на симетричні криптоалгоритми.
3. Класифікація атак на асиметричні криптоалгоритми.
4. Основні поняття диференціального криптоаналізу.
5. Основні поняття лінійного криптоаналізу.
6. Сутність атаки дешифрування ітераціями.
7. Сутність атаки на спільний модуль.
8. Сутність атаки на електронний підпис.
9. Сутність атаки на основі обраного тексту (chosen-text attack).
10. Сутність атаки на основі обраного шифротексту (chosen-ciphertext attack).

Розділ 9. Основні напрями розвитку сучасної криптографії

9.1. Нові асиметричні алгоритми на основі еліптичних кривих

Більшість продуктів і стандартів, у яких для шифрування і цифрових підписів застосовується метод криптографії з відкритим ключем, базується на алгоритмі RSA. Кількість бітів ключа, необхідна для надійної захищеності RSA, за останні роки різко зросла, що обумовило відповідне зростання завантаження систем у додатках, що використовують RSA. Це породило множину проблем, особливо для вузлів, що спеціалізуються на електронній комерції, де потрібен захист великої кількості транзакцій. Але є підхід, який може конкурувати з RSA – це криптографія на основі еліптичних кривих (ECC – Elliptic curve cryptography). Уже зараз вживають спроби стандартизації цього підходу, наприклад, як у стандарті IEEE P1363 за криптографією з відкритим ключем [23; 27; 33; 43; 45].

Привабливість підходу на основі еліптичних кривих порівняно з RSA полягає в тому, що з використанням еліптичних кривих забезпечується еквівалентний захист при дуже невеликій кількості розрядів, внаслідок чого зменшується завантаження процесору.

У цьому підрозділі подані основи використання еліптичних кривих і криптографії з їх застосуванням.

9.1.1. Еліптичні криві

Перевага підходу на основі еліптичних кривих порівняно із завданням факторизації числа, яке використовується в RSA, або завданням цілочисельного логарифмування, що застосовується в алгоритмі Діффі – Хеллмана і в DSS, полягає в тому, що в цьому випадку забезпечується еквівалентний захист при меншій довжині ключа [23; 45].

У загальному випадку рівняння еліптичної кривої E має вигляд:

$$y^2 + axy + by = x^3 + cx^2 + dx + e.$$

Як приклад розглянемо еліптичну криву E, рівняння якої має вигляд: $y^2 + y = x^3 – x^2$.

На цій кривій лежать тільки чотири точки, координати яких є цілими числами. Це числа A (0, 0), B (1, –1), З (1, 0) і D (0, –1) (рис. 9.1).

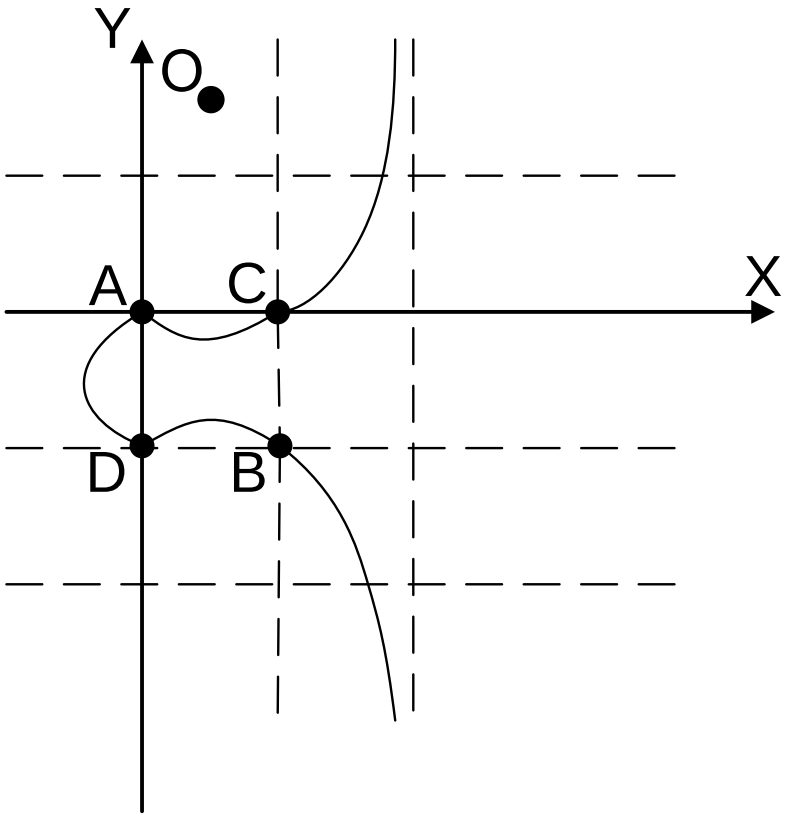


Рис. 9.1. **Приклад еліптичної кривої із чотирма точками**

Для визначення операції додавання для точок на еліптичній кривій зробимо такі припущення (рис. 9.2):

На площині існує нескінченно вилучена точка $0 \in E$, у якій сходяться всі вертикальні прямі.

Будемо вважати, що дотична до кривої перетинає точку торкання два рази.

Якщо три точки еліптичної кривої лежать на прямій лінії, то їх сума дорівнює 0.

Введемо правила додавання точкок на еліптичній кривій:

точка 0 виступає в ролі нульового елемента. Так, $0 = – 0$ і для будь-якої точки P на еліптичній кривій $P + 0 = P$.

Вертикальна лінія перетинає криву у двох точках з однією і тією ж координатою x – скажемо, $S = (x, y)$ і $T = (x, – y)$. Ця пряма перетинає криву і в нескінченно вилученій точці. Тому $P_1 + P_2 + 0 = 0$ і $P_1 = – P_2$.

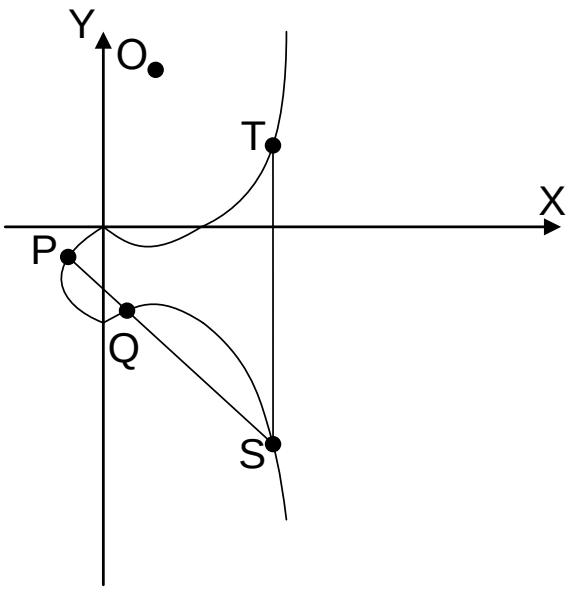


Рис. 9.2. **Додавання точок на еліптичній кривій**

Щоб скласти дві точки P і Q (див. рис. 9.2) з різними координатами x, слід провести через ці точки пряму і знайти точку перетинання її з еліптичною кривою. Якщо пряма не є дотичною до кривої в точках P або Q, то існує тільки одна така точка, позначимо її S. Згідно з припущенням:

$$P + Q + S = O.$$

Отже,

$$P + Q = – S \text{ або } P + Q = T.$$

Якщо пряма є дотичною до кривої в якій-небудь із точкок P або Q, то в цьому випадку слід покласти S = P або S = Q відповідно.

Щоб подвоїти точку Q, слід провести дотичну в точці Q і знайти іншу точку перетинання S з еліптичною кривою. Тоді:

$$Q + Q = 2 \times Q = – S.$$

Введена таким способом операція додавання підкоряється всім звичайним правилам додавання, зокрема комутативному і асоціативному законам. Множення точки P еліптичної кривої на позитивне число k визначається як сума k точкок P.

У криптографії з використанням еліптичних кривих всі значення обчислюються за модулем p, де p є простим числом. Елементами даної еліптичної кривої є пари невід’ємних цілих чисел, які менше p і задовольняють приватному виду еліптичної кривої:

$$y^2 \equiv x^3 + ax + b \pmod{p}.$$

Таку криву будемо позначати Ep (a, b). При цьому числа a і b повинні бути менше p і задовольняти умову:

$$4a^3 + 27b^2 \pmod p \neq 0.$$

Множина точкок на еліптичній кривій обчислюється в такий спосіб.

1. Для кожного такого значення x, що $0 \leq x \leq p$ обчислюється $x^3 + ax + b \pmod p$.

2. Для кожного з отриманих на попередньому кроці значень з'ясовується, чи має це значення квадратний корінь за модулем p. Якщо ні, то в Ep (a,b) немає точкок з цим значенням x. Якщо корінь існує, є два значення y, які відповідні до операції добування квадратного кореня (виключенням є випадок, коли єдиним значенням виявляється y = 0). Ці значення (x, y) і будуть точками E_p (a, b).

Множина точкок E_p (a, b) має такі властивості:

1. P + 0 = P.

2. Якщо P = (x, y), то P + (x,– y) = 0. Точка (x, – y) є від'ємним значенням точки P і означається – P. Зазначимо, що (x,– y) лежить на еліптичній кривій і належить E_p (a, b).

3. Якщо $P = (x_1, y_1)$ і $Q = (x_2, y_2)$, де P ≠ Q, то $P + Q = (x_3, y_3)$ визначається за такими формулами:

$$x_3 \equiv \lambda_2 - x_1 - x_2 \pmod p;$$
$$y_3 \equiv \lambda (x_1 - x_3) - y_1 \pmod p,$$

де $(y_2 - y_1)/(x_2 - x_1)$, якщо P ≠ Q, $\lambda = (3x_{12} + a)/2y_1$, якщо P = Q.

Число λ є кутовим коефіцієнтом січної, яка проведена через точки $P = (x_1, y_1)$ і $Q = (x_2, y_2)$. При P = Q січна перетворюється в дотичну, ніж і пояснюється наявність двох формул для обчислення λ.

Завдання, яке повинен розв'язати в цьому випадку атакуючий, є свого роду завданням "дискретного логарифмування на еліптичній кривій", і формулюється в такий спосіб.

Дані точки P і Q на еліптичній кривій E_p (a,b). Необхідно знайти коефіцієнт k < p такий, що P = k × Q.

Відносно легко обчислити P за даними k і Q, але досить важко обчислити k, знаючи P і Q.

Розглянемо три способи використання еліптичних кривих у криптографії.

Аналог алгоритму Діффі – Хеллмана обміну ключами

Обмін ключами з використанням еліптичних кривих може бути виконаний у такий спосіб. Спочатку обирається просте число $p \approx 2^{180}$ і параметри a і b для рівняння еліптичної кривої. Це задає множину точкок E_p (a,b). Потім в E_p (a,b) обирається генеруюча точка $G = (x_1, y_1)$. При виборі G важливо, щоб найменше значення n, при якому $n \times G = 0$, виявилось набагато більшим простим числом. Параметри E_p (a,b) і G криптосистеми є параметрами, відомими всім учасникам [47].

Обмін ключами між користувачами A і B проводиться за такою схемою:

1. Абонент A вибирає ціле число n_A, менше n. Це число є закритим ключем абонента A. Потім абонент A обчислює відкритий ключ $P_A = n_A \times G$, який становить деяку точку на E_p (a,b).

2. Таким же чином абонент B обирає закритий ключ n_B і обчислює відкритий ключ P_B.

3. Абоненти обмінюються відкритими ключами, після чого обчислюють загальний секретний ключ K.

Абонент A: $K = n_A \times P_B$.

Абонент B: $K = n_B \times P_A$.

Слід зазначити, що загальний секретний ключ становить пару чисел. Якщо цей ключ передбачається використовувати в якості сеансового ключа для алгоритму симетричного шифрування, то із цієї пари необхідно створити одне значення.

Шифрування/розшифрування з використанням еліптичних кривих

Розглянемо найпростіший підхід до шифрування/розшифрування з використанням еліптичних кривих. Завдання полягає в тому, щоб зашифрувати повідомлення M, яке може бути презентовано у вигляді точки на еліптичній кривій P_m (x,y).

Як і у випадку обміну ключем, у системі шифрування/розшифрування в якості параметрів розглядається еліптична крива E_p (a,b) і точка G на ній. Абонент B обирає закритий ключ n_B і обчислює відкритий ключ $P_B = n_B \times G$. Щоб зашифрувати повідомлення P_m, використовується від-критий ключ одержувача B P_B. Абонент A вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій: $C_m = \{k \times G, P_m + k \times P_B\}$.

Щоб розшифрувати повідомлення, учасник B множить першу координату точки на свій закритий ключ і віднімає результат із другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m.$$

Абонент A зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Противнику для відновлення повідомлення доведеться обчислити k, знаючи G і $k \times G$. Зробити це буде нелегко.

Одержувач також не знає k, але йому в якості підмови посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач отримає значення, яке було додане відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

9.2. Математичні моделі нелінійних вузлів замін у термінах булевої алгебри

Основним і найбільш важливим елементом сучасних симетричних криптографічних засобів захисту інформації є нелінійні вузли замін (блоки ускладнення, нелінійні блоки підстановок), криптографічні властивості яких безпосередньо впливають на ефективність розроблювальних механізмів забезпечення безпеки інформаційних технологій [21; 23; 33; 45; 50 – 52]. Перспективним напрямком у їхньому розвитку є підхід, що базується на використанні розвиненого математичного апарату булевої алгебри [21; 23] і, зокрема, методів формування криптографічних булевих функцій [23; 43; 48]. Він дозволяє використовувати єдиний підхід конструювання та оцінки ефективності нелінійних вузлів замін як блокових, так і поточних симетричних засобів захисту інформації. Крім того, такі показники як нелінійність, кореляційний імунітет, степінь критерію розповсюдження і автокореляція послідовності булевої функції дозволяють адекватно оцінювати стійкість нелінійного вузла заміни до диференціальних, лінійним та іншим методам криптографічного аналізу [40; 41].

Основна відмінність між блоковими і потоковими методами криптографічного перетворення (рис. 9.3.) полягає в такому: блочні методи застосовують одне постійне перетворення до фіксованих блокам даних відкритого тексту [40; 41]; потокові методи застосовують змінюються

в часі перетворення до окремих символів відкритого тексту. Як показує аналіз відкритої літератури та результати проведених криптографічних конкурсів [23; 60; 61], на сьогоднішній день реалізовано велику кількість різних методів – як поточних, так і блокових: DES, ORYX, RC2, RC4, RC5 (США), ГОСТ 28147-89 (Росія); Rijndael (Бельгія); SEAL, B-Crypt (Великобританія); A5, Camellia, Snow (Європа) і багато інших. При цьому переважна більшість блокових методів перетворення інформації поділяється на дві великі групи методів:

методи, побудовані на основі використання ланцюгів Фейстеля (DES, DEAL, E2, LOKI97, RC6, Twofish, MARS);

методи, побудовані на основі чергування процедур перестановок і підстановок, так званих SPN – конструкцій (Square, Rijndael, SAFER, Serpent, CRYPTON).

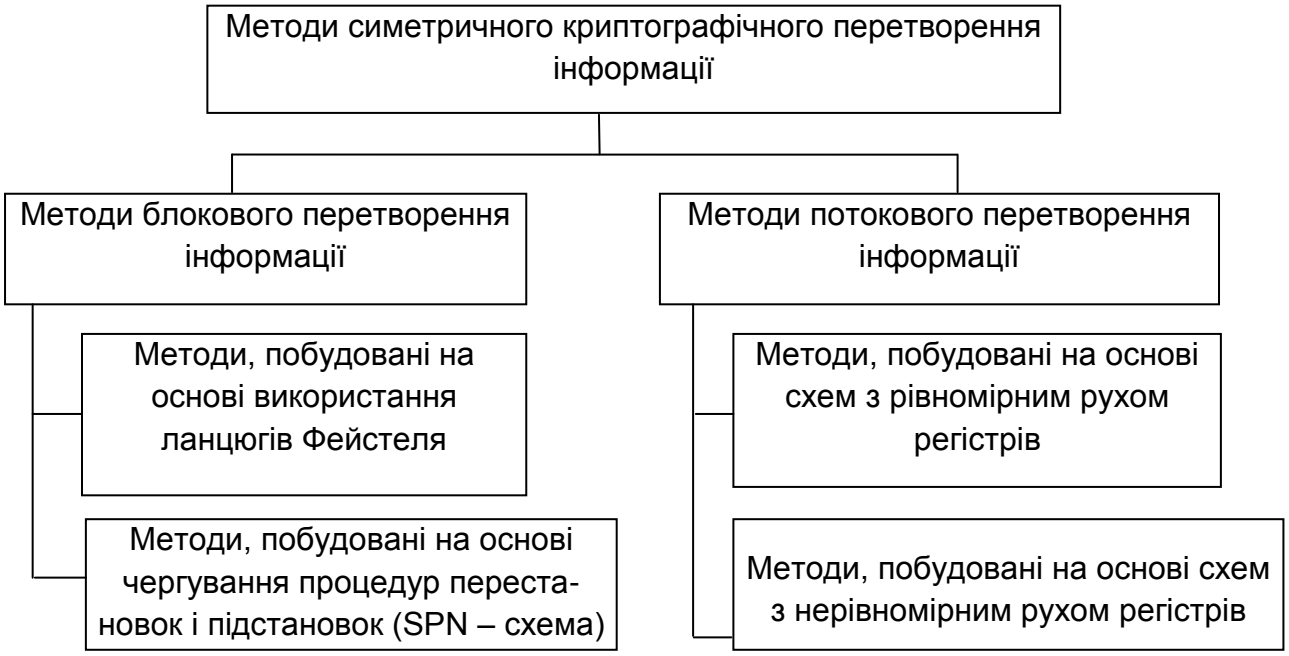


Рис. 9.3. **Класифікація методів симетричного криптографічного перетворення інформації**

Характерною особливістю обох конструкцій блочного криптографічного перетворення інформації є використання в якості шифруючих функцій операцій нелінійного перемішування (нелінійних S-блоків, або нелінійних вузлів заміни). Зазначимо, що в якості шифруючих функцій можуть також додатково використовуватися операції розсіювання (бітові P-перестановки) і операції введення секретності (складання з ключем), однак нелінійні вузли замін є вузлами, визначальними стійкість перетворення інформації і їх використання характерно як для методів, що вико-

ристовують ланцюги Фейстеля, так і для методів, що використовують SPN-структури. В цілому можна констатувати [23; 40; 41; 48; 50 – 52; 61], що стійкість методів блочного криптографічного перетворення інформації базується, насамперед, на стійкості нелінійних вузлів замін.

Стійкість нелінійних вузлів замін, які здійснюють незворотні/важкозворотні нелінійні перетворення, визначають стійкість використовуваних методів в цілому. Тому першочергове значення для підвищення ефективності симетричних криптографічних засобів захисту інформації набуває розробка математичних моделей і обчислювальних методів формування нелінійних вузлів замін з покращеними властивостями.

Подана проблема не є характерною для методів потокового перетворення інформації. Ці методи засновані на використанні як нелінійних вузлів замін булевих функцій, методи формування яких широко обговорюються у відкритій пресі. З точки зору авторів, такий стан – відкритість методів формування нелінійних вузлів замін для методів потокового перетворення інформації і закритість методів формування нелінійних вузлів замін для методів блочного перетворення інформації – можна пояснити двома причинами. Перша причина полягає в тому, що до недавнього часу прерогатива розробки методів потокового перетворення інформації належала виключно європейським розробникам, у яких грань між закритими і відкритими дослідженнями є дуже тонкою, внаслідок чого багато розробки в області захисту інформації є загальнодоступними, а використовуваний математичний апарат широко відомий і повсюдно використовуваний. Друга ж причина полягає в тому, що методи блокового перетворення інформації розроблялися переважно у США в стінах науково-дослідних підрозділів спецслужб, що саме по собі виключало можливість проникнення розробок у відкриту пресу. Можна також припустити, що оскільки задача формування нелінійних вузлів замін для блокових методів значно складніше завдання побудови нелінійних булевих функцій, то досягнення в галузі побудови таких вузлів є чинниками, що визначають степінь захищеності інформаційних ресурсів корпорації/держави. Отже, володіння такими методами є одним з факторів, що визначають степінь успіху/могутності корпорації/країни в цілому, що позиціонує ці методи як стратегічно важливий інтелектуальний продукт, що не підлягає розголошенню.

Для потокових методів перетворення інформації ефективність вузла заміни оцінюється ефективністю безпосередньо нелінійної булевої

функції і не викликає проблем. Для блокових методів перетворення інформації ефективність вузла заміни може оцінюватися двома рівноцінними способами:

1. На основі оцінки ефективності протистояння вузла заміни криптоаналітичних атакам у термінах диференційних характеристик (ДХ) [23; 40; 41; 48; 50 – 52; 61]. Використовуються такі критерії, як граничне (максимальне) значення ймовірності r-циклової ДХ, ітеративна ДХ і т. д.

2. На основі оцінки ефективності протистояння вузла заміни криптоаналітичних атакам у термінах булевих функцій [23; 40; 41; 48; 50 – 52; 61] використовуються такі критерії, як нелінійність функції, її збалансованість і т. д.

Нелінійні вузли замін блокових симетричних криптографічних засобів захисту інформації подаються у вигляді обладнання перетворення, реалізованого за допомогою двох комутаторів (рис. 9.3). При цьому один комутатор перетворює набір з n бітів $(a_1,a_2,...,a_n)$ в одну цифру за підставою 2^n, інший комутатор виконує зворотне перетворення, тобто утворює набір з n бітів $(b_1,b_2,...,b_n)$. Таке обладнання потенційно може замінити будь-який вхідний набір даних $(a_1,a_2,...,a_n)$ на будь-який вихідний набір $(b_1,b_2,...,b_n)$**.** Нелінійний вузол заміни (S-блок), схема перетворення даних якого подана на рис. 9.4, містить 2^n внутрішніх станів комутаторів, які можуть бути виконані $2^n!$ різними способами. Це означає, що існує $2^n!$ різних варіантів відповідних нелінійних вузлів (таблиць) замін.

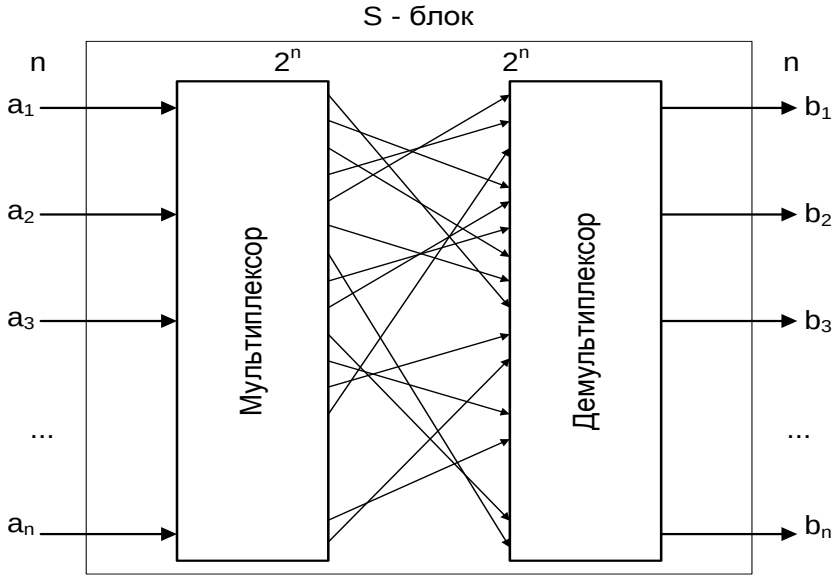


Рис. 9.4. **Схема перетворення даних у нелінійному вузлі заміни**

Формалізуємо процес перетворення даних у нелінійному вузлі замін за допомогою математичної моделі, що дозволяє описати його внутрішню структуру і адекватно оцінити основні показники його ефективності.

Як видно з рис. 9.4, процес перетворення даних у нелінійному вузлі замін у загальному випадку можна представити у вигляді деякого відображення:

$$\varphi : A \to B, \qquad (9.1)$$

де A – множина можливих наборів $(a_1,a_2,...,a_n)$, що надходять на вхід S-блоку;

B – множина можливих наборів $(b_1,b_2,...,b_n)$, що надходять на вихід S-блоку.

Властивості відображення (9.1) задаються внутрішньою структурою вузла заміни, тобто набором конкретних комутацій (з'єднань) виходів мультиплексора і входів демультиплексора.

Для формального аналітичного опису внутрішньої структури й оцінки властивостей нелінійного вузла замін пропонується математична модель як абстрактна сукупність наступних елементів (математичної символіки):

1. Множина вхідних векторів:

$$A = \{A_1, A_2, ..., A_i, ..., A_{2^m}\}, A_i = \left(a_1^{(i)}, a_2^{(i)}, ..., a_m^{(i)}\right) \in A, a_j^{(i)} \in GF(2), i = 1,...,2^m,$$
$$j = 1,...,m, \ |A| = 2^m.$$

2. Множина вихідних векторів:

$$B = \{B_1, B_2, ..., B_i, ..., B_{2^n}\}, B_i = \left(B_1^{(i)}, B_2^{(i)}, ..., B_n^{(i)}\right) \in B, B_j^{(i)} \in GF(2), i = 1,...,2^m,$$
$$j = 1,...,m, \ |B| = 2^n.$$

3. Множина відображень $\Phi = \{\varphi_1, \varphi_2, ..., \varphi_i, ..., \varphi_{2^m!}\}$ вхідних векторів з множини A в множину вихідних векторів B, тобто:

$$\varphi_i : A \to B, \varphi_i \in \Phi, i = 1,...,2^m!, \ |\Phi| = 2^m!.$$

4. Кожне відображення $\varphi_i : A \to B$ з множини Φ параметризується сукупністю компонентних криптографічних булевих функцій:

$$F = \{f_1(x_1,\dots,x_m), f_2(x_1,\dots,x_m),\dots, f_u(x_1,\dots,x_m),\dots, f_n(x_1,\dots,x_m)\}, u = 1,\dots,n, |F| = n.$$

Кожний елемент множини запишемо в алгебраїчній нормальній формі:

$$\begin{cases} f_1(x_1,\dots,x_m) = c_{1,0} \oplus \bigoplus_{i=1}^{m} c_{1,i}x_i \oplus \bigoplus_{1\le i<j\le m} c_{1,ij}x_ix_j \oplus \dots \oplus c_{1,12\dots m}x_1x_2\dots x_m; \\ f_2(x_1,\dots,x_m) = c_{2,0} \oplus \bigoplus_{i=1}^{m} c_{2,i}x_i \oplus \bigoplus_{1\le i<j\le m} c_{2,ij}x_ix_j \oplus \dots \oplus c_{2,12\dots m}x_1x_2\dots x_m; \\ \dots \\ f_u(x_1,\dots,x_m) = c_{u,0} \oplus \bigoplus_{i=1}^{m} c_{u,i}x_i \oplus \bigoplus_{1\le i<j\le m} c_{u,ij}x_ix_j \oplus \dots \oplus c_{u,12\dots m}x_1x_2\dots x_m; \\ \dots \\ f_n(x_1,\dots,x_m) = c_{n,0} \oplus \bigoplus_{i=1}^{m} c_{n,i}x_i \oplus \bigoplus_{1\le i<j\le m} c_{n,ij}x_ix_j \oplus \dots \oplus c_{n,12\dots m}x_1x_2\dots x_m. \end{cases}, \quad (9.2)$$

5. Система обмежень:

$$\left\{ \begin{array}{c} Сб_f = Сб_{f_1} \wedge Сб_{f_2} \wedge \dots \wedge Сб_{f_n} = Сб_{тр} \\ N_f = \min\{N_{f_1}, N_{f_2}, \dots, N_{f_n}\} \ge N_{тр} \\ КІ_f(k) = \min\{КІ_{f_1}(k), КІ_{f_2}(k), \dots, КІ_{f_n}(k)\} \ge КІ_{тр}(k) \\ КП_f(k) = \min\{КП_{f_1}(k), КП_{f_2}(k), \dots, КП_{f_n}(k)\} \ge КП_{тр}(k) \\ AC_f = \max\{AC_{f_1}, AC_{f_2}, \dots, AC_{f_n}\} \le AC_{тр} \end{array} \right\}, \quad (9.3)$$

де $Сб_f, N_f, КІ_f(k), КП_f(k), AC_f$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції, відповідно, оцінені за критерієм мінімального ризику;

$Сб_{f_i}, N_{f_i}, КІ_{f_i}(k), КП_{f_i}(k), AC_{f_i}$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції булевої функції $f_i(x_1,\dots,x_m)$, $i = 1,2,\dots,n$; $Сб_{тр}, N_{тр}, КІ_{тр}(k), КП_{тр}(k), AC_{тр}$ –

необхідні значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції відповідно.

6. Множина компонентних нелінійних булевих функцій $\bar{F} = \{\bar{f}_1(x_1,...,x_m), \bar{f}_2(x_1,...,x_m),...,\bar{f}_u(x_1,...,x_m),...,\bar{f}_{2^n-1}(x_1,...,x_m)\}$ як повна множина функцій, отриманих лінійною комбінацією булевих функцій $f_i(x_1,...,x_m)$ з множини F, $u = 1,...,2^n$, $|\bar{F}| = n$:

$$\begin{cases} \bar{f}_1(x_1,...,x_m) = f_1(x_1,x_2,...,x_m); \\ \bar{f}_2(x_1,...,x_m) = f_2(x_1,x_2,...,x_m); \\ ... \\ \bar{f}_n(x_1,...,x_m) = f_n(x_1,x_2,...,x_m); \\ \bar{f}_{n+1}(x_1,...,x_m) = f_1(x_1,x_2,...,x_m) \oplus f_2(x_1,x_2,...,x_m); \\ ... \\ \bar{f}_{2^n-1}(x_1,...,x_m) = f_1(x_1,x_2,...,x_m) \oplus f_2(x_1,x_2,...,x_m) \oplus ... \oplus f_n(x_1,x_2,...,x_m). \end{cases} \quad (9.4)$$

7. Система обмежень:

$$\left\{ \begin{array}{c} Сб^*_f = Сб_{\bar{f}_1} \wedge Сб_{\bar{f}_2} \wedge ... \wedge Сб_{\bar{f}_{2^n-1}} = Сб^*_{тр} \\ N^*_f = \min\{N_{\bar{f}_1}, N_{\bar{f}_2}, ..., N_{\bar{f}_{2^n-1}}\} \geq N^*_{тр} \\ КІ^*_f(k) = \min\{КІ_{\bar{f}_1}(k), КІ_{\bar{f}_2}(k), ..., КІ_{\bar{f}_{2^n-1}}(k)\} \geq КІ^*_{тр}(k) \\ КП^*_f(k) = \min\{КП_{\bar{f}_1}(k), КП_{\bar{f}_2}(k), ..., КП_{\bar{f}_{2^n-1}}(k)\} \geq КП^*_{тр}(k) \\ АС^*_f = \max\{АС_{\bar{f}_1}, АС_{\bar{f}_2}, ..., АС_{\bar{f}_{2^n-1}}\} \leq АС^*_{тр} \end{array} \right\}, \quad (9.5)$$

де $Сб^*_f, N^*_f, КІ^*_f(k), КП^*_f(k)$, $АС^*_f$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції компонентних булевих функцій, відповідно, оцінені за критерієм мінімального ризику;

$Сб_{\bar{f}_i}, N_{\bar{f}_i}, КІ_{\bar{f}_i}(k), КП_{\bar{f}_i}(k)$, $АС_{\bar{f}_i}$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції компонентної булевої функції $\bar{f}_l(x_1,...,x_m)$, $i = 1,2,...,2^n - 1$;

$Сб^*_{тр}, N^*_{тр}, КІ^*_{тр}(k), КП^*_{тр}(k)$, $АС^*_{тр}$ – необхідні значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції до компонентних булевих функцій відповідно.

Усі функції з множини F задовольняють систему обмежень (9.1). Вузли замін, відповідно до запропонованої моделі, задаються значеннями функцій $f_u(x_1,...,x_m)$, $u = 1,...,n$ з множини F, тобто значення на виході S-блоку задаються сукупністю рівностей:

$$\begin{cases} b_1^{(l)} = f_1\left(a_1^{(l)},...,a_m^{(l)}\right) = c_{1,0} \oplus \bigoplus_{i=1}^{m} c_{1,i}a_i^{(l)} \oplus \bigoplus_{1\le i<j\le m} c_{1,ij}a_i^{(l)}a_j^{(l)} \oplus ... \oplus c_{1,12..m}a_1^{(l)}a_2^{(l)}...a_m^{(l)}; \\ b_2^{(l)} = f_2\left(a_1^{(l)},...,a_m^{(l)}\right) = c_{2,0} \oplus \bigoplus_{i=1}^{m} c_{2,i}a_i^{(l)} \oplus \bigoplus_{1\le i<j\le m} c_{2,ij}a_i^{(l)}a_j^{(l)} \oplus ... \oplus c_{2,12..m}a_1^{(l)}a_2^{(l)}...a_m^{(l)}; \\ ... \\ b_u^{(l)} = f_u\left(a_1^{(l)},...,a_m^{(l)}\right) = c_{u,0} \oplus \bigoplus_{i=1}^{m} c_{u,i}a_i^{(l)} \oplus \bigoplus_{1\le i<j\le m} c_{u,ij}a_i^{(l)}a_j^{(l)} \oplus ... \oplus c_{u,12..m}a_1^{(l)}a_2^{(l)}...a_m^{(l)}; \\ ... \\ b_n^{(l)} = f_n\left(a_1^{(l)},...,a_m^{(l)}\right) = c_{n,0} \oplus \bigoplus_{i=1}^{m} c_{n,i}a_i^{(l)} \oplus \bigoplus_{1\le i<j\le m} c_{n,ij}a_i^{(l)}a_j^{(l)} \oplus ... \oplus c_{n,12..m}a_1^{(l)}a_2^{(l)}...a_m^{(l)}. \end{cases}$$

Будь-яка функція:

$$\bar{f}_u(x_1,...,x_m) \in \bar{F} = \left\{\bar{f}_1(x_1,...,x_m), \bar{f}_2(x_1,...,x_m), ..., \bar{f}_{2^n-1}(x_1,...,x_m)\right\},$$
$$u = 1,2,...,2^n - 1,$$

отримана лінійною комбінацією функцій:

$$\bar{f}_l(x_1,x_2,...,x_m) = f_i(x_1,x_2,...,x_m) \oplus f_j(x_1,x_2,...,x_m) \oplus ... \oplus f_s(x_1,x_2,...,x_m),$$
$$f_i(x_1,x_2,...,x_m), f_j(x_1,x_2,...,x_m), ..., f_s(x_1,x_2,...,x_m) \in F,$$

задовольняє систему обмежень (9.1) із граничними значеннями $Сб^*_{тр}, N^*_{тр}, КІ^*_{тр}(k), КП^*_{тр}(k)$, $АС^*_{тр}$.

Таким чином, математична модель нелінійних вузлів замін блокових симетричних засобів захисту інформації на основі аналітичного опису основних структурних компонентів системи, що накладає обмеження з нелінійності, збалансованості, кореляційного імунітету, критерію поширення і автокореляції, дозволяє термінами булевої алгебри описувати внутрішню структуру нелінійних вузлів замін і оцінювати основні показники їх ефективності.

9.3. Проблеми генерування випадкових та псевдовипадкових послідовностей

9.3.1. Вимоги до генераторів випадкових та псевдовипадкових послідовностей

Генерування випадкових та псевдовипадкових послідовностей – це необхідний і надзвичайно важливий елемент криптографічних застосувань. Більше того, від якості генераторів, апаратних чи програмних, багато в чому залежить якість самої криптосистеми. Неякісний генератор випадкових послідовностей може піддаватися атакам зловмисників, бути причиною утворення слабких криптографічних ключів. Тому важливість якісного генерування криптографічно стійких послідовностей важко переоцінити.

Криптографічні послідовності використовують в разі:

генерування ключів симетричних та асиметричних криптосистем;

генерування цифрових підписів;

реалізації переважної кількості криптографічних протоколів;

автентифікації, що ґрунтується на криптографічних засобах;

потокового шифрування

інших криптографічних застосувань.

Послідовності можна розділити на 2 класи: ***істинно випадкові*** (далі будемо називати їх ***випадковими***) та ***псевдовипадкові***.

Випадкові послідовності, як правило, породжуються апаратними або комбінованими, програмно-апаратними засобами, які використовують в якості генеруючого пристрою датчики фізичних величин. Дані, які постачають датчики фізичних величин, обробляються або електронними засобами, або програмними рішеннями, які, власне, і виробляють двійкові послідовності.

Псевдовипадкові послідовності, як правило, генеруються програмним чином.

Відмінності між істинно випадковими та псевдовипадковими послідовностями дуже значні. Основна відмінність, найважливіша, полягає в тому, що псевдовипадкова послідовність – періодична. Іншою важливою відмінністю є те, що за однакових початкових умов псевдовипадкова послідовність буде однаковою.

Два ці недоліки псевдовипадкових послідовностей необхідно враховувати при проектуванні криптографічних систем, що використовують такі генератори.

Проектування криптографічно стійких псевдовипадкових послідовностей – дуже важлива криптографічна задача, що стала вже цілою індустрією. Прийнято вважати [23; 33; 43; 45], що стійкий псевдовипадковий генератор, який можна сміливо використовувати у найскладніших криптографічних застосуваннях, повинен мати період $\geq 2^{256}$.

Генератори випадкових послідовностей. На ринку немає дешевих апаратних засобів генерування справді випадкових послідовностей, що використовують результати вимірювання певної фізичної величини. Як правило, для побудови таких генераторів можна використати датчик практично довільної фізичної величини, що вимірює її значення з великою точністю:

температури навколишнього середовища;

рівня радіоактивності;

рівня сейсмічної активності;

рівня освітленості;

опору еталонного резистора;

інших фізичних величин.

Схема обробки опитує датчик з деякою періодичністю та використовує отримані дані для формування двійкової випадкової послідовності.

У подальшому сигнал може використовувати програмна (або апаратна) реалізація криптографічного пристрою для різних використань.

Перевагами фізичних генераторів випадкових послідовностей можна вважати таке:

1) відсутність періодичності послідовності, що генерується;

2) неможливість отримання однакових послідовностей навіть при однакових зовнішніх умовах.

Отже, проектування та реалізація генератора випадкових послідовностей такої схеми – складне і нетривіальне інженерне завдання, причому найскладнішою частиною можна вважати саме датчик фізичної величини, який повинен вимірювати цю величину з максимально можливою точністю.

Однак в якості датчика можна використати будь-який комп'ютер.

Непоганими схемами можна вважати генерування випадкових послідовностей з використання клавіатури або мишки [23; 43].

Перша схема працює таким чином. Програмне забезпечення "просить" користувача набирати на клавіатурі будь-яку інформацію (краще беззмістовний набір символів) на деякий проміжок часу. ПЗ аналізує дії користувача і може використовувати їх, як мінімум, у два різних способи. Перший спосіб полягає в тому, що для генерування випадкової послідовності використовуються проміжки часу, виміряні або у "тіках" процесора, або у мілісекундах між послідовними натисканнями на клавіші. Цей час у двійковому представленні обробляється певним чином, наприклад, виконується додавання усіх розрядів за модулем 2 і на вихід подається черговий біт послідовності. При такому використанні не має значення, що саме набирає користувач на клавіатурі. Другий спосіб обробки отриманої інформації полягає у тому, що ПЗ використовує саме символи, які набирає користувач. У цьому разі буде краще, якщо користувач набиратиме невпорядкований набір символів. Двійкове представлення чергового символу піддається деякій обробці (у найпростішому випадку це може бути та сама сума усіх розрядів за модулем 2, чи вибірка молодших або старших розрядів цього представлення з наступним їх додаванням за модулем 2 тощо) та на вихід подається біт (або кілька бітів) випадкової двійкової послідовності.

Мишку можна використати у такий спосіб. При необхідності генерування випадкової послідовності, наприклад, для створення криптографічного ключа, на екран комп'ютера виводиться форма, на якій у випадковому порядку з'являється об'єкт. Користувача просять клацнути мишкою на цьому об'єкті. ПЗ вимірює проміжки часу між послідовними клацаннями мишкою та, обробляючи двійкове представлення цього проміжку, генерує черговий біт випадкової послідовності. Після генерування потрібної кількості двійкових знаків форма зникає.

Перевагами таких способів генерування двійкових послідовностей є відносна простота їх реалізації. Для цього окрім комп'ютера, програмного

забезпечення та користувача нічого не потрібно. Водночас якість таких послідовностей практично ідентична фізичним генераторам випадкових послідовностей.

До недоліків таких способів можна віднести неможливість (або, в крайньому, разі практичну складність) отримання безмежних послідовностей для потокового шифрування, хоча для інших цілей обидва способи достатньо зручні.

Програмні генератори псевдовипадкових послідовностей. На перший погляд в якості псевдовипадкової послідовності можна використати таку, яка генерується програмним забезпеченням для програмування на зразок вбудованих у мови програмування генераторів. Однак такі послідовності не будуть криптографічно стійкими, тобто вони не можуть бути використані у криптографії. Взагалі кажучи, вони взагалі не вважаються випадковими, оскільки період їх дуже малий. Наприклад, у популярній мові програмування C++ функція RAND() генерує "випадкові" значення за формулою:

$$N_{i+1} \equiv (N_i * 1103515245 + 12345) \bmod 4294967296.$$

Окрім цього, що ще важливіше, вони на різних комп'ютерах при однакових вхідних умовах обов'язково дають однакові послідовності. Це робить їх абсолютно непридатними для криптографічних застосувань, адже зловмисник може запустити цей генератор на своєму комп'ютері, змоделювавши ті ж самі вхідні умови, та отримати таку ж послідовність, зламавши усю криптосистему.

Таким чином, для подолання цього недоліку, необхідно накласти додаткові умови на генератори псевдовипадкових послідовностей, що використовуються у криптографічних застосуваннях.

По-перше, необхідно, щоб псевдовипадкові послідовності якомога менше відрізнялися від справді випадкових. Це означає таке. Нехай є дві бінарні послідовності, X_n та Y_n, а також A – імовірнісний алгоритм, який, отримавши на вході двійкову послідовність видає на виході один двійковий біт. Нехай $[P(X_n)=1]$ – імовірність того, що, отримавши на вході двійкову послідовність X_n, алгоритм на виході видає 1.

Вважається, що послідовності X_n та Y_n не розрізняються за поліноміальний час, якщо справджується нерівність:

$$[P(X_n)=1] - [P(Y_n)=1] < 1/n^C$$

для довільної сталої *C* за достатньо великих *n*. Це й буде означати, що псевдовипадкова послідовність відрізняється від випадкової незначно.

По-друге, псевдовипадкова послідовність не повинна бути передбачуваною. Це означає, що знання усіх попередніх бітів послідовності не дозволяє передбачити наступний біт з імовірністю >50 %.

9.3.2. Криптографічно стійкі генератори псевдовипадкових послідовностей

Розглянемо основні генератори псевдовипадкових бінарних послідовностей, які вважаються криптографічно стійкими.

1. Генератор BBS. Програмний генератор двійкових послідовностей BBS (назву утворено від перших літер його авторів – Ленори та Мануеля Блум та Майка Шуба, Blum-Blum-Shub) вважають одним з найсильніших програмних генераторів псевдовипадкових послідовностей. Він вважається криптографічно стійким, і може використовуватися у серйозних криптографічних застосуваннях.

Нехай є два простих числа, *p* і *q*, причому $p \equiv q \equiv 3 \bmod 4$. Добуток цих чисел $n=pq$ називається цілим числом Блума. Оберемо ще одне випадкове число, *x*, взаємно просте з *n* та обчислимо $x_0 \equiv x \bmod n$. Це число вважається стартовим числом генератора.

Далі можна обчислити такі біти послідовності за формулою: $x_i \equiv x_{i-1}^2 \bmod n$ та $s_i \equiv x_i \bmod 2$. Останнє визначає, що в якості виходу генератора обирається молодший біт числа x_i. Отже, можна записати:

$$x_0 = x^2 \bmod n$$
$$\text{for } i = 1 \text{ to } \infty$$
$$x_i = (x_{i-1})^2 \bmod n$$
$$s_i = x_i \bmod 2.$$

Найцікавішою властивістю генератора BBS є те, що для визначення значення *i*-го біту зовсім необов'язково знати усі попередні $i-1$ бітів. Для безпосереднього обчислення значення *i*-го біту достатньо знати *p* та *q*.

Безпека цієї схеми ґрунтується на складності розкладання *n* на прості множники. Число *n* можна опублікувати так, що кожен зможе генерувати біти за допомогою цього генератора. Однак поки крипто-

аналітик не розкладе n на множники, він не зможе передбачити вихід генератора.

Більше того, генератор BBS непередбачуваний як в правому, так і в лівому напрямах. Це означає, що отримавши послідовність бітів, криптоаналітик не зможе передбачити ні наступний, ні попередній біти послідовності. Причиною цього є не якийсь заплутаний механізм генерації, а математика розкладання n на множники.

Приклад: $p = 19$; $q = 23$; $p = q \equiv 3 \bmod 4$; $n = 437$; $x = 233$.

I	0	1	2	3	4	5	6	7
X_i	101	150	213	358	123	271	25	188
S_i	1	0	1	0	1	1	1	0

Обов'язковою умовою, що накладається на зародок x, повинно бути таке:

а) x – просте; б) x не ділиться на p і на q.

Цей генератор повільний, але є спосіб його прискорити. Як вказано у роботі [32], в якості бітів псевдовипадкової послідовності можна використовувати не один молодший біт, а $\log_2 m$ молодших бітів, де m – довжина числа x_i. Порівняна повільність цього генератора не дозволяє використовувати його для потокового шифрування (цей недолік зі зростанням швидкодії комп'ютерів стає менш актуальним), а от для високонадійних застосувань, як наприклад, генерування ключів, він вважається кращим за багато інших.

2. Генератор Блум – Мікалі. Безпека цього генератора базується на проблемах обчислення дискретного логарифма у кінцевому полі.

Для його реалізації генерують два простих числа: g та p. Зародок x_0 породжує такий процес генерації: $x_{i+1} \equiv g^{x_i} \bmod p$. Виходом генератора буде 1, якщо $x_i < (p - 1)/2$ і 0 – якщо ця нерівність не виконується.

Якщо модуль p достатньо великий для того, щоб обчислення дискретного логарифму було обчислювальним складним завданням, цей алгоритм безпечний.

3. Генератор RSA. Цей генератор використовує алгоритм RSA для утворення псевдовипадкової послідовності.

При цьому використовується публічний ключ (e, n):

1) оберемо випадкове число $X_0 < n$;

2) обчислимо $X_i = (X_{i-1})^e \bmod n$.

For $i = 1$ to ∞

$X_i = (X_{i-1})^e \bmod n$

$B_i = X_i \bmod 2$

Безпека цього генератора ґрунтується на складності розкладання великого числа на множники, тобто на тій самій задачі, яка лежить в основі криптостійкості власне системи RSA.

4. Використання симетричних алгоритмів для генерування псевдовипадкових послідовностей також може бути виправданим, оскільки вони сконструйовані так, щоб максимально перемішувати і розсіювати особливості вхідного тексту. Як показує аналіз, наприклад, алгоритму DES, вже при 8 циклах бінарна послідовність на виході практично мало чим відрізняється від випадкової.

Що стосується алгоритму ГОСТ 28147-89, то навіть у рекомендаціях щодо його використання є режим, що називається "гаммування зі зворотним зв’язком", де сам алгоритм використовується для генерування так званої "гамми" (бінарного ключового потоку), яка побітово накладається на вхідний текст.

Контрольні запитання

1. Нові асиметричні алгоритми на основі еліптичних кривих. Основні переваги та недоліки.

2. Математичні моделі нелінійних вузлів замін (S-box). Основні математичні операції при їх побудові.

3. Основи булевої алгебри щодо побудови нелінійних вузлів замін (S-box).

4. Генерування випадкових та псевдовипадкових послідовностей щодо використання в механізмах безпеки інформаційних систем.

5. Вимоги до генераторів випадкових та псевдовипадкових послідовностей.

6. Побудова криптографічностійкіх генераторів псевдовипадкових послідовностей.

7. Сучасні алгоритми побудови каскадних геш-функцій на універсальних класах.

Розділ 10. Механізми та протоколи керування ключами в (PKI) інформаційної системи

10.1. Основні положення керування ключами. Життєвий цикл криптографічного ключа

Перша частина стандарту ISO/IEC 11770-1 присвячена розгляду таких основних питань:

мета, погрози і політика керування ключами;

основні поняття і визначення керування ключами;

життєвий цикл ключів і функції керування ключами;

класифікація ключів;

моделі розподілу ключів і методи захисту ключів;

сертифікація ключів;

інформаційні об’єкти керування ключами (Key Management Information Objects), структури даних, що містять ключ (ключі) і пов’язану з ними інформацію (такі структури також визначені в ASN.1), життєвий цикл сертифіката керування ключами.

Під *керуванням ключами* розуміють множину методів і процедур, що здійснюють встановлення і керування ключовими взаєминами між авторизованими об’єктами. Керування ключами включає методи і процедури, які підтримують:

ініціалізацію системних користувачів усередині домену безпеки;

генерацію, розподіл й інсталяцію ключового матеріалу;

керування і контроль використання ключового матеріалу;

відновлення, анулювання і знищення ключового матеріалу;

зберігання, резервування/відновлення та архівування ключового матеріалу.

Метою керування ключами є запобігання таких основних погроз:

компрометація конфіденційності секретних ключів;

компрометація автентичності секретних і відкритих ключів;

неавторизоване використання секретних і відкритих ключів.

Керування ключами здійснюється на основі спеціальної політики безпеки, яка прямо або побічно визначає погрози безпеки. Політика також визначає:

заходи і процедури, що випливають із застосування технічних і організаційних аспектів керування ключами як автоматичного, так і ручного;

права, обов’язки і відповідальність кожної зі сторін, що брали участь у керуванні ключами;

тип і зміст записів, внесених у контрольні журнали (журнали контролю безпеки) щодо настання яких-небудь подій, пов’язаних з безпекою керування ключами.

Для подальшого розгляду положень стандарту наведемо деякі найбільш важливі визначення, які вводяться ISO/IEC 11770-1.

Орган (адміністратор) сертифікації (certification authority) – довірчий центр створення й закріплення сертифікатів відкритих ключів. Додатково орган сертифікації може створювати і призначати ключі об’єктам.

Неявна автентифікація ключа об’єкта А – завірення одного об’єкта *А* в тому, що тільки інший ідентифікований об’єкт може мати правильний (відповідний) ключ.

Ключ – послідовність символів, які управляють процесом криптографічного перетворення.

Центр розподілу ключів (ЦРК) (key distribution centre) – довірча сторона, яка генерує або здобуває, а потім розподіляє ключі об’єктам, що мають загальний ключ з ЦРК.

Підтвердження приймання ключа (key confirmation) – гарантії одного об’єкта того, що інший ідентифікований об’єкт володіє правильним ключем.

Контроль над ключем (key control) – здатність вибрати ключ або параметри, які використовуються при обчисленні ключів.

Центр передачі ключів (ЦПК) (key translation center) – довірча сторона передачі ключів між об’єктами, кожний з яких має загальний ключ з ЦПК.

Запровадження в дію ключа (key establisment) – процес, що робить доступним загальний секретний ключ одному або більше об’єктам. Запровадження в дію ключа включає угоду про ключ і доставку ключа.

Угода про ключ (key agreement) – процес уведення в дію загального секретного ключа між об’єктами таким шляхом, що ніхто з них не може визначити значення цього ключа. Це значить, що жоден з об’єктів не управляє ключем.

Доставка ключа (key transport) – процес (можливо захищений) передачі ключа від одного об’єкта іншому.

Класифікація ключів, ієрархія ключів і криптоперіод.

Ключі розділяються залежно від того, у якій криптографічній системі вони використовуються – симетричній або несиметричній. Як відомо, в симетричних криптографічних системах здійснюється два перетворення: одне на передавальній стороні, інше – на прийомній стороні, які використовують той самий ключ.

У несиметричних криптосистемах одне перетворення (відкрите) виконується на одному (відкритому) ключі, а закрите перетворення виконується на іншому (особистому) ключі. У зв'язку з цим усі ключі можна розділити на такі типи.

Особистий ключ (private key) – ключ із пари несиметричних ключів об'єкта, який повинен використовуватися тільки даним об'єктом.

Відкритий ключ (public key) – ключ із пари несиметричних ключів об'єкта, який зроблений загальнодоступним.

Особистий і відкритий ключі використовуються в несиметричних криптографічних системах.

Симетричний ключ (symmetric key) – ключ, що використовується у симетричних (одноключових) криптографічних системах.

Секретний ключ (secret key) – ключ, який тримається в секреті і використовується тільки певною множиною об'єктів. Секретними ключами звичайно є симетричні й особисті ключі.

Як симетрична, так і несиметрична криптосистема в загальному випадку реалізують послуги конфіденційності (системи шифрування) і автентифікації об'єктів і повідомлень (системи автентифікації). У цьому випадку можна розглядати ключі, які потрібні для реалізації завдань цих систем [23].

Стандарт ISO/IEC 10770 здійснює класифікацію ключів за такими класифікаційними ознаками (рис. 10.1).

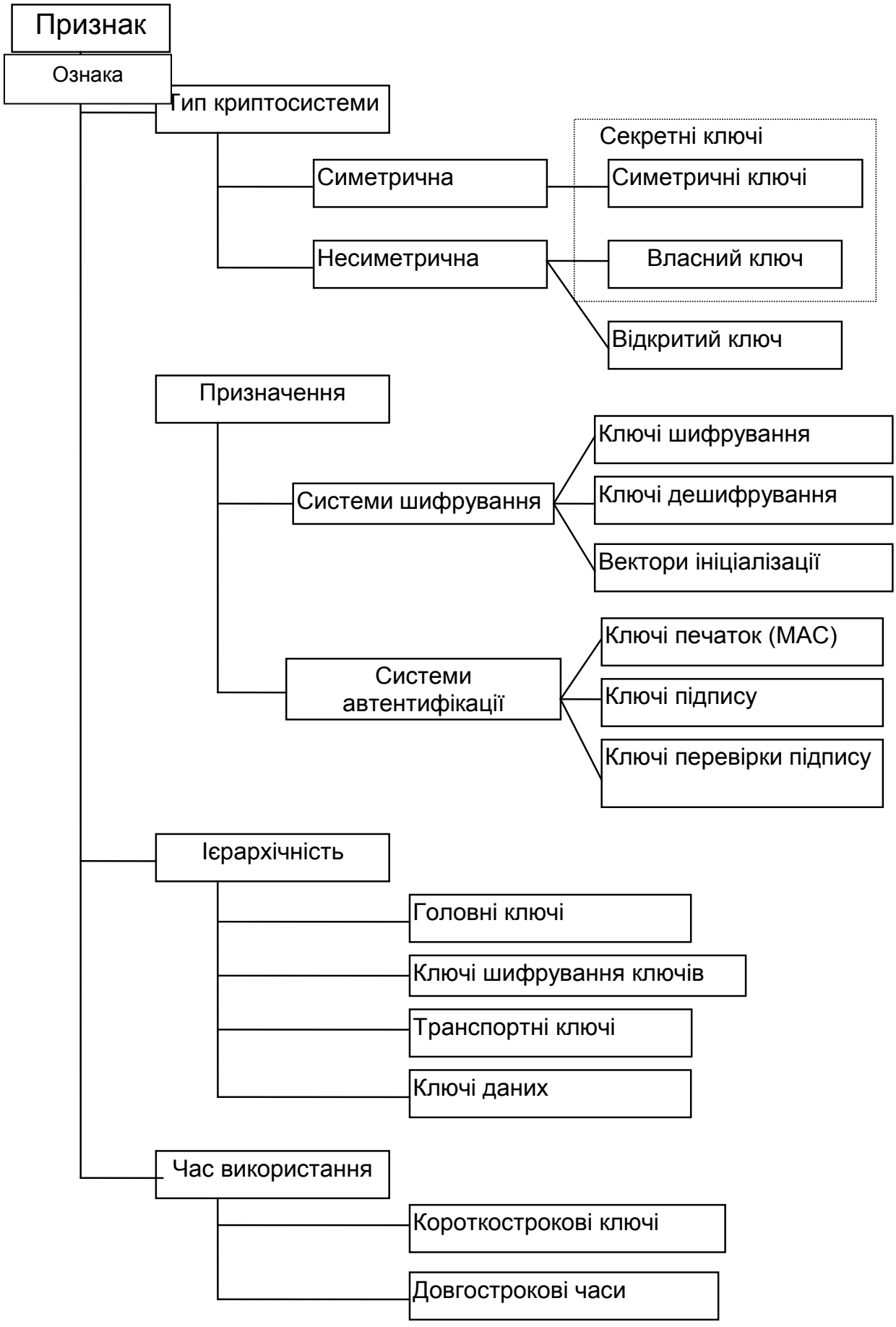


Рис. 10.1. **Класифікація ключів**

Системи автентифікації містять у собі механізми забезпечення цілісності повідомлень (MAC-коди, MDC-коди та ін.) та цифрові підписи.

Дані механізми допускають виконання таких дій, як:

постановка цифрової печатки (seal), тобто обчислення симетричної криптографічної контрольної суми (MAC-код);

підпис – обчислення цифрового підпису;

перевірка цифрової печатки – повторне обчислення криптографічної контрольної функції;

перевірка підпису – перевірка несиметричного цифрового підпису.

Таким чином, системи автентифікації використовують три основних типи ключів:

ключ печатки (sealing key) – симетричний секретний ключ;

ключ підпису (signature key) – особистий ключ;

ключ перевірки (verification key) – відкритий ключ (для несиметричних систем), або симетричний секретний ключ (для симетричних систем).

Системи шифрування включають симетричні й несиметричні системи і основними типами ключів є:

ключ шифрування – секретний або відкритий ключ;

ключ розшифрування – секретний або особистий ключ.

Завдання забезпечення конфіденційності можна розбити на дві підзадачі, за типом підлягаючої закриттю інформації – забезпечення конфіденційності даних користувача й забезпечення конфіденційності ключового матеріалу. У зв’язку з цим ключі зазвичай організовуються в ієрархію ключів (key hierarchies).

На верхньому рівні ієрархії розташовується *головний* або *майстер-ключ* (master key). Головні ключі не можуть бути криптографічно захищені. Вони захищаються прийняттям відповідних організаційних заходів при їх розподілі, зберіганні, запровадженні в дію. Розподіл здійснюється ручним способом через довірених кур’єрів або спеціальною поштою, чи шляхом безпосередньої інсталяції в криптографічне встаткування при його виготовленні або поставці. Захист у цьому випадку здійснюється через фізичну або електричну ізоляцію обладнання зберігання ключа.

На наступному рівні ієрархії перебувають *ключі шифрування ключів* (key-encrypting keys). Це симетричні або відкриті ключі шифрування, які використовуються для зберігання і передачі інших ключів. Якщо такі

ключі застосовуються в протоколах доставки ключів, то такі ключі ще називаються *транспортними ключами* (key-transport keys).

Нарешті на нижньому рівні ієрархії розташовуються *ключі даних* (data keys), які застосовуються для реалізації різних криптографічних операцій над даними користувача. Найчастіше до таких ключів відносяться симетричні короткочасні ключі. До таких ключів можна віднести і ключі генерації цифрового підпису, які найчастіше є довгочасними ключами.

Для захисту ключів одного рівня ієрархії можуть використовуватися ключі тільки вищого рівня ієрархії. Безпосередньо для реалізації послуг забезпечення безпеки даних використовуються ключі тільки найнижчого рівня ієрархії. Такий підхід дозволяє обмежити використання конкретного ключа, тим самим зменшити ризик розкриття ключа і затруднити проведення криптоаналітичних атак. Наприклад, компрометація одного ключа даних (тобто ключа на нижньому рівні ієрархії) приведе до компрометації тільки захищених цим ключем даних. Розкриття головного ключа потенційно дає можливість розкрити або маніпулювати всіма захищеними цим ключем ключами (тобто всіма ключами ієрархії). Отже, бажано мінімізувати доступ до цього ключа. Можливо побудувати систему таким чином, щоб жоден з користувачів не мав доступу до значення головного ключа.

Наступною ознакою класифікації ключів є строк їх дії або криптоперіод.

Криптоперіод – це певний період часу, протягом якого конкретний криптографічний ключ затверджується до використання, або протягом якого криптографічні ключі залишаються в силі для даної системи й придатні для використання авторизованими сторонами.

Криптоперіод служить для того, щоб:

обмежити кількість інформації, пов’язаної із ключами, доступної для криптоаналізу;

зменшити ризик порушення безпеки у випадку компрометації ключа;

обмежити використання конкретних методів і технологій для ефективної оцінки значення ключа протягом його життєвого циклу;

обмежити доступний обчислювальний час для проведення інтенсивних криптоаналітичних атак (це стосується тільки короткочасних ключів).

За часовою ознакою ключі можна розділити на два класи – довгострокові і короткострокові.

До *довгострокових ключів* відносяться головні ключі, а також можуть бути віднесені ключі шифрування ключів і ключі, використовувані в схемах угоди про ключі.

Короткостроковими ключами є ключі, що вводяться в дію транспортними ключами або шляхом застосування схем угоди про ключі. Найчастіше короткочасні ключі використовуються як ключі даних або сеансових ключів (session key), використовуваних на один сеанс зв'язку.

З погляду застосування короткострокові ключі звичайно використовують у комунікаційних додатках тоді, як довгочасні ключі використовуються в додатках, які орієнтовані на зберігання даних, а також для захисту короткострокових ключів.

Терміни коротко- і довгостроковий відносяться тільки до безпосереднього використання ключа для реалізації криптографічної операції, виконуваної авторизованими сторонами. Ці терміни не можна віднести до більш загальної системної характеристики ключа, а саме – до життєвого циклу ключа. Так, ключ шифрування, використовуваний для одного сеансу зв'язку, тобто короткочасний ключ, може вимагати забезпечення такої захищеності, щоб протистояти криптонападу досить довгий період. З іншого боку, якщо сигнатура буде перевірена негайно й у подальшій перевірці вона не матиме потреби, то і ключ підпису може бути захищений тільки на відносно короткий строк.

Життєвий цикл ключа, функції керування ключами, життєвий цикл керування ключами.

Криптографічний ключ може перебувати в різних станах, які визначають його життєвий цикл. Стандарт ISO/IEC 11770 розрізняє основні перехідні стани. Основними станами є:

стан очікування (черговий стан) (pending active) – стан, у якому ключ не використовується для звичайних операцій;

активний стан (active) – стан, у якому ключ використовується для криптографічної обробки інформації;

постактивний стан (post active) – стан, у якому ключ може використовуватися тільки для дешифрування або верифікації. Якщо буде потреба використання ключа за призначенням, він переводиться з постактивного стану в активний. Ключ, про який відомо, що він скомпрометований, повинен бути негайно переведений у постактивний стан.

При переході з одного основного стану в інший ключ може перебувати в одному з перехідних станів (transition). Такими перехідними станами є:

генерація – процес генерації ключа, у ході якого відповідно до запропонованих правил генерується ключ;

активізація (activation) – процес або сукупність процесів, у ході яких ключ робиться придатним для використання, тобто переводиться зі стану очікування в активний стан;

деактивизація (deactivation) – процес або сукупність процесів, що обмежують використання ключа, наприклад, через закінчення терміну дії ключа або його анулювання, що і переводять ключ із активного в постактивний стан;

реактивізація (reactivation) – процес або сукупність процесів, які дозволяють перевести ключ з постактивного в активний стан для повторного використання;

знищення (destruction) – завершує життєвий цикл ключа.

На рис. 10.2 схематично поданий взаємозв'язок основних і перехідних станів.

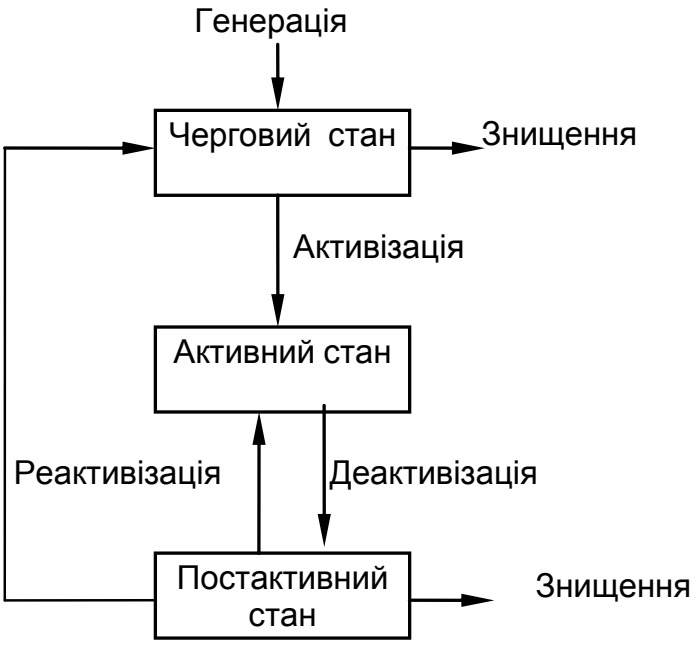


Рис. 10.2. **Взаємозв'язок основних і перехідних станів**

Життєвий цикл ключа підтримується одинадцятьма функціями керування ключами (key management services). Коротко охарактеризуємо ці функції [23; 43].

1. *Генерація ключа* – забезпечує генерацію криптографічного ключа із заданими властивостями для конкретних криптографічних додатків.

2. *Реєстрація ключа* – зв'язує ключ із об'єктом (звичайно тільки відповідні секретні ключі). Об'єкт, що бажає зареєструвати ключ, контактує з адміністратором реєстрації.

3. *Створення сертифіката ключа* – гарантує взаємозв'язок відкритого ключа з об'єктом і забезпечується вповноваженим органом сертифікації (crtification authority), який генерує відповідні сертифікати.

4. *Розподіл ключа* (distribute key) – множина процедур безпечного (секретного) забезпечення ключами і пов'язаної з ними інформації вповноважених об'єктів.

5. *Інсталяція ключа* (install-key) – розміщення ключа в устаткуванні керування ключами безпечним чином і готовим до використання.

6. *Зберігання ключа* (store-key) – безпечне зберігання ключів для подальшого використання або відновлення ключа для повторного використання.

7. *Похідна ключа* (derive-key) – формування великої кількості ключів, які називаються похідними ключами, шляхом комбінування секретного вихідного ключового матеріалу, називаного *ключем деривації*, з несекретними даними на основі використання необоротних процесів.

8. *Архівування ключа* (archive-key) – забезпечення безпечного зберігання ключів після їх використання. Ця функція використовує функцію зберігання ключа й інші засоби, наприклад, зовнішні сховища.

9. *Скасування (анулювання) ключа* (revoke-key) (відоме як видалення ключа (delete key)) – у випадках компрометації ключа функція забезпечує безпечну деактивізацію ключа.

10. *Дереєстрація ключа* (deregister-key) – функція реалізується повноважним органом реєстрації, який забирає запис про те, що даний секретний ключ пов'язаний з об'єктом.

11. *Знищення ключа* (destroy-key) – забезпечує безпечне знищення ключів, у яких минув термін дії. Ця функція включає і знищення всіх архівних копій ключа.

Послідовність станів, у яких може перебувати ключовий матеріал, функції керування ключами, інші функції і процеси, що протікають протягом усього життєвого циклу ключів, утворюють життєвий цикл керування ключами. На рис 10.3 поданий життєвий цикл керування ключами і його взаємозв'язок з основними станами ключів.

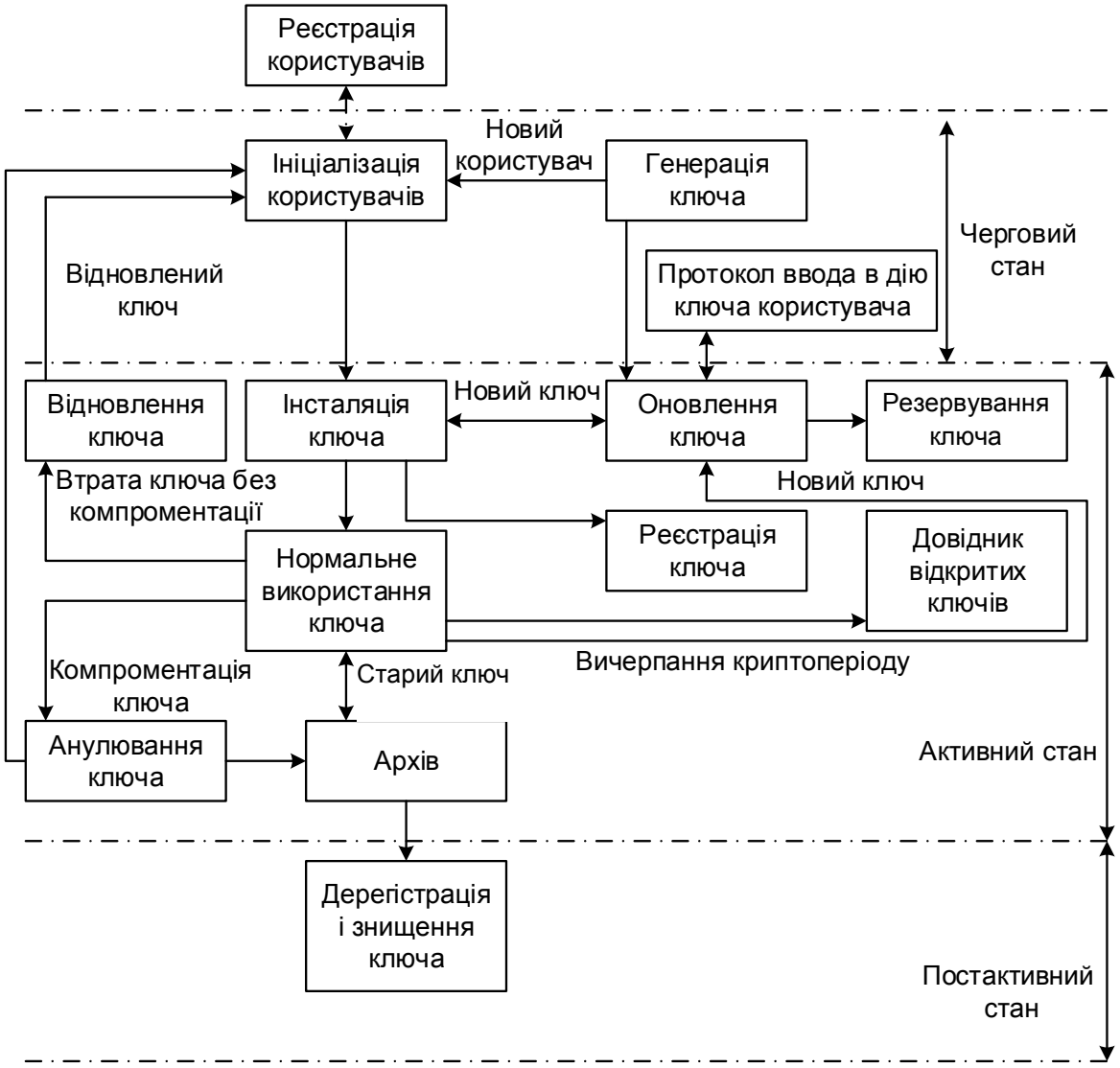


Рис. 10.3. **Життєвий цикл керування ключами**

Життєвий цикл керування ключами містить такі основні етапи й процеси [23; 43].

1. *Реєстрація користувача* – процес, у ході якого об’єкт стає авторизованим членом домену безпеки. Це допускає придбання (створення) і обмін первинним ключовим матеріалом між користувачем і доменом безпеки, наприклад, поділюваним паролем або персональним ідентифікаційним номером (PIN). Всі дії в ході реєстрації здійснюються безпечними одноразовими способами, наприклад, через особистий обмін, замовлення поштою, довіреним кур’єром.

2. *Ініціалізація користувача* – процес, у ході якого об’єкт ініціалізує свій криптографічний додаток (наприклад, інсталює та ініціалізує програмне або апаратне забезпечення), включаючи використання або інсталяцію первинного ключового матеріалу, який отриманий під час реєстрації користувача.

3. *Генерація ключа.* Генерація криптографічних ключів обов'язково повинна включати заходи, спрямовані на забезпечення відповідних властивостей ключа і його випадковості. Ці властивості забезпечуються шляхом використання методів генерації випадкових або псевдовипадкових чисел. Об'єкт може генерувати собі ключі або самостійно, або запитувати їх у довірчої сторони.

4. *Інсталяція ключа*. Ключовий матеріал інсталюється в програмне або апаратне забезпечення об'єкта за допомогою різних методів, наприклад, ручне введення пароля або PIN, передача даних з використанням диска, постійного запам'ятовувального пристрою, чіп-карти або інших апаратних обладнань (наприклад, завантажника ключа). Первинний ключовий матеріал може служити для організації безпечного on-line сеансу зв'язку, за допомогою якого вводяться в дію основні робочі криптографічні ключі. Надалі відновлення новим ключовим матеріалом замість використовуваного повинне здійснюватися за допомогою безпечних on-line-методів відновлення.

5. *Реєстрація ключа* здійснюється в тісному зв'язку з інсталяцією ключа і полягає в тому, що ключовий матеріал офіційно заноситься під унікальним іменем об'єкта в закриту базу ключів адміністратором реєстрації. Для відкритих ключів адміністратор сертифікації створює сертифікати відкритих ключів, які виступають у ролі гарантів дійсності й цілісності ключа. Сертифікати містяться в довідниках відкритих ключів і є загальнодоступними.

6. *Нормальне використання*. На даному етапі ключі перебувають в оперативній доступності для стандартних криптографічних додатків, включаючи контроль використання криптографічних ключів. За нормальних умов функціонування системи в період нормального використання ключів збігається із криптоперіодом ключів. У несиметричних криптосистемах ключі з однієї пари можуть перебувати на різних етапах свого існування. Наприклад, у деякий момент часу відкритий ключ шифрування може вважатися недійсним у той час, як закритий особистий ключ залишається в активному стані і може використовуватися для розшифрування.

7. *Резервування ключа* становить резервування ключового матеріалу в незалежному, безпечному сховищі з метою здійснення, якщо буде потреба, відновлення ключа. Резервні копії ключа в основному відправляються на короткострокове зберігання під час нормального використання ключа.

8. Відновлення ключа. Наприкінці криптоперіоду оперативний ключовий матеріал замінюється новим. Це допускає використання комбінації

функції генерації ключів, вироблення або установки нових ключів, реалізації двосторонніх протоколів запровадження в дію ключів або організації зв'язку із залученням довірчої третьої сторони. Для відкритих ключів відновлення й реєстрація нових ключів зазвичай допускає реалізацію безпечних комунікаційних протоколів за участю адміністратора сертифікації.

9. *Архівування* ключового матеріалу, який надалі не буде використовуватися для здійснення криптографічних операцій, здійснюється з метою забезпечення можливості пошуку ключа при виникненні особливих умов, наприклад, вирішення спорів, включаючи реалізацію функції причетності. Архівування допускає довгострокове автономне зберігання ключа, який виводиться при цьому в постактивний стан.

10. Перед *знищенням ключа* здійснюється його *дереєстрація*, тобто видалення відповідного запису з довідника, у результаті чого знищується взаємозв'язок значення конкретного ключа з об'єктом. Якщо здійснюється знищення секретного ключа, необхідно забезпечити безпечне і надійне знищення всіх його слідів (залишків).

11. *Відновлення ключа* здійснюється у випадку, якщо ключовий матеріал був загублений у результаті не примусової (ненавмисної) компрометації (збої, несправність устаткування, забування (втрата) пароля). У цьому випадку використовуються резервні копії ключа.

12. *Видалення (анулювання) ключа* має на увазі виведення ключа з активного стану в постактивний в результаті, наприклад, його компрометації. При цьому безпосередньо ключ не знищується. Для відкритих ключів у цьому випадку здійснюється видалення (анулювання) сертифіката.

Представлений життєвий цикл керування ключами є найбільш загальним і більш застосовуваним до несиметричних криптосистем. У симетричних криптосистемах керування ключами в загальному випадку менш складне. Так, сеансові ключі можуть не реєструватися, не резервуватися, не віддалятися й не архівуватися.

Сертифікація ключів

Стандартами керування ключами сертифікація ключів розглядається як основний засіб забезпечення автентифікації ключів. У ISO/IEC 11770-1 обговорюються такі питання сертифікації ключів:

роль і місце органу сертифікації (Certification Authority);

модель процесу сертифікації;

розподіл, використання та анулювання сертифікатів.

Сертифікат відкритого ключа (public key certificate) – список даних, пов'язаних з окремим користувачем, включаючи відкритий ключ (або ключі) цього користувача, підписаний органом сертифікації. Таким

чином, сертифікат складається із двох полів – поля даних і поля підпису. Поле даних містить, як мінімум, відкритий ключ користувача й ідентифікаційну інформацію користувача (наприклад, ідентифікатор користувача). Поле підпису містить підпис адміністратора сертифікації, яка є поручництвом за автентичність відкритого ключа користувача. Сертифікат також може містити й іншу додаткову інформацію, наприклад, із вказівкою того, яким чином може бути використаний той або інший ключ. Кожний користувач повинен бути приписаний до конкретного адміністратора сертифікації і мати довірену копію ключа перевірки підпису цього адміністратора.

Для виконання своїх функцій адміністратор повинен мати свою ключову пару формування/перевірки підпису. Відкритий ключ із цієї пари доступний для всіх зареєстрованих авторизованих користувачів системи і дозволяє всім користувачам системи перевірити автентичність відкритих ключів у будь-якому сертифікаті адміністратора. Дійсність відкритого ключа АС може бути забезпечена як криптографічними, так і не криптографічними методами, наприклад, може бути використана замовлена пошта, кур'єр або особистий контакт. Кожного разу забезпечення дійсності, на відміну від конфіденційності, є обов'язковим.

Стандарт визначає коло відповідальності адміністратора (органу) сертифікації. Зокрема АС відповідає за:

визначення об'єктів, інформація і відкриті ключі повинні бути включені в сертифікат;

якість власної ключової пари, яка використовується для генерації сертифіката;

безпека процесу генерації сертифіката й особистого ключа, який використовується в процесі генерації сертифікатів.

На сьогодні існує кілька стандартів, що визначають структуру і зміст сертифіката. Найбільш важливими з них є рекомендації ITU-T X.509 – 1993 і відповідний міжнародний стандарт ISO/IEC 9594-8:1995. Відносно недавно в ці стандарти були внесені виправлення, у результаті чого з'явилась так звана третя версія формату сертифіката X.509.

Зазвичай сертифікат містить у собі таку інформацію: ім'я користувача; дату закінчення терміну дії сертифіката і ключа (період повноважень); порядковий номер або ідентифікатор ключа, що ідентифікують сертифікат або ключ; один або більше відкритих ключів, що належать користувачу; алгоритм присвоєння ідентифікатора (ідентифікаторів) відкритому ключу (ключам); інформація, що стосується політики безпеки, під керуванням якої був створений сертифікат; додаткова інформація про

користувача (наприклад, адреса або мережна адреса); додаткова інформація про ключ (наприклад, алгоритм і передбачуване використання); інформація, що сприяє верифікації підпису (наприклад, ідентифікатор алгоритму підпису); статус відкритого ключа.

Сертифікати відкритих ключів служать для гарантії дійсності відкритого ключа, але не сертифікують додаткової інформації. Для сертифікації додаткової інформації можуть використовуватись *сертифікати атрибутів*. Сертифікат атрибутів схожий із сертифікатом відкритого ключа й дозволяє передавати специфічну інформацію (атрибути), пов'язану з АС, об'єктами або відкритими ключами. Сертифікати атрибутів можуть бути асоційовані з конкретним відкритим ключем через зв'язок атрибутивної інформації із ключем. Такий зв'язок забезпечується за допомогою методів ідентифікації ключа, наприклад, можуть використовуватися порядковий номер відповідного сертифіката відкритого ключа, геш-код відкритого ключа або сертифіката. Сертифікат атрибутів підписується адміністратором сертифікації атрибутів і створюється разом з адміністратором реєстрації атрибутів.

Модель процесу сертифікації.

Модель процесу сертифікації характеризує взаємозв'язок між основними логічними об'єктами, які беруть участь у формуванні і керуванні сертифікатами. Модель, запропонована в ISO/IEC 11770-1, визначає такі логічні об'єкти, які можуть бути залучені в процес сертифікації (рис. 10.4):

Рис. 10.4. **Об'єкти процесу сертифікації**

орган сертифікації або *адміністратор сертифікації* відповідає за організацію та здійснення процесу сертифікації відкритих ключів користувачів, а також ручається за дійсність відкритих ключів. Це включає виконання таких дій, як призначення відкритим ключам різних імен за допомогою підписаних сертифікатів, керування призначенням порядкових номерів сертифікатам і анулювання сертифікатів;

сервер імен (name server) відповідає за керування простором імен користувачем з метою забезпечення кожного користувача унікальним іменем;

довідник сертифікатів (certificate directory) забезпечує підтримку сертифікатів в on-line режимі, тобто в стані повної готовності до використання їх користувачами. Звичайно це база даних або сервер, доступні для користувачів у режимі тільки для читання. Поповнення й підтримку довідника здійснює адміністратор сертифікації. Користувачі також можуть мати свої довідники сертифікатів. У цьому випадку за їх підтримку відповідає сам користувач;

центр генерації ключів здійснює генерацію пар відкритого/закритого ключа, а також генерацію симетричних ключів і паролів. Центр генерації ключів може бути частиною встаткування користувача, якщо користувач самостійно генерує собі ключ, або частиною органу сертифікації, або взагалі може бути окремою довірчою системою;

орган реєстрації або *адміністратор реєстрації* (registration authority) відповідає за авторизацію об’єктів, що відрізняються унікальними іменами, у якості члена домену безпеки. Реєстрація користувача звичайно допускає зв’язок ключового матеріалу з об’єктом.

Зовсім не обов’язково, що в конкретній реалізації системи сертифікації ці об’єкти будуть роздільними. В окремих випадках деякі з них взагалі можуть не існувати.

Створення, використання, верифікація, розподіл і анулювання сертифікатів відкритих ключів.

Перед створенням сертифіката відкритого ключа для деякого об’єкта *A* з метою перевірки дійсності об’єкта і факту приналежності відкритого ключа об’єкта, який сертифікується, АС повинен здійснити ряд заходів, які часто носять не криптографічний, а організаційний характер.

Генерація ключової пари об’єкта *A* може здійснюватися як довіреною стороною, так і самостійно.

У першому випадку довірена сторона генерує ключову пару, співвідносить її з конкретним об’єктом і включає відкритий ключ й ідентифікаційну інформацію об’єкта в сертифікат. Об’єкт одержує копію відповідного особистого ключа безпечним (автентичному і конфіденційному) каналом після того, як доведе свою дійсність (наприклад, особисто показавши паспорт). Усі сторони, що надалі використовують сертифікат, по суті делегують відповідальність за встановлення дійсності об’єкта довіреній стороні.

У другому випадку після генерації ключової пари об’єкт безпечним чином передає відкритий ключ довірчій стороні або особисто, або використовуючи захищений канал. За результатами перевірки дійсності джерела відкритого ключа довірена сторона генерує сертифікат відкритого ключа. При цьому адміністратор сертифікації повинен зажадати докази знання об’єктом відповідного особистого ключа, для того щоб запобігти шахрайству серед легітимних користувачів, що прагнуть для злочинних дій одержати на своє ім’я сертифікат відкритого ключа іншого користувача.

Для того щоб об’єкт В міг використовувати автентичний відкритий ключ об’єкта А, він повинен виконати такі дії:

1. Одержати автентичний відкритий ключ адміністратора сертифікації. Це однократна операція, яка здійснюється при реєстрації об’єкта органом сертифікації.

2. Одержати ідентифікаційну інформацію, яка однозначно ідентифікує передбачуваний об’єкт А.

3. Одержати по незахищеному каналу, наприклад, з довідника сертифікатів відкритих ключів або безпосередньо від А, сертифікат відкритого ключа об’єкта А, погоджений з ідентифікаційною інформацією об’єкта А.

4. Перевірити поточне значення часу і дати та співвіднести їх з терміном дії ключа, зазначеним у сертифікаті. Перевірити дійсність відкритого ключа АС. Перевірити підпис під сертифікатом, використовуючи відкритий ключ АС. Перевірити, що сертифікат не є анульованим.

Існує три моделі розподілу сертифікатів.

Якщо розглядати довідник сертифікатів як відкриту базу даних, то реалізується запитальна модель розподілу сертифікатів або pull-модель. У цьому випадку користувачі витягають сертифікати з бази даних у міру необхідності.

Інша модель розподілу сертифікатів носить примусовий характер (push-модель). У цій моделі сертифікати примусово розподіляються всім користувачам або періодично, або у міру створення сертифіката. Така модель найбільш прийнятна для реалізації в закритих системах.

Нарешті третій підхід до розподілу сертифікатів полягає в тому, що окремі користувачі за необхідності самостійно забезпечують своїми сертифікатами інших користувачів, наприклад, розподіляють сертифікати відкритих ключів для перевірки підпису.

Оскільки довідник сертифікатів звичайно розглядається як незахищена третя сторона, то необхідно реалізувати систему керування доступом до довідника. Для всіх користувачів дозволена тільки функція зчитування з довідника, а для авторизованих об'єктів найчастіше такими є адміністратори сертифікації й реєстрації, реалізуються функції підтримки і відновлення довідника. Крім того, самі сертифікати захищаються підписом, у результаті чого вони й можуть розподілятися відкритими каналами зв'язку. Виключенням є on-line сертифікати, які формуються адміністратором сертифікації в режимі реального часу за запитом користувача. Такі сертифікати мають короткий термін дії й розподіляються через довірчі сторони, які надають гарантії того, що даний сертифікат не анульований.

Сертифікати можуть бути анульовані ще до закінчення строку своєї дії. Це відбувається при компрометації ключів, за запитом будь-якого об'єкта про анулювання сертифіката, по витіканню терміну дії повноважень об'єкта, при зміні функціональних обов'язків об'єкта в організації й з інших причин. У зв'язку з цим необхідні засоби, які дозволяють ефективно інформувати інших користувачів про недійсні сертифікати. Це особливо важливо при компрометації ключів, оскільки зменшити можливий збиток можна шляхом виключення подальшого використання скомпрометованого ключа. Існують різні методи розв'язання завдання анулювання ключів і сертифікатів.

1. Включення дати закінчення терміну дії ключа (сертифіката) в одне з полів сертифіката. Крайнім випадком є on-line-сертифікати, які анулюються відразу ж після їх використання.

2. Попередження ручним способом, коли всі користувачі системи інформуються про анулювання ключа з використанням бездротових засобів або спеціальними каналами. Такий метод зручний у малих або закритих системах.

3. Використання відкритих файлів анульованих ключів, у яких утримуються ідентифікатори анульованих ключів. Перед використанням ключа всі користувачі перевіряють відсутність ідентифікатора даного ключа у файлі.

4. Створення списку анульованих сертифікатів (Certificate Revocation List (CRL)). CRL є списком порядкових номерів або інших ідентифікаторів сертифікатів, які були анульовані визначеним органом сертифікації.

5. Використання сертифікатів анулювання, що є альтернативою CRL. Сертифікат анулювання може розглядатися як сертифікат відкритого ключа, що містить час і прапор анулювання, і службовець для скасування відповідного сертифіката. Оригінальний сертифікат виключається з довідника сертифікатів і заміняється сертифікатом анулювання.

Стандарти рекомендують вирішувати завдання анулювання ключів через ведення списку анульованих сертифікатів. Цей список обов’язково повинен бути позначений міткою часу і підписується відповідним органом сертифікації. Крім списку порядкових номерів анульованих сертифікатів список може містити й іншу інформацію, наприклад, причину анулювання того або іншого сертифіката. Підпис списку гарантує його автентичність і формується тим адміністратором сертифікації, який відповідає за формування сертифікатів. Звичайно CRL містить ім’я цього AC. Відновлення списку повинне здійснюватися регулярно, навіть якщо список не був змінений, з обов’язковою установкою дати відновлення. Це необхідно для того, щоб надати можливість користувачам перевірити "свіжість" списку. Крім того, необхідно передбачити засоби і механізми ефективного й своєчасного розподілу списків анульованих сертифікатів.

10.2. Керування ключами на основі симетричних методів

Друга частина стандарту ISO/IEC 11770-2 була опублікована в 1996 році і визначає механізми встановлення ключа, що використовують методи симетричної криптографії [23; 36; 43]. В основному використовується симетричне шифрування, але також розглядається використання й криптографічних контрольних функцій. Здебільшого розглянуті в ISO/IEC 11770-2 механізми засновані на використанні протоколів автентифікації, розглянутих в ISO/IEC 9798-2 [23; 36; 43].

ISO/IEC 11770-2 включає 13 механізмів встановлення ключа для:

розподілу сеансового ключа між парою об'єктів із передвстановленим загальним головним ключем;

розподілу ключа між парою сторін через ЦРК;

розподілу ключа між парою сторін через ЦПК.

Таким чином, визначені механізми для всіх трьох моделей, визначених в ISO/IEC 11770-1 для випадку, коли два об'єкти належать тому самому домену безпеки. Механізми для випадку, коли об'єкти довіряють різним адміністраторам, можуть бути отримані з механізмів, викладених ISO/IEC 11770-2.

При розгляді механізмів будемо використовувати такі позначення:

A і B – два об'єкти, що бажають встановити новий секретний ключ;

ID_A і ID_B – ідентифікатори об'єктів;

$E_K(X)$ позначає шифрування блоку даних X з використанням секретного ключа K (зазначимо, що спосіб шифрування в стандарті не визначається). Якщо спосіб шифрування забезпечує цілісність даних і автентифікацію джерела даних, то мається на увазі, що за даними будуть обчислені MDC- або MAC-коди і приєднані до даних перед шифруванням;

X||Y позначає конкатенцію полів даних X і Y у точно встановленому порядку;

поля Text містять додаткову допоміжну інформацію.

Механізм встановлення ключа № 3.

Умовою реалізації механізму № 3 є те, що об'єкти A і B уже мають у своєму розпорядженні загальний секретний ключ K_{AB}. A і B також повинні підтримувати синхронізацію в системі, використовуючи синхронізований годинник або синхропослідовність. Цей механізм реалізує модель об'єктів, що безпосередньо зв'язуються, засновану на однопрохідному (однобічному) протоколі автентифікації (ISO/IEC 9798-2 п. 5.11) і забезпечує однобічну автентифікацію об'єкта A об'єктом B, а також неявну автентифікацію ключа об'єктом A. Об'єкт A вибирає ключ і, отже, має контроль над ключем.

Механізм має такий протокол.

Резюме. Об'єкт A взаємодіє з об'єктом B.

Результат. Об'єкт A передає секретний ключовий матеріал (наприклад, сеансовий ключ) об'єкта B.

1. Початкові умови. Об'єкти A і B мають загальний ключ K_{AB}.

2. Повідомлення протоколу:

$$A \rightarrow B : E_{K_{AB}} (T / N \,||\, ID_B \,||\, K \,||\, Text).$$

3. Дії за протоколом:

а) об’єкт А генерує ключовий матеріал і формує пакет даних, у якому поле T/N містить або мітку часу T, або порядковий номер повідомлення N, поле K містить новий ключовий матеріал, поле Text містить іншу додаткову інформацію, наприклад, тип алгоритму шифрування. Сформований пакет об’єкта шифрує на загальному секретному ключі $К_{АВ}$ і відправляє зашифроване повідомлення об’єкту В;

б) при одержанні шифрованого повідомлення об’єкт В розшифровує його, потім перевіряє наявність свого ідентифікатора ID_B, правильність мітки часу або порядкового номера і витягає необхідний ключовий матеріал з поля К.

Механізм установки ключа № 6.

Механізм № 6 використовує випадкові непередбачені числа, тобто застосовується механізм "запит – відповідь". Він опирається на трипрохідний протокол автентифікації (ISO/IEC 9798-2 п.5.2.2) і також реалізує модель об’єктів, що безпосередньо зв’язуються. Механізм забезпечує взаємну автентифікацію між А і В. Причому жоден з об’єктів не має контролю над ключем.

Протокол механізму установки ключа № 6.

Резюме. Об’єкта взаємодіє з об’єктом В.

Результат. Об’єкт А передає секретний ключовий матеріал (наприклад, сеансовий ключ) об’єкту В.

1. Початкові умови. Об’єкти А і В мають загальний ключ $К_{АВ}$.

2. Повідомлення протоколу:

$$B \rightarrow A : ID_B \,||\, R_B$$
$$A \rightarrow B : E_{K_{AB}}(R_A \,||\, R_B \,||\, ID_B \,||\, К_A \,||\, \text{Text 1})$$
$$B \rightarrow A : E_{K_{AB}}(R_B \,||\, R_A \,||\, К_B \,||\, \text{Text 2}).$$

3. Дії за протоколом.

а) об’єкт В формує непередбачене випадкове число R_B і відправляє його об’єкту А разом зі своїм ідентифікатором ID_B;

б) об’єкт А генерує непередбачене випадкове число R_A і ключовий матеріал, який зберігається, в поле $К_A$. Потім формує пакет, шифрує його на загальному ключі $К_{АВ}$ і відправляє об’єкту В;

в) об’єкти А і В обчислюють свій новий загальний ключ як необоротну функцію від ключового матеріалу $К_A$ і $К_B$. Стандарт допускає або $К_A$, або $К_B$ встановити в нуль. Однак якщо використовуються і $К_A$, і $К_B$, то функція, яка використовується для комбінації цих величин, повинна мати властивості, при яких жоден з об’єктів не матиме керування над ключем.

Механізм встановлення ключа № 9.

Механізм заснований на п’ятипрохідному протоколі автентифікації (ISO/IEC 9798-2 п. 6.2) і реалізує модель встановлення ключа через ЦРК, коли об’єкт А посилає ключ об’єкту В. У цьому протоколі вводиться довірча третя сторона ЦРК, яка довіряє і А і В. ЦРК має загальні секретні ключі $К_{АЦРК}$ і $К_{ВЦРК}$ з об’єктами А й В та має контроль над ключами. Механізм забезпечує взаємну автентифікацію між А і В.

Протокол механізму встановлення ключа № 9.

Резюме. Об’єкт А взаємодіє з ЦРК і об’єктом В.

Результат. А передає секретний ключовий матеріал $К_{АВ}$ об’єкту В.

1. Початкові умови. Об’єкти А і В мають відповідні загальні ключі $К_{АЦРК}$ і $К_{ВЦРК}$ із центром розподілу ключів.

2. Повідомлення протоколу.

$В \rightarrow А : R_B$.

$А \rightarrow ЦРК : ID_A \,||\, R_A \,||\, R_B \,||\, ID_B$.

$ЦРК \rightarrow А : E_{К_{АЦРК}} (R_A \,||\, К_{АВ} \,||\, ID_B \,||\, Text\ 1) \,||\, E_{К_{ВЦРК}} (R_B \,||\, К_{АВ} \,||\, ID_A \,||\, Text\ 2)$ $В : E_{К_{ВЦРК}} (R_B \,||\, К_{АВ} \,||\, ID_A \,||\, Text\ 2) \,||\, E_{К_{АВ}}(R_A \,||\, R_B \,||\, Text\ 3)$.

$В \rightarrow А : E_{К_{АВ}} (R_B \,||\, R_A \,||\, Text\ 4)$.

3. Дії за протоколом:

а) В генерує випадкове число R_B і відправляє його А;

б) А генерує випадкове число R_A і разом з R_B й ідентифікатором об’єкта В відправляє в ЦРК;

в) центр розподілу ключів формує ключ $К_{АВ}$, який буде встановлений між А і В. Повідомлення із ключем $К_{АВ}$ ЦРК шифрує на ключах $К_{АЦРК}$ і $К_{ВЦРК}$ й відправляє об’єкту А;

г) об’єкт А передає частину отриманого від ЦРК повідомлення, зашифрованого на ключі $К_{ВЦРК}$, об’єкту В. Крім того, він шифрує додаткове повідомлення, що містить значення раніше отриманих чисел R_A і R_B на новому ключі $К_{АВ}$;

д) об’єкт В, при одержанні повідомлення від А, розшифровує його, витягає значення випадкових чисел, формує нове повідомлення і зашифровує на ключі $К_{АВ}$;

е) об’єкт А, одержавши і розшифрувавши повідомлення, порівнює значення випадкових чисел з наявними у нього. У випадку їх збігу він робить висновок про те, що ключ між ним і об’єктом В встановлений правильно.

10.3. Керування ключами на основі асиметричних методів

Третя частина стандарту ISO/IEC 11770-3 визначає механізми встановлення ключа, засновані на застосуванні методів несиметричної криптографії. Стандарт описує механізми, що забезпечують [23; 36; 43]:

узгодження загального секретного ключа між об'єктами (сім механізмів узгодження ключа);

передачу секретного ключа від одного об'єкта до іншого (шість механізмів передачі ключа);

забезпечення доступності секретного ключа одного об'єкта іншому об'єкту способом, який перевіряється (три механізми передачі секретних ключів).

Перші два типи механізмів включають використання протоколів визначених в ISO/IEC 9798-3, а також включає різні методи несиметричної криптографії, наприклад, ключовий обмін Діффі – Хеллмана.

Третій тип механізмів включає використання сертифікатів.

Усі сім механізмів узгодження ключа, визначених в ISO/IEC 11770-3, використовують спеціальну однобічну функцію F

$$F : H \times G \to G,$$

де G і H – множини.

Функція задовольняє такі властивості:

1. $F(h, g) = F(h', F(h', g))$ для будь-яких $h, h' \in H$ і $g \in G$.

2. За даними $F(h, g)$, $F(h', g)$ і g обчислювально неможливо визначити $F(h, F(h, g))$. Це означає, що $F(\bullet, g)$ є однобічною криптографічною функцією для будь-якого g. Об'єкти A і B повинні мати загальний елемент $g \in G$ (який може бути і секретним).

Для прикладу розглянемо механізм узгодження ключів № 5. Якщо об'єкти A і B бажають використовувати цей механізм для установки нового загального секретного ключа, то:

об'єкт A повинен мати особистий ключ $h_A \in H$, відомий тільки A, і відкритий ключ $p_A = F(h_A, g)$, відомий об'єкту B;

об'єкт B повинен мати особистий ключ $h_B \in H$, відомий тільки B, і відкритий ключ $p_B = F(h_B, g)$, відомий об'єкту A.

Механізм забезпечує взаємну неявну автентифікацію ключа, але не дозволяє, однак, об'єктам A і B вибрати значення або форму K_{AB} заздалегідь, тобто ні об'єкт A, ні об'єкт B не мають контролю над ключем.

Отже, цей механізм не застосовується в системах, де необхідно мати спеціальний ключ виду – у таких випадках необхідно використовувати механізм передачі ключа.

Перед застосуванням механізму об'єкта вибирає випадкове (секретне) число $R_A \in H$, а В вибирає випадкове (секретне) число $R_B \in H$. Протокол обміну повідомленнями має вигляд:

$$A \rightarrow B : F(R_B, g) \,||\, \text{Text } 1$$
$$B \rightarrow A : F(R_B, g) \,||\, \text{Text } 2.$$

Після одержання першого повідомлення В обчислює загальний секретний ключ, як:

$$K_{AB} = \omega(F(hb, F(R_A, g)), F(R_B, pa)),$$

а об'єкт А, після одержання другого повідомлення, також обчислює загальний секретний ключ згідно з виразом:

$$K_{AB} = \omega(F(R_A, pb), F(R_B, g)).$$

де ω – загальна однобічна криптографічна функція узгодження.

Одним із прикладів однобічної криптографічної функції F(h, g), визначеної в стандарті, є дискретний логарифм. Ця функція лежить в основі ключового обміну Діффі – Хеллмана, у якому:

G = Z_p^* (мультиплікативна група цілих чисел за модулем простого числа p);

$$H = \{1, 2, \ldots, p - 2\};$$

g – примітивний елемент Z_p^*;

F(h, g) = ghmod p.

Реалізація механізму узгодження ключа № 5 з використанням дискретного логарифма має такий вигляд:

Узгодження ключів Діффі – Хеллмана

Резюме. А і В обмінюються повідомленнями відкритим каналом зв'язку .

Результат. Між А і В узгоджується загальний секретний ключ K_{AB}.

1. Початкові умови. Обирається відповідне просте число p і первісний елемент α групи Z_p^*. Дані параметри є відкритими.

2. Повідомлення протоколу:

$$A \rightarrow B: \alpha^x \bmod p$$
$$B \rightarrow A: \alpha^y \bmod p.$$

3. Дії за протоколом. Дані дії виконуються при кожному узгодженні ключа:

а) А вибирає випадкове число x, $1 \le x \le p - 2$, зберігає його в секреті і посилає повідомлення об'єкту В;

б) В вибирає випадкове число y, $1 \le y \le p - 2$, зберігає його в секреті і посилає повідомлення об'єкту А;

в) В одержує α^x і обчислює загальний ключ як $К_{АВ} = (\alpha^x)^y \bmod p$;

г) А одержує α^y і обчислює загальний ключ як $К_{АВ} = (\alpha^y)^x \bmod p$.

Розглянутий протокол забезпечує взаємну автентифікацію ключа. Фіксовані значення $\alpha^x \bmod p$ і $\alpha^y \bmod p$ виступають у ролі довгочасних відкритих ключів відповідних сторін, що розподіляються з використанням підписаних сертифікатів. Якщо такі сертфикати відомі в апріорі, то одержимо схему узгодження ключів без передачі секретної інформації. Однак інваріантність у часі загального ключа є недостачею протоколу. Протокол Діффі – Хеллмана може виконуватись на елементах мультиплікативної групи Z_p^* поля Z_p, а також на елементах мультиплікативної групи поля $GF(2^m)$ і на групі точок, визначених на еліптичній кривій над кінцевим полем. Більш докладно такі схеми викладені в стандарті IEEE P1363 [89; 90].

Передачу ключа від одного об'єкта до іншого реалізують за допомогою транспортних механізмів. Одним з таких механізмів є транспортний механізм № 5. Даний механізм передає ключ із використанням випадкових чисел і заснований на 3-прохідному протоколі автентифікації (ISO/IEC 9798-3 п. 5.2.2).

Механізм забезпечує взаємну автентифікацію і додаткове підтвердження одержання ключа, надаване об'єкту В (key confirmation). У ході реалізації протоколу встановлюються два ключа (передача у двох напрямах).

Протокол передачі ключа має вигляд.

Резюме. А взаємодіє з В.

Результат. Об'єкти А і В обмінюються новими секретними ключами $К_А$ і $К_В$.

1. Початкові умови. А і В мають пари перетворення підпис/перевірки (S_A, Y_A) і (S_B, Y_B) відповідно. Обидві сторони повинні мати доступ до відкритого перевірочного перетворення один одного. А і В мають пари перетворень шифрування/розшифрування (E_A, D_A) і (E_B, D_B) відповідно. Обидві сторони повинні мати доступ до відкритого перетворення, що шифрує, один одного.

2. Повідомлення протоколу:

$A \rightarrow B : ID_A \,||\, R_A \,||$ Text 1.

$B \rightarrow A : S_B(R_B \,||\, R_A \,||\, ID_A \,||\, E_A(ID_B \,||\, K_B \,||$ Text 2) $||$ Text 3 $||$ Text 4.

$A \rightarrow B : S_A(R_A \,||\, R_B \,||\, ID_B \,||\, E_B(ID_A \,||\, K_A \,||$ Text 5) $||$ Text 6 $||$ Text 7.

3. Дії за протоколом:

а) об'єкт А надсилає запит (Text 1) об'єкту В на обмін новими секретними ключами спільно зі своїм ідентифікатором і випадковим числом R_A.

б) об'єкт В генерує новий секретний ключ $К_B$, шифрує його на відкритому ключі шифрування об'єкта А, а потім зашифрований блок разом з іншою інформацією підписує, використовуючи свій особистий ключ підпису і алгоритм підпису S_B;

в) об'єкт А генерує новий ключ шифрування $К_A$, шифрує його на відкритому ключі шифрування об'єкта В, а потім підписує його разом з іншою інформацією на своєму особистому ключі підпису. При цьому для підтвердження правильності приймання ключа $К_B$ у поле Text 6 об'єкт А поміщає контрольну суму ключа $К_B$. Взаємний контроль ключів може бути досягнутий шляхом обчислення обох ключів $К_A$ і $К_B$ із використанням однобічної функції.

10.4. Безпека керування ключами

Жоден ключ шифрування не можна використовувати нескінченно. Час його дії повинен минати автоматично, подібно паспортам і ліцензіям. Причини цього в такому:

чим довше використовується ключ, тим більша ймовірність його компрометації. Якщо ключ використовується протягом року, то ймовірність його компрометації значно вища, ніж якби його використовували тільки один день;

чим довше використовується ключ, тим більші втрати при компрометації ключа. Якщо ключ використовується для шифрування одного документа, то втрата ключа означає компрометацію тільки цього документа. Якщо той же самий ключ використовується для шифрування більших обсягів інформації, то його компрометація призводить до значних втрат;

чим довше використовується ключ, тим більша спокуса прикласти необхідні зусилля для його розкриття. Розкриття ключа, який використ-

товується протягом доби, дозволить прочитати всі повідомлення, передані протягом доби;

розкриття ключа, який використовується протягом року, дозволить противнику одержати доступ до секретної інформації, переданої протягом року;

у ряді випадків трудомісткість криптоаналізу визначається кількістю шифротекстів, отриманих у результаті шифрування на одному ключі.

Для будь-якого криптографічного додатка необхідна стратегія, що визначає допустимий час життя ключа. У різних ключів можуть бути різні періоди життя. Наприклад, для криптографічного телефону має сенс використовувати ключ тільки протягом телефонної розмови, а для нової розмови – використовувати новий ключ. У ключів повинна бути відносно коротка тривалість життя, залежно від значимості і кількості даних, зашифрованих протягом заданого періоду. Ключ для каналу зв'язку зі швидкістю передачі 1 Гбіт/с, можливо, доведеться міняти частіше, ніж для модемного каналу швидкістю 9600 біт/с. Якщо існує ефективний метод передачі нових ключів, сеансові ключі повинні мінятись щодня. Ключі шифрування ключів так часто міняти не потрібно. При цьому для криптоаналітика утворюється небагато шифротексту, а відповідний відкритий текст (випадкові ключі) неможливо вгадати. Однак якщо ключ шифрування скомпрометований, потенційні втрати катастрофічні. У деяких додатках ключі шифрування ключів заміняються тільки раз на місяць або навіть раз у рік. Необхідно збалансувати ризики, пов'язані з заміною ключа або використанням фіксованого ключа.

Ключі шифрування, які використовуються при шифруванні даних довгострокового зберігання, не можна міняти часто. Щоденне розшифрування й повторне шифрування на новому ключі може призвести до негативних наслідків – подібна процедура призводить до нагромадження криптоаналітиком робочого матеріалу для наступної атаки. Розв'язком може послужити шифрування даних на деякому унікальному ключі (або ключах) з наступним його шифруванням на ключі шифрування ключів. Ключ шифрування ключів повинен зберігатися в безпечному місці. Втрата цього ключа означає втрату всіх ключів, які були на ньому зашифровані. Тривалість життя секретних ключів асиметричної криптографії залежить від додатків. Секретні ключі для цифрових підписів можуть використовуватись роками (навіть протягом людського життя). У багатьох криптомережах термін дії секретних ключів обмежений двома

роками. Старий ключ також повинен зберігатися в секреті на випадок підтвердження підпису, зробленого під час дії старого ключа. Але для підписання нових документів повинен використовуватись новий ключ. Такий підхід дозволяє зменшити кількість матеріалу, яку криптоаналітик зможе використовувати для розкриття ключа.

10.5. Структура та призначення PKI

10.5.1. Структура

Загальними цілями сучасних архітектур безпеки є захист і поширення інформації в розподіленому середовищі, де користувачі, ресурси і посередники розділені в часі і просторі. Інфраструктура відкритих ключів забезпечує сервіси, необхідні для безперервного управління ключами в розподіленій системі, пов'язує відкриті ключі з власниками відповідних секретних ключів і дозволяє користувачам перевіряти справжність цих зв'язків. PKI підтримує електронний документообіг і забезпечує ведення електронного бізнесу, гарантуючи, що:

1) особа або процес, ідентифікований як відправник електронного повідомлення або документа, дійсно є ініціатором відправлення;

2) особа або процес, який виступає одержувачем електронного повідомлення або документа, дійсно є тим одержувачем, якого мав на увазі відправник;

3) цілісність переданої інформації не порушена.

Зі зростанням числа додатків, що використовують криптографію, із збільшенням кількості її користувачів зростає кількість різнорідних сертифікатів, як у всій інформаційній системі, так і у кожного конкретного клієнта. Завдання однакової організації сервісу управління сертифікатами успішно вирішує інфраструктура відкритих ключів (PKI). На даний час відомо п'ять основних підходів до реалізації PKI в таких системах:

1) інфраструктура відкритих ключів, заснована на сертифікатах X.509 - PKIX;

2) проста інфраструктура відкритих ключів SPKI/SDS1;

3) захищена система доменних імен DNS;

4) система захищеної пошти PGP;

5) система захищених електронних транзакцій SET.

10.5.2. Компоненти і сервіси інфраструктури відкритих ключів

Основні компоненти PKI

Інфраструктура відкритих ключів є комплексною системою, що забезпечує всі необхідні сервіси для використання технології відкритих ключів. Мета PKI полягає в управлінні ключами та сертифікатами, за допомогою якого корпорація може підтримувати надійну мережеву середу. PKI дозволяє використовувати сервіси шифрування і вироблення цифрового підпису узгоджено з широким колом додатків, що функціонують в середовищі відкритих ключів.

Основними компонентами ефективної PKI є: засвідчує центр; реєстраційний центр; реєстр сертифікатів; архів сертифікатів; кінцеві суб'єкти (користувачі).

У складі PKI повинні функціонувати підсистеми анулювання сертифікатів, створення резервних копій і відновлення ключів, підтримки неможливості відмови від цифрових підписів, автоматичного коректування пар ключів і сертифікатів, управління "історією" ключів та підтримки взаємної сертифікації, прикладне програмне забезпечення користувачів повинно взаємодіяти з усіма цими підсистемами безпечним, погодженими і надійним способом.

Засвідчувальний центр

Безпосереднє використання відкритих ключів вимагає додаткової їх захисту та ідентифікації для визначення зв'язку з секретним ключем. Без такої додаткового захисту зловмисник може видавати себе як за відправника підписаних даних, так і за одержувача зашифрованих даних, замінивши значення відкритого ключа або порушивши його ідентифікацію. Все це призводить до необхідності верифікації відкритого ключа. Всі користувачі PKI повинні мати зареєстроване посвідчення, визнане співтовариством користувачів законним і надійним. Ці посвідчення зберігаються в цифровому форматі, відомому як сертифікат з відкритим ключем. Електронний сертифікат є цифровим документом, який пов'язує відкритий ключ з його власником. Для запевнення електронного сертифіката використовується електронний цифровий підпис засвідчує центру (ЗЦ), в цьому сенсі засвідчує центр уподібнюється нотаріальній конторі, так як підтверджує справжність сторін, що беруть участь в обміні електронними повідомленнями або документами.

Засвідчувальний центр відомий користувачам за двома атрибутами: за назвою та відкритим ключем. ЗЦ включає своє ім'я в кожен випущений ним сертифікат і список анульованих сертифікатів (САС) і підписує їх за допомогою власного секретного ключа. Користувачі можуть легко ідентифікувати сертифікати за іменем засвідчувального центру і переконатися в їх достовірності, використовуючи його відкритий ключ.

Засвідчувальний центр, головний керуючий компонент PKI, виконує такі основні функції:

формує власний секретний ключ і самопідписаний сертифікат;

випускає (тобто створює і підписує) сертифікати підлеглих засвідчувальних центрів та сертифікати відкритих ключів користувачів;

веде базу всіх виданих сертифікатів та формує список анульованих сертифікатів з регулярністю, визначеної регламентом;

публікує інформацію про статус сертифікатів і САС.

При необхідності засвідчує центр може делегувати деякі функції інших компонентів PKI. Випускаючи сертифікат відкритого ключа, засвідчувальний центр тим самим підтверджує, що особа, пойменована в сертифікаті, володіє особистим ключем, який відповідає цьому відкритому ключу. Включаючи в сертифікат додаткову інформацію, засвідчувальний центр підтверджує її належність цьому суб'єкту. Додаткова інформація може бути контактною (наприклад, адреса електронної пошти) або містить відомості про типи програм, які можуть працювати з даними сертифікатом. Коли суб'єктом сертифіката є інший засвідчувальний центр, видавець підтверджує надійність випущених цим центром сертифікатів.

Дії засвідчувального центру обмежені політикою застосування сертифікатів (ПЗС), яка диктує йому, яку інформацію повинен містити сертифікат. Засвідчувальний центр виконує адекватний захист свого таємного ключа і відкрито публікує свою політику, щоб користувачі могли переконатися у відповідності їй сертифікатів. Ознайомившись з політикою застосування сертифікатів і вирішивши, що довіряють засвідчувальному центру та його діловим операціям, користувачі можуть покладатися на сертифікати, випущені цим центром. Таким чином, у PKI засвідчують центри виступають як довірена третя сторона.

Реєстраційний центр

Реєстраційний центр (РЦ) є необов'язковим компонентом PKI. Зазвичай РЦ отримує уповноваження від центра реєструвати користу-

вачів, забезпечувати їх взаємодію з засвідчувальним центром та перевіряти інформацію, яка заноситься в сертифікат. Сертифікат може містити інформацію, надану суб'єктом, що подає заявку на сертифікат і пред'являє документ (наприклад, паспорт, водійські права, чекову книжку тощо) або третьою стороною (кредитним агентством про кредитний ліміт пластикової карти). Іноді в сертифікат включається інформація з відділу кадрів або дані, що характеризують повноваження суб'єкта в компанії (наприклад, право підпису документів певної категорії). РЦ агрегує цю інформацію і надає її засвідчувальному центру.

Засвідчувальний центр може працювати з декількома реєстраційними центрами, в цьому випадку він підтримує список акредитованих реєстраційних центрів, тобто тих, які визнані надійними. Засвідчувальний центр видає сертифікат реєстраційному центру. РЦ виступає як об'єкт, підлеглий засвідчувальному центру, і повинен адекватно захищати свій секретний ключ. РЦ відомий засвідчувальному центру за іменем та відкритим ключем. Перевіряючи підпис РЦ на повідомленні або документі, що засвідчується, центр покладається на надійність наданої РЦ інформації.

РЦ об'єднує комплекс програмного і апаратного забезпечення і людей, що працюють на ньому. У функції РЦ може входити генерація та архівування ключів, повідомлення про анулювання сертифікатів, публікація сертифікатів і CAC в мережевому довіднику LDAP й ін. Але РЦ не може випускати сертифікати і списки анульованих сертифікатів. Іноді засвідчує центр сам виконує функції реєстраційного центру.

Реєстр сертифікатів

Реєстр сертифікатів – спеціальний об'єкт PKI, база даних, де зберігаються діючі сертифікати і списки анульованих сертифікатів. Термін "реєстр" є синонімом раніше використаних в літературі назв сховища сертифікатів: каталог, репозиторій, довідник та інші. Цей компонент PKI значно спрощує управління системою та доступ до ресурсів. Реєстр надає інформацію про статус сертифікатів, забезпечує зберігання та розповсюдження сертифікатів та списків анульованих сертифікатів, управляє внесеннями змін у сертифікати. До реєстру висуваються такі вимоги: простота і стандартність доступу; регулярність оновлення інформації; вбудована захищеність; простота управління; сумісність з іншими сховищами (необов'язкове вимога).

Реєстр зазвичай розміщується на сервері каталогів, відповідному міжнародному стандарту X.500 і його підмножині. Більшість серверів

каталогів і прикладне програмне забезпечення користувачів підтримують протокол полегшеного доступу до каталогів LDAP. Такий уніфікований підхід дозволяє забезпечувати функціональну сумісність додатків PKI і дає можливість довіряє сторонам отримувати інформацію про статус сертифікатів для верифікації ЕЦП.

Архів сертифікатів. На архів сертифікатів покладена функція довготривалого зберігання (від імені засвідчувального центру) і захисту інформації про усі видані сертифікати. Архів підтримує базу даних, яка використовується при регулюванні суперечок з приводу надійності електронних цифрових підписів, якими в минулому завірялися документи. Архів підтверджує якість інформації в момент її отримання і забезпечує цілісність даних під час зберігання. Інформація, яка надається засвідчувальним центром архіву, повинна бути достатньою для визначення статусу сертифікатів та їх видавця. Архів захищає інформацію відповідними технічними засобами і процедурами.

Користувачі. Кінцеві суб'єкти, або користувачі, PKI діляться на дві категорії: власники сертифікатів та довіряють боку. Вони використовують деякі сервіси і функції PKI, щоб отримати сертифікати або перевірити сертифікати інших суб'єктів. Власником сертифіката може бути фізична або юридична особа, додаток, сервер і т. д. Довіряють боку запитують і покладаються на інформацію про статус сертифікатів і відкритих ключах ЕЦП своїх партнерів по діловому спілкуванню.

10.5.3. Сервіси PKI

Криптографічні сервіси

Генерація пар ключів. За допомогою цього сервісу генерується пара ключів (відкритий ключ/закритий ключ), секретний ключ зберігається у файлі, захищеному паролем або іншими засобами, наприклад, на смарт-карті або за допомогою іншого апаратного чи програмного засобу, що гарантує конфіденційність секретного ключа. У PKI повинні підтримуватися дві пари ключів для кожного користувача. У будь-який момент часу користувач повинен мати одну пару ключів для шифрування й розшифрування повідомлення, а іншу пару ключів для вироблення або перевірки цифрового підпису.

Вироблення цифрового підпису. Цей сервіс полягає в генерації дайджеста повідомлення і підпису його цифровим чином.

Верифікація (перевірка) цифрового підпису. За допомогою цього сервісу встановлюється достовірність повідомлення і відповідної йому цифрового підпису.

Сервіси управління сертифікатами

Сервіси управління сертифікатами утворюють ядро інфраструктури відкритих ключів.

Випуск сертифіката. Сертифікати випускаються для користувачів (фізичних та юридичних осіб), для сертифікаційних центрів, які знаходяться на більш низьких рівнях ієрархії довіри, а також для інших сертифікаційних центрів в разі взаємної сертифікації.

Управління життєвим циклом сертифікатів і ключів. Якщо секретний ключ користувача втрачено, викрадено або скомпрометований або є ймовірність настання таких подій, дія сертифіката повинно бути припинена. Після отримання підтвердження запиту користувача про анулювання сертифіката засвідчує центр повідомляє про анулювання всі зацікавлені сторони, використовуючи список анульованих сертифікатів. Аналогічно анотування здійснюється призупинення дії сертифіката. Воно полягає в однократної скасування сертифіката на деякий час протягом періоду його дії. Після цього дія сертифіката поновлюється автоматично або ж сертифікат анулюється. Призупинення дії сертифіката здійснюється у тих ситуаціях, коли неможливо встановити автентичність особи, що звертається із запитом про анулювання.

Підтримка реєстру. Випущений сертифікат або САС включається до реєстру (відповідно до специфікацій стандарту X.500 або інших вимог, щоб треті сторони могли мати до нього доступ. Зазвичай реєстр контролюється засвідчувальним центром, в деяких випадках – третьою стороною. Доступ до реєстру може бути обмежений. Якщо необхідно дотримання прав приватності користувачів, застосовуються заходи захисту даних від осіб, які не мають повноважень доступу.

Зберігання сертифікатів і САС в архіві. Випускаються сертифікати і списки анульованих сертифікатів зберігаються в архіві тривалий час, обумовлене правилами зберігання завірених ЕЦП документів.

Допоміжні сервіси

У PKI можуть підтримуватися також різні додаткові сервіси.

Реєстрація. Реєстраційні сервіси забезпечують реєстрацію і контроль інформації про суб'єктів, а також аутентифікацію, необхідну для

випуску або анулювання сертифікатів (від імені засвідчувального центру). Фактичний випуск сертифікатів здійснюється засвідчувальним центром.

Зберігання інформації в архіві. Сервіси зберігання інформації в архіві призначені для довготривалого зберігання та управління електронними документами та іншою інформацією. Сервіси забезпечують створення резервних копій і відновлення інформації у разі знищення або старіння середовища зберігання.

Нотаріальна автентифікація. Нотаріальна автентифікація включає автентифікацію відправника повідомлення, підтвердження цілісності та юридичної сили електронних документів.

Створення резервних копій та відновлення ключів. Засвідчувальний центр повинен мати можливість відновити зашифровану інформацію в разі втрати користувачами їх ключів шифрування. Це означає, що засвідчувальному центру необхідна система створення резервних копій і відновлення цих ключів. Цей процес відомий як комерційне створення резервних копій і відновлення ключів, і він відрізняється від примусового депонування ключів третьою стороною (зазвичай правоохоронними органами), яка отримує доступ до ключів для розшифровки необхідної інформації. Комерційні сервіси відновлення ключів забезпечують завчасне засекречування копії ключа на випадок втрати ключа користувачем, його відходу з роботи, забування пароля, необхідного для доступу до ключа, і відновлення ключа у відповідь на запит користувача або його роботодавця.

Невідмовність. Власноручні підписи традиційно свідчать про згоду або ознайомлення людини, яка його підписала, з текстом документа і не дозволяють відмовитися від факту підписання документу. Сучасні електронні технології дозволили замінити власну підпис на цифрову. Найголовніша вимога для запобігання відмови від цифрового підпису полягає в тому, що ключ підпису повинен генеруватися і безпечно зберігатися під контролем його власника. Коли користувачі забувають свої паролі або втрачають свої ключі підпису, на резервування або відновлення попередньої пари ключів підпису не накладається ніяких технічних обмежень (на відміну від аналогічної ситуації з парами ключів шифрування повідомлень). У таких випадках допускається генерація і подальше використання нових пар ключів підпису. Паралельне функціонування систем резервного копіювання і відновлення ключів і сервісу неспростовності викликає певні проблеми. При резервному копіюванні та

відновленні ключів повинні створюватися копії секретних ключів користувача. Щоб забезпечити неможливість відмови від цифрового підпису, не повинні створюватися резервні копії секретних ключів користувача, що використовуються для вироблення цифрового підпису. Для дотримання цих вимог у PKI повинні підтримуватися дві пари ключів для кожного користувача. У будь-який момент часу користувач повинен мати одну пару ключів для шифрування й розшифрування, а іншу пару ключів для вироблення або перевірки цифрового підпису.

Авторизація. Сертифікати можуть використовуватися для підтвердження особи користувача і завдання повноважень, якими він наділений. У числі повноважень суб'єкта сертифіката може бути, наприклад, право переглядати інформацію або дозвіл вносити зміни в матеріал, представлений на web-сервері.

Коригування ключів і керування історіями ключів. У найближчому майбутньому користувачі будуть мати величезну кількість пар ключів, які повинні будуть підтримуватися як криптографічні ключі, навіть якщо ніколи не будуть використовуватися. Ключі шифрування повинні з часом оновлюватися і повинні підтримувати історію всіх ключів, використаних раніше (наприклад, для розшифрування інформації багаторічної давності і перевірки цифрового підпису на старому контракті).

Процес коректування пар ключів повинен бути "прозорий" для користувача. Це означає, що користувачі не повинні піклуватися про оновлення ключів або отримувати відмова в обслуговуванні через недійсності своїх ключів. Для задоволення цієї вимоги пари ключів користувача повинні автоматично оновлюватися до закінчення терміну їх дії. При оновленні пари ключів підпису попередній ключ підпису безпечно знищується. Тим самим запобігає несанкціонований доступ до ключа та усувається необхідність зберігання попередніх ключів.

Інші сервіси. У ряді випадків необхідні й інші сервіси, наприклад, сервіси генерації пар ключів і запису їх на смарткарти, якщо ключі зберігаються на смарт-картах.

10.5.4. Архітектура і топологія PKI

Типи архітектури. PKI зазвичай складається з багатьох засвідчувальних центрів та користувачів, пов'язаних між собою різними способами, що дозволяють вибудувати шляхи довіри. *Шлях довіри*

пов’язує сторону, що довіряє з однією або багатьма третіми довіреними сторонами і дозволяє конфіденційно перевірити законність використовуваного довіряють стороною сертифіката. Одержувач завіреного цифровим підписом повідомлення, що не має зв’язку з центром, що засвідчує – видавцем сертифіката відкритого ключа підпису, може перевірити сертифікат відправника повідомлення, простежуючи ланцюжок сертифікатів між двома центрами: своїм і відправника. Для розгортання PKI традиційно використовують ієрархічну або мережеву архітектуру, останнім часом зі зростанням масштабів PKI і необхідності об’єднання різнорідних інфраструктур одержує поширення гібридна архітектура.

Ієрархічна архітектура

Засвідчувальні центри організуються ієрархічно під управлінням, так званого кореневого засвідчувального центру, який випускає самопідписаний сертифікат та сертифікати для підлеглих засвідчувальних центрів. Підлеглі засвідчувальні центри можуть випускати сертифікати для засвідчувальних центрів, що знаходяться нижче них за рівнем ієрархії, або для користувачів. В ієрархічній PKI кожна довіряюча сторона знає відкритий ключ підпису кореневого засвідчувального центру. Будь-який сертифікат може бути перевірений шляхом вибудовування ланцюжка сертифікатів від кореневого центру, що засвідчує, і її *верифікації*, тобто перевірки пов’язаності суб’єкта сертифіката та його відкритого ключа. Процедура верифікації ланцюжка сертифікатів передбачає, що всі "правильні" ланцюжки починаються з сертифікатів, виданих одним кореневим засвідчувальним центром. Щоб покладатися на сертифікат, сторона, що довіряє повинна пересвідчитися, що:

кожен сертифікат у ланцюжку підписаний за допомогою відкритого ключа наступного сертифіката у ланцюжку;

термін дії сертифіката не закінчився і сертифікат не анульований;

кожен сертифікат задовольняє ряд критеріїв, що задаються сертифікатами, розташованими вище в ланцюжку.

Вибудовування ланцюжка наведено на рис. 10.1. Відправник повідомлення А перевіряє сертифікат одержувача повідомлення В, випущений ЗЦ 4, потім – сертифікат ЗЦ 3, випущений ЗЦ 2, а потім сертифікат ЗЦ 2, випущений ЗЦ 1 – кореневим засвідчувальним центром, відкритий ключ підпису відомий відправнику.

Існують два способи отримання довіряючою стороною сертифікатів для перевірки ланцюжка: модель з проштовхуванням і модель з витягом. Модель з проштовхуванням припускає, що відправник передає одержувачу разом зі своїм сертифікатом всі сертифікати ланцюжка і одержувач

може негайно їх перевірити. При використанні моделі з витягом надсилається тільки сертифікат відправника, а одержувач сам повинен витягти сертифікат засвідчувальному центру. Оскільки кожен сертифікат містить ім'я видавця, одержувачу відомо, де перевірити сертифікат.

Перевагою ієрархічної архітектури є те, що не всі сторони повинні автоматично довіряти всім засвідчувальним центрам. Фактично єдиним центром, що засвідчує, якому необхідно довіряти, є кореневий засвідчувальний центр.

Мережна архітектура

Незалежні засвідчувальні центри взаємно сертифікують один одного, тобто випускають сертифікати одне для одного, і об'єднуються в пари взаємної сертифікації. Взаємна сертифікація дозволяє взаємодіяти засвідчувальним центрам та кінцевим суб'єктам з різних доменів PKI. У результаті між ними формується мережа довіри, яку наведена на рис. 10.2.

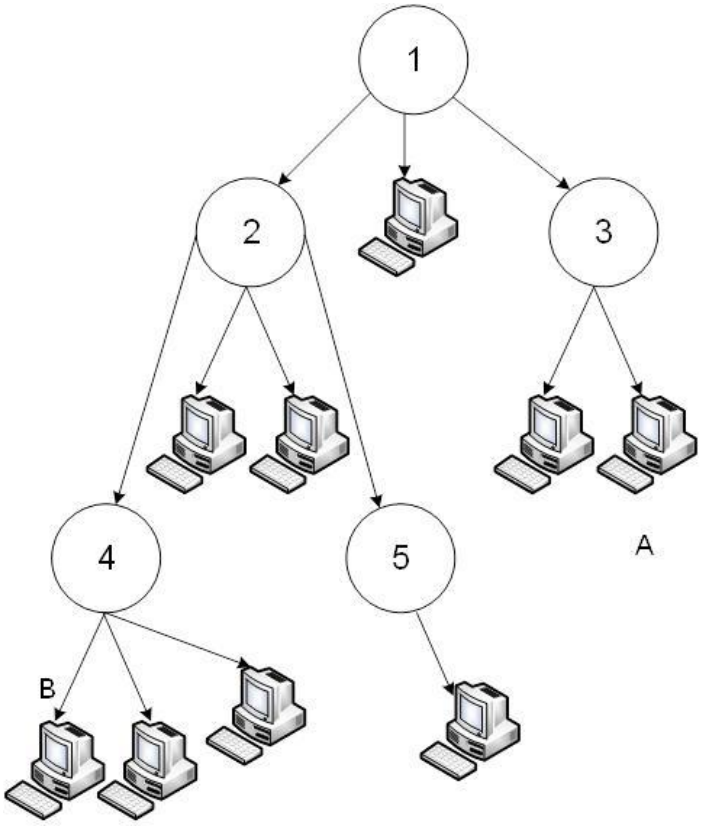


Рис. 10.1. **Ієрархічна архітектура PKI**

Сторона, що довіряє, знає відкритий ключ найближчого до неї засвідчувального центру, зазвичай того, який випустив для неї сертифікат. Сторона, що довіряє перевіряє сертифікат, вибудовуючи ланцюжок довіри від відомого їй засвідчувального центру, якому вона довіряє.

Наприклад, відправник повідомлення А знає відкритий ключ підпису ЗЦ 3, в той час як одержувачу повідомлення В відомий відкритий ключ ЗЦ 4. Існує кілька ланцюжків сертифікатів від одержувача до відправника повідомлення. Найкоротша ланцюжок: відправник повідомлення перевіряє сертифікат одержувача, випущений ЗЦ 4, потім – сертифікат ЗЦ 4, випущений ЗЦ 5, і, нарешті, сертифікат ЗЦ 5, випущений ЗЦ 3. ЗЦ 3 – це засвідчувальний центр, якому довіряє відправник і знає його відкритий ключ.

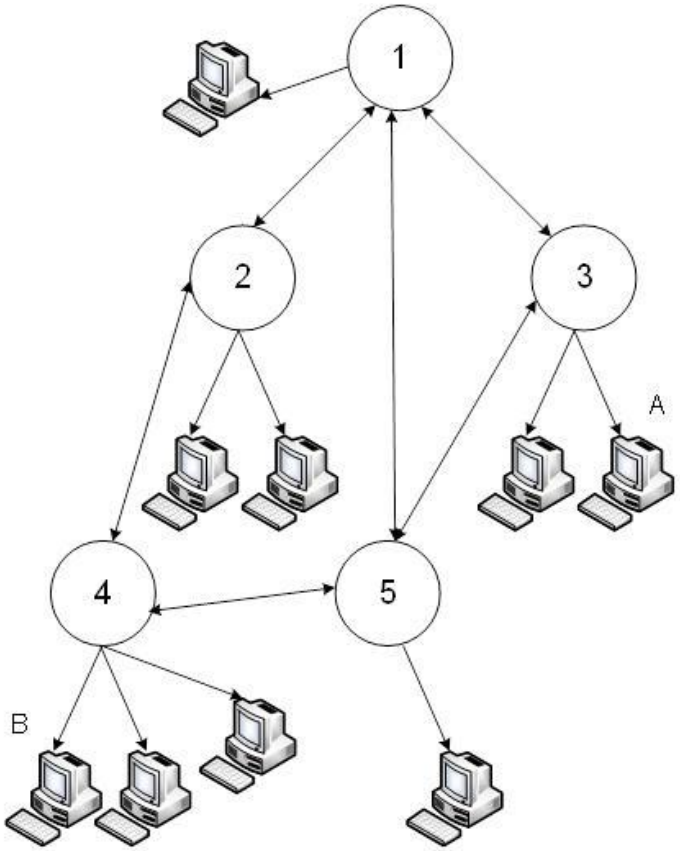


Рис. 10.2. **Мережна архітектура PKI**

Перевага мережної архітектури полягає в тому, що компрометація одного центру в мережі центрів, що засвідчують не обов’язково веде до втрати довіри до всієї PKI.

Гібридна архітектура

Для зв’язування різнорідних інфраструктур відкритих ключів нещодавно була запропонована гібридна (змішана) або "мостова" архітектура. З’єднання корпоративних PKI незалежно від їх архітектури досягається введенням нового центра, названого мостовим, єдиним призначенням якого є встановлення зв’язків між ними.

На відміну від мережного центра, що засвідчувальний мостовий центр не випускає сертифікати безпосередньо для користувачів, а на відміну від кореневого засвідчувального центру у ієрархічній PKI – не виступає в якості вузла довіри.

Усі користувачі PKI розглядають мостовий засвідчувальний центр як посередника. Мостовий засвідчувальний центр встановлює відносини "рівний з рівним" між різними корпоративними PKI. Ці відносини слугують своєрідним мостом довіри між користувачами різнорідних PKI. Якщо область довіри реалізована як ієрархічна PKI, то мостовий засвідчує центр встановлює зв’язок з кореневим засвідчувальним центром. Якщо ж область довіри – це мережа PKI, то мостовий засвідчувальний центр взаємодіє тільки з одним із центрів, що засвідчують мережі. У будь-якому випадку засвідчувальний центр, який вступає у відносини довіри з мостовим ЗЦ, називається головним.

На рис. 10.3 мостовий засвідчує центр встановлює зв’язки з трьома корпоративними PKI: перша – це PKI користувачів A і B, друга – ієрархічна PKI користувача 3 і третя – мережева PKI користувача D. Жоден з користувачів не довіряє безпосередньо мостовому засвідчувальному центру. Користувачі A і B довіряють засвідчувальному центру, видало їх сертифікати, і побічно довіряють мостовому засвідчувальному центру, для якого їх ЗЦ випустив взаємний сертифікат.

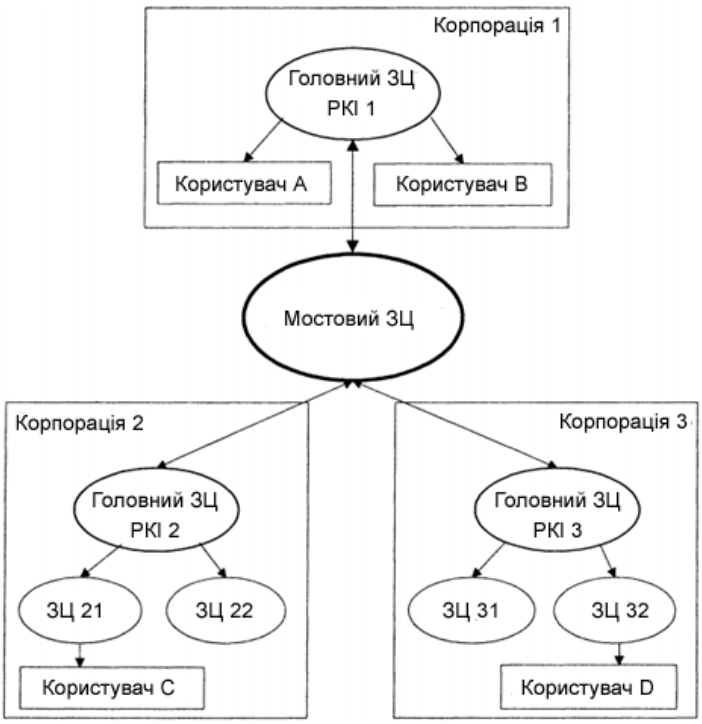


Рис. 10.3. **Мостовий ЗЦ і різнорідні PKI**

Користувач З довіряє мостовому засвідчувальному центру тільки тому, що для нього випустив сертифікат кореневої засвідчувальний центр в ієрархічній PKI, суб'єктом якої є сам користувач С. Аналогічно користувач D довіряє мостовому засвідчувальному центру тільки тому, що існує законна ланцюжок сертифікатів від засвідчувального центра, що випустив сертифікат для користувача D, до мостового засвідчувального центру. Користувачі А і В можуть використовувати міст довіри через мостовий засвідчувальний центр, щоб встановити контакти з користувачами С і D.

Використання взаємної сертифікації замість або разом з ієрархіями сертифікатів представляється більш захищеним рішенням, ніж чисто ієрархічна модель. Якщо в ієрархії довіри скомпрометований секретний ключ кореневого засвідчувального центру, то не слід покладатися на сертифікати всіх підлеглих йому засвідчувальних центрів. На противагу цьому, в мережній або гібридній архітектурі компрометація одного засвічуючого центру не обов'язково підриває довіру до всієї інфраструктурі відкритих ключів.

Фізична топологія

Система PKI крім виконання цілого ряду функцій – випуску сертифікатів, генерації ключів, управління безпекою, автентифікації, відновлення даних повинна забезпечувати інтеграцію із зовнішніми системами. PKI необхідно взаємодіяти з множиною найрізноманітніших систем і додатків – це і програмне забезпечення групової роботи, і електронна пошта, і системи управління доступом, і каталоги користувачів, і віртуальні приватні мережі, і різноманітні операційні системи, і служби безпеки, і web-додатки, і широкий спектр корпоративних систем. Рис. 10.4 ілюструє взаємодію компонентів PKI.

Функціональні компоненти PKI (ЗЦ, РЦ та ін) можуть бути реалізовані програмно і апаратно різними способами, наприклад, розташовуватися на одному або декількох серверах. Системи, що виконують функції посвідчувального і реєстраційного центрів, часто називають серверами сертифікатів та реєстрацією відповідно.

Основними серверними компонентами PKI є сервер сертифікатів, сервер каталогів і сервер відновлення ключів. На сервер сертифікатів покладаються функції випуску та управління сертифікатами, захищеного зберігання секретного ключа засвідчувального центра підтримки життєвого циклу сертифікатів і ключів, відновлення даних, ведення контрольного журналу і реєстрації всіх операцій засвідчувального центру.

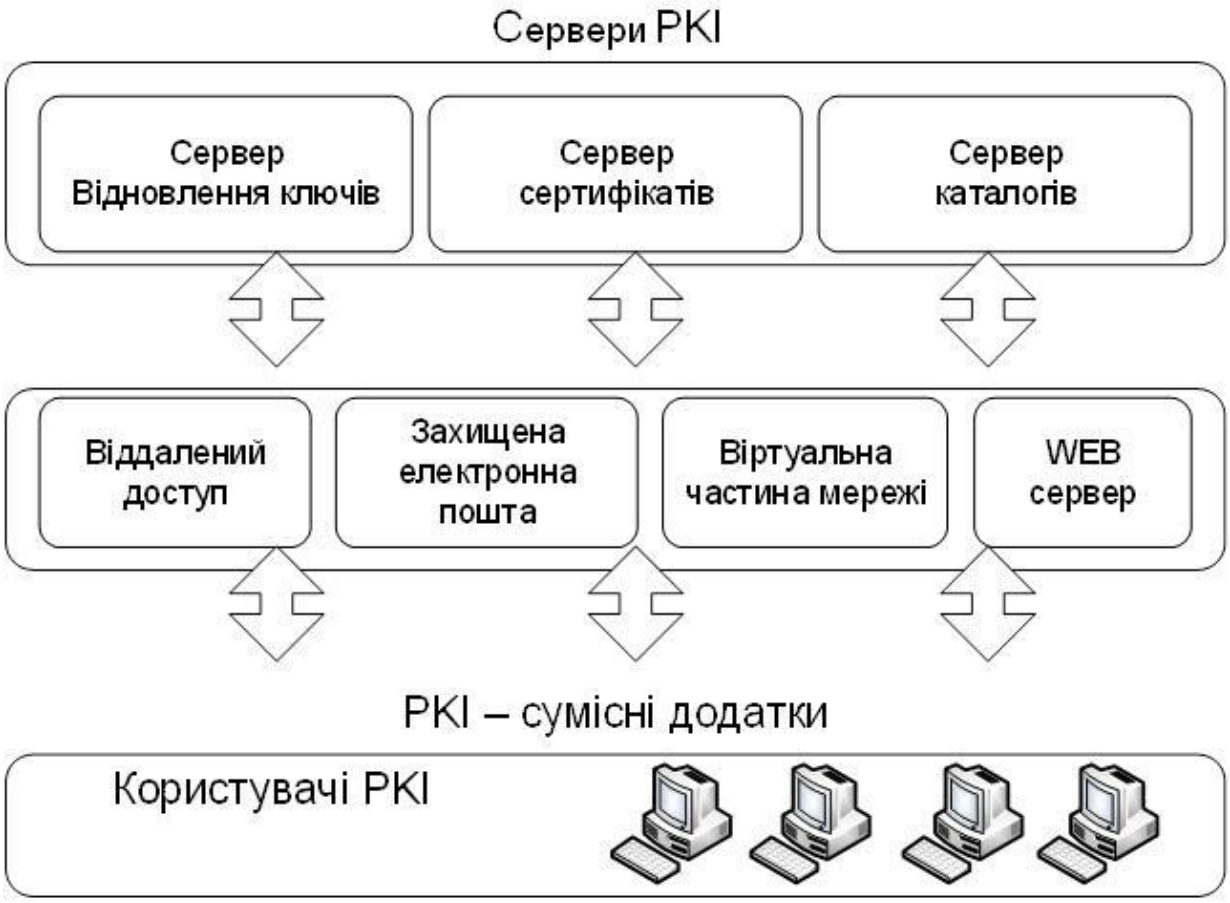


Рис. 10.4. **Взаємодія компонентів PKI**

Сервер каталогів містить інформацію про сертифікати та атрибути суб'єктів сертифікатів відкритих ключів. Через протокол LDAP додатка стандартним чином звертаються до записів каталогів, наприклад, до адрес електронної пошти, номерів телефонів і сертифікатів користувачів.

Сервер каталогів повинен забезпечувати:

мережну автентифікацію через IP-адреси або DNS-імена і автентифікацію кінцевих суб'єктів за іменами та паролями або за сертифікатами відкритих ключів;

управління доступом суб'єктів до інформації залежно від їх прав на виконання операцій читання, запису, знищення, пошуку або порівняння;

конфіденційність (за допомогою протоколу SSL) і цілісність повідомлень для всіх видів зв'язку.

Сервер відновлення ключів підтримує створення резервних копій і відновлення ключів шифрування кінцевих суб'єктів. Серед усіх компонентів PKI сервер відновлення ключів повинен бути найбільш захищений і забезпечувати сувору автентифікацію адміністратора

і користувачів, підтримку конфіденційності і цілісності повідомлень, безпечне зберігання всіх компонентів ключів.

PKI управляє ключами і сертифікатами, що використовуються для реалізації криптографічних операцій в web-браузерах, web-серверах, додатках електронної пошти, електронного обміну повідомленнями та даними, в додатках, що підтримують захищені мережні транзакції і сеанси зв'язку через Worid Wide Web або в віртуальних приватних мережах на базі протоколів S/MIME, SSL і IPSec, а також для запевнення цифровим підписом електронних документів або програмного коду. Поряд з перерахованими додатками, PKI-сумісними можуть бути і корпоративні додатки, розроблені усередині організації.

Програми електронної пошти і обміну повідомленнями використовують пари ключів для шифрування повідомлень і файлів і завірення їх цифровими підписами. Системи електронного обміну даними підтримують транзакції, що вимагають автентифікації сторін, забезпечення конфіденційності і цілісності даних. Браузери і web-сервери використовують шифрування для автентифікації, забезпечення конфіденційності і для таких додатків, як онлайнове надання банківських послуг та електронна комерція. Шифрування і автентифікація застосовуються також для створення віртуальних приватних мереж (Virtual Private Networks – VPN) на основі мереж загального користування, для захисту комунікацій "сайт-сайт" або віддаленого доступу (клієнт – сервер). Засвідчення цифровим підписом програмних кодів і файлів дає можливість користувачам підтвердити джерело вивантажуваних програм і файлів і цілісність їх вмісту, це важливо і для контролю вірусного зараження.

Існує безліч варіантів фізичної конфігурації корпоративної PKI. З метою безпеки зазвичай рекомендується, щоб основні компоненти PKI були реалізовані у вигляді окремих систем. Оскільки системи містять конфіденційні дані, вони повинні розміщуватися за корпоративним міжмережним екраном. Особливо важлива система засвідчувального центру, оскільки її компрометація потенційно здатна зруйнувати функціонування PKI і викликати необхідність повного оновлення системи. Бажано, щоб система засвідчувального центру розміщувалася за додатковим міжмережним екраном, тільки тоді вона буде захищена від внутрішніх атак і вторгнень через Інтернет. Очевидно, що міжмережний екран не повинен заважати взаємодії системи засвідчувального центру з іншими компонентами PKI.

У зв’язку з необхідністю оперативного отримання користувачами інформації про статус сертифікатів і даних з реєстрів сертифікатів засвідчувальних центрів різних PKI сервери каталогів повинні бути доступні через Інтернет. Однак деякі організації використовують сервер каталогів ширше, ніж просто сховище сертифікатів, зокрема для зберігання корпоративних даних, у тому числі і конфіденційних. У цьому випадку проблема захищеності даних вирішується таким чином: корпоративні користувачі отримують доступ до основного сервера каталогів, а всі інші зовнішні користувачі, системи та організації – до прикордонного сервера (рис. 10.5).

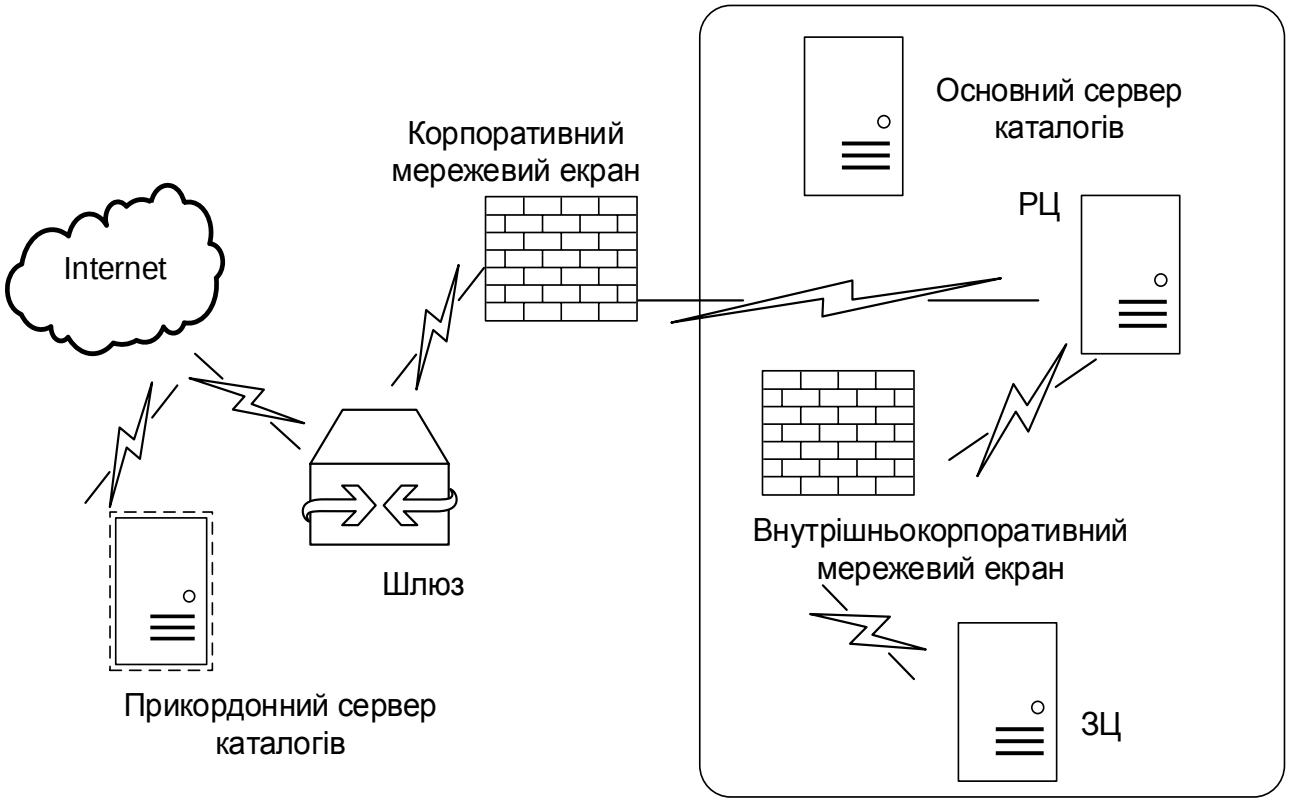


Рис. 10.5. **Фізична топологія PKI**

До основних загальних вимог безпеки корпоративної PKI належать:

продумана політика безпеки;

надійне програмне забезпечення компонентів PKI;

безпечний/надійний зв’язок між компонентами (наприклад, по протоколах IPSec, SSL тощо).

Кожен компонент, щоб бути частиною PKI, повинен задовольняти критерій безпеки. Цей критерій характеризує необхідний для цілей бізнесу рівень захищеності в межах допустимого рівня ризику.

Механізми безпеки, що забезпечують заданий рівень захищеності, зазвичай підрозділяють на механізми захисту апаратних засобів, комп'ютерної платформи, мережі і додатків. РК1-сумісні програми не дозволяють забезпечити повну безпеку корпоративної мережі і повинні бути доповнені іншими засобами захисту, наприклад, міжмережевим екранами, сервісами автентифікованих імен (службами імен) та суворим контролем адміністратора мережі.

10.5.5. Стандарти і специфікації PKI

Стандарти в області PKI

Стандарти відіграють істотну роль у розгортанні та використанні PKI. Стандартний підхід особливо важливий при регулюванні процедур реєстрації та вироблення ключа, завданні формату сертифіката й списку анульованих сертифікатів, формату криптографічно захищених даних і описі онлайнових протоколів. Стандартизація в області PKI дозволяє різним додаткам взаємодіяти між собою з використанням єдиної PKI.

Системи, що використовують сертифікати та PKI Результатом зусиль технічних фахівців щодо підвищення безпеки Інтернет стала розробка групи протоколів безпеки, таких, як S/MIME, TLS і IPSec (рис. 10.6).

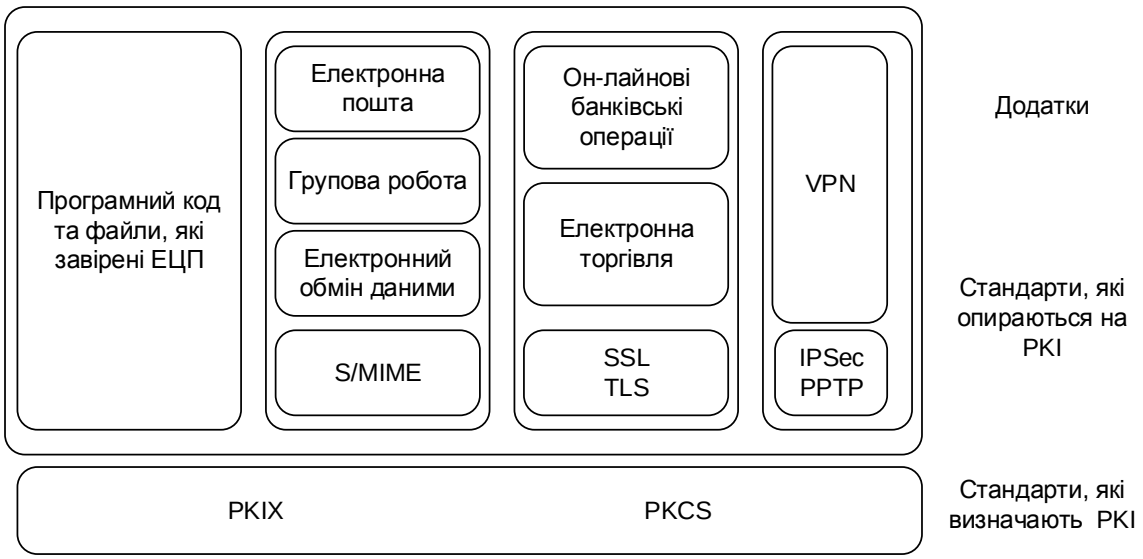


Рис. 10.6. **Взаємозв'язок стандартів в області PKI**

Усі ці протоколи використовують криптографію з відкритими ключами для забезпечення сервісів конфіденційністю, цілісністю даних, автентифікацією джерела даних і неспростовністю.

Мета PKI полягає в забезпеченні надійного і ефективного управління ключами і сертифікатами відкритих ключів. Користувачі систем, заснованих на PKI, повинні бути впевнені, що в будь-який момент часу при комунікації з певним суб'єктом вони покладаються на відкритий ключ, пов'язаний з секретним ключем, власником якого є саме цей суб'єкт.

Ця впевненість виникає в результаті використання сертифікатів відкритих ключів, що зв'язують значення відкритих ключів з їх власниками.

Зв'язування відбувається в результаті перевірки довіреною центром, що засвідчує особистості суб'єкта та запевнення цифровим підписом кожного сертифіката відкритого ключа.

Згідно зі стандартами PKIX, PKI є комплексом програмного і апаратного забезпечення, кадрів, а також політик і процедур, необхідних для створення, управління, зберігання, поширення та анулювання сертифікатів відкритих ключів. Функціональність і компоненти PKI представлені табл. 10.1 і 10.2.

Сертифікат відкритого ключа має обмежений період дії, який зафіксований у змісті сертифіката. Оскільки клієнт повинен мати можливість самостійно перевірити підпис сертифіката відкритого ключа та його термін дії, сертифікати повинні відкрито зберігатися в системах, що використовують сертифікати, і можуть розповсюджуватися через ненадійних комунікацій і систем серверів.

Таблиця 10.1

Функціональність PKI

<table>
<tr><td colspan="3">Функції PKI</td></tr>
<tr><td>Реєстрація</td><td>Реєстрація</td><td>Реєстрація</td></tr>
<tr><td>Ініціалізація</td><td>Ініціалізація</td><td>Ініціалізація</td></tr>
<tr><td>Сертифікація</td><td>Контроль періоду дії ключів</td><td rowspan="2">Публікація і розповсюдження сертифікатів і повідомлень про анулювання</td></tr>
<tr><td>Генерація ключів</td><td>Компрометація ключів</td></tr>
</table>

Таблиця 10.2

Компоненти PKI

Компонент	Опис
Засвідчуючі центри (ЗЦ)	Випускають і анулюють сертифікати
Реєстраційні центри (РЦ)	Підтверджують зв'язування відкритих ключів та особистостей власників сертифікатів та інших атрибутів
Власники сертифікатів	Підписують і шифрують електронні документи
Клієнти	Перевіряють справжність цифрових підписів і відповідних ланцюжків сертифікатів за допомогою відкритого ключа довіреної ЗЦ
Реєстри	Зберігають і роблять доступними сертифікати і CAC

Сертифікати відкритих ключів використовуються в процесі валідації (підтвердження) завірених цифровим підписом даних, коли одержувач перевіряє, щоб:

1) інформація, що ідентифікує відправника, відповідала даним, що містяться в сертифікаті;

2) жоден сертифікат з ланцюжка сертифікатів не був анульований, і в момент підписання повідомлення всі сертифікати були дійсними;

3) сертифікат використовувався відправником за призначенням;

4) дані не були змінені з моменту створення ЕЦП.

У результаті перевірок одержувач може прийняти дані, підписані відправником.

Загальна схема функціонування PKI наведена на рис. 10.7. Кінцевий суб'єкт відправляє запит на сертифікат в реєстраційний центр (транзакція управління). Якщо запит фактично схвалений, то направляється безпосередньо в засвідчувальний центр для запевнення цифровим підписом. Засвідчувальний центр перевіряє запит на сертифікат, і якщо той проходить верифікацію, то підписується і випускається сертифікат. Для публікації сертифікат направляється до реєстру сертифікатів, залежно від конкретної конфігурації PKI ця функція може бути покладена на реєстраційний або засвідчувальний центр.

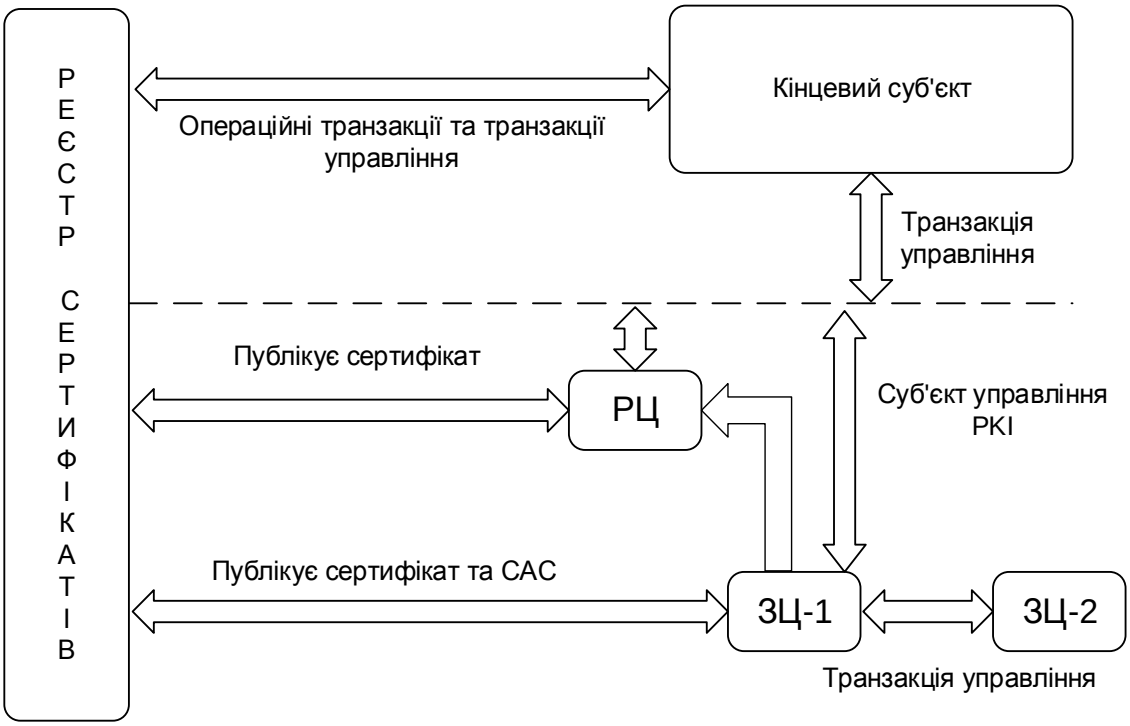


Рис. 10.7. **Схема функціонування PKI**

На рис. 10.7 показані всі можливі комунікації між кінцевим суб'єктом і засвідчувальним центром. Процес анулювання сертифіката аналогічний процесу його генерації. Кінцевий суб'єкт запрошує засвідчуючий центр про анулювання свого сертифіката, реєстраційний центр приймає рішення і направляє запит про анулювання в ЗЦ. Засвідчувальний центр вносить зміни у список анульованих сертифікатів та публікує його в реєстрі. Кінцеві суб'єкти можуть перевірити дійсність конкретного сертифіката через операційний протокол.

Операційні протоколи – це протоколи для доставки сертифікатів (або інформації про їх статус) і списків анульованих сертифікатів до клієнтських системам, що використовують сертифікати. Існують різноманітні механізми поширення сертифікатів і CAC з використанням протоколів LDAP, HTTP і FTP. Наприклад, пошук CAC для перевірки статусу сертифіката здійснює операційний протокол.

Протоколи управління потрібні для підтримки взаємодій в онлайновому режимі між користувачем PKI і суб'єктами управління.

Протоколи управління використовуються при:

1) реєстрацію суб'єкта для отримання сертифіката;
2) ініціалізації (наприклад, генерації пари ключів);
3) випуску сертифіката;
4) відновлення пари ключів;
5) оновленні пари ключів по закінченні терміну дії сертифіката;
6) звернення із запитом про анулювання сертифіката;
7) взаємної сертифікації, коли два засвідчують центру обмінюються інформацією для генерації взаємного сертифіката.

Політика застосування сертифікатів і регламент засвідчувального центру містяться в документах, що описують зобов'язання сторін і правила використання сертифікатів.

10.5.6. Управління сертифікатами

Вибір способу управління сертифікатами. Засвідчувальний центр відповідає за публікацію в реєстрі сертифікатів списку анульованих сертифікатів, сертифікати можуть публікуватися в реєстрі засвідчувальним центром, реєстраційним центром або кінцевим суб'єктом. У PKI може бути як один центральний сервіс каталогів, що надає сертифікати користувачам, так і декілька пунктів розповсюдження сертифікатів та

списків анульованих сертифікатів. Організація, яка використовує PKI, може відокремити сервіси автентіфікації від сервісів управління сертифікатами, в цьому випадку вона, діючи як реєстраційний центр, самостійно виконує автентифікацію користувачів і підтримує захищеність бази даних про своїх службовців, а частина функцій PKI з видачі сертифікатів, оновленню ключів і поновленню сертифікатів передає третій стороні. У цьому випадку відбувається і передача відповідальності за виконання цих функцій PKI, і організація мінімізує свою активність по адмініструванню інфраструктури.

Для функціонування PKI надзвичайно важливо правильне керування списками анульованих сертифікатів. Списки анульованих сертифікатів забезпечують єдиний спосіб перевірки дійсності використовуваного сертифіката, так як дата закінчення терміну дії, що вказується в сертифікаті, не може служити підтвердженням того, що даний сертифікат є дійсним.

Порядок обробки запитів про анулювання. При формуванні політики та розгортанні PKI повинен бути встановлений порядок обробки запитів про анулювання та встановлено коло осіб, які мають право звертатися з такими запитами. Зазвичай запит про анулювання сертифіката направляє його власник при втраті або компрометації секретного ключа. У деяких випадках із запитом про анулювання може звертатися не власник сертифіката, а інша особа. Наприклад, при звільненні службовця з компанії запит про анулювання його сертифіката може надійти від начальника підрозділу, в якому працював службовець. Крім того, запит про анулювання сертифіката може бути направлений з засвідчувального центру, який випустив сертифікат, або іншого підтверджуючого центра з мережі взаємної сертифікації, якщо виявляється, що власник сертифіката порушив вимоги політики безпеки або регламенту.

Після отримання запиту про анулювання сертифіката та автентифікації особи, яка направила запит, засвідчувальний центр відповідає за публікацію списку анульованих сертифікатів та внесення до нього змін. Для керування сертифікатами у відносно невеликій PKI зазвичай застосовується пряма публікація анульованих сертифікатів в CAC і забезпечується доступ до нього додатків, перевіряючих статус сертифіката. Деякі програми зберігають в пам’яті комп’ютера останню версію списку, що дозволяє додатку працювати в автономному режимі і підвищує його продуктивність. Збільшення масштабу PKI і необхідність керувати серти-

фікатами з декількох доменів породжує проблеми зберігання і обробки великих списків анульованих сертифікатів. У процесі вироблення політики і проектування PKI ці обставини повинні бути враховані, а також обрані спосіб публікації, пункти розповсюдження і тип списку анульованих сертифікатів.

Вибір способу публікації CAC. При виборі способу публікації слід оцінити переваги та недоліки кожного з трьох можливих способів (публікація з опитуванням наявності змін, примусова розсилка змін і онлайнова верифікація), характер РК1-транзакцій і степінь операційного ризику.

Публікація CAC з опитуванням наявності змін ("pull") виконується в певні заплановані моменти часу і може привести до ситуації, коли анульований сертифікат деякий час не включається до CAC, а користувачі продовжують покладатися на його дійсність. Цей спосіб підходить в більшості випадків, але піддає серйозному ризику клієнтів, які використовують критичні для ведення бізнесу додатки, навіть якщо плановані поновлення виконуються досить часто.

Спосіб примусової розсилки змін ("push") CAC підходить для PKI невеликих організацій, що використовують обмежену кількість PKI-додатків, і не годиться для PKI, обслуговуючих велика спільнота користувачів та численні додатки. Поширення списку цим способом вимагає вирішення проблем розпізнавання додатків, яким розсилається інформація про оновлення CAC, синхронізації випуску списку, а також отримання зазначеної інформації додатками, якщо останні були недоступні в момент розсилки.

Важливою перевагою способу онлайнової верифікації є своєчасність доставки (в реальному часі) інформації про анулювання сертифікатів, цей спосіб переважний для обслуговування додатків, що вимагають обов’язкової перевірки сертифікатів до виконання транзакції. Спосіб онлайнової верифікації встановлює жорсткі вимоги постійної захищеності сервера OCSP-респондеров і підписання всіх запитів до засвідчувальному центру цифровими підписами, що може створити "вузькі місця" при їх обробці.

Можливий шлях вирішення проблем поширення списку анульованих сертифікатів при розгортанні PKI – диференціювати сертифікати за призначенням і застосовувати різні способи публікації CAC для різних типів сертифікатів: онлайнову верифікацію для сертифікатів, використо-

вуваних у додатках, критичних для ведення бізнесу (наприклад, в електронній комерції), і "pull"-спосіб для сертифікатів інших типів.

Управління ключами. Управління ключами – суттєвий аспект успішного розгортання PKI. Проблеми управління ключами особливо актуальні для масштабних PKI з великою кількістю власників сертифікатів і користувачів. Політика управління ключами регулює порядок:

1) генерації, розповсюдження та використання;
2) поновлення, знищення та зберігання;
3) відновлення/резервного копіювання, зберігання в архіві;
4) депонування ключів.

Для досягнення високого рівня безпеки PKI необхідний продуманий вибір способу і процедури генерації ключів, а також довжини ключа.

Вибір способу генерації пари ключів. Генерація ключів може здійснюватися централізовано, засвідчувальним центром або за його дорученням реєстраційним центром, або індивідуально, тобто кінцевим суб'єктом. У більшості випадків пари ключів створюються кінцевими суб'єктами, які повинні мати програмні або апаратні засоби для створення надійних ключів. Цей спосіб дозволяє суб'єкту домогтися більшої конфіденційності при відношенні з довіряючим сторонами, оскільки секретний ключ власник зберігає сам і ніколи не пред'являє. На жаль, більшість користувачів не приймає достатніх заходів для захисту своїх секретних ключів, що піддає PKI серйозному ризику.

До переваг централізованої генерації можна віднести швидкість створення ключів, використання спеціалізованих засобів генерації високоякісних ключів, контроль відповідності алгоритмів генерації встановленим стандартам, а також зберігання резервних копій секретних ключів на випадок їх втрати користувачами. Якщо ключі генеруються централізовано, то політикою безпеки PKI повинні бути передбачені засоби їх захищеної транспортування інших компонентів PKI, а також гарантії того, що не буде здійснюватися паралельне несанкціоноване копіювання секретних ключів.

Вибір довжини ключа. Вибір довжини ключа залежить від типу використовуваних додатків та обчислювальної потужності (доступної в момент проектування і прогнозованою в найближчому майбутньому) комп'ютерної бази конкретної організації, розгортається PKI. Довжини ключів, зазвичай використовуваних в PKI, складають 512, 768 і 1024 біта, такі ключі іноді називають ключами відповідно нижчого, середнього і

вищого ступені в стійкості. Нещодавно доведено, що 512-розрядні ключі можуть бути зламані за наявності достатньої обчислювальної потужності комп'ютерної техніки, яка поки недоступна для більшості організацій. Тим не менше, ключі нижчого ступеня стійкості ще кілька років можуть вважатися досить хорошими для використання в додатках, що працюють з несекретними даними, наприклад повідомленнями електронної пошти. В інших випадках при проектуванні PKI слід орієнтуватися на застосування ключів середнього і вищого ступенів стійкості.

Політика PKI повинна регулювати частоту звернення користувачів за новою парою ключів. Ключі доцільно оновлювати через такі проміжки часу, які дозволяють уникнути проблем, що виникають при одночасному закінчення строків дії великої кількості сертифікатів, і дають можливість кожному користувачеві володіти декількома сертифікатами з різними термінами дії. Для ключів, які використовуються в критичних для ведення бізнесу додатках, потрібно більш часте оновлення.

Вибір терміну дії ключа. Як правило, пари ключів діють більш тривалий час, ніж сертифікати, тим не менше, з ряду причин термін дії ключів слід обмежувати. З часом ключі стають більш уразливими для атак з боку криптоаналітиків і можуть бути скомпрометовані. Тривале використання ключа шифрування піддає ризику розкриття велику кількість документів, зашифрованих за його допомогою. При виборі терміну дії ключа слід враховувати той факт, що через деякий час у зв'язку з новими науково-технічними досягненнями його захищеність може виявитися істотно нижче, ніж очікувалося при генерації ключа. Так, наприклад, нещодавно було продемонстровано, що ключ, згенерований відповідно до стандарту DES, в умовах нових технологій не є достатньо надійним, хоча в 1976 році його надійність не піддавалася сумніву.

Порядок поновлення ключів. Політикою PKI повинен бути визначений порядок дій у разі поновлення пар ключів. Пари ключів можуть оновлюватися вручну й автоматично. При ручному оновленні відповідальність за своєчасне формування запиту про оновлення покладається на кінцевого суб'єкта, який повинен пам'ятати дату закінчення терміну дії сертифіката. Якщо запит про оновлення не буде вчасно спрямований у засвідчувальний центр, суб'єкт позбудеться сервісу PKI. При автоматичному оновленні система PKI сама відслідковує дату закінчення терміну дії сертифіката й ініціює запит про оновлення ключа відповідному засвідчувальному центру.

Політика безпеки організації може передбачати, наприклад, щоб всі документи, зашифровані старими ключами, розшифровувалися і знову зашифровувати за допомогою нових ключів або щоб будь-які документи, підписані раніше старим ключем, підписувалися за допомогою нового ключа. Раціональна політика управління ключами допускає п'ятирічний (і навіть більше) термін дії пари ключів, але може обмежувати період дії ключів шифрування суворо конфіденційних даних декількома місяцями. Іноді конкретний термін дії ключів не встановлюється, а ключі замінюються в разі необхідності, наприклад, при втраті секретного ключа. У цьому випадку слід переоцінювати рівень захищеності використовуваної пари ключів по закінченню п'яти років або при появі нових криптографічних алгоритмів або інших технологічних досягнень.

Вибір способу зберігання секретного ключа. При проектуванні PKI повинен бути вибраний спосіб зберігання криптографічних ключів, він, як правило, залежить від специфіки діяльності конкретної організації. Для обмеження доступу до секретних ключах застосовуються такі механізми:

1. Захист за допомогою пароля. Пароль або PIN-код використовується для шифрування секретного ключа, який зберігається на локальному жорсткому диску. Цей метод вважається найменш безпечним, оскільки проблема доступу до ключа вирішується підбором пароля.

2. Карти PCMCIA. Ключ захищено зберігається на карті з мікрочіпом, але при введенні в систему "покидає" карту, отже, стає вразливим для розкрадання.

3. Пристрої зберігання секрету. Секретний ключ зберігається в зашифрованому вигляді в спеціальному пристрої і витягується тільки за допомогою одноразового коду доступу, наданого пристроєм. Цей метод безпечніший, ніж згадані вище, але вимагає доступності пристрої зберігання кінцевого суб'єкта і не виключає втрати пристрою.

4. Біометричні засоби. Ключ захищається біометричними засобами автентіфікації власника ключа, при цьому забезпечується той же самий рівень захисту, що й у попередньому випадку, але суб'єкт позбавляється необхідності мати при собі пристрій зберігання секрету.

5. Смарт-карти. Ключ зберігається на смарт-карті з чіпом, який забезпечує можливість виконувати операції шифрування і цифрового підпису. Ключ ніколи не покидає карту, тому ризик його компрометації

низький. Однак власник ключа повинен носити смарт-карту з собою і піклуватися про її збереження. При втраті смарт-карти зашифровані за допомогою секретного ключа дані можуть виявитися невідновними.

Порядок відновлення, резервного копіювання та зберігання ключів в архіві. Дуже важливими аспектами управління ключами є створення резервних копій і відновлення ключів, оскільки суб'єктам будь-яких PKI властиво втрачати свої секретні ключі. У разі втрати секретного ключа кінцевого суб'єкта засвідчувальний центр повинен анулювати відповідний сертифікат відкритого ключа, після цього повинна бути згенерована нова пара ключів і створений новий сертифікат відкритого ключа. Сервер відновлення ключів забезпечує копіювання секретних ключів у момент їх створення та відновлення їх згодом. В екстримальній ситуації при втраті ключа підпису самого засвідчувального центру стають неможливими випуск сертифікатів та підписання списку анульованих сертифікатів, тобто компрометується весь домен довіри. Політикою безпеки резервного копіювання і відновлення повинен бути визначений формат резервних копій ключів (звичайний текст, зашифрований текст або ключ по частинах) і визначений порядок роботи з персоналом, відповідальним за процедури резервного копіювання і відновлення, ведення контрольних журналів, матеріалів архіву, підтримки секретних ключів посвідчує та реєстраційного центрів і кінцевих суб'єктів.

При розробці процедур зберігання ключів та іншої інформації в архіві повинні бути вибрані об'єкти, що підлягають зберіганню, період зберігання і особи, відповідальні за архів і мають доступ до нього, детально описані події, що фіксуються в контрольних журналах, способи пошуку і захисту від спотворень архівної інформації, процедури проставлення позначки часу. В силу типовості операцій створення резервних копій, архівування та копіювання до будь копіям даних повинні застосовуватися ті ж строгі правила, які поширюються на оригінал.

Вибір способу і агента депонування ключів. При розгортанні PKI на додаток до функцій резервного копіювання і відновлення ключів може бути запланована підтримка депонування ключів. Під депонуванням ключів розуміється надання копій секретних ключів третій стороні і дозвіл користуватися ними при певних обставинах, в якості третьої сторони найчастіше виступають урядові установи і правоохоронні органи. Депонування ключів може бути покладено на незалежний підрозділ усередині організації, розгортається PKI, або на зовнішнє агентство.

Один із способів депонування ключів і підтримки високого рівня безпеки полягає в шифруванні секретних ключів відкритим ключем агента депонування і передачі їх на локальне зберігання під контроль власників ключів або іншої уповноваженої особи. При необхідності відновити секретний ключ зашифрований ключ знову передається агенту депонування для розшифрування за допомогою його секретного ключа.

Альтернативним способом депонування всередині організації є розподіл ключа на дві частини, шифрування кожної частини відкритими ключами різних осіб (наприклад, офіцерів безпеки) і локального зберігання під контролем власників ключів або уповноваженої особи. Крім того, для депонування і роздільного зберігання двох частин секретного ключа підпису користувача можна використовувати смарт-карти.

Вибір способу і агента депонування здійснюється з урахуванням фінансових можливостей, вимог безпеки і особливостей діяльності організації, розгортається PKL.

Життєвий цикл сертифікатів і ключів. Політикою PKI має бути чітко визначено, в який момент часу сертифікати та ключі стають дійсними і як довго зберігають свій статус, а також коли необхідно їх заміняти або відновлювати.

Найважливішим питанням у розумінні можливих правових наслідків застосування електронного цифрового підпису є питання: коли сертифікат стає дійсним. Випуск сертифіката відкритого ключа та підписання його засвідчувальним центром після автентіфікації особи, що звертається із запитом про видачу сертифіката, не є достатньою умовою для додання до сертифіката статусу дійсного. Як зазначалося раніше, сертифікат стає дійсним тільки після його відкритої публікації в реєстрі засвідчувального центру, і навпаки, сертифікат втрачає статус дійсного після його включення в список анульованих сертифікатів та публікації останнього. Деякі засвідчуючі центри вимагають, щоб користувач, який звертається із запитом про видачу сертифіката, став передплатником засвідчувального центру, перш ніж сертифікат цього користувача буде опублікований в реєстрі.

На практиці сертифікат може бути відправлений користувачеві разом з договором передплатника з умовою, що користувач не буде використовувати сертифікат, поки формально не підпише договір. Як тільки засвідчує центр отримує згоду користувача, то публікує сертифікат

у відкритому реєстрі, надаючи сертифікату правову силу, після цього сертифікат і відкритий ключ стають загальнодоступними в PKI.

Користувачі мають потребу в сертифікатах різних за рівнем безпеки та додатковим можливостям управління сертифікатами. Зазвичай користувач має, принаймні, дві пари ключів: одну пару для шифрування, а іншу – для електронного цифрового підпису. Політикою PKI повинні бути визначені типи випускаючих сертифікатів та їх терміни дії. Взагалі кажучи, теоретично сертифікати можуть діяти протягом тривалого часу, але з практичних міркувань багато сертифікати мають обмежений термін дії, який дозволяє зменшити ризик їх неправильного вживання.

На практиці більшість персональних сертифікатів діють протягом одного-двох років після випуску, а сертифікати серверів зазвичай зберігають свою силу два роки і більше. Для цілей архівування та довготривалого шифрування використовуються спеціальні сертифікати з тривалим періодом дії.

Рис. 10.8 ілюструє життєвий цикл сертифіката, стрілки, що відображають нормальний життєвий цикл, виділені більш яскраво на відміну від тих стрілок, якими позначені моменти втручання посвідчувально або реєстраційного центрів. Так, наприклад, в корпоративній PKI, де власниками сертифікатів є службовці організації, втручання засвідчувального центру у нормальний життєвий цикл сертифіката вимагається у випадках:

1) анулювання сертифіката при звільненні службовця, що володіє цим сертифікатом;

2) анулювання сертифіката при втраті службовцем свого секретного ключа чи пароля доступу до секретного ключа;

3) призупинення дії сертифіката, випущеного для службовця, який в даний момент часу звільняється або перебуває під слідством;

4) поновлення сертифіката службовця при відмові від звільнення або після прояснення обставин судової справи тощо.

Іноді в PKI випускаються сертифікати з різними термінами дії для службовців залежно від їх статусу, наприклад, службовці, що працюють за контрактом, можуть мати сертифікати на період їх запланованої роботи, а постійні працівники – сертифікати, поновлювані через кожні 12 місяців.

Рис. 10.8. **Життєвий цикл сертифіката**

Зразкові сценарії управління життєвим циклом сертифікатів і ключів. Розглянемо можливі сценарії управління життєвим циклом сертифікатів і ключів на прикладі інфраструктури відкритих ключів, припускаючи, що політикою застосування сертифікатів встановлений термін дії сертифіката відкритого ключа – 1 рік, секретного ключа – 10 років, цифрового підпису – 25 років з моменту підписання електронного документа.

У прикладі 1 на рис. 10.9 секретний ключ використовується для підписання ділових контрактів.

Рис. 10.9. **Секретний ключ використовується для підписання ділових контрактів**

Оскільки термін дії секретного ключа 10 років, і він створювався на початку 2000 року, то повинен зберігатися до початку 2010 року. На рисунку символом X в середині 2001 року позначений момент підписання документа, який буде діяти до середини 2026 року.

Цифровий підпис цього документа залишається дійсним після закінчення терміну дії секретного ключа, використаного для створення цього підпису, тому відкритий ключ повинен зберігатися довше секретного, оскільки він продовжує використовуватися для верифікації цифрового підпису та після закінчення дії секретного ключа. Цілком імовірно, що інший електронний документ буде підписаний в кінці 2009 року безпосередньо перед тим, як закінчиться термін дії секретного ключа, отже, відкритий ключ повинен зберігатися, принаймні, до 2035 року, бо він може знадобитися для верифікації цифрового підпису через 25 років після підписання документа.

У період 2010 – 2035 років секретний ключ не може бути скомпрометований, оскільки знищується або зберігається в архіві захищеним. Таким чином, немає необхідності встановлювати більш тривалий термін зберігання сертифіката відкритого ключа. Альтернативою довготривалого зберігання в PKI ключів, використовуваних для верифікації електронного цифрового підпису, може бути організація роботи надійної системи, спроектованої за типом нотаріальної.

На рис. 10.10 наведений інший приклад, більш складний: компрометація особистого ключа підпису (цей момент позначений символом X на початку 2002 року). Якщо останній документ був підписаний за допомогою секретного ключа (до його компрометації) на початку 2002 року, то відкритий ключ повинен залишатися доступним до початку 2027 року, отже, сертифікат відкритого ключа повинен бути доступний, незважаючи на його публікацію в списку анульованих сертифікатів.

При розгортанні PKI і виробленні політики слід аналізувати всі можливі сценарії управління життєвим циклом сертифікатів і ключів і оцінювати наслідки компрометації ключів. Політика PKI, повинна визначати типи користувачів сертифікатів, типи додатків, в яких будуть використовуватися сертифікати, життєвий цикл сертифікатів та умови втручання в нього посвідчує або реєстраційного центрів, а також враховувати вимоги функціонування організації або ведення бізнесу.

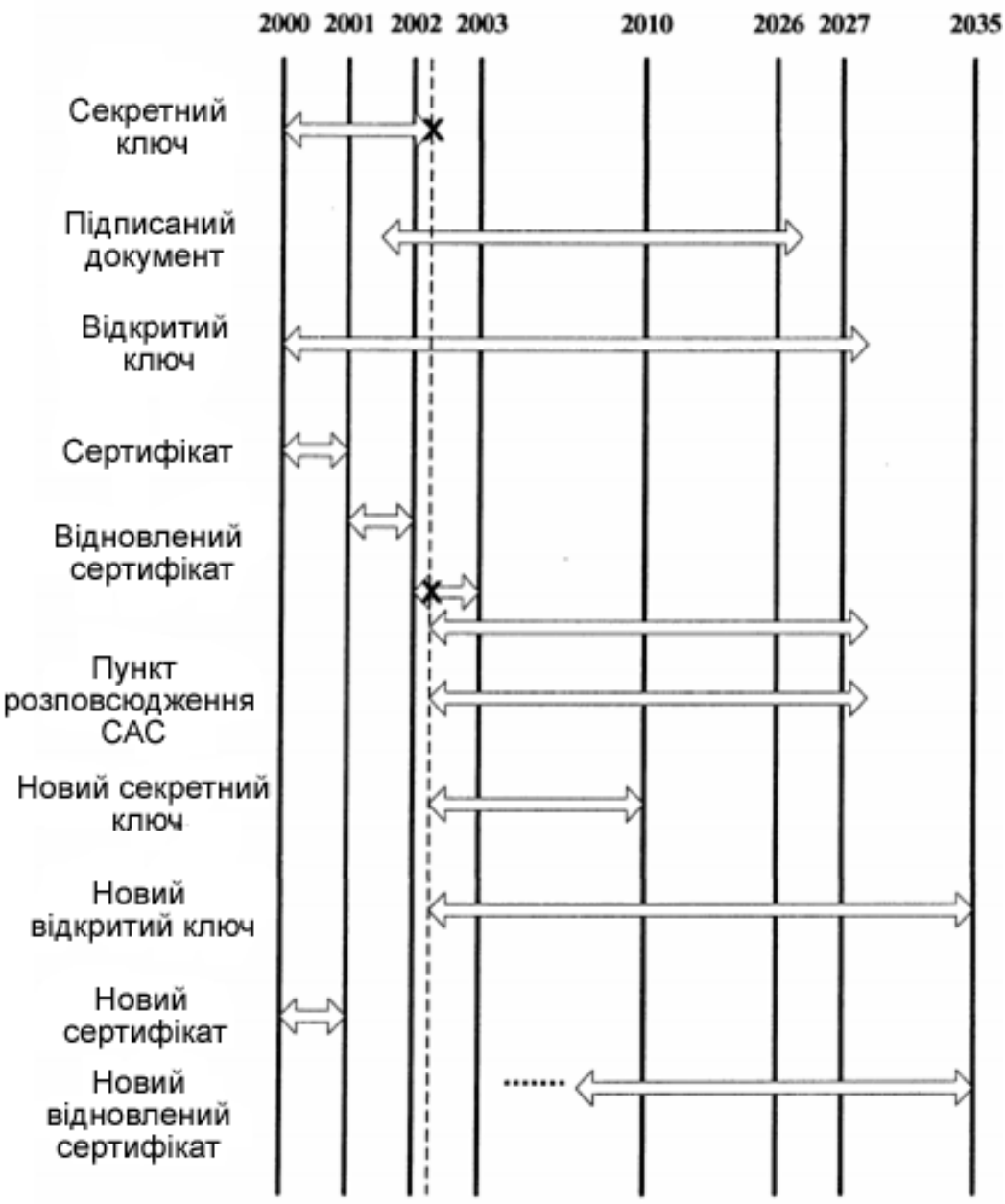


Рис. 10.10. **Компрометація особистого ключа підпису**

Контрольні запитання

1. Основні положення керування ключами. Життєвий цикл криптографічного ключа.

2. Керування ключами на основі симетричних методів.

3. Керування ключами на основі симетричних методів.

4. Безпека керування ключами.Протоколи забезпечення безпеки ключів.

5. Структура та призначення PKI. Життєвий цикл сертифікатів і ключів.

Розділ 11. Методи та пристрої забезпечення захисту і безпеки

11.1. Основні принципи захисту інформації при підключенні до мережі Інтернет

Для підключення будь-якої організації до мережі Інтернет необхідно прийняти ряд певних організаційно-технічних заходів для її захисту.

При побудові захисту варто виходити з того, що будь-який захист ускладнює використання системи, що, за прямим призначенням обмежує функціональні можливості, споживає обчислювальні й трудові ресурси, вимагає фінансових витрат на створення та експлуатацію. Чим вище захист, тим дорожчою у побудові та обслуговуванні стає система і тим менш зручною для безпосередніх користувачів [1; 4; 5; 9; 10; 17 – 23; 26; 32 – 35; 43; 45]. Тому, захищаючи мережу, варто виходити з доцільної вартості захисту. Тобто витрати на захист повинні бути пропорційні цінності ресурсу, що захищає. Існує ряд основних принципів, що дозволяють організувати досить безпечне підключення до Інтернет порівняно простими засобами.

Firewall (Брандмауер).

Основним загальновизнаним засобом такого захисту є міжмережний екран (Брандмауер) [23; 26; 43].

Міжмережний екран установлюється між мережею та Інтернет і виконує роль мережного фільтра (рис. 11.1).

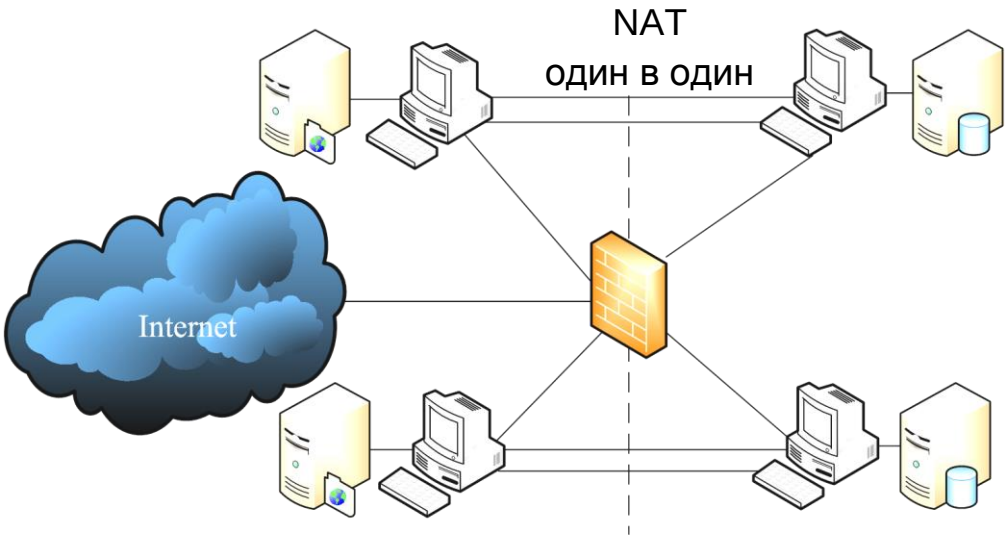


Рис. 11.1. **Встановлення брандмауера у локальній мережі**

Він настроюється таким чином, щоб пропускати допустимий трафік від користувачів мережі до служб Інтернет і назад, і обмежити трафік з боку Інтернет до мережі, яка потребує захисту, тільки необхідними службами, наприклад: smtp, dns, ntp.

Допустимість того або іншого трафіка визначається мережним адміністратором відповідно до політики інформаційної безпеки організації. Наприклад, може бути дозволений доступ із частини комп'ютерів мережі до web та ftp-серверів Інтернет і двонаправлений доступ між Інтернет та поштовим сервером, але при цьому заборонені всі інші протоколи й напрями трафіка.

Таким чином, міжмережний екран фізично розташовується на місці мережного шлюзу (маршрутизатора), логічно доцільно сполучити їх функції в одному пристрої. Це дозволяє одним засобом захистити й локальну мережу і безпосередньо сам шлюз. Така опція передбачена для маршрутизаторів компанії Cisco Systems (Firewall Feature Set). Однак дане правило є необов'язковим і міжмережний екран може бути поданий окремим пристроєм.

У найпростішому випадку виконання функцій міжмережного екрана можна організувати за допомогою мережного фільтра на основі аркушів доступу (access-lists). Аркуші доступу визначають правила, за якими або дозволяється, або забороняється проходження трафіка з певними ознаками від одного мережного інтерфейсу маршрутизатора до іншого усередині самого маршрутизатора. Як ознаки можуть використовуватися IP-адреси або діапазон, IP-адреса джерела й приймача, тип протоколу, номер порту призначення або відправлення, ряд інших службових ознак IP-пакета.

Відмінність і недолік аркушів доступу порівняно із сьогоднішнім міжмережним екраном полягає у тому, що вони дозволяють створити статичний однобічний фільтр, тоді як мережне з'єднання становить динамічний процес. Аркуші доступу не дозволяють контролювати параметри IP-пакета, що залежать від попередніх пакетів. Звідси виникає складність застосування аркушів доступу для тонкого настроювання фільтрації трафіка в точній відповідності із прийнятою політикою безпеки. Зокрема, із цієї причини аркуші доступу не в змозі захистити від такого різновиду мережної атаки, як "викрадення з'єднання", або "хайджекінг".

У Firewall Feature Set зазначені проблеми вирішуються за допомогою того, що він відслідковує кожне мережне з’єднання окремо і контролює весь процес у динаміку. При встановленні нового TCP-сеансу міжмережний екран створює для нього новий процес, що контролює правильність з’єднання до самого моменту його завершення. При цьому кожний пакет на транспортному рівні перевіряється на відповідність попередній, а всі "підозрілі" пакети відбраковуються. Завдяки цьому стає можливим досить легко організувати фільтр для доступу внутрішнього комп’ютера до зовнішнього, але не дозволяє зовнішньому комп’ютеру самостійно звернутися до внутрішнього.

Іншими словами, у настроюваннях міжмережного екрана задаються правила для проходження трафіка від одного інтерфейсу до іншого, для кожного напряму й кожного тракту окремо. Якщо правило дозволяє проходження IP-пакета від інтерфейсу внутрішньої мережі до Інтернет-інтерфейсу, то на підставі такого пакета формується логічний тунель у маршрутизаторі, через який уже можуть пройти відповідні пакети від зовнішнього одержувача. Як тільки з’єднання закрите, або вичерпаний час очікування, тунель закривається, і обіг ззовні до внутрішнього комп’ютера буде відкинутий. З цієї ж причини екран не пропустить пакети у зворотному напрямі, якщо ініціатором з’єднання є зовнішній комп’ютер.

Крім того, міжмережний екран, на відміну від аркушів доступу, може контролювати зміст IP-пакетів у полі даних і відбраковувати пакети, що містять потенційно-небезпечні коди, наприклад, java-апліти. Є міжмережні екрани, здатні виявити в IP-пакетах ознаки відомих мережних атак і перервати таке з’єднання, але це вже досить дорогі системи.

З найбільш дешевих систем слід зазначити *Firewall* на основі ядра операційної системи Linux версії 2.4.20 і вище й засоби керування iptables. Через те, що Linux є безкоштовною ОС, витрати на побудову такого міжмережного екрана зводяться до придбання звичайного персонального комп’ютера із двома мережними інтерфейсами. Проте Linux дозволяє побудувати досить надійний і гнучкий мережний фільтр, що розпізнає окремі прапори в службових полях IP-пакета.

NAT.

Другою цеглинкою забезпечення захищеності мережі є "заміна мережної адреси" – Network Address Translation, або NAT. Вона становить заміну в IP-пакеті реальної адреси комп’ютера внутрішньої мережі на будь-яку іншу задану адресу при посиланні його в зовнішню

мережу. Завдяки цьому для внутрішньої мережі стає можливим використання діапазонів адрес, які не застосовуються в Інтернет (наприклад, 10.0.0.0 – 10.255.255.255). Це дозволяє запобігти прямому обігу ззовні до внутрішніх комп'ютерів і приховує структуру мережі. Існує кілька різновидів NAT [23; 26; 43].

Найпростіша й найбільш марна з погляду захисту – це трансляція фіксованої внутрішньої адреси у фіксовану зовнішню. При цьому противник безперешкодно "бачить" такий комп'ютер у зовнішній мережі, тому що йому однозначно відповідає певна зовнішня адреса. Однак вона необхідна при організації сервера, до якого потрібно забезпечити доступ ззовні (рис. 11.2).

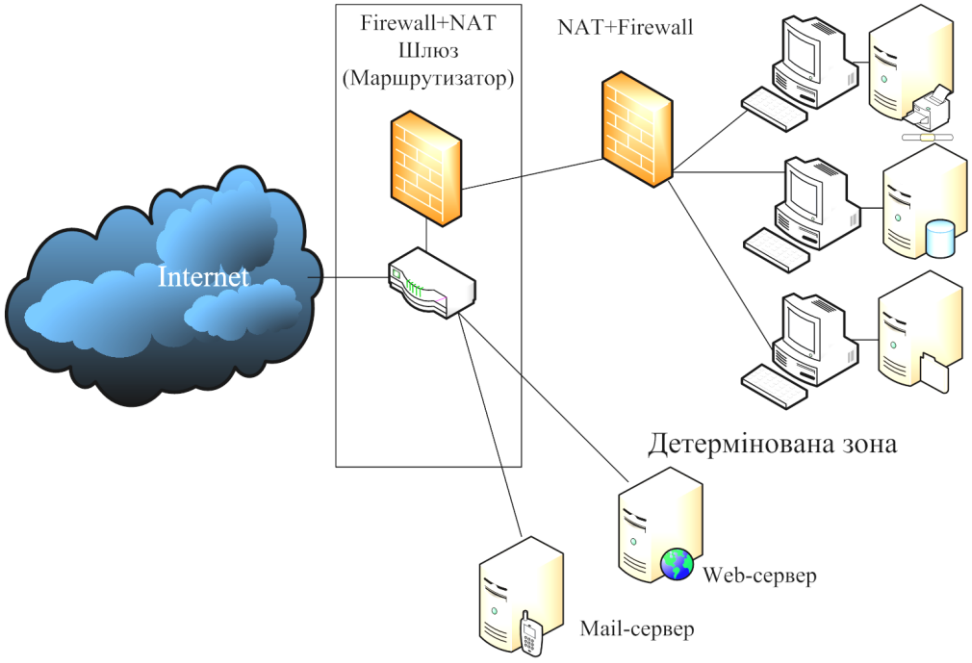


Рис. 11.2. **Трансляція фіксованої внутрішньої адреси у фіксовану зовнішню**

Друга форма NAT – це трансляція групи внутрішніх адрес в одну зовнішню. При цьому всі внутрішні комп'ютери можуть працювати з Інтернетом одночасно, а маршрутизатор розрізняє, кому яка відповідь перетрансльовується за службовими даними *TCP*-з'єднання. У зовнішній мережі створюється враження, що до неї звертається тільки один комп'ютер. Така заміна істотно ускладнює життя противнику, тому що повністю приховує внутрішні комп'ютери й перешкоджає "обчисленню" жертви (рис. 11.3). Противник, навіть бачучи трафік, що виходить

із внутрішньої мережі, не може визначити, від якого комп'ютера він виходить. Крім того, це виключає можливість ініціативного обігу ззовні до внутрішнього комп'ютера, тому що для маршрутизатора в цьому випадку відсутнє правило прив'язки зовнішньої адреси до внутрішньої. Зокрема виключається можливість сканування ззовні внутрішньої мережі.

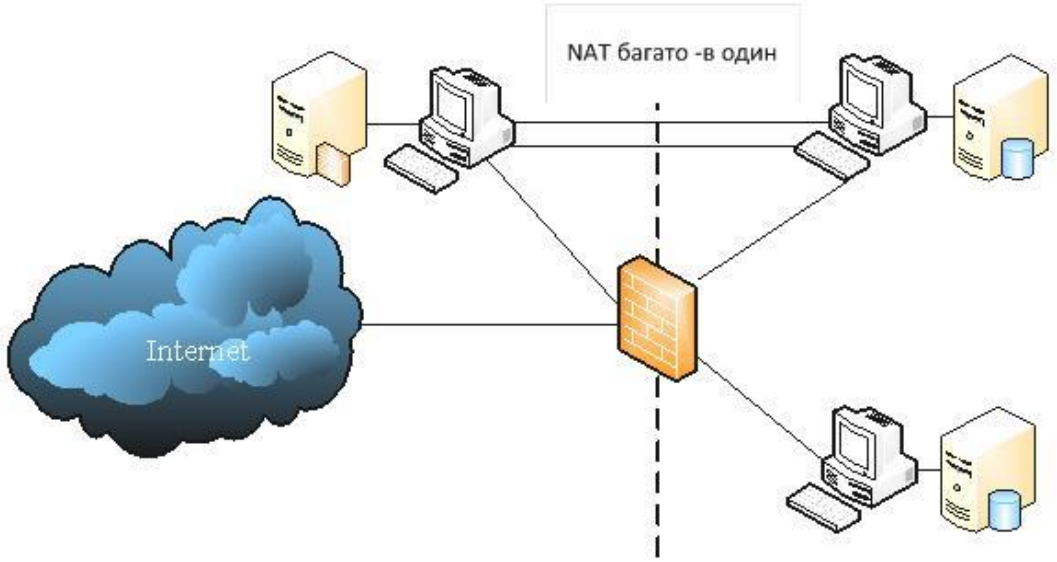


Рис. 11.3. **Трансляція групи внутрішніх адрес в одну зовнішню**

Третя форма NAT – використання для заміни внутрішніх адрес не однієї адреси, а будь-якої з виділених адрес. Тобто внутрішній комп'ютер, виходячи в Інтернет, одержує вільну у цей момент адресу з бази даних (БД). При цьому адреси підмінюються динамічно, і кожне нове *TCP*-з'єднання може бути встановлене з іншою IP-адресою. Це також створює додаткові труднощі противнику, тому що позбавляє його можливості атакувати будь-який внутрішній комп'ютер прицільно. Сказане відносно другої форми NAT є справедливим і для третьої форми. Якщо запит приходить ззовні, то маршрутизатор не в змозі зв'язати адресу з БД з адресою мережі. Тому такий запит не досягне мети.

Демілітаризована зона.

Як правило, організації потрібно мати у себе деякі мережні ресурси, до яких відкритий доступ з мережі Інтернет. Звичайно це поштовий, dns і web-сервери [23; 26; 43]. Механізм їх роботи допускає, що до них повинен бути дозволений вільний або слабко обмежений обіг з Інтернету. Відповідно ймовірність їх зламу вища, ніж інших комп'ютерів мережі. Із цієї причини розміщати їх усередині зони, яка захищається, недоцільно

з погляду безпеки, тому що у випадку зламу вони можуть стати воротами для атаки внутрішніх комп'ютерів. Для мінімізації ризику і збереження функціональності такі сервери встановлюють за основним шлюзом мережі, але перед міжмережним екраном, що забезпечує захист внутрішніх комп'ютерів. Логічну область їх розміщення називають демілітаризованою зоною (рис. 11.4).

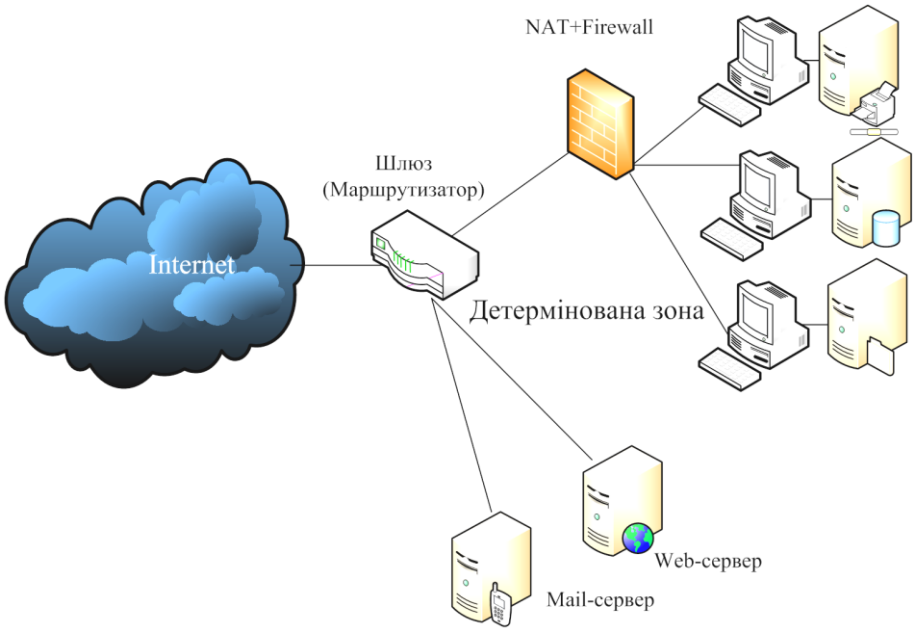


Рис. 11.4. **Демілітаризована зона**

Другий firewall.

З рис. 11.4 видно, що ніщо не заважає встановити другий *Firewall* на основному шлюзі мережі. Це є логічним рішенням і дозволяє одночасно підвищити рівень захисту внутрішньої мережі й захистити сервери демілітаризованої зони. При правильному настроюванні обох міжмережних екранів противнику буде вже набагато сутужніше дістатися до внутрішньої мережі організації. Наявність другого міжмережного екрана ускладнює конфігурування мережного встаткування й настроювання роботи всіх елементів мережі. Для додаткового підвищення захищеності можна використати *Firewall*-и різних виробників. Тоді якщо в одному з них буде виявлена вразливість, інший не дозволить противнику безперешкодно проникнути у мережу, як це мало б місце при використанні *Firewall*-ів одного типу [23; 26; 43].

Особливо варто підкреслити, що можливість мережного доступу до шлюзів і до міжмережних екранів, щоб уникнути зловмисного використ-

тання, повинна бути відключена. З погляду безпеки пристрої, які знаходяться на сторожі мережі, повинні конфігуруватися й адмініструватися тільки через консольний порт локально (рис. 11.5).

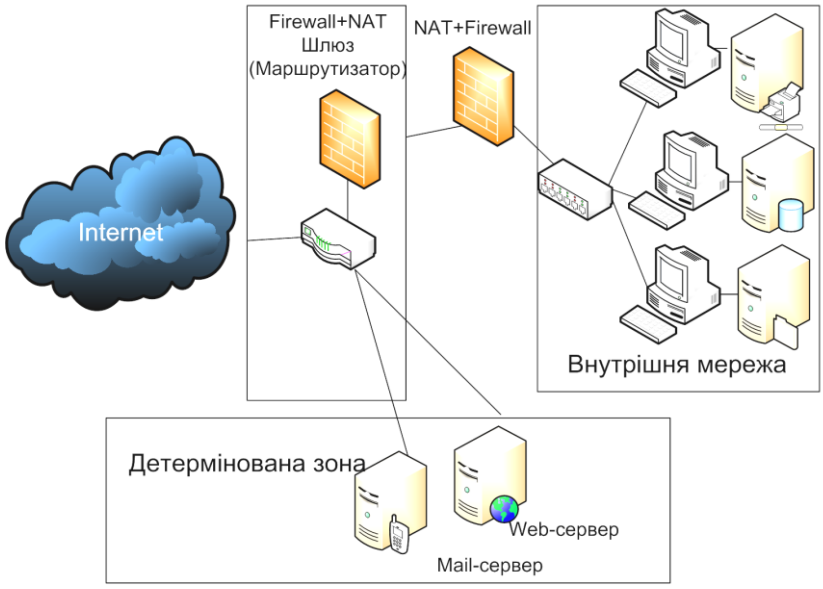


Рис. 11.5. **Локально-консольний порт для серверів**

Схема, запропонована на рис. 11.5, може бути дещо вдосконалена. Для цього необхідно використати граничний маршрутизатор із двома Ethernet-портами (див. рис. 11.6).

Proxy-сервер.

Використання так званого "посередника" (proxy-сервера) також підвищує рівень захищеності мережі, тому що виключає необхідність прямого виходу в Інтернет комп'ютерів користувачів. При цьому також стає можливим більш строгий контроль за даними в IP-пакетах на рівні мережних додатків. Proxy-сервер працює як посередник між користувальницьким додатком і вилученим мережним ресурсом в Інтернет. Схематично сутність його роботи показана на рис. 11.6.

Proxy-сервер складається ніби із двох частин – клієнтської і серверної. Клієнтська частина дивиться у бік Інтернету, серверна – у бік клієнтського комп'ютера. Коли клієнтський комп'ютер звертається до вилученого сайта через proxy-сервер, його клієнтський мережний додаток взаємодіє із серверною частиною proxy-сервера. При цьому *proxy*-сервер на рівні додатка передає клієнтський запит своєї клієнтської частини, і вона вже від імені proxy-сервера надсилає даний запит на вилучений сайт. Тобто *IP*-пакет, що відправлений, має адресу *proxy*-сервера.

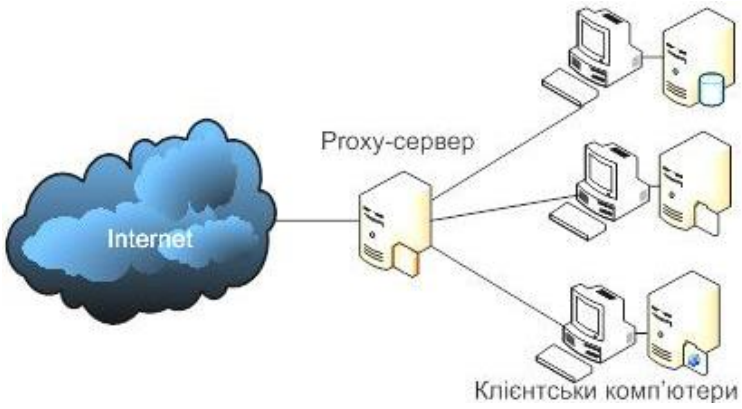


Рис. 11.6. **Proxy-сервер**

Потім отримана відповідь передається у зворотну сторону від клієнтської частини *proxy*-сервера його серверної частини, з якою безпосередньо взаємодіє користувальницький комп'ютер. Таким чином, пряме з'єднання клієнтських комп'ютерів з вилученим сайтом виключається. Усередині *proxy*-сервера передача даних між клієнтською частиною й серверною відбувається вже не на транспортному рівні, а на рівні протоколу додатка, ніж забезпечується легкість контролю команд і даних на відповідність установленим стандартам. Крім того, це дозволяє забезпечити досить надійний контроль проти передачі зловмисних кодів усередині даних. Навіть у випадку успішної атаки з боку Інтернет за відкритими протоколами у цьому випадку буде ушкоджений тільки proxy-сервер, що не представляє інформаційної цінності, а користувальницькі комп'ютери будуть залишатися в безпеці ще якийсь час.

Через те, що proxy-сервер працює тільки за декількома відомими протоколами (HTTP, FTP та інших) і не пропускає через себе інші пакети, він сильно обмежує можливості противника з використання мережних "троянських коней" для закріплення на будь-якому з користувальницьких комп'ютерів.

Другий mail-сервер.

Залишати *mail*-сервер у демілітаризованій зоні з одного боку небажано, бо на ньому фактично зберігається поштова база даних з перепискою локальних користувачів, а демілітаризована зона не може забезпечити належного рівня захисту мережним ресурсам. З іншого боку, якщо сховати mail-сервер усередині локальної мережі, то він або не зможе взаємодіяти із зовнішнім середовищем, або буде становити ворота із зовнішнього середовища у внутрішню локальну мережу, якими потенційно зможе скористатися противник [23; 26; 43].

Внаслідок цього доречним рішенням є використання двох поштових серверів. Основний сервер установлюється усередині мережі, яка захищена, і його не видно для зовнішнього світу. Всі локальні користувачі поштової системи заводяться на нього і мають до нього прямий доступ. Відповідно вся вхідна кореспонденція зберігається на ньому у поштових скриньках локальних користувачів. Відправлення електронної пошти також здійснюється через нього.

Другий, або зовнішній, поштовий сервер установлюється в демілітаризованій зоні й забезпечує взаємодію по e-mail з мережою Інтернет. Він настроюється таким чином, щоб всю пошту, що приходить на ім'я користувачів організації, відразу пересилати на внутрішній поштовий сервер. У такий спосіб у його поштовій базі даних немає ні одного облікового запису користувачів організації і жоден аркуш не відкладається для довгострокового зберігання. Тому якщо він виявиться зламаним зловмисником, то противник не одержить доступу до накопиченої переписки. Проте після зламу противник одержує можливість перехоплення й читання транзитної пошти. Тому потрібен ретельний контроль за подібною ситуацією й негайне вживання заходів при підозрі на НСД (несанкційного доступу).

Перевагою такої схеми є те, що навіть зі зламаного зовнішнього поштового сервера не так просто дістатися до внутрішньої захищеної мережі. Обмін даними між зовнішніми й внутрішнім поштовими серверами відбувається через міжмережний екран з єдиним дозволеним портом (SMTP) за єдиною дозволеною парою адрес. Звертання до інших комп'ютерів і по інших протоколах буде блокуватися. Тому впливати з нього прямо на комп'ютери користувачів внутрішньої мережі неможливо.

Антивірусний захист поштової системи.

Операційна система Windows дуже вразлива перед деякими різновидами поштових вірусів. Користувачу буває досить встановити покажчик на інфікований конверт, щоб вірус активізувався. Але більш небез-печним є те, що механізм роботи поштових вірусів може бути використаний зловмисником для закидання в область мережного "троянського коня", якими захищається. Він дозволить противнику таємно скачувати дані мережі та здобути всю інформацію, що цікавить. Тому забезпеченню антивірусного захисту тракту доставки пошти у внутрішню мережу варто приділити досить серйозну увагу [23; 26; 43].

Існує ряд програмних засобів, призначених для контролю кореспонденції на поштових серверах на предмет наявності в ній вірусів у процесі прийому й пересилання електронної пошти.

Принцип її роботи полягає в тому, що вся пошта, що проходить через сервер, спочатку перенаправляється спеціальному користувачу, у ролі якого виступає антивірусний процес. Він сканує зміст кожного аркуша на наявність у ньому фрагментів відомих вірусів. Якщо аркуш містить щось схоже на вірус, воно вилучається із процесу передачі й, залежно від настроювань антивірусу, піддається заданій обробці. Повідомлення про виявлений вірус відсилаються відправнику й одержувачу інфікованого аркушу, а також на ім'я зазначених адміністраторів системи. Після перевірки аркуші, що невикликають підозри, відсилаються за призначенням.

Тим самим на рівні поштового сервера ставиться надійний захист відомим вірусам у електронній пошті. Через те, що антивірусна програма розпізнає тільки віруси, сигнатури яких перебувають у її базі даних, необхідно регулярно обновляти антивірусну базу даних з офіційного сайта. Інакше мережа може стати вразливою для знову створених вірусів.

Log-сервер.

Загальновідомий механізм протоколювання системних подій на серверах і клієнтських робочих станціях. Розроблювачі програмного забезпечення включають у свої продукти фрагменти коду, які на ту або іншу подію генерують відповідні текстові повідомлення, що посилають операційній системі. Система збирає дані повідомлення в log-файлах, які потім можуть аналізуватися адміністратором або користувачем з метою з'ясування, які події відбувалися в системі деякий час потому. Це дозволяє, наприклад, з'ясувати, чому не запускається та або інша програма, або чому припинив функціонувати певний сервіс. Дуже корисні log-файли для пошуку слідів зламу системи й відвідування її несанкціонованими гостями. Через те, що злом, як правило, супроводжується множиною заборонених дій, це породжує велику кількість системних повідомлень, що осідають в log-файлах [23; 26; 43]. Із цієї причини противник завжди прагне стерти сліди своєї присутності, або видаливши log-файли, або їх підчистивши. В обох випадках адміністратору після цього буде важко зрозуміти, що ж відбулося в системі насправді – яким чином у неї проникнули, як довго в ній перебували, ніж встигли покористуватися. Або навіть просто переконатися, що все добре.

Тому обов’язковою умовою для мережі, підключеної до Інтернету, є наявність у ній окремого log-сервера.

Принцип його роботи полягає в тому, що кожна операційна система може посилати повідомлення про системні події за UDP-протоколом на вилучений сервер. Це можуть робити також маршрутизатори й міжмережні екрани.

Збираючи такі повідомлення на спеціально виділеному сервері, забезпечується їм схоронність від рук противника. Тому для мінімізації ймовірності зламу log-сервер повинен бути призначений тільки для збору log-повідомлень. Він не повинен виконувати будь-яких інших функцій і виконувати інші мережні додатки, крім syslogd.

У цьому випадку після зламу будь-яких комп’ютерів мережі на log-сервері залишаться відповідні повідомлення, знищити які противник уже не зможе. Таким чином, у результаті найбільш оптимальною є наступна схема підключення локальної мережі до Інтернет (рис. 11.7).

Таким чином, проведений аналіз способів захисту комп’ютерних мереж при підключені їх до глобальної мережі Інтернет показав, що для забезпечення захисту при обміні інформацією абоненти локальної мережі повинні використовувати принципи і засоби безпеки в комплексі з організаційними заходами.

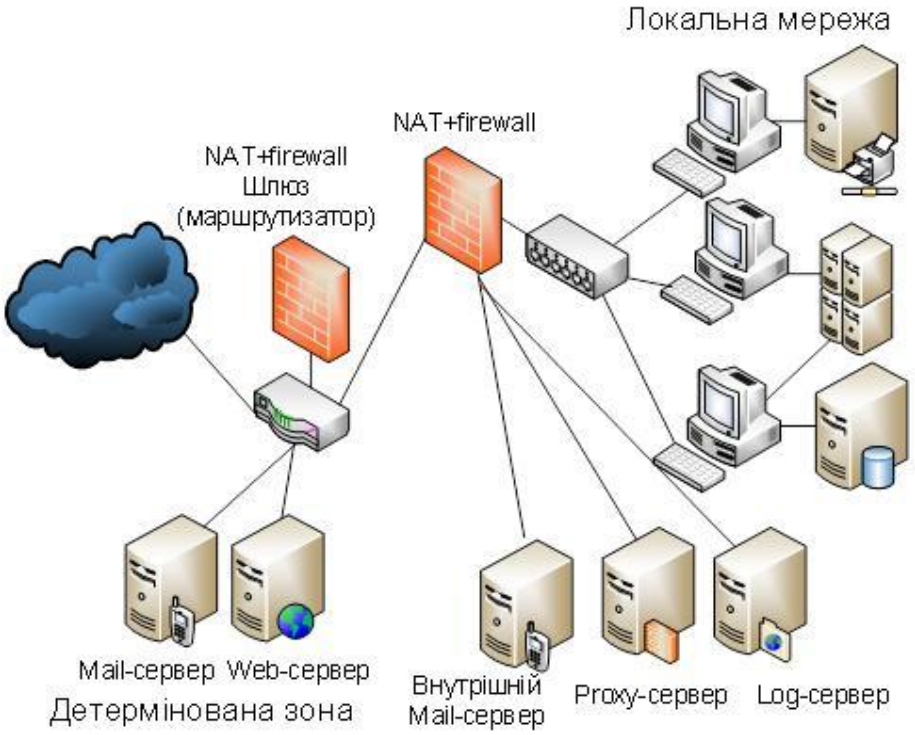


Рис. 11.7. **Схема з *Log*-сервером**

Це дозволить надійно захистити від атак як активних, так і пасивних противників.

11.2. Захист інформації за допомогою міжмережних екранів

Серед *програмно-апаратних* і *програмних засобів забезпечення автентичності,* розподілених КМ, можна виділити міжмережеві екрани (ММЕ), засоби аналізу захищеності й засоби виявлення атак.

Міжмережеві екрани (брандмауери, firewall) реалізують набір правил, які визначають умови проходження пакетів даних з однієї частини розподіленої КМ (відкритої) в іншу (захищену). Залежно від рівня взаємодії об'єктів мережі основними різновидами ММЕ є фільтруючі *маршрутизатори, шлюзи сеансового й прикладного рівнів*. Основною функцією фільтруючих маршрутизаторів, що працюють на мережному рівні еталонної моделі, є фільтрація пакетів даних, що входять у захищену частину мережі або вихідних з неї. Правила фільтрації визначають, дозволяється або блокується проходження через ММЕ пакета із правилами, що задаються цими параметрами.

До основних переваг фільтруючих маршрутизаторів відносяться простота їх створення, установка й конфігурування; прозорість для додатків користувачів КМ і мінімальний вплив на їх продуктивність; невисока вартість.

Недоліками фільтруючих маршрутизаторів є:

відсутність автентифікації на рівні користувачів КМ;

уразливість для підміни *IP*-адреси в заголовку пакета;

незахищеність від погроз порушення конфіденційності й цілісності переданої інформації;

сильна залежність ефективності набору правил фільтрації від рівня знань адміністратора ММЕ конкретних протоколів;

відкритість *IP*-адрес комп'ютерів захищеної частини мережі.

Шлюзи сеансового рівня призначені для контролю віртуального з'єднання між робочою станцією захищеної частини мережі й хостом її незахищеної частини і трансляції *IP*-адрес комп'ютерів захищеної частини мережі.

У процесі виконуваного шлюзом сеансового рівня процедури трансляції *IP*-адрес відбувається їхнє перетворення в одну *IP*-адресу, асоційовану із ММЕ. Це виключає пряму взаємодію між хостами захищеної й відкритої мереж і не дозволяє порушнику здійснювати атаку шляхом підміни *IP*-адрес.

До переваг шлюзів сеансового рівня відносяться їх простота й надійність програмної реалізації. Недоліком є відсутність можливості перевіряти вміст переданої інформації. Це дозволяє порушнику намагатися передати пакети зі шкідливим програмним кодом і звернутися потім прямо до одного із серверів, що атакується КМ.

Шлюзи прикладного рівня не тільки виключають пряму взаємодію між уповноваженим користувачем із захищеної частини мережі й хостом з її відкритої частини, але й фільтрують усі вхідні й вихідні пакети даних на прикладному рівні (на основі аналізу змісту переданих даних).

Основні функції шлюзів прикладного рівня:

ідентифікація й автентифікація користувача КМ при спробі встановити з'єднання;

перевірка цілісності переданих даних;

розмежування доступу до ресурсів захищеної й відкритої частин розподіленої КМ;

фільтрація і перетворення переданих повідомлень (виявлення шкідливого програмного коду, шифрування й розшифрування та ін.);

реєстрація подій у спеціальному журналі;

кешування запитуваних ззовні даних, розміщених на комп'ютерах внутрішньої мережі (для підвищення продуктивності КМ).

Перевагами шлюзів прикладного рівня також є:

прихованість структури захищеної частини мережі для інших хостів;

надійна автентифікація й реєстрація минаючих повідомлень;

більш прості правила фільтрації пакетів на мережному рівні, відповідно до яких маршрутизатор повинен пропускати тільки трафік, призначений для шлюзу прикладного рівня, і блокувати весь інший трафік;

можливість реалізації додаткових перевірок.

Основними недоліками шлюзів прикладного рівня є більш висока вартість, складність розробки, установки й конфігурування, зниження продуктивності КМ, "непрозорість" для додатків користувачів КМ.

Міжмережеві екрани є основою для створення віртуальних приватних мереж (Virtual Private Network, VPN), які призначені для приховання топології внутрішніх мереж організацій, що обмінюються інформацією з мережею Інтернет, і захисту трафіка між ними. При цьому використовуються спеціальні системи маршрутизації.

Загальним недоліком ММЕ будь-якого виду є те, що ці програмно-апаратні засоби захисту в принципі не можуть запобігти багатьох видів

атак, наприклад, погрози несанкціонованого доступу до інформації з використанням неправильного сервера служби доменних імен мережі Інтернет, погрози аналізу мережного трафіка, погрози відмови в обслуговуванні. Порушнику реалізувати погрозу доступності інформації в КМ, що використовує ММЕ, може виявитися навіть простіше, тому що досить атакувати тільки хост із ММЕ для фактичного відключення від зовнішньої мережі всіх комп'ютерів захищеної частини мережі.

11.3. Захист інформації на мережному рівні

11.3.1. Протокол IPSec

Розглянемо протоколи мережевої безпеки IPSec, досліджуємо застосовувані механізми забезпечення цілісності, автентичності та конфіденційності даних.

Інтернет Protocol Security (IPSec) – це узгоджений набір відкритих стандартів, що має на сьогоднішній день конкретну специфікацію, який, в той же час, може бути доповнений новими протоколами, алгоритмами та функціями мережевої безпеки .

Основне призначення протоколів IPSec – забезпечення безпечної передачі даних IP-мережами. Їх застосування забезпечує:

цілісність, тобто здатність телекомунікаційної мережі забезпечувати передачу даних без спотворення, втрати або дублювання;

автентичність, тобто здатність телекомунікаційної мережі забезпечувати передачу даних з можливістю підтвердити їх достовір-ність, тобто дійсність того, що дані передані саме тим відправником, за кого він себе видає;

конфіденційність, тобто здатність телекомунікаційної мережі забезпечувати передачу даних у формі, що запобігає їх несанкціонованому перегляду.

Специфікація IP Security (відома сьогодні як IPSec) розробляється робочою групою IP Security Protocol IETF. Спочатку IPsec включав 3 алгоритмо-незалежні базові специфікації, опубліковані в якості RFC-документів "Архітектура безпеки IP", " Автентифікований заголовок (AH)", "Інкапсуляція зашифрованих даних (ESP)" (RFC1825, 1826 і 1827). У листопаді 1998 року робоча група IP Security Protocol запропонувала нові версії цих специфікацій, що мають в даний час статус попередніх стандартів, це RFC2401 – RFC2412. Версії RFC1825-27 впродовж останніх декількох років вважаються застарілими і реально не використ-

товуються. Робоча група IP Security Protocol розробляє також і протоколи управління ключовою інформацією. Завданнями цієї групи є розробка Інтернет Security Association and Key Management Protocol (ISAKMP), протоколу управління ключами прикладного рівня, не залежного від використовуваних протоколів забезпечення безпеки.

Основними компонентами IPsec є:

RFC2402 "IP Authentication Header" (AH), призначений для контролю цілісності та автентичності пакетів даних в IP-мережах;

RFC2406 "IP Encapsulation Security Payload" (ESP), призначений для забезпечення конфіденційності, контролю цілісності та автентичності пакетів даних у IP-мережах;

RFC2408 "Інтернет Security Association and Key Management Protocol" (ISAKMP), призначений для забезпечення узгодження параметрів, створення, зміни, знищення контекстів захищених з'єднань (Security Association, SA) і управління ключами в IP-мережах;

RFC2409 "The Інтернет Key Exchange" (IKE), є подальшим розвитком і адаптацією ISAKMP, призначений для роботи з протоколами IPSec.

Ядро IPSec складають три протоколи (рис. 11.8): протокол автентичності (Authentication Header, AH), протокол шифрування (Encapsulation Security Payload, ESP) і протокол обміну ключами (Інтернет Key Exchange, IKE). Функції з підтримання захищеного каналу розподіляються між цими протоколами таким чином:

протокол AH забезпечує цілісність і автентичність даних;

протокол ESP шифрує дані, що передаються, гарантуючи конфіденційність, але він також може підтримувати автентифікацію та цілісність даних;

протокол IKE вирішує допоміжну задачу автоматичного надання секретних ключів, необхідних для роботи протоколів автентифікації і шифрування даних.

Можливості протоколів AH і ESP частково перекриваються. Протокол AH відповідає тільки за контроль цілісності і автентифікації даних, в той час, як протокол ESP дозволяє шифрувати дані, та виконувати функції протоколу AH (в обмеженому вигляді). Для забезпечення цілісності та автентифікації пакетів даних використовуються спеціальні механізми контролю цілісності та автентичності, які засновані присвоєнням у дані, що передаються спеціально сформованої надмірності (коди контролю цілісності та автентичності).

Для забезпечення ефективного функціонування протоколів AH і ESP використовується протокол IKE, який встановлює між двома кінцевими точками логічне з'єднання, яке в IPSec носить назву "безпечна асоціація" (Security Association, SA) .

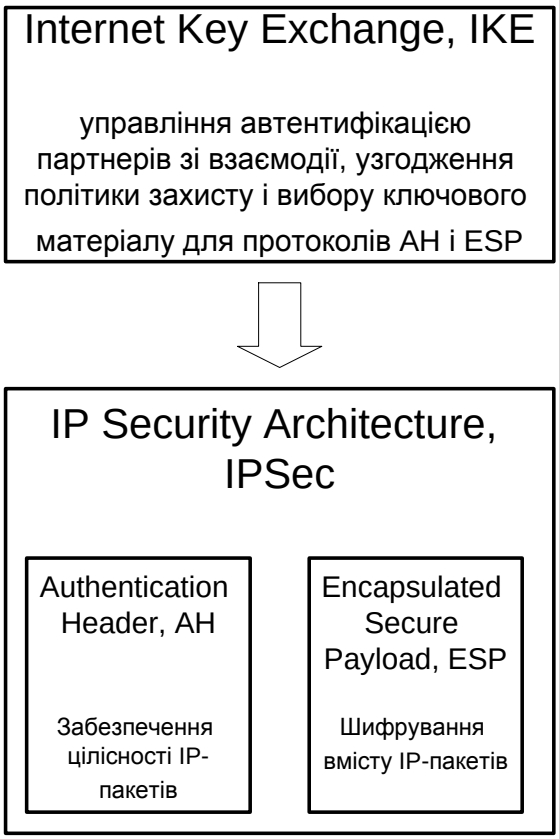


Рис. 11.8. **Основні компоненти протоколу мережевої безпеки IPSec**

Встановлення SA починається з взаємної автентифікації сторін. Обрані далі параметри SA визначають, який з двох протоколів, AH чи ESP, застосовується для захисту даних, які функції виконує протокол захисту: наприклад, тільки автентифікацію та перевірку цілісності або, крім того, ще й захист від помилкового відтворення.

Протоколи AH і ESP забезпечують захист даних у двох режимах: транспортному і тунельний (рис. 11.9, 11.10).

У транспортному режимі (рис. 11.9) передача IP-пакета виконується за допомогою оригінального заголовка цього пакета даних. Перевагою такого режиму є істотно менші обчислювальні та комунікаційні витрати. В той же час, з точки зору забезпечення безпеки телекомунікаційної мережі, для транспортного режиму функціонування протоколів AH і ESP притаманні такі недоліки: протокол ESP в транспортному режимі не захищає

заголовок пакета даних; неможливо приховати топологію мережі, оскільки заголовки пакетів даних передаються у відкритому (не захищеному) вигляді.

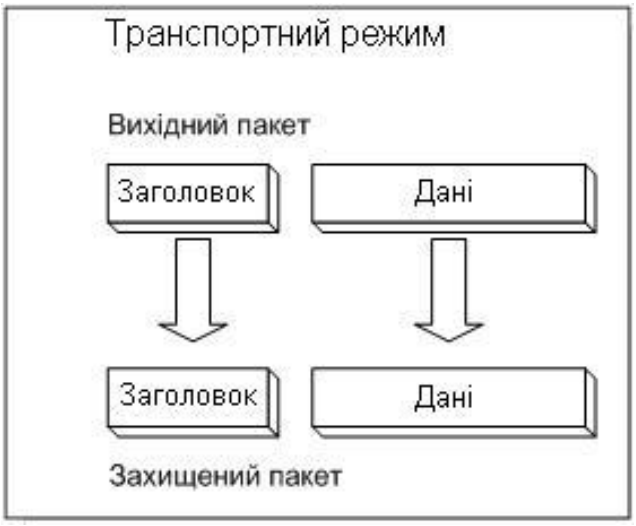


Рис. 11.9. **Схема захисту пакета даних у транспортному режимі функціонування протоколів АН і ESP**

У тунельному режимі (рис. 11.10) вихідний IP-пакет поміщається в новий, після чого здійснюється передача даних мережою виконується на підставі заголовка нового IP-пакета.

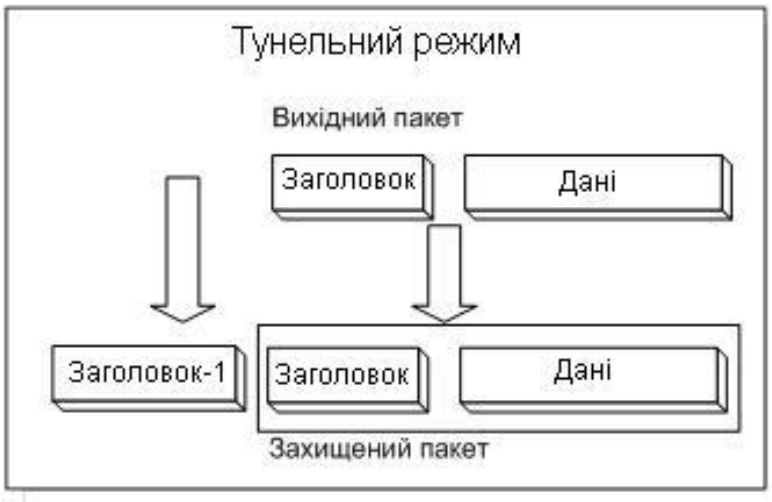


Рис. 11.10. **Схема захисту пакета даних у тунельному режимі функціонування протоколів АН і ESP**

Цей режим забезпечує захист заголовка пакета даних, у результаті чого ховається топологія мережі, що є безумовною перевагою при побудові захищених телекомунікаційних систем і мереж. У тож же час

реалізація тунельного режиму вимагає великих обчислювальних і комунікаційних ресурсів.

Застосування того чи іншого режиму залежить від вимог, що висуваються до захисту даних, а також від ролі, яку відіграє в мережі вузол, завершальний захищений канал. Так, вузол може бути хостом (кінцевим вузлом) або шлюзом (проміжним вузлом). Відповідно, є три схеми застосування IPSec: "хост-хост", "шлюз-шлюз" і "хост-шлюз".

У першій схемі захищений канал (безпечна асоціація, SA), встановлюється між двома кінцевими вузлами телекомунікаційної мережі (рис. 11.11). Протокол IPSec в цьому випадку працює на кінцевому вузлі і захищає дані, що надходять на нього. Для схеми "хост-хост" найчастіше використовується транспортний режим захисту, хоча дозволяється і тунельний.

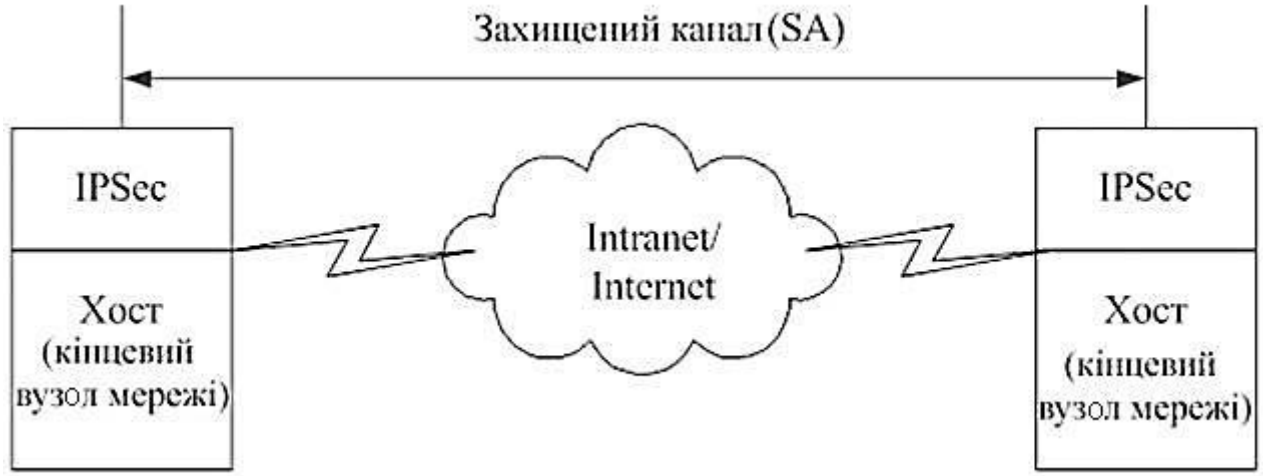


Рис. 11.11. **Схема організації захищеного каналу "хост-хост"**

Рис. 11.12. **Схема організації захищеного каналу "шлюз-шлюз"**

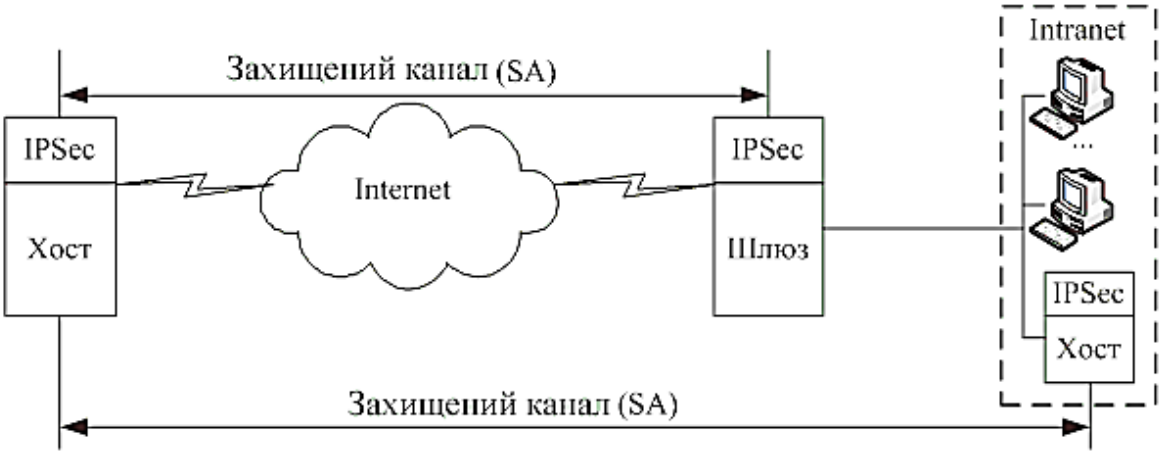


Рис. 11.13. **Схема організації захищеного каналу "хост-шлюз" з додатковим каналом "хост-хост"**

Відповідно до другої схеми (рис. 11.12), захищений канал встановлюється між двома проміжними вузлами, так званими шлюзами безпеки (Security Gateway, SG), на кожному з яких працює протокол IPSec. Захищений обмін даними може відбуватися між будь-якими двома кінцевими вузлами, підключеними до мереж, які розташовані позаду шлюзів безпеки. Від кінцевих вузлів підтримка протоколу IPSec не потрібна, вони передають свій трафік у незахищеному вигляді через мережу Intranet, яка заслуговує довіри підприємств. Трафік, що направляється в загальнодоступну мережу, проходить через шлюз безпеки, який і забезпечує його захист за допомогою IPSec, діючи від свого імені. Шлюзи можуть використовувати тільки тунельний режим роботи.

Схема "хост-шлюз" (див. рис. 11.13) часто застосовується при віддаленому доступі. Тут захищений канал організується між віддаленим хостом, на якому працює IPSec, і шлюзом, який захищає трафік для всіх хостів, що входять в мережу Intrenet організації. Віддалений хост може використовувати при відправці пакетів шлюзу як транспортний, так і тунельний режим, в свою чергу шлюз відправляє пакет хосту тільки в тунельному режимі. Цю схему можна ускладнити, створивши паралельно ще один захищений канал – між віддаленим хостом і яким-небудь хостом, що належить внутрішній мережі, який захищається шлюзом. Таке комбіноване використання двох SA дозволяє надійно захистити трафік і у внутрішній мережі.

Розглянемо реалізацію захисту пакетів даних з використанням протоколів AH і ESP в IP-мережах.

11.3.1.1. Забезпечення цілісності та автентичності даних в IP-мережах з використанням протоколу AH (IPSec)

Автентифікований заголовок (AH) є звичайним опціональним заголовком і, як правило, розташовується між основним заголовком пакета IP і полем даних. Наявність AH ніяк не впливає на процес передачі інформації транспортного і більш високого рівнів. Основним і єдиним призначенням AH є забезпечення контролю цілісності та автентичності пакетів даних для захисту від атак, пов'язаних з несанкціонованою зміною вмісту пакета, і в тому числі від підміни вихідного адреси мережевого рівня. Протоколи більш високого рівня повинні бути модифіковані з метою здійснення перевірки автентичності отриманих даних. Формат AH представлений на рис. 11.14, він складається з 96-бітового заголовка і даних змінної довжини, що складаються з 32-бітових слів. Назви полів достатньо ясно відображають їх вміст: Next Header указує на наступний заголовок, Payload Len представляє довжину пакета, SPI є покажчиком на контекст безпеки і Sequence Number Field містить послідовний номер пакета.

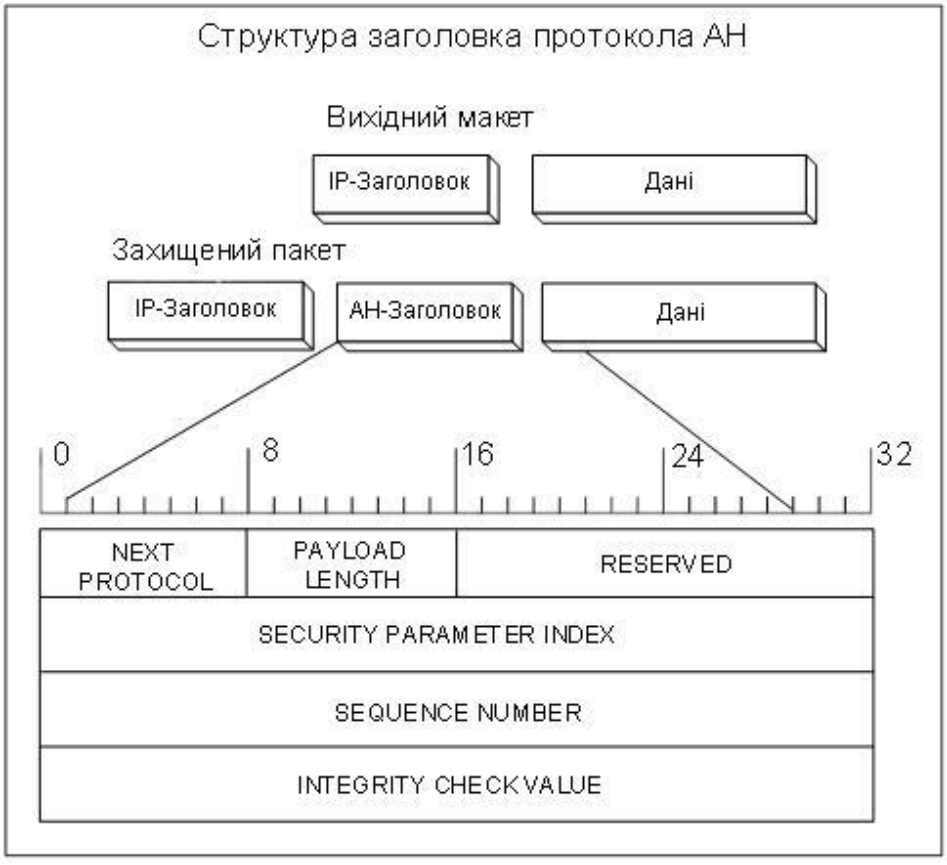


Рис. 11.14. **Формат заголовка AH**

На відміну від алгоритмів обчислення контрольної суми, вживаних у протоколах передачі інформації комутованими лініями зв'язку або каналами локальних мереж і орієнтованих на виправлення випадкових помилок середовища передачі, механізми контролю цілісності та автентичності даних призначені для захисту від внесення цілеспрямованих змін. Застосування механізмів контролю цілісності та автентичності для кожного IP-пакета є ефективним засобом боротьби з атаками типу перехоплення, модифікації трафіку і підміни IP-адрес. Крім того, забезпечується захист від повторної передачі пакетів у IP-мережах. При цьому реалізація протоколу AH не припускає забезпечення конфіденційності даних, тобто інформаційні дані кожного пакета передаються в незашифрованому вигляді. Як видно з розглянутої структури заголовка AH при розрахунку значень кодів контролю цілісності та автентичності пакетів даних (integrity check value, ICV) не використовуються змінні IP-заголовка оригінального пакета. Заголовок AH збільшує довжину оригінального IP-пакета приблизно на 24 байти (196 біт), основну частину цих додатково внесених надлишкових даних займає код контролю цілісності та автентичності ICV.

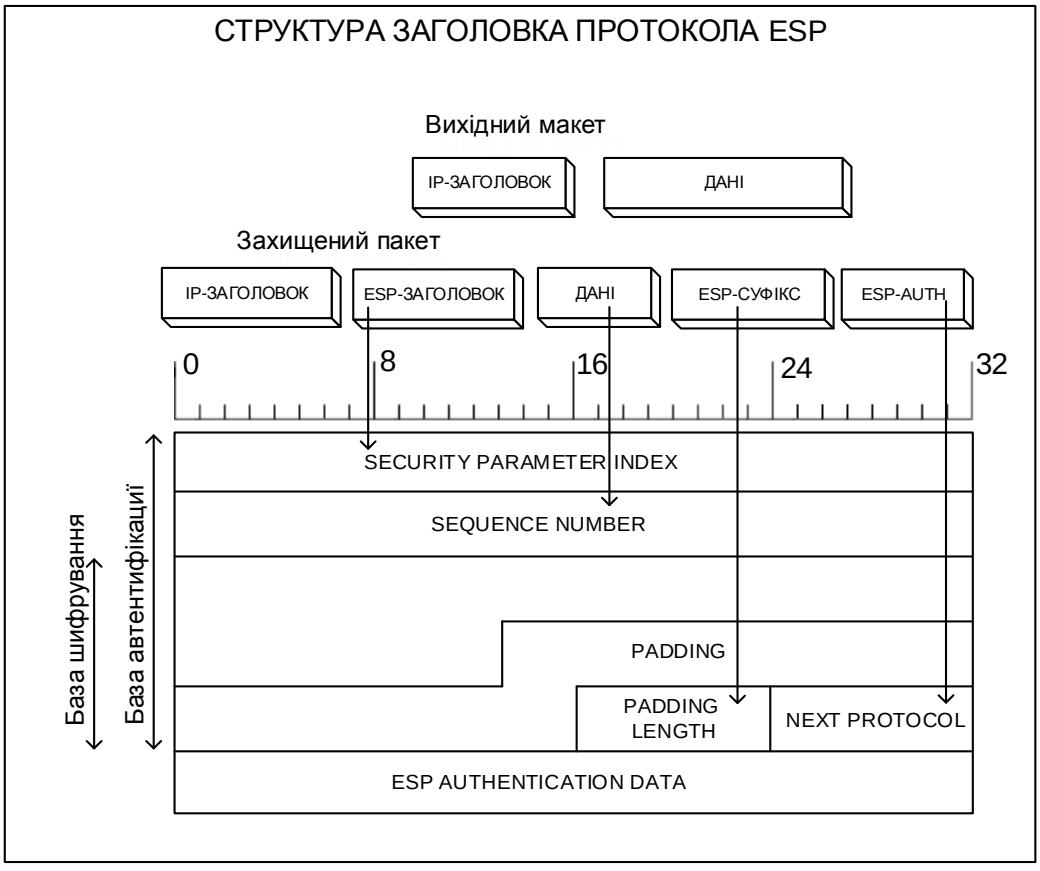


Рис. 11.15. **Формат заголовка ESP**

Для формування кодів контролю цілісності та автентичності ICV використовуються спеціальні механізми безпеки інформації: коди виявлення маніпуляцій (MDC – Manipulation Detection Code), коди автентифікації повідомлень (MAC – Message Authentication Code). Їх формування засноване на використанні гешируючих функцій, які дозволяють для даних у загальному випадку довільної довжини формувати геш-код (геш-значення) строго заданої довжини. Зазначені механізми застосовуються за замовчуванням з метою забезпечення цілісності та автентичності пакетів даних у всіх реалізацій мереж IPv6. При цьому для формування ICV в протоколі AH передбачені обов'язкові алгоритми (для забезпечення сумісності програмних продуктів різних виробників), такі, наприклад, як HMAC-MD5-96 (описаний в стандарті RFC 2403), HMAC-SHA-1-96 (описаний в стандарті RFC 2404). Крім того, передбачено деякі інші (додаткові) алгоритми для формування ICV, наприклад, DES-MAC, HMAC-ГОСТ Р 34.11-94, HMAC-ГОСТ Р 34.11-2001. Специфікацією протоколу AH передбачено також використання нових, більш ефективних алгоритмів формування ICV, які розробляються або будуть розроблені в найближчому часі, що дозволяє говорити про високу гнучкості протоколу AH і простоті його подальшої модернізації.

Протокол AH може використовуватися як в тунельному, так і в транспортному режимі, самостійно і в комбінації з протоколом ESP.

11.3.1.2. Забезпечення конфіденційності, цілісності та автентичності даних у IP-мережах з використанням протоколу ESP (IPSec)

У разі використання інкапсуляції зашифрованих даних заголовок ESP є останнім у ряду опціональних заголовків, "видимих" у пакеті. Формат ESP (див. рис. 11.15) так само, як і формат AH, може зазнавати значні зміни залежно від використовуваних криптографічних алгоритмів. Проте, у форматі ESP можна виділити такі обов'язкові поля: SPI, яке вказує на контекст безпеки і Sequence Number Field, що містить послідовний номер пакета. Поле "ESP Authentication Data" (контрольна сума), не є обов'язковим в заголовку ESP. Одержувач пакета ESP розшифровує ESP заголовок і використовує параметри і дані застосованого алгоритму шифрування для декодування інформації транспортного рівня.

Протокол ESP реалізує: шифрування даних IP-пакетів для забезпечення конфіденційності інформації; додатково (аналогічно протоколу AH) автентифікацію джерела кожного пакета, цілісність даних кожного пакета, захист від повторної передачі пакетів.

Заголовок ESP так само, як і заголовок AH збільшує довжину оригінального IP-пакета приблизно на 24 байти (196 біт), основну частину цих додатково внесених надлишкових даних займає код контролю цілісності та автентичності ESP. Для його формування в протоколі ESP передбачено використання спеціальних механізмів контролю цілісності та автентичності даних (аналогічних тим, які використовуються протоколом AH): HMAC-MD5-96, HMAC-SHA-1-96, DES-MAC, HMAC-ГОСТ Р 34.11-94, HMAC-ГОСТ Р 34.11-2001 з можливістю заміни їх на більш ефективні.

Для забезпечення конфіденційності даних IP-пакетів передбачено використання криптографічних алгоритмів шифрування, серед яких передбачені обов'язкові алгоритми (для забезпечення сумісності програмних продуктів різних виробників), такі, наприклад, як DES-CBC (описаний в стандарті RFC 2405), NULL (описаний в стандарті RFC 2410). Крім того, передбачено деякі інші (додаткові) алгоритми шифрування, наприклад, CAST-128, IDEA, 3DES (описані в стандарті RFC 2451), а також національний стандарт шифрування США AES-128, 192, 256 (FIPS-197) і вітчизняний стандарт ГОСТ-28147-89.

Протокол ESP може використовуватися як в тунельному, так і в транспортному режимі, самостійно і в комбінації з протоколом AH.

11.3.1.3. Застосування протоколів AH і ESP в транспортному і тунельний режимах

Транспортний режим використовується для захисту поля даних IP пакета, що містить протоколи транспортного рівня (TCP, UDP, ICMP), яке, в свою чергу, містить інформацію прикладних служб. Схема проходження IP-пакета даних з використанням протоколів безпеки AH і ESP в транспортному режимі наведена на рис. 11.16.

Прикладом застосування транспортного режиму є передача електронної пошти. Всі проміжні вузли на маршруті пакета від відправника до одержувача використовують тільки відкриту інформацію мережевого рівня і, можливо, деякі опціональні заголовки пакета (в IPv6). Недоліком

транспортного режиму є відсутність механізмів приховання конкретних відправника і одержувача пакета, а також можливість проведення аналізу трафіку. Результатом такого аналізу може стати інформація про об'єми і напрями передачі інформації, області інтересів абонентів, розташування керівників.

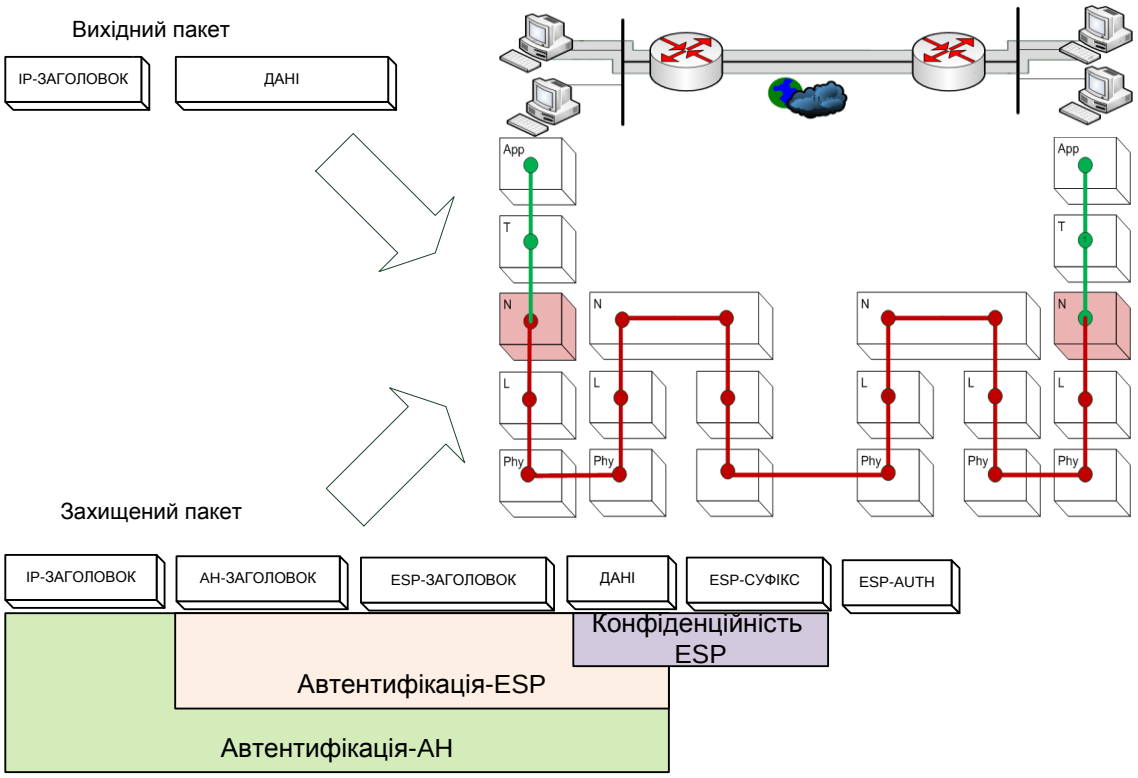


Рис. 11.16. **Схема проходження IP-пакета даних з використанням протоколів безпеки АН і ESP в транспортному режимі**

Тунельний режим (рис. 11.17) передбачає захист (у тому числі шифрування) всього пакета, включаючи заголовок мережевого рівня. Тунельний режим застосовується у разі потреби приховання інформаційного обміну організації із зовнішнім світом. При цьому, адресні поля заголовка мережевого рівня пакета, що використовує тунельний режим, заповнюються міжмережевим екраном організації і не містять інформації про конкретного відправника пакета. При передачі інформації із зовнішнього світу в локальну мережу організації як адреса призначення використовується мережева адреса міжмережевого екрану. Після розшифрування міжмережевим екраном початкового заголовка мережевого рівня пакет направляється одержувачу.

Таким чином, проведений аналіз сучасних протоколів мережевої безпеки, застосовуваних у IP-мережах для забезпечення цілісності,

автентичності та конфіденційності передачі даних, дозволяє зробити такі висновки:

застосування механізмів захисту інформації на верхніх рівнях (рівня прикладного процесу, рівня представлень або сеансового рівня) моделі OSI дозволяє ефективно реалізувати функції безпеки конкретних мережевих служб. Такий спосіб захисту інформації не залежить від того, які мережі (IP або IPX, Ethernet або ATM) застосовуються для транспортування даних, що є безперечною перевагою такого підходу. У той же час спостерігається залежність реалізації мережевих служб і конкретних додатків від версії протоколу мережевої безпеки. Зниження рівня (по специфікації моделі OSI) підвищує універсальність застосовуваних засобів захисту для будь-яких додатків і протоколів прикладного рівня, однак виникає залежність протоколу захисту від конкретної мережевої технології;

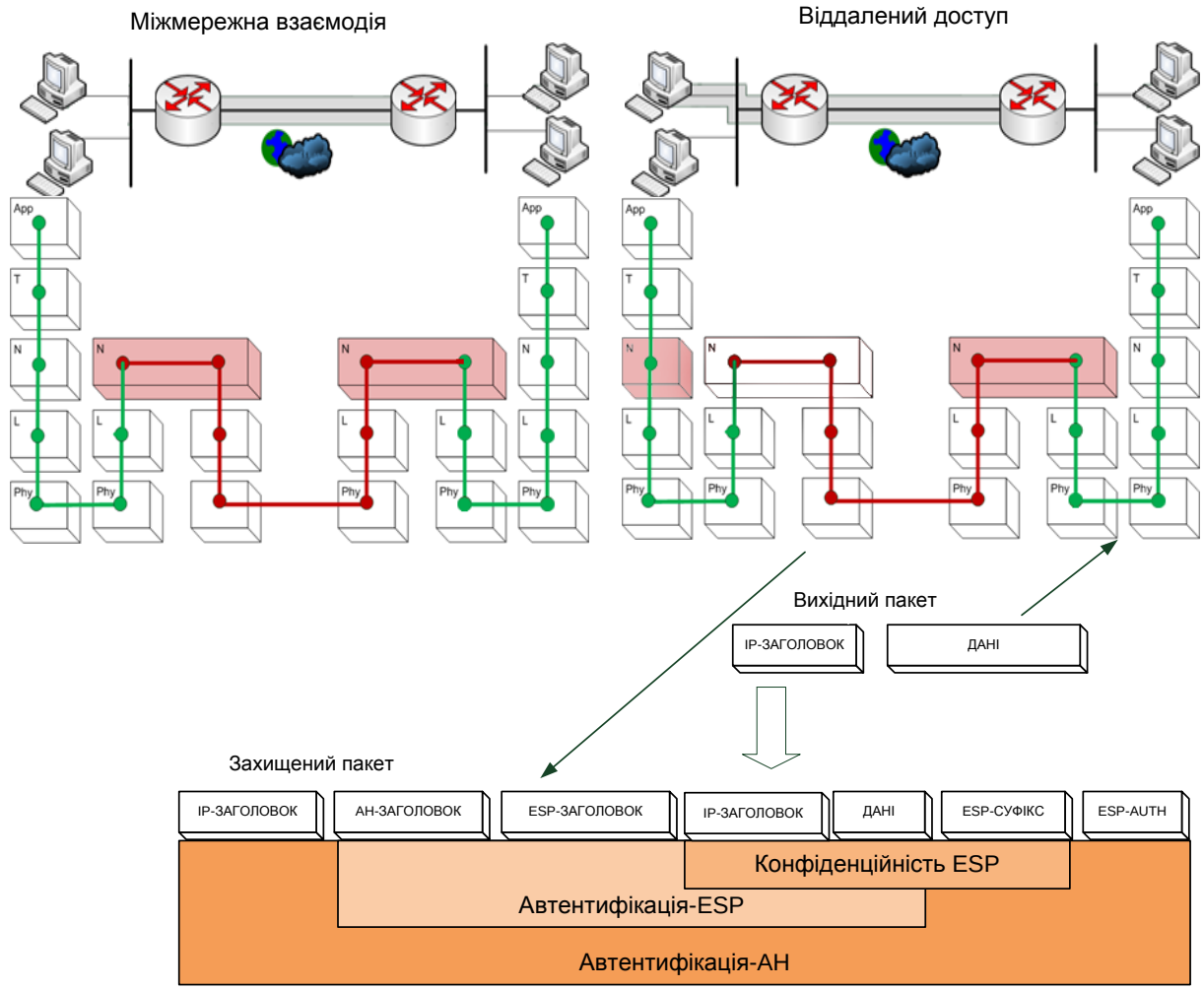


Рис. 11.17. **Схема проходження IP-пакета даних з використанням протоколів безпеки АН і ESP в тунельному режимі**

компромісним варіантом є протоколи мережевої безпеки IPSec, що функціонують на мережевому рівні. З одного боку, вони "прозорі" для додатків, а з іншого – можуть працювати практично у всіх мережах, оскільки засновані на широко розповсюдженому протоколі IP. Протоколи IPSec домінують на сьогоднішній день у більшості реалізацій віртуальних приватних мереж і реалізовуються як програмному вигляді (наприклад, протоколи реалізовані в операційній системі Windows компанії Microsoft), так і у вигляді програмно-апаратних реалізацій (рішення Cisco, Nokia). Незважаючи на велике число різних рішень, все реалізації володіють високою сумісністю один з одним;

для контролю цілісності та автентичності пакетів даних у протоколах IPSec застосовуються спеціальні механізми захисту. Їх застосування дозволяє, за рахунок внесення в дані, що передаються спеціально сформованої надмірності (MDC, MAC) ефективно вирішувати задачі захисту пакетів даних від випадкового і зловмисного зміни. Формування кодів контролю цілісності та автентичності пакетів даних засноване на використанні ключових (MAC) та безключових (MDC) гешуючих функцій. Зазначені механізми застосовуються за замовчуванням у протоколах IPSec з метою забезпечення цілісності та автентичності пакетів даних у всіх реалізацій мереж IPv6.

11.3.2. Протокол SSL

Протокол SSL (secure socket layer) розроблений для забезпечення надійного захисту наскрізної передачі даних з використання протоколу TCP. SSL становить не один протокол, а два рівні протоколів, як показано на рис. 11.18.

<table>
<tr><td>Протокол квантування SSL</td><td>Протокол змінипараметрів шифрування SSL</td><td>Протокол сповіщанняSSL</td><td>HTTP</td></tr>
<tr><td colspan="4">Протокол записуSSL</td></tr>
<tr><td colspan="4">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

Рис. 11.18. **Стік протоколів SSL**

Протокол SSL пропонує базовий набір засобів захисту, які застосовуються протоколами більш високих рівнів, і забезпечує конфіденційність каналу комунікацій і автентифікацію користувача.

Протокол діалогу SSL має дві основні фази. Перша фаза використовується для встановлення конфіденційного каналу комунікацій. Друга – служить для автентифікації користувача.

11.3.3. Протокол TLS

Протокол TLS призначений для забезпечення конфіденційності й цілісності даних. Він має два рівні: протокол записів TLS і протокол діалогу TLS. Протокол записів TLS забезпечує конфіденційність даних з використанням симетричних алгоритмів шифрування DES, RC4 і цілісність даних з використанням геш-функцій SHA-1 або MD5. Протокол діалогу TLS забезпечує цифровий підпис, заснований на підході RSA або DSS.

Контрольні запитання

1. Основні принципи захисту інформації при підключенні до мережі Інтернет.
2. Захист інформації в інформаційних системах за допомогою міжмережевих екранів.
3. Захист інформації на мережному рівні за допомогою протоколів TLS, SSL, IPSec.
3. Основні режими використання мережних протколів.
4. Схема проходження IP-пакета даних з використанням протоколів безпеки AH і ESP в тунельному режимі.
5. Схема проходження IP-пакета даних з використанням протоколів безпеки AH і ESP в транспортному режимі.
6. Забезпечення конфіденційності, цілісності та автентичності даних в IP-мережах з використанням протоколу ESP (IPSec).
7. Забезпечення безпеки даних в інформаційних системах за допомогою Log-сервера.
8. Забезпечення безпеки даних в інформаційних системах за допомогою Proxy-сервера.
9. Забезпечення безпеки даних в інформаційних системах за допомогою демілітаризованої зони.
10. Забезпечення електронної пошти за допомогою антивірусних програм.

Розділ 12. Моделі захисту. Захист пам’яті

При виборі криптосистеми необхідно проводити аналіз загроз безпеки в конкретній комп’ютерній системі, що передбачає оцінку стійкості до досить різноманітних типів криптоаналітичних нападів.

Модель загроз можна розглядати як композицію моделі противника, моделі атак і моделі криптосистеми.

12.1. Аналіз умов функціонування та загроз інформації в комп’ютерних системах та мережах

Аналіз умов функціонування локальних і глобальних обчислювальних систем показав, що головною вимогою, яка висувається до них, є забезпечення користувачам потенційної можливості доступу до поділюваних ресурсів усіх комп’ютерів, об’єднаних у мережу [1; 4; 5; 9; 10; 17 – 23; 26; 32 – 35; 43; 45]. До основних вимог функціонування глобальних обчислювальних систем (ГОС) відносяться: продуктивність, надійність, сумісність, керованість, захищеність, розширюваність і масштабованість. Основні вимоги і їх складові наведені на рис. 12.1.

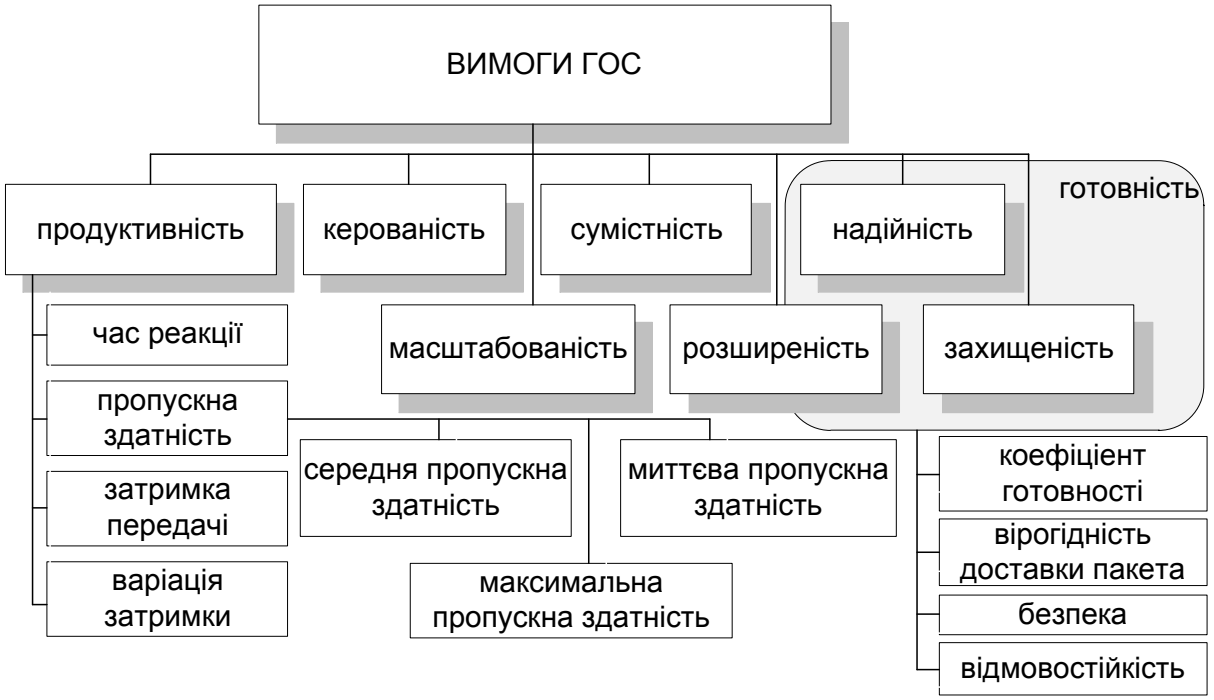


Рис. 12.1. **Вимоги, які висуваються до обчислювальних мереж та систем**

У цей час для оцінки функціонування локальних обчислювальних систем (ЛОС) і ГОС уведене поняття "якість обслуговування" (Quality of Service, Qps) комп'ютерної мережі, що включає тільки дві найважливіші характеристики – продуктивність і надійність [23; 43].

Проведений аналіз показника якості обслуговування мережі визначає два підходи до його забезпечення [23; 43; 45].

Перший підхід полягає в гарантованому забезпеченні користувачеві дотримання деякої числової величини показника якості обслуговування (забезпечення встановленого показника середньої пропускної здатності, показника часу затримки передачі і т. д.). Так, наприклад, технології Frame Relay і ATM дозволяють будувати мережі, що гарантують якість обслуговування за продуктивністю (показники середньої пропускної здатності, часу реакції, часу затримки та ін.).

Другий підхід полягає в пріоритетному обслуговуванні користувачів відповідно до встановленої ієрархії мережі. Таким чином, якість обслуговування залежить від ступеня привілейованості користувача або групи користувачів, до якої він належить. Для вповноважених користувачів ГОС якість обслуговування не гарантується, а гарантується тільки рівень їх привілеїв. Таке обслуговування називається обслуговуванням best effort – з найбільшим старанням. Проведений аналіз функціонування локальних мереж показує, що за таким принципом працюють ЛОС, побудовані на комутаторах з пріоритетом кадрів.

Для забезпечення необхідного показника якості обслуговування ГОС необхідно забезпечити продуктивність і надійність. Під продуктивністю [23; 43] розуміється властивість, що забезпечує можливість розпаралелювання робіт між декількома комп'ютерами мережі.

Основними характеристиками продуктивності є: час реакції, пропускна здатність, затримка передачі і її варіація.

Час реакції [23; 43] є інтегральною характеристикою продуктивності мережі й визначається як інтервал часу між виникненням запиту користувача до якої-небудь мережної служби й одержанням відповіді на цей запит.

Проведений аналіз даного показника показує, що його значення залежить тільки від типу служби, до якої звертається користувач, статусу користувача в мережі, типу сервера, а також від поточного стану елементів ГОС – завантаженості сегментів, комутаторів і маршрутизаторів, через які проходить запит, завантаженості сервера та ін. Час

реакції мережі розподіляється на час підготовки запитів на клієнтському комп'ютері, час передачі запитів між клієнтом і сервером через комунікаційне встаткування, час обробки запитів на сервері, час передачі відповідей від сервера клієнту і час обробки одержуваних від сервера відповідей на клієнтському комп'ютері.

Для визначення обсягу переданих даних за одиницю часу використовується пропускна здатність та її похідні (миттєва, максимальна і середня пропускні здатності).

При цьому середня пропускна здатність [23; 43] визначається як співвідношення загального обсягу переданих даних до часу їх передачі за тривалий проміжок часу (доба, місяць, рік), миттєва пропускна здатність [23; 43] визначається за дуже маленький проміжок часу (від 0,1 до 10^{-3} с) і максимальна пропускна здатність визначається як найбільша миттєва пропускна здатність, зафіксована протягом періоду спостереження.

Аналіз функціонування ГОС показує, що для проектування, настроювання й оптимізації використовуються такі показники, як середня й максимальна пропускні здатності. Для визначення якості мережі в цілому, не диференціюючи його за окремими сегментами або обладнаннями, використовується загальна пропускна здатність мережі, яка визначається як середня кількість інформації, переданої між усіма вузлами мережі в одиницю часу. Для визначення якості мережі так само використовують кількісний показник максимальної затримки передачі і її варіації.

Затримка передачі визначається як час знаходження пакета в будь-якому мережному обладнанні або частині мережі. Цей параметр продуктивності за змістом близький до реакції мережі, але відрізняється тим, що завжди характеризує тільки мережні етапи обробки даних, без затримок обробки комп'ютерами ЛОС.

Проведений аналіз створення розподілених систем і експлуатації ЛОС і ГОС показує, що для забезпечення їх надійності застосовуються характеристики складних систем: готовність або коефіцієнт готовності, що означає проміжок часу, протягом якого система може бути використана; вірогідність даних, тобто захист їх від викривлень; погодженість (несуперечність) та їх ідентичність.

Для опису передачі пакетів між кінцевими вузлами використовуються імовірнісні характеристики каналу зв'язку : імовірність доставки

пакета вузлу призначення без викривлень, імовірність втрати пакета (за кожною із причин – переповнення буфера маршрутизатора, через розбіжність контрольної суми, через відсутність працездатного шляху до вузла призначення тощо), імовірність викривлення окремого біти переданих даних.

У показник загальної надійності включається безпека – здатність системи захистити дані від несанкціонованого доступу й відмовостійкість – здатність системи сховати від користувача її окремі елементи. При проектуванні і модернізації ЛОС ураховуються додаткові вимоги до обчислювальних мереж [23; 43]:

Розширюваність (extensibility) – можливість порівняно легкого додавання окремих елементів мережі (користувачів, комп'ютерів, додатків, служб), нарощування довжини сегментів мережі і заміни існуючої апаратури могутнішою.

Масштабованість (scalability) – можливість мережі нарощувати кількість вузлів і довжину зв'язків у дуже широких межах, при цьому зберігається показник продуктивності мережі.

Прозорість (transparency) – можливість роботи з вилученими ресурсами з використанням тих же команд і процедур, що й для роботи з локальними ресурсами.

Комп'ютерні мережі споконвічно призначені для спільного доступу користувача до ресурсів комп'ютерів: файлів, принтерів тощо.

Проведений аналіз працездатності ЛОС (ГОС) показує, що особливу складність представляє сполучення в одній мережі традиційного комп'ютерного та мультимедійного трафіка. Для обліку складеного трафіка використовуються такі додаткові параметри мережі.

Керованість – можливість централізовано контролювати стан основних елементів мережі, виявляти й розв'язувати проблеми, що виникають при роботі мережі, виконувати аналіз продуктивності й планувати розвиток мережі.

Планування – можливість прогнозування змін вимог користувачів до мережі, застосування нових додатків і мережних технологій.

Сумісність або *інтегрованість* – можливість включення у ЛОС (ГОС) найрізноманітнішого програмного і апаратного забезпечення (різні операційні системи, що підтримують різні стеки комунікаційних протоколів, апаратні засоби й додатка від різних виробників).

Таким чином, аналіз основних вимог, які висуваються до ЛОС і ГОС, показує, що для виконання головного завдання забезпечення користувачам потенційної можливості доступу до поділюваних ресурсів усіх комп'ютерів, об'єднаних у мережу, необхідно виконати вимоги двох основних характеристик показника "якості обслуговування" – продуктивності і надійності.

Для оцінки надійності мережі використовуються основні характеристики складних систем: коефіцієнт готовності – проміжок часу, протягом якого система може бути використана; безпека – здатність системи захистити дані від несанкціонованого доступу й відмовостійкість – здатність системи працювати в умовах відмови деяких її елементів.

Проблема захисту комп'ютерних мереж від несанкціонованого доступу набула особливої гостроти.

Розвиток комунікаційних технологій дозволяє будувати мережі розподіленої архітектури, що поєднують велику кількість сегментів, розташованих на значній відстані один від одного. Все це викликає збільшення числа вузлів мереж і кількості різних ліній зв'язку між ними, що, у свою чергу, підвищує ризик несанкціонованого підключення до мережі й доступу до важливої інформації [23; 43].

Збільшення обсягів оброблюваних і переданих даних у комп'ютерних системах і мережах, насамперед, у банківських системах, у системах управління великими фінансовими і промисловими організаціями, підприємствами енергетичного сектору, транспорту нових підходів до побудови протоколів і механізмів забезпечення безпеки інформаційних систем.

Природня вимога до безпеки і вірогідності оброблюваної та переданої інформації в таких системах постає дуже гостро, оскільки відмова системи або вихід за встановлені обмеження зазначених властивостей може привести до значних фінансових і матеріальних втрат, збитку екології, життя і здоров'я людей.

Аналіз показує [; 4; 5; 9; 10; 17 – 23; 26; 32 – 35; 43; 45], що за останній час загальний обсяг оброблюваної й переданої інформації в комп'ютерних системах і мережах збільшився у декілька разів (на два – три порядки кожні п'ять – десять років) і загальні тенденції свідчать, що така динаміка збереглася.

Сучасні криптографічні засоби захисту інформації повинні забезпечувати своєчасну обробку величезних обсягів даних (десятки – сотні Мбіт/с) і задовольняти твердим вимогам з вірогідності і безпеки інформації.

Крім того, сучасний розвиток інформаційних технологій, високий рівень комп'ютеризації й інформатизації сучасного суспільства обумовили виникнення нових загроз безпеки інформації [; 4; 5; 9; 10; 17 – 23; 26; 32 – 35; 43; 45].

12.2. Побудова моделі загроз у сучасних комп'ютерних мережах та системах

Моделювання процесу реалізації загроз КМіС доцільно здійснювати на основі розгляду логічного ланцюжка: "загрози – джерело загрози – метод реалізації – уразливість – наслідки". На рис. 12.2 подана структурна схема моделі реалізації загроз інформаційних ресурсів у КМіС.

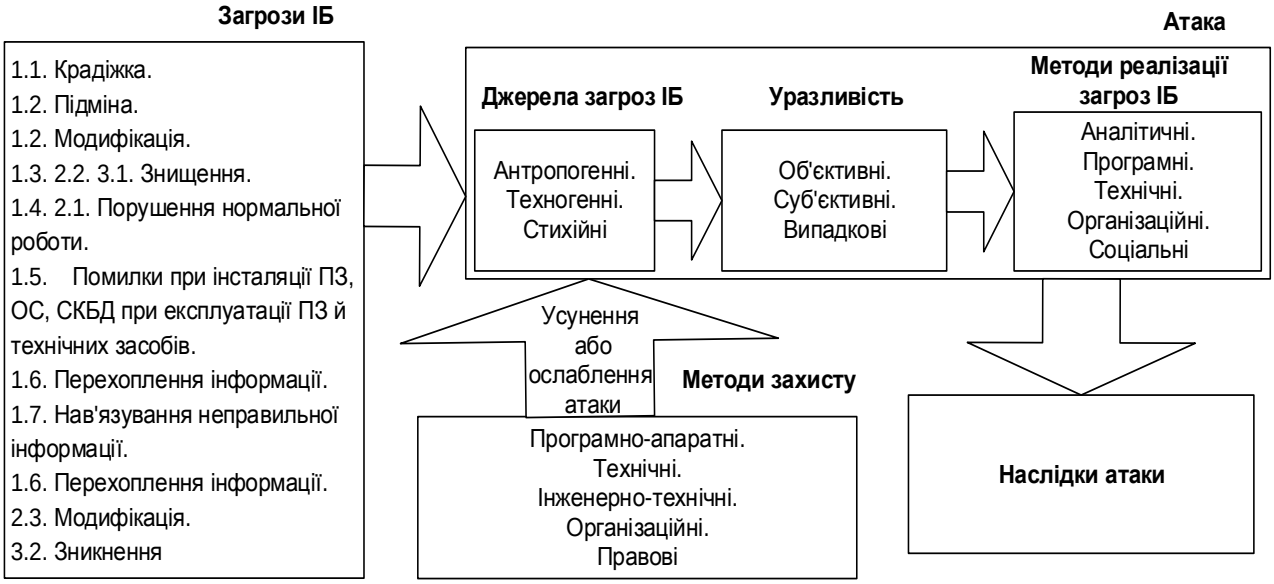


Рис. 12.2. **Структурна схема моделі реалізації загроз інформаційних ресурсів у КМіС**

Аналіз негативних наслідків реалізації загроз припускає обов'язкову ідентифікацію (наприклад, присвоєння унікального коду) можливих джерел загроз, уразливостей, що сприяють їх прояві та методів реалізації, тобто класифікацію (рис. 12.3).

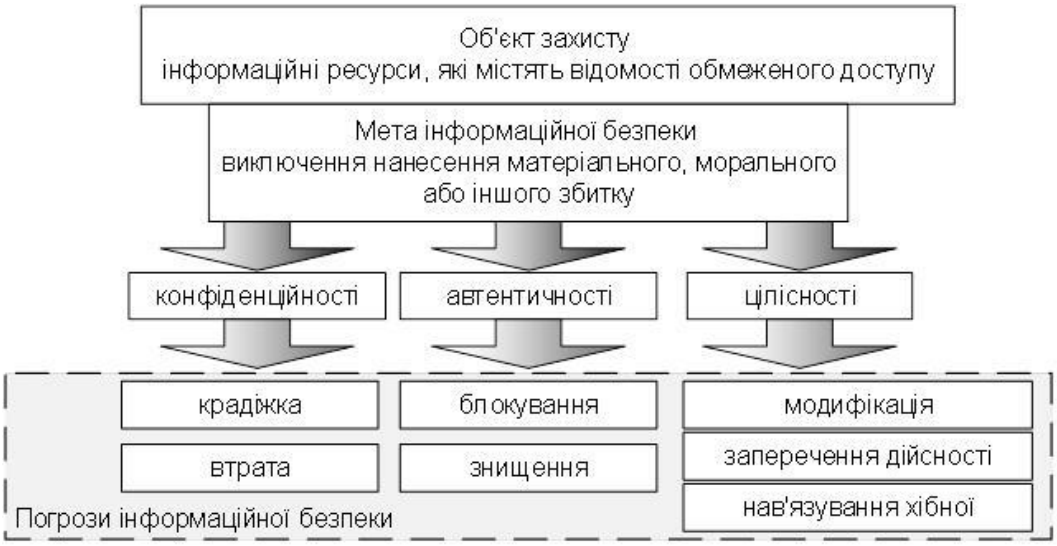


Рис. 12.3. **Мета і загрози безпеки інформації**

Для опису моделі реалізації загроз інформаційних ресурсів зафіксуємо кінцеву множину суб'єктів, взаємодіючих з інформаційною системою – S, параметр N – кількість уразливих до атаки комп'ютерів (ПК); параметр D містить початкове значення середньої кількості атакованих комп'ютерів за обрану одиницю часу. Вважаємо, що D є константою, обчислення враховують, що комп'ютер не може бути атакований двічі; $a(t)$ – пропорція вразливих ПК, які були успішно атаковані за час t, $N \cdot a(t)$ – загальна кількість успішно атакованих комп'ютерів, за час t зроблено не більше $D(1-a(t))$ нових успішних атак. Кількість захоплених комп'ютерів за період часу $d(t)$ дорівнює:

$$n = aN \cdot D(1-a)dt.$$

Ураховуючи, що N – константа і $n = d(Na) = Nda$, правильне наступне рівняння:

$$Nda = aN \cdot D(1-a)dt,$$

в диференціальному вигляді:

$$\frac{da}{dt} = Da(1-a),$$

має таке рішення:

$$a = \frac{e^{D(t-T)}}{1+e^{D(t-T)}},$$

де T – часовий параметр, що характеризує найбільше зростання атак.

Для побудови загальної структури підсистеми безпеки інформаційної безпеки в КМіС і моделей атак обраний функціональний тип математичних моделей, який називається моделями "чорної скриньки". Математична модель є моделлю об’єкта, процесу або явища, що становить математичні закономірності, за допомогою яких описані основні характеристики об’єкта, який моделюємо, процесу або явища [23]. Загальним для опису даних математичних моделей є процес формування криптограми.

Для цього зафіксуємо кінцеву множину $I = \{I_1, I_2, ..., I_m\}$ переданих пакетів, причому кожному пакета відповідає ймовірність $P^*(I_j)$. Розподіл ймовірностей випадкового процесу задається сукупним розподілом ймовірностей випадкових величин, тобто множиною ймовірностей $P^*_o = \{P^*(I_1), P^*(I_2), ..., P^*(I_m)\}$. Джерело ключів породжує потік ключів з множини К і/або К*. Кожному ключу $K_i \in K = \{K_1, K_2, \ldots, K_k\}$ відповідає деяка ймовірність $P^*(K_i)$, а кожному $K_i^* \in K^* = \{K_1^*, K_2^*, \ldots, K_k^*\}$ відповідає ймовірність $P^*(K^*_i)$. Випадковий процес вироблення ключів задається множиною ймовірностей:

$$P^*_K = \{P^*(K_1), P^*(K_2), ..., P^*(K_k)\}$$
$$P^*_{K^*} = \{P^*(K_1^*), P^*(K_2^*), ..., P^*(K_k^*)\}.$$

Вибір ключа K_i визначає конкретне відображення φ_i з множини відображень і криптограма формується: $E_I = \varphi_i(K_i, I_j)$.

Відмінність між активними і пасивними атаками полягає в тому, що при виконанні атак першого типу (активні атаки) порушник здійснює активні дії, тобто дії, пов’язані зі зміною потоку даних або зі створенням фальшивих потоків (імітація, відтворення, модифікація повідомлень або перешкоди в обслуговуванні). Метою другого типу атак (пасивні атаки) є одержання переданої інформації (розкриття вмісту повідомлень і аналіз потоку даних).

Оцінка ступеня ефективності атаки може бути здійснена за рахунок проведення аналізу даних, якими володіє противник, його можливостей та інших параметрів пасивних атак. Основним методом оцінки можливостей противника при атаці є створення моделей атак.

12.3. Побудова моделі порушника у сучасних комп'ютерних мережах та системах

Спроба одержати несанкціонований доступ до комп'ютерних мереж з метою ознайомитися з ними, залишити записку, виконати, знищити, змінити або викрасти програму або іншу інформацію кваліфікується як *комп'ютерне піратство. Я*к соціальне явище, подібні дії прослідковуються в останні 10 років, але при цьому спостерігається тенденція до їх стрімкого зростання в міру збільшення кількості побутових комп'ютерів.

Зростання кількості комп'ютерних порушень очікується в тих країнах, де вони широко рекламуються у фільмах і книгах, а діти в процесі ігор рано починають знайомитися з комп'ютерами. Разом з тим зростає кількість й серйозних навмисних злочинів [23; 43].

Однак комп'ютерні злочинці не цікавляться, наскільки добре здійснюється в цілому контроль у комп'ютерній системі; вони шукають єдину лазівку, яка приведе їх до бажаної мети. Для одержання інформації вони проявляють винахідливість, використовуючи психологічні фактори, детальне планування та активні дії. Необхідно розділити два визначення: хакер (hacker) і кракер (cracker). Основна відмінність полягає в постановці мети зламу комп'ютерних систем: перші ставлять дослідницькі завдання з оцінки і знаходження уразливостей з метою подальшого підвищення надійності комп'ютерної системи. Кракери же вторгаються в систему з метою руйнування, крадіжки, псування, модифікації інформації і роблять правопорушення з корисливими намірами швидкого збагачення.

Для запобігання можливих загроз необхідно не тільки забезпечити захист операційних систем, програмного забезпечення і контроль доступу, але й виявити категорії порушників і ті методи, які вони використовують.

Залежно від мотивів, цілей і методів, дії порушників безпеки інформації можна розділити на чотири категорії:

шукачі пригод;

ідейні хакери;

хакери-професіонали;

ненадійні (неблагополучні) співробітники.

Шукач пригод, як правило, це студент, у якого рідко є продуманий план атаки. Він вибирає мету випадковим чином і зазвичай відступає, зіштовхнувшись із труднощами. Знайшовши діру в системі безпеки, він намагається зібрати закриту інформацію, але практично ніколи не намагається її таємно змінити. Своїми перемогами такий шукач пригод ділиться тільки зі своїми близькими друзями-колегами.

Ідейний хакер – це той же шукач пригод, але більш митецький. Він уже вибирає собі конкретні цілі (хости та ресурси) на підставі своїх переконань. Його улюбленим видом атаки є зміна інформаційного наповнення web-сервера або, у більш рідких випадках, блокування роботи ресурсу, що атакуються. Порівняно з шукачем пригод, ідейний хакер розповідає про успішні атаки набагато більш широкій аудиторії, звичайно розміщаючи інформацію на хакерському web-вузлі або в конференціях Usenet.

Хакер-професіонал має чіткий план дій і націлюється на конкретні ресурси. Його атаки добре продумані й звичайно здійснюються в кілька етапів. Спочатку він збирає попередню інформацію (тип операційної системи (ОС), сервіси, що надаються, заходи захисту, що застосовуються). Потім він становить план атаки з урахуванням зібраних даних і підбирає (або навіть розробляє) відповідні інструменти. Провівши атаку, він одержує закриту інформацію, і нарешті, знищує всі сліди своїх дій. Такий атакуючий професіонал звичайно добре фінансується й може працювати самому або в складі команди професіоналів.

Ненадійний (неблагополучний) співробітник своїми діями може доставити стільки ж проблем (буває й більше), скільки промисловий шпигун, до того ж його присутність звичайна складніше виявити. Крім того, йому доводиться долати не зовнішній захист мережі, а тільки, як правило, менш твердий внутрішній захист. Він не так витончений у способах атаки, як промисловий шпигун, і тому частіше допускає помилки й тим самим може видати свою присутність. Однак у цьому випадку небезпека його несанкціонованого доступу до корпоративних даних багато вище, чим будь-якого іншого зловмисника. Перераховані категорії порушників безпеки інформації можна згрупувати за їх кваліфікацією: *початківець* (шукач пригод), *фахівець* (ідейний хакер, ненадійний співробітник), *професіонал* (хакер-професіонал).

Порушник безпеки інформації, як правило, будучи фахівцем визначеної кваліфікації, намагається довідатися про комп’ютерні системи

й мережі і, зокрема, про засоби їх захисту. Тому модель порушника визначає [23; 43]: категорії осіб, у числі яких може виявитися порушник;

можливі цілі порушника і їх градації за ступенями важливості й небезпеки;

припущення про його кваліфікації;

оцінка його технічної озброєності;

обмеження й припущення про характер його дій.

На рис. 12.4 наведена узагальнена модель порушника безпеки інформації.

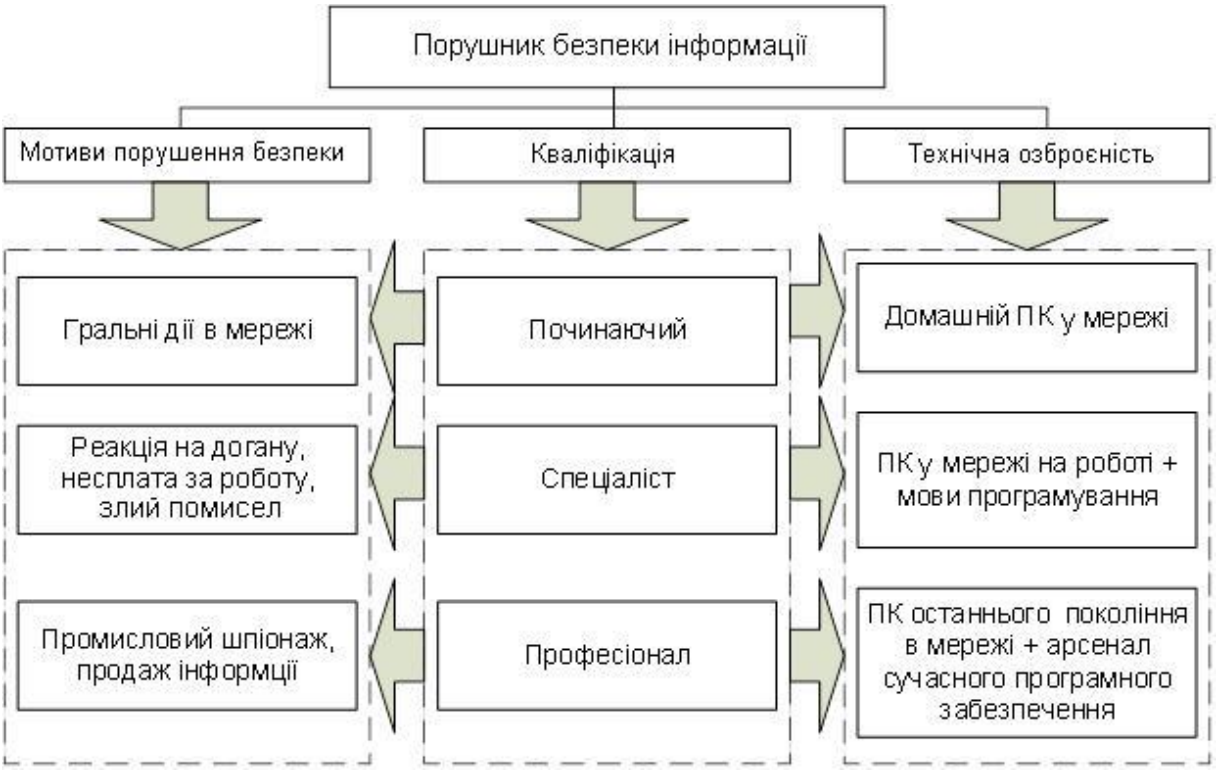


Рис. 12.4. **Модель порушника безпеки інформації**

Діапазон спонукальних мотивів одержання доступу до системи досить широкий: від бажання випробувати емоційний підйом при грі з комп'ютером до відчуття влади над ненависним менеджером. Займаються цим не тільки новачки, що бажають побавитися, але й професійні програмісти. Паролі вони добувають або в результаті підбору або шляхом обміну з іншими хакерами.

Частина з них, однак, починає не тільки переглядати файли, але й проявляти інтерес саме до їх змісту, а це вже є серйозною загрозою, оскільки в цьому випадку важко відрізнити невинне гру від злочинних дій.

Донедавна викликали занепокоєння випадки, коли незадоволені керівником службовці, зловживаючи своїм положенням, псували системи, допускаючи до них сторонніх або залишаючи системи без догляду в робочому стані. Спонукальними мотивами таких дій є:

реакція на догану або зауваження з боку керівника;

невдоволення тим, що фірма не оплатила понаднормові години роботи, хоча найчастіше понаднормова робота виникає через неефективне використання робочого часу;

злий намір у якості реваншу з метою послабити фірму як конкурента якої-небудь фірми.

Незадоволений керівником службовець створює одну з найбільших загроз обчислювальним системам колективного користування.

Професійні хакери – це комп'ютерні фанати, що прекрасно знають обчислювальну техніку і системи зв'язку. Вони затратили масу часу на обмірковування способів проникнення в системи й експериментуючи із самими системами. Для входження в систему професіонали найчастіше використовують деяку систематику й експерименти, а не розраховують на удачу або здогад. Їх мета – виявити й подолати захист, вивчити можливості обчислювальної установки й потім вийти, ствердившись у можливості досягнення своєї мети.

Завдяки високій кваліфікації ці люди розуміють, що ступінь ризику малий, оскільки відсутні мотиви руйнування або розкрадання. До категорії хакерів-професіоналів зазвичай відносять таких осіб:

які належать до злочинних групувань, що переслідують політичні цілі;

прагнучих одержати інформацію з метою промислового шпигунства;

хакерів або угрупування хакерів, що прагнуть до наживи.

Для здійснення несанкціонованого доступу в інформаційну систему потрібно, як правило, провести два підготовчі етапи:

зібрати відомості про систему;

виконати пробні спроби входження в систему.

Збір відомостей. Залежно від особистості зловмисника і його нахилів можливі різні напрями збору відомостей:

підбір співучасників;

аналіз періодичних видань, відомчих бюлетенів і документації;

перехоплення повідомлень електронної пошти;

підслуховування розмов, телексів, телефонів;

перехоплення інформації й електромагнітного випромінювання; організація крадіжок;

вимагання й хабар.

Багато власників систем часто не уявляють, яку підготовчу роботу повинен провести порушник, щоб проникнути в ту або іншу комп'ютерну систему. Тому вони самовпевнено вважають, що єдине, що необхідно зробити, – це захистити файл, указавши йому пароль, і забувають, що будь-яка інформація про ті або інші слабкі місця системи може допомогти зломщику знайти лазівку й обійти пароль, одержавши доступ до файла. Таким чином, інформація стає легкодоступною, якщо зловмисник знає, де й що дивитися.

Підбір співучасників. Підбір співучасників заснований на підслуховуванні розмов у барах, фойє готелів, ресторанах, таксі, підключенні до телефонів і телексам, вивченні змісту загублених портфелів і документів. Більшу й корисну інформацію можна витягти, якщо трапляється можливість підсісти до групи програмістів. Цей спосіб часто використовують репортери й професійні агенти.

Витяг інформації з періодичних видань. Зловмисники можуть почерпнути багато корисної інформації з газет і інших періодичних видань. *Перехоплення повідомлень електронної пошти.* Звичайно для підключення до електронної пошти використовується побутовий комп'ютер з модемом для зв'язку з державною телефонною мережею [23; 43]. Телефонний канал доступу в таку систему звичайно вільний, хоча останнім часом системні оператори вимагають установки обладнань реєстрації користувачів електронної пошти. Аж до недавнього часу багато довідкових систем були оснащені блоками, через які зломщики могли витягати більші обсяги даних, а також ідентифікатори й паролі користувачів. Зараз немає нічого незвичайного в тому, що блоки, установлені кракерами, можуть бути зашифровані й тільки окремі члени злочинних угруповань можуть зчитувати з них інформацію.

Зав'язування знайомств. З метою одержання інформації про обчислювальну систему або отримання службових паролів зломщики можуть використовувати різноманітні способи. Наприклад, знайомлячись, вони представляються менеджерами; використовують запитальники, роздаючи їх у фойє фірми й детально розпитуючи співробітників про комп'ютерну систему; дзвонять системноми адміністратору в обідній

час із проханням нагадати нібито забутий пароль; прогулюються в будинку, спостерігаючи за доступом до системи; установлюють контакти з незайнятими в цей момент службовцями охорони, яким відвідувачі при вході в будинок фірми повинні пред'являти ідентифікаційний код або пароль.

Більш зловмисним, але, можливо, і більш успішним є метод "полювання за разумами", коли на фірму приходить людина, яка бажає працювати системним програмістом або інженером зв'язку, і просить дати йому консультацію. Дивно, як багато інформації може передати службовець, який не має перспективи зростання, але вважає себе гідним більш важливої й високооплачуваної посади; він може розкрити коди користувачів, паролі, указати слабкі місця в мережах зв'язку і т. д.

Аналіз роздруківок. Деякі зломщики одержали доступ, до ЕОМ просто вивчаючи роздруківки, і це один з найбільш ефективних і найменш ризикованих шляхів одержання конфіденційної інформації. Численні фірми втрачають інформацію зі своїх комп'ютерних систем, по-перше, помилково думаючи, що вона не містить конфіденційної інформації, і, по-друге, помилково вважаючи, що всі чорнові роздруківки сумлінно знищуються. Саме таким способом зломщики змогли одержати досить повну картину організації комп'ютерної системи, використовуючи викинуті роздруківки й незатребувані протоколи роботи системи, які співробітникам обчислювального центру представлялися невинними папірцями.

Перехоплення повідомлень в каналах зв'язку. На сьогодняшній час кількість фірм, оснащених обчислювальною технікою, постійно зростає, тому перехоплення повідомлень стало досить реальною загрозою й для комерційного світу. Спектр можливих перехоплень досить широкий – перехоплення усних повідомлень із використанням радіопередавачів, мікрофонів і мікрохвильових обладнань; підслуховування повідомлень, переданих з телефону, телексу й іншими каналами передачі даних; контроль електромагнітного випромінювання від ПК; перехоплення супутникових або мікрохвильових передач.

Установкою радіопередавачів, мікрофонів і мікрохвильових обладнань або прослуховуванням ліній зв'язку звичайно займаються професійні зломщики, а також аматори й фахівці зі зв'язку. Останнім часом кількість випадків установки таких обладнань зросла. Улюбленими точками безконтрольного доступу також є телефонні лінії.

Передача даних з комутацією пакетів або з використанням широкосмугових ліній зв'язку зі швидкостями в тисячу й мільйони бод викликає інтерес у зломщиків і може бути перехоплена, щоб викрасти передані повідомлення, модифікувати їх зміст, затримати або вилучити.

12.4. Організація захисту пам'яті в ПК

Захист пам'яті – паратні і програмні засоби для запобігання запису або відтворення інформації з недозволених адрес пам'яті обчислювальної системи або машини. Сутність захисту пам'яті полягає в тому, що пам'ять ЦОМ програмно або апаратно розбивається на ряд ділянок і кожній ділянці або групі ділянок присвоюється код-ключ, який запам'ятовується в тій самій або спеціальній пам'яті. При зверненні до пам'яті визначається її ділянка і відповідний ключ, який порівнюється з дозволеним ключем захисту пам'яті, зазначеним в самій команді або диспетчером-програмою. Невідповідність ключів розглядається як порушення захисту пам'яті і виконання програми переривається. Переривання програми організовується так, щоб зміст захищеної області пам'яті залишився без змін. Захист пам'яті функціонує при кожному зверненні до пам'яті або в режимі запису інформації, або в режимі відтворення інформації, або в обох режимах. Захист пам'яті виконує такі функції: захист вмісту певних областей пам'яті від втрати інформації під час виконання програм через помилкові засилки інформації, які викликані відмовами і збоями устаткування або диспетчер-програми ЦОМ, помилками програміста або користувача; захист інформації від попадання її в руки стороннього користувача при несанкціонованому випадковому або навмисному втручанні.

До апаратних засобів захисту пам'яті відносяться: запам'ятовуючий пристрій ключів захисту, ємність якого відповідає числу ділянок, які захищаються, а швидкодія на порядок більше, ніж в основній пам'яті ЦОМ; схеми порівняння ключів захисту, переривання та індикації при порушенні захисту пам'яті. До програмних засобів захисту пам'яті відносяться: програми контролю ділянок пам'яті, їх кодування і складання таблиць відповідності; програми динамічного перерозподілу захисту пам'яті за розпорядженнями споживачів, за параметрами одночасно розв'язуваних завдань; програми аналізу причин порушень захисту

пам’яті і прийняття рішень щодо їх усунення. Захист пам’яті підвищує ефективність роботи ЦОМ, скорочуючи тимчасові витрати на пошук помилок, несправностей і на повторні обчислення через втрату інформації. Захист пам’яті необхідний при одночасному рішенні декількох завдань однієї ЦОМ у режимі поділу часу, одночасному обслуговуванні декількох користувачів, наявності бібліотек програм, архівів, що належать певним споживачам, одночасній роботі декількох пристроїв у складі ЦОМ.

12.4.1. Організація захисту пам’яті в ЕОМ

При мультипрограмному режимі роботи ЕОМ у її пам’яті одночасно можуть перебувати кілька незалежних програм. Тому необхідні спеціальні заходи по запобіганню або обмеженню звернень однієї програми до областей пам’яті, які використовуються іншими програмами. Програми можуть також містити помилки, які, якщо цьому не перешкодити, призводять до спотворення інформації, що належить іншим програмам. Наслідки таких помилок особливо небезпечні, якщо руйнуються програми операційної системи. Іншими словами, треба виключити вплив програми користувача на роботу програм інших користувачів і програм операційної системи. Слід захищати і самі програми від помилок, які знаходяться в них самих.

Таким чином, засоби захисту пам’яті повинні запобігати [23; 43]:

недозволеній взаємодії користувачів один з одним,

несанкціонованому доступу користувачів до даних,

пошкодженню програм і даних через помилки в програмах,

навмисним спробам зруйнувати цілісність системи,

використанню інформації в пам’яті не відповідно до її функціонального призначення.

Щоб перешкодити руйнуванню одних програм іншими, досить захистити область пам’яті даної програми від спроб запису в неї з боку інших програм, а в деяких випадках і своєї програми (захист від запису), при цьому допускається звернення інших програм до цієї області пам’яті для зчитування даних.

В інших випадках, наприклад при обмеженнях на доступ до інформації, що зберігається в системі, необхідно забороняти іншим програмам будь-яке звернення до деякої області пам’яті як на запис, так

і на зчитування. Такий захист від запису і зчитування допомагає в налагодженні програми, при цьому здійснюється контроль кожного випадку звернення за область пам’яті своєї програми.

Для полегшення налагодження програм бажано виявляти і такі характерні помилки в програмах, як спроби використання даних замість команд або команд замість даних у власній програмі, хоча ці помилки можуть і не руйнувати інформацію (невідповідність функціонального використання інформації).

Якщо порушується захист пам’яті, виконання програми призупиняється і виробляється запит переривання за порушенням захисту пам’яті.

Захист від вторгнення програм в чужі області пам’яті може бути організована різними методами. Але при будь-якому підході реалізація захисту не повинна помітно знижувати продуктивність комп’ютера і вимагати дуже великих апаратурних витрат.

Методи захисту базуються на деяких класичних підходах, які отримали свій розвиток в архітектурі сучасних ПК. До таких методів можна віднести захист окремих комірок, метод граничних регістрів, метод ключів захисту [23; 43].

Захист окремих комірок пам’яті організовується в ПК, призначених для роботи в системах управління, де необхідно забезпечити можливість налагодження нових програм без порушення функціонування робочих програм, керуючих технологічним процесом, які знаходяться в пам’яті. Це може бути досягнуто виділенням в кожній комірці пам’яті спеціального "розряду захисту". Установка цього розряду в "1" забороняє проводити запис в дану комірку, що забезпечує збереження робочих програм. Недолік такого підходу – велика надмірність в кодуванні інформації через надмірно дрібний рівень об’єкта, що захищається (комірка).

У системах з мультипрограмною обробкою доцільно організовувати захист на рівні блоків пам’яті, що виділяються програмами, а не окремих комірок.

Метод граничних регістрів (рис. 12.5) полягає у введенні двох граничних регістрів, які вказують верхню і нижню межі області пам’яті, куди програма має право доступу.

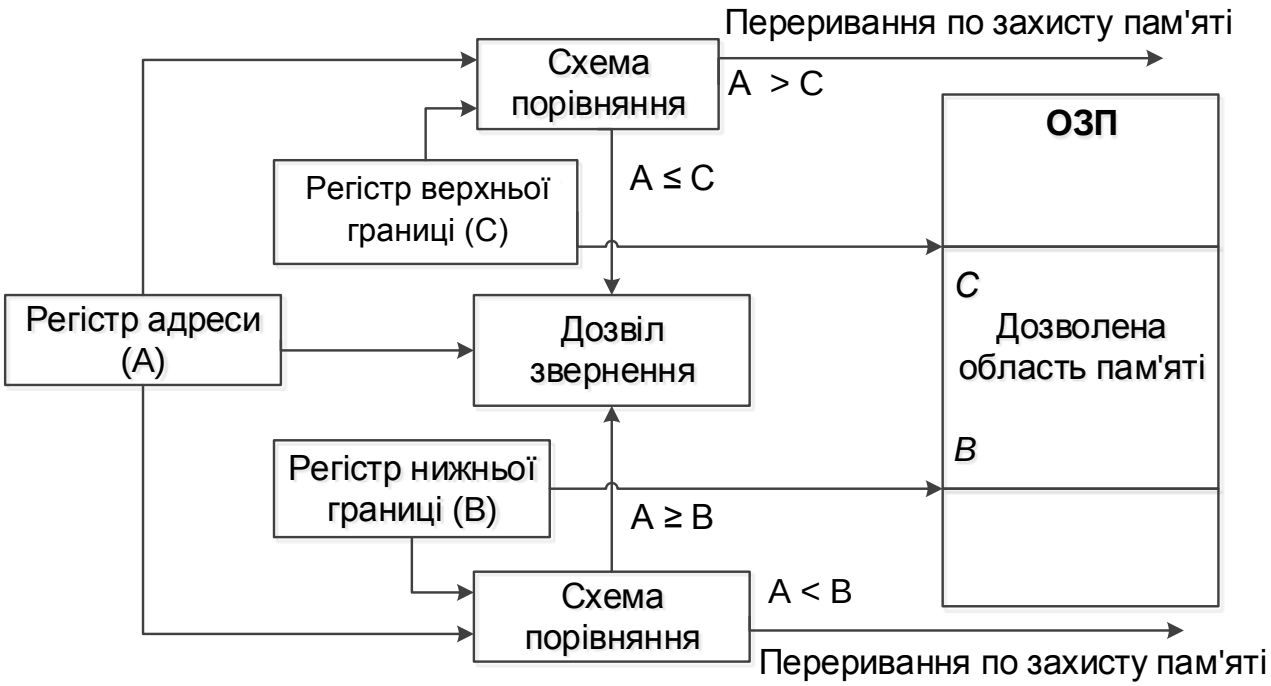


Рис. 12.5. **Захист пам’яті методом граничних регістрів**

При кожному зверненні до пам’яті перевіряється, чи знаходиться використовувана адреса в установлених межах. При виході за межі звернення до пам’яті не виконується, а формується запит переривання, що передає керування операційній системі. Зміст граничних регістрів встановлюється операційною системою при завантаженні програми в пам’ять.

Модифікація цього методу полягає в тому, що один регістр використовується для вказівки адреси початку області, яка захищається, а інший містить довжину цієї області.

Метод граничних регістрів, володіючи безсумнівною простотою реалізації, має і певні недоліки. Основним з них є те, що цей метод підтримує роботу лише з безперервними областями пам’яті.

Метод ключів захисту, на відміну від попереднього, дозволяє реалізувати доступ програми до областей пам’яті, організованим у вигляді окремих модулів, що не є єдиним масивом. Пам’ять у логічному відношенні ділиться на однакові блоки, наприклад, сторінки. Кожному блоку пам’яті ставиться у відповідність код, званий **ключем захисту пам’яті**, а кожній програмі, яка бере участь у мультипрограмній обробці, присвоюється код **ключа програми**. Доступ програми до даного блоку пам’яті для читання і запису дозволений, якщо ключі збігаються (тобто даний блок пам’яті відноситься до даної програми) або один з них має

код 0 (код 0 присвоюється програмам операційної системи і блокам пам’яті, до яких мають доступ всі програми: загальні дані, підпрограми, які спільно використовуються тощо). Коди ключів захисту блоків пам’яті і ключів програм встановлюються операційною системою.

У ключі захисту пам’яті передбачається додатковий розряд режиму захисту. Захист діє тільки при спробі запису в блок, якщо в цьому розряді стоїть 0, і при будь-якому зверненні до блоку, якщо стоїть 1. Коди ключів захисту пам’яті зберігаються в спеціальній пам’яті ключів захисту, більш швидкодіючої, ніж оперативна пам’ять. Функціонування цього механізму захисту пам’яті пояснюється схемою на рис. 12.6.

Рис. 12.6. **Захист пам’яті методом ключів захисту**

Призверненні до пам’яті група старших розрядів адреси ОЗП, що відповідає номеру блоку, до якого здійснюється звернення, використовується як адреса для вибірки з пам’яті ключів захисту коду ключа захисту, присвоєного операційною системою даному блоку.

Схема аналізу порівнює ключ захисту блоку пам’яті і ключ програми, що знаходиться в регістрі слова стану програми (ССП), і виробляє сигнал "Звернення дозволено" або сигнал "Переривання по захисту пам’яті". При цьому враховуються значення режиму звернення до ОЗП (запис або зчитування), що вказується тригерним режимом обіг і режиму захисту, встановленого в розрядірежиму обігу (РРО) ключа захисту пам’яті.

12.5. Засоби захисту памяті в персональній ЕОМ

Захист пам’яті в персональній ЕОМ [4] ділиться на захист при управлінні пам’яттю і захист за привілеями.

Засоби **захисту при управлінні пам’яттю** здійснюють перевірку:

перевищення ефективною адресою довжини сегмента;

прав доступу до сегмента на запис або тільки на читання;

функціонального призначення сегмента.

Перший механізм базується на методі граничних регістрів. При цьому початкові адреси того чи іншого сегмента програми встановлюються операційною системою. Для кожного сегмента фіксується його довжина. При спробі звернення за відносною адресою, що перевищує довжину сегмента, виробляється сигнал порушення захисту пам’яті.

При перевірці функціонального призначення сегмента визначаються операції, які можна проводити над розташованими в ньому даними. Так, сегмент, що є стеком програми, повинен допускати звернення як на запис, так і на читання інформації. До сегмента, що містить програму, можна звертатися тільки на виконання. Будь-яке звернення на запис або читання даних з цього сегмента буде сприйнято як помилкове. Тут спостерігається певний відхід від принципів Неймана в побудові ЕОМ, в яких стверджується, що будь-яка інформація, що знаходиться в ЗП, функціонально не поділяється на програму і дані, а її ідентифікація проводиться лише на стадії застосування даної інформації. Очевидно, що такий розвиток викликано багато в чому появою мультипрограмування і необхідністю більш уважного розгляду питань захисту інформації.

Захист по привілеям фіксує більш тонкі помилки, пов’язані, в основному, з розмежуванням прав доступу до тієї чи іншої інформації.

Якоюсь мірою захист по привілеях можна порівняти з класичним методом ключів захисту пам’яті. Різним об’єктам, а саме програмам, сегментам пам’яті, запитам на звернення до пам’яті і до зовнішніх пристроїв, які повинні бути розпізнані процесором, присвоюється ідентифікатор, званий рівнем привілеїв. Процесор постійно контролює, чи має поточна програма достатні привілеї, щоб:

виконувати деякі команди;

виконувати команди введення-виведення на тому чи іншому зовнішньому пристрої;

звертатися до даних інших програм;

викликати інші програми.

На апаратному рівні в процесорі розрізняються 4 рівні привілеїв. Найбільш привілейованими є програми на рівні 0.

Число програм, які можуть виконуватися на більш високому рівні привілеїв, зменшується від рівня 3 до рівня 0. Програми рівня 0 діють як ядро операційної системи. Тому рівні привілеїв зазвичай зображаються у вигляді чотирьох кілець захисту (рис. 12.7).

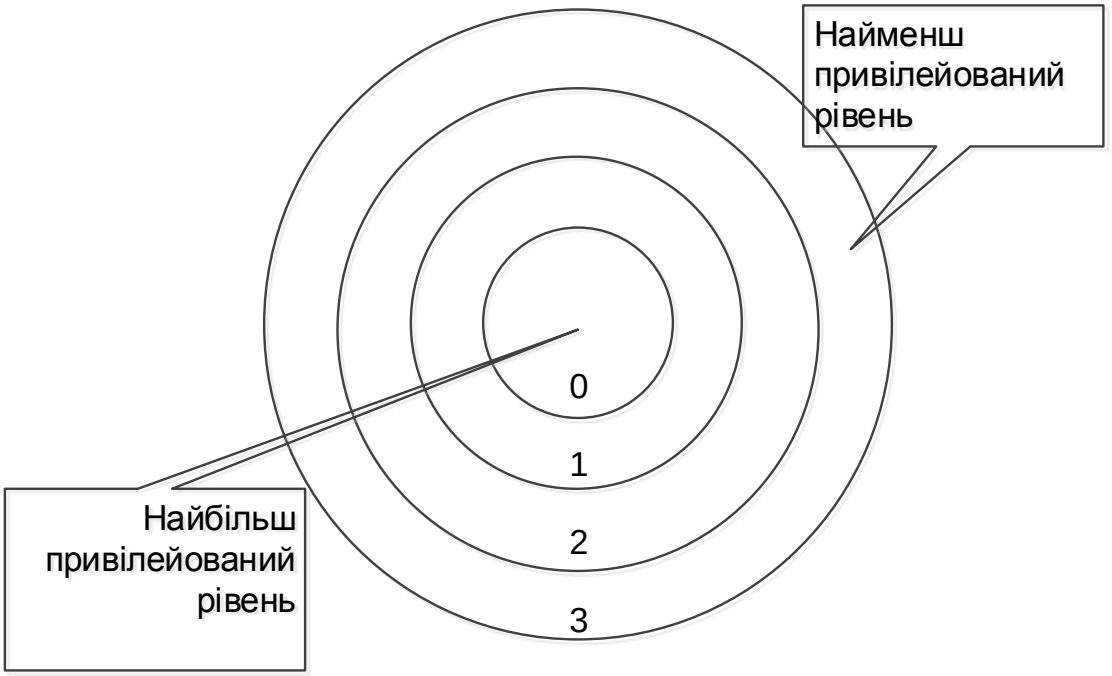


Рис. 12.7. **"Кільця захисту"**

Як правило, **розподіл програм по кільцях захисту** має такий вигляд:

рівень 0 – ядро ОС, що забезпечує ініціалізацію роботи, управління доступом до пам'яті, захист і ряд інших життєво важливих функцій, порушення яких повністю виводить з ладу процесор;

рівень 1 – основна частина програм ОС (утиліти);

рівень 2 – службові програми ОС (драйвери, СУБД, спеціалізовані підсистеми програмування та ін);

рівень 3 – прикладні програми користувача.

Конкретна операційна система не обов'язково повинна підтримувати всі чотири рівня привілеїв. Так, ОС UNIX працює з двома кільцями захисту: супервізор (рівень 0) та користувач (рівні 1,2,3). Операційна система OS / 2 підтримує три рівні: код ОС працює в кільці 0,

спеціальні процедури для звернення до пристроїв введення-виведення діють в кільці 1, а прикладні програми виконуються в кільці 3.

Просту незахищену систему можна цілком реалізувати в одному кільці 0 (в інших кільцях це зробити неможливо, оскільки деякі команди доступні тільки на цьому рівні).

Рівень привілеїв сегмента визначається полем DPL рівня привілеїв його дескриптора. Рівень привілеїв запиту до сегмента визначається рівнем привілеїв RPL, закодованому в селекторі. Звернення до сегмента дозволяється тільки тоді, коли рівень привілеїв сегмента не вище рівня запиту. Звернення до програм, які знаходяться на більш високому рівні привілеїв (наприклад, до утиліт операційної системи з програм користувача), можливо за допомогою спеціальних апаратних механізмів – шлюзів виклику.

При сторінковому перетворенні адреси застосовується простий дворівневий механізм захисту: користувач (рівень 3) / супервізор (рівні 0,1,2), що указується в полі U / S відповідної таблиці сторінок.

При сегментно-сторінковій перетворенні адреси спочатку перевіряються привілеї при доступі до сегмента, а потім – при доступі до сторінки. Це дає можливість встановити більш високий степінь захисту окремих сторінок сегмента.

Моделі безпеки, що застосовуються при побудові захисту в СУБД

При побудові СУБД широко використовується рольова модель безпеки. Її загальний вигляд у СУБД розглянемо далі.

На додаток до прав доступу, видаваним окремим користувачам, СУБД може надавати можливість призначення:

• певних прав доступу ролі, яка потім видається іншим;

• певних вбудованих груп прав доступу користувачам.

Звичайно, термінологія не дотримується строго у всіх основних СУБД. Деякі СУБД називають ролі групами і навпаки. Як адміністратору баз даних, вам необхідно розуміти, як кожна СУБД, якою ви керуєте, реалізує ролі і групи і як кожна з цих властивостей може бути використана для спрощення адміністрування безпеки бази даних.

Ролі

Як тільки вона визначена, роль може бути використана для видачі одного або декількох заздалегідь встановлених прав доступу користувачу. Роль по суті є набором прав доступу. Адміністратор баз даних

може створити роль і призначити певні права доступу для цієї ролі. Потім роль може бути присвоєна одному або декільком користувачам. Адміністрування безпеки бази даних таким чином спрощується. Наприклад, розглянемо таку послідовність операторів:

```
CREATE role MANAGER;
COMMIT;
GRANT select , insert, update, delete on employee to MANAGER;
GRANT select, insert, update, delete on job_title to MANAGER;
GRANT execute on payroll to MANAGER;
COMMIT;
GRANT MANAGER to userl;
COMMIT;
```

Цей сценарій створює нову роль під назвою MANAGER, видає права доступу до певних таблиць і процедурам для ролі, а потім привласнює користувачеві userl роль MANAGER.

Роль MANAGER може бути призначена й іншим користувачам, а адміністратору баз даних не потрібно буде пам'ятати про видачу для кожного з них окремого оператора GRANT, оскільки їм уже була привласнена роль MANAGER.

Групи

Право доступу рівня групи схоже з роллю. Проте кожна СУБД надає вбудовані групи, які не можуть бути змінені. Кожна СУБД реалізує безпеку бази даних на рівні групи різними шляхами, під різними назвами і з різними правами доступу. Однак існують деякі подібні елементи в усіх СУБД. Далі перелічені групи є загальними для основних СУБД.

Системний адміністратор. Іноді скорочується як SA або SYSADM, група системного адміністратора є найбільш потужною в СУБД. Користувач, якому надається право доступу рівня системного адміністратора, звичайно може виконувати всі команди бази даних і отримувати доступ до всіх баз даних і об'єктів. Системний адміністратор зазвичай відповідає за установку СУБД і вважається власником системних ресурсів і таблиць системного каталогу.

Адміністратор баз даних. Іноді скорочується як DBADM або DBA, група адміністратора баз даних дає всі права доступу до визначеної бази даних плюс здатність отримувати доступ, але не модифікувати дані в таблицях в межах цієї бази даних. Користувачі, яким присвоєно право

доступу рівня адміністратора баз даних, можуть видаляти або змінювати будь-які об’єкти в базі даних (табличні області, таблиці та індекси).

Обслуговування бази даних. Іноді скорочується як DBMAINT, група обслуговування баз даних включає певні права доступу до бази даних для обслуговування об’єктів баз даних (як наприклад, здатність запускати утиліти і видавати команди). Подібно групі адміністраторів баз даних, право доступу рівня DBMAINT надається на основі бази даних.

Адміністратор безпеки. Роль адміністратора безпеки має набір прав доступу, що дозволяє видачу та скасування безпеки бази даних у СУБД.

Будь-яка діяльність, пов’язана з безпекою бази даних, може виконуватися адміністратором безпеки, включаючи адміністрування реєстраційних імен і паролів, аудит, конфігурування властивостей безпечності, а також видачу GRANT і REVOKE. Іншою загальною назвою ролі адміністратора безпеки є SSO.

Контроль операцій. Іноді звана OPER або SYSOPR, роль контролю операцій має право виконувати завдання працюючої бази даних, такі, як резервування і відновлення або завершення завдань: які вийшли з-під контролю.

Обмеження кількості користувачів-системних адміністраторів

Організація повинна обмежувати кількість користувачів, яким назначається роль системного адміністратора або право доступу рівня групи. Користувач з можливостями системного адміністратора має дуже потужну владу. Тільки корпоративні адміністратори баз даних і системні програмісти повинні отримувати права доступу такого рівня. Кінцеві користувачі, менеджери і персонал розробки додатків не потребують правах доступу системного адміністратора для виконання своєї роботи.

Безпека рівня групи і каскадні REVOKE. Залежно від групи деякі користувачі, яким були призначені права доступу рівня групи, можуть видавати права доступу іншим користувачам. Якщо право доступу рівня групи скасовується для цього користувача, будь-які права доступу, видані цим користувачем, також будуть скасовані. Це подібно каскадним REVOKE, які відбуваються в результаті вибору WITH GRANT OPTION.

Перед скасуванням права доступу рівня групи для користувача переконайтеся, що встановили вплив каскадних REVOKE і готові до того, щоб встановити повторно необхідні права доступу, які будуть видалені через ефект каскадування.

Використання транзакції для ізолювання дій користувачів

Визначення транзакцій

Транзакція – це атомарна частка роботи щодо відновлення і узгодженості. Логічна транзакція виконує повний діловий процес зазвичай від особи онлайн-користувача. Вона може складатися з декількох етапів та містити більше однієї фізичної транзакції. Результати запуску транзакції реєструють вплив ділового процесу – завершеного ділового процесу. Після виконання транзакції дані в базі даних повинні бути правильними і коректними.

Коли всі етапи, які складають певну транзакцію, будуть виконані, буде виданий COMMIT. COMMIT сигналізує про те, що вся робота починаючи з останньої фіксації (COMMIT), була коректна і повинна бути поміщена в базу даних. У будь-якій точці транзакції може бути прийнято рішення про зупинку і відкат впливів всіх змін, починаючи з останнього COMMIT. При відкаті транзакції дані в базі даних будуть перезаписані в первинний стан до запуску транзакції. СУБД зберігає журнал реєстрації транзакцій для відстеження змін бази даних.

Транзакції мають властивості АУІС (АУІС). АСІД – це абревіатура таких властивостей, як атомарність, узгодженість, ізольованість і стійкість. Кожне з цих чотирьох властивостей необхідно для правильного проектування транзакції.

Атомарність означає, що транзакція повинна представляти поведінку "все або нічого". Або всі з вказівок відбуваються в транзакції, або ні одне з них. Атомарність зберігає повноту виконання ділового процесу.

Узгодженість відноситься до стану даних як до, так і після виконання транзакції. Транзакція підтримує узгодженість стану даних. Іншими словами, після запуску транзакції всі дані в базі даних є "коректними".

Ізольованість означає, що транзакції можуть працювати одночасно. Будь транзакції, запущені паралельно, мають ілюзію, що паралельності немає. Іншими словами, здається, що система запускає тільки одну транзакцію одночасно. Жодну з інших паралельних транзакцій не видно незавершеним модифікаціям бази даних, виконуваним іншими транзакціями. Щоб досягти ізольованості, потрібно використовувати механізм блокування.

Стійкість відноситься до впливу простою або збою в роботі на діючу транзакцію. Стійка транзакція не буде впливати на стан даних, якщо робота транзакції припиниться неправильно. Дані переживуть будь-які збої.

Отримання несанкціонованого доступу до конфіденційної інформації шляхом логічних висновків

Захист СУБД є одним з найпростіших завдань. Це пов'язано з тим, що СУБД мають чітко визначену внутрішню структуру, і операції над елементами СУБД задані досить чітко. Є чотири основних дії – пошук, вставка, видалення і заміна елемента. Інші операції є допоміжними і застосовуються досить рідко. Наявність чіткої структури і чітко визначених операцій спрощує рішення задачі захисту СУБД. У більшості випадків хакери вважають за краще зламувати захист комп'ютерної системи на рівні операційної системи і отримувати доступ до файлів СУБД за допомогою засобів операційної системи. Однак у випадку, якщо використовується СУБД, що не має достатньо надійних захисних механізмів, або погано протестована версія СУБД, що містить помилки, або якщо при визначенні політики безпеки адміністратором СУБД були допущені помилки, то стає цілком ймовірним подолання хакером захисту, реалізованого на рівні СУБД.

Крім того, існує два специфічних сценарії атаки на СУБД, для захисту від яких потрібно застосовувати спеціальні методи. У першому випадку результати арифметичних операцій над числовими полями СУБД округляються в меншу сторону, а різниця підсумовується в деякому іншому записі СУБД (як правило, цей запис містить особовий рахунок хакера в банку, а числові поля, які округляються, відносяться до рахунків інших клієнтів банку). У другому випадку хакер отримує доступ до полів записів СУБД, для яких доступна тільки статистична інформація. Ідея хакерської атаки на СУБД – так хитро сформулювати запит, щоб безліч записів, для якого збирається статистика, складалося тільки з одного запису.

Особливості застосування криптографічних методів

Для полегшення використання баз даних існує безліч менеджерів (засобів управління базами даних), для забезпечення криптографічного захисту в засоби управління додані функції по шифруванню і дешифруванню даних. Такі функції в основному працюють практично за кадром тобто користувачу або адміністратору досить вказати що дані є приватними або подібними мітками позначити криптографічний захист. Ось деякі приклади:

Захищеність (Security) в Microsoft Message Queue Server. Сервер черг використовує списки прав доступу (ACL) Windows NT для визначення прав доступу до черги. Застосовуються також інші системи захисту, засновані на використанні відкритих (Public) ключів (система

захисту заснована на використанні API з RSA-провайдером). Для перевірки цілісності повідомлень застосовуються контрольні суми. Сервер черг використовує алгоритми шифрування RC2 і RC4; за замовчуванням вибирається алгоритм RC2. Для шифрування повідомлення додаток простопривласнює йому атрибут "private", все інше буде зроблено автоматично.

Технології віддаленого доступу до систем баз даних, тиражування та синхронізація в розподілених системах баз даних

В умовах, коли швидко розвиваються технології, доступність інформації є дуже важливим фактором, в умовах віддаленого доступу хорошим способом є кошти мережі Інтернет, розглянемо доступ до сервера СУБД через web-сервер. Для доступу web-навігатора до сервера СУБД через web-сервер використовується система програмних шлюзів. Програмний шлюз, отримавши запит від web-сервера, виступає в якості посередника між сервером web і сервером СУБД. Програмні шлюзи розробляються відповідно до визначених стандартів, визначальними способи виклику web-сервером прикладних програм або функцій динамічних бібліотек, а також способи обміну інформацією з цими програмними об’єктами. Одними з найбільш поширених стандартів даного типу є інтерфейс CGI (Common Gateway Interface – загальний інтерфейс шлюзів), а також його вдосконалена специфікація, названа як FastCGI (прискорений CGI).

Інтерфейс CGI. Для доступу web-навігатора до сервера СУБД через Web-сервер за стандартом CGI необхідна відповідна CGI-програма, що виконує роль програмного шлюзу між web-сервером і сервером СУБД (рис. 12.8).

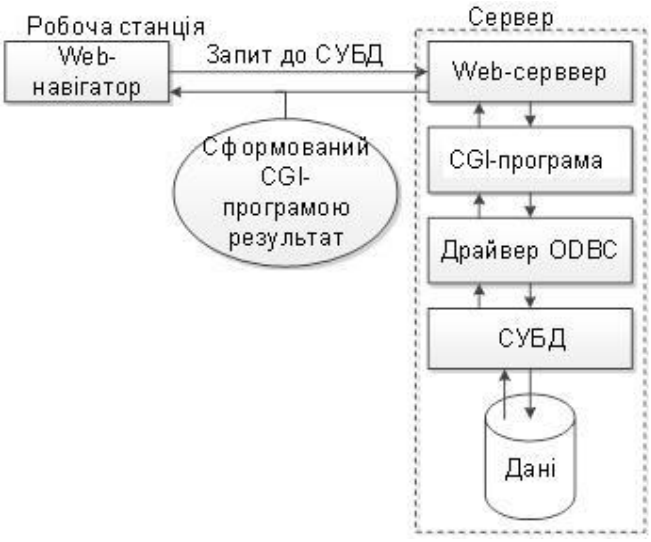


Рис 12.8. **Схема доступу до СУБД через CGI-програму**

CGI-додатки працюють незалежно від web-сервера, а їх запуск здійснюється за викликом з web-документа при його обробці web-навігатором. CGI-програма взаємодіє з web-сервером за допомогою двостороннього обміну змінними середовища через стандартні канали вводу/виводу даного додатка.

Оскільки CGI-програми працюють незалежно від web-сервера і мають простий спільний інтерфейс, розробники web-документів мають можливість створювати свої CGI-програми будь-якою мовою, що підтримує стандартні файлові операції вводу/виводу. Крім того, при незалежній розробці можна створювати такі додатки, які з легкістю переносяться з одного на інший web-сервер. Існують і стандартні CGI-програми, спеціально розроблені для взаємодії web-серверів з різними СУБД, наприклад, програма WebDBC.

У якості інтерфейсу між web-навігатором і сервером СУБД у складі web-документів застосовують HTML-форми, які дозволяють формулювати запити до бази даних. CGI-програма отримує інформацію від web-сервера або через змінні оточення, або через стандартний ввід. Все залежить від методу доступу, який використовується при обміні даними між web-навігатором і web-сервером. Далі CGI-програма через драйвер ODBC (Open DataBase Connectivity) звертається до сервера СУБД і повертає web-серверу відповідь на запит через стандартний вивід.

Драйвер ODBC забезпечує уніфікований спосіб доступу до різних СУБД за допомогою стандартної мови запитів SQL. Завдяки стандарту ODBC прикладні програми можуть використовувати єдиний діалект SQL і взаємодіяти з різними СУБД. Можна обійтися і без драйвера ODBC, але в цьому випадку CGI-програма повинна бути написана з орієнтацією на конкретну СУБД, що функціонує на сервері.

Таким чином, розробнику CGI-додатки не треба нічого знати про те, як влаштований web-сервер. Більше того, йому зовсім не обов’язково використовувати складні мови типу C++. CGI-програма може бути написана і командною мовою, наприклад Perl. Головне – витримати всі угоди, що накладаються стандартом CGI. Такий підхід істотно полегшує розробку прикладного програмного забезпечення для web взагалі і для сполучення баз даних з web-сервером зокрема.

Стандарт CGI володіє і недоліком – зниження швидкості обробки заптів при збільшенні інтенсивності їх надходження. При кожному виклику CGI-програми її доводиться завантажувати з диска (тобто

запускати новий процес). По завершенні роботи програми потрібно звільнити ресурси, які використовувалися нею. Такі операції створюють помітне додаткове навантаження на сервер, що позначається на його продуктивності. До того ж запуск нового процесу при кожному запиті знижує ефективність постійних процесів і доступність даних. Інформацію, яка згенерована в ході обробки одного запиту, неможливо використовувати при обробці іншого.

Інтерфейси API і FastCGI

Для того, щоб обійти проблеми, пов'язані зі швидкодією CGI, багато постачальників web-серверів, включаючи Microsoft і Netscape, розробили відповідні інтерфейси прикладного програмування (API). Корпорацією Microsoft був розроблений інтерфейс ISAPI (Інтернет Server API), а корпорацією Netscape – інтерфейс NSAPI (Netscape Server API).

Ці інтерфейси тісно інтегровані з web-сервером, дозволяючи зберігати доступність постійно використовуваних процесів і даних. Програми з інтерфейсом ISAPI компілюються у файли, динамічно підключаються бібліотек DLL. Вони завантажуються в пам'ять під час першого звернення до них і тому для повторного виклику цих програм не потрібно породжувати новий процес. Функції інтерфейсу NSAPI завантажуються в серверний простір процесів. Відповідно при виклику цих функцій також не породжуються додаткові процеси. Завдяки API-інтерфейсу програма, яка його використовує, може залишати з'єднання з СУБД відкритим, тому наступному запиту до бази даних не доведеться витрачати час на відкриття і закриття з'єднання.

Однак API-інтерфейси web-серверів – хоч і непогане, але нестандартне рішення. Більшість додатків можна переносити з одного API на інший, і дуже рідко вдається перенести додатки на інші платформи. Крім того, більшість додатків для web-серверів все ще створюються для інтерфейсу CGI, тому перехід до додатків на базі API не є економічно виправданим.

Тому стали з'являтися способи побудови деякого проміжного варіанта, який, з одного боку, задовольняв би вимогам мобільності, незалежності та простоти програмування, а з іншого боку – був би досить ефективним. Одним з таких рішень є специфікація FastCGI. Ідея цієї специфікації у тому, що прикладна програма використовує спосіб передачі параметрів і даних, який використовується в CGI, але при цьому не видаляється з пам'яті, а залишається резидентною, обробляючи запити, що надходять.

Таким чином, додатки на базі FastCGI, подібно CGI-програмами, працюють незалежно від web-сервера і запускаються через стандартні посилання в web-документах. Але, як і програми на базі API, програми для FastCGI є постійно діючими. Коли програма закінчує обробку чергового запиту, її процес залишається відкритим в очікуванні нового запиту.

При доступі web-навігатора до реляційної бази даних через інтерфейс FastCGI виходить схема, в якій фактично використовуються три сервера: web-сервер, FastCGI-програма і сервер бази даних. web-сервер отримує запит web-навігатора і передає його FastCGI-програмі, яка в свою чергу звертається до сервера баз даних. Результат повертається за зворотнім ланцюжком. Наведемо приклад дещо іншої технології віддаленого доступу: RMI-додатки.

RMI (Remote Method Invocation) – виклик віддалених методів, забезпечує засоби комунікації між Java-програмами, навіть якщо вони виконуються на різних комп'ютерах, що знаходяться в протилежних точках.

Важлива особливість RMI полягає в тому, що він представляє програмуємий інтерфейс, який програмується, для роботи з мережами на відміну від сокетів TCP. Головна перевага його в тому, що він пропонує вам інтерфейс більш високого рівня, заснований на викликах методів так, як якщо б видалений об'єкт оброблявся локально. RMI більш зручний і природний, ніж інтерфейс, заснований на сокетах, але він вимагає виконання Java-програм на обох кінцях з'єднання. Мережне з'єднання досягається використанням все того ж TCP-IP-протоколу.

Кластерна організація серверів баз даних

Дублювання серверів

Описані способи забезпечення надійного зберігання даних можна використовувати, тільки коли сам сервер працює надійно. У разі несправності сервера його клієнти будуть на якийсь час позбавлені можливості доступу до даних. І це триватиме тим довше, чим більше часу буде потрібно адміністратору на відновлення самого сервера або даних з стрічки на іншому сервері. Тому виникає необхідність резервування інформації в реальному масштабі часу.

Фірмою Octopus Technologies Inc випускається продукт Octopus Real Time Data Protection. З його допомогою можна організувати дублювання файлів або каталогів з одного сервера на один або декілька

інших серверів або робочих станцій. Інформацією, що посилається в дублікати каталогів, є тільки оновлення файлів і каталогів. Наприклад, якщо використовувати Octopus для резервування бази даних, по мережі реально передаються тільки оновлення (видалені або додані записи). Якби кожного разу передавалася вся база, то це призвело б до різкого зростання трафіка в мережі і зниження її продуктивності.

Ще одна перевага полягає в тому, що сервери пов'язані звичайною мережею, і при цьому не вимагається особливий протокол. Досить встановити серверну частину на комп'ютер, дані якого повинні резервуватися, і клієнтську частину – на комп'ютери, куди будуть надходити дані. Можливі такі комбінації резервування: з одного сервера на інший, з одного на кілька інших, з декількох на кілька і з декількох на один. Оновлення файлів виконуються на рівні транзакцій.

У разі краху сервера при використанні стандартної версії Octopus адміністратор системи повинен або відновити вихідний сервер, або перемкнути всіх користувачів на той, який був дублікатом. Останні версії Octopus мають додаткову функцію автоматичного перемикання, що дозволяє відразу ж перемикати користувачів на дублікат сервера при краху основного. У будь-якому випадку останні версії файлів залишаться збереженими.

Незважаючи на простоту і відносно невисоку вартість, у такої системи є мінус: користувачам необхідно переключатися на новий сервер, перезапускати процеси і заново відкривати файли. До того ж при цьому втрачається і вміст оперативної пам'яті.

Кластери серверів

Вища надійність досягається за допомогою кластерних систем, що забезпечують не тільки надійність зберігання даних, але і безперервність процесів, що виконуються в кластері, а також продуктивність, обумовлену сукупністю всіх серверів, що входять у кластер.

Кластером називається група незалежних систем, які працюють як єдине ціле. Клієнт взаємодіє з кластером як з одним сервером. Кластери використовуються як для підвищення доступності, так і для нарощуваності.

Доступність: коли одна з систем, складових кластера, виходить з ладу, програмне забезпечення кластера розподіляє роботу, виконувану нею, між іншими системами кластера.

Нарощуваність: коли загальне завантаження системи досягає межі можливостей систем, які складають кластер, його можна наростити, додавши додаткову систему. Раніше для цього потрібно було здобувати нові комп'ютери, що дозволяють встановити додаткові процесори, диски і пам'ять. За допомогою кластерів можна збільшити продуктивність в міру необхідності, просто додаючи нові системи.

Наприклад, можна навести роботу сучасного супермаркету. Касові апарати в його розрахункових центрах повинні бути постійно підключені до бази даних магазину, що зберігає інформацію про продукти, коди, назви і ціни. Якщо зв'язок рветься, то втрачається можливість обслуговування клієнтів, падає прибуток, і, що найгірше – підривається репутація підприємства.

У цьому випадку кластерна технологія використовується для підвищення доступності системи. В даному випадку можна запропонувати використання двох систем, підключених до багатопортового дискового масиву, на якому розташовується база даних. У разі збою одного з серверів резервна система (інший сервер) автоматично підхопить з'єднання так, що користувачі і не помітять, що стався збій. Таким чином, комбінуючи стандартну технологію забезпечення підвищеної надійності роботи з диском (чергування, дублювання і тощо) з кластерною технологією, можна забезпечити гарантовану достовірність інформації в системі.

Нарощуваність системи дозволяє відмовитися від дорогого устаткування ПК і використовувати широко розповсюджену систему на звичайних апаратних платформах навіть у таких ресурсоємних додатках, як фінансові або банківські системи. Нарощування потужності досягається додаванням ще однієї системи в кластер.

Традиційна архітектура високої надійності та нарощуваності

Підвищення надійності комп'ютерних систем досягається кількома способами. Самий типовий – дублювання систем з повністю повторюваними компонентами. Програмне забезпечення такої системи повністю відстежує стан працюючої системи, а друга система весь цей час простоює. У разі збою першої відбувається перемикання на другу. Такий підхід, з одного боку, значно підвищує вартість устаткування без підвищення продуктивності системи в цілому, а з іншого – не забезпечує захисту від помилок у додатках.

Нарощуваність сьогодні забезпечується декількома способами. Створити систему з нарощуваною продуктивністю можна, наприклад, використавши симетричну мультипроцесорну обробку (СМПО). У таких системах декілька процесорів використовують одну загальну пам’ять і пристрої введення/виводу. У традиційній моделі "спільного використання пам’яті" виконується одна копія операційної системи, а процеси прикладних задач працюють так, ніби в системі лише один процесор. При запуску в такій системі додатків, що не використовують загальні дані, досягається високий степінь нарощуваності. Основним гальмуючим фактором використання систем з симетричною обробкою є фізичні обмеження швидкості роботи шини і доступу до пам’яті. У міру збільшення швидкості роботи процесорів зростає їх вартість.

Архітектура кластера

Кластери можуть мати різні форми. Наприклад, в якості кластера може виступати кілька комп’ютерів, зв’язаних мережею Ethernet. Кластерами високого рівня є високопродуктивні багатопроцесорні СМПО-системи, пов’язані високошвидкісною шиною. В обох випадках збільшення обчислювальної потужності досягається невеликими кроками при додаванні чергової системи. З точки зору клієнта, кластер представляється у вигляді одного сервера або образу однієї системи, хоча реально складається з декількох комп’ютерів. Сьогодні в кластерах використовуються в основному дві моделі: з загальними дисками і без загальних компонент.

Модель із загальними дисками

У моделі з загальними дисками програмне забезпечення, що виконується на будь-якій з систем, що входять в кластер, має доступ до ресурсів систем кластера. Якщо двом системам потрібні одні і ті ж дані, то вони або двічі зчитуються з диска, або копіюються з однієї системи на іншу. У СМПО-системах додаток повинен синхронізувати і перетворити в послідовний вид доступ до загальних даних. Зазвичай для організації синхронізації використовується Диспетчер розподілених блокувань (DLM). Служба DLM дозволяє додаткам відслідковувати звернення до ресурсів кластера. Якщо до одного ресурсу звертається більше двох систем одночасно, Диспетчер розподілених блокувань розпізнає і запобігає потенційний конфлікт. Процеси DLM можуть призводити до додаткового трафіку повідомлень у мережі і знизити продуктивність. Один із способів уникнути цього ефекту – програмна модель без загальних компонент.

Модель без загальних компонент

У моделі без загальних компонент кожна система, що входить в кластер, володіє підмножиною ресурсів кластера. У кожен момент часу тільки одна система має доступ до певного ресурсу, хоча при збоях інша динамічно обумовлена система може вступити у володіння цим ресурсом. Запити від клієнтів автоматично перенаправляються до тих систем, які володіють необхідним ресурсом.

Якщо в запиті клієнта міститься звернення до ресурсів, що знаходяться у володінні кількох систем, то одна вибирається для обслуговування запитів (її називають хост-системою). Потім вона аналізує запит і передає підзапити відповідним підсистемам. Вони виконують отриману частину запиту і результат повертають хост-системі, яка формує остаточний результат і відсилає його клієнту. Додаток, розподілений між декількома системами, що входять в кластер, дозволяє подолати технічні обмеження, властиві одному комп'ютеру. Моделі із загальним диском і модель без загальних компонент можуть працювати в межах одного кластера. Деякі програми найбільш просто використовують можливості кластера в рамках моделі із загальним диском. До таких додатків належать завдання, що вимагають інтенсивного доступу до даних, а також завдання, які важко розділити на частини. Додатки, для яких важлива нарощуваність, повинні використовувати модель без загальних компонент.

Контрольні запитання

1. Основні вимоги, які висуваються до обчислювальних мереж та інформаційних систем.
2. Модель реалізації загроз інформаційних ресурсів в інформаційних системах.
3. Моделі порушника в сучасних глобальних (локальних) мережах та інформаційних системах.
4. Організація захисту пам'яті в сучасних ПК.
5. Захист пам'яті ПК методом граничних регістрів.
6. Захист пам'яті методом ключей захисту.
7. Моделі безпеки, що застосовуються при побудові захисту в СУБД.

Розділ 13. Використання паролів і механізмів контролю за доступом

13.1. Формальні моделі доступу. Дискреційний та мандатний доступ до інформації

Схему класифікації та взаємозв'язки математичних моделей безпеки комп'ютерних систем подано на рис. 13.1.

Розглянемо деякі моделі доступу детальніше.

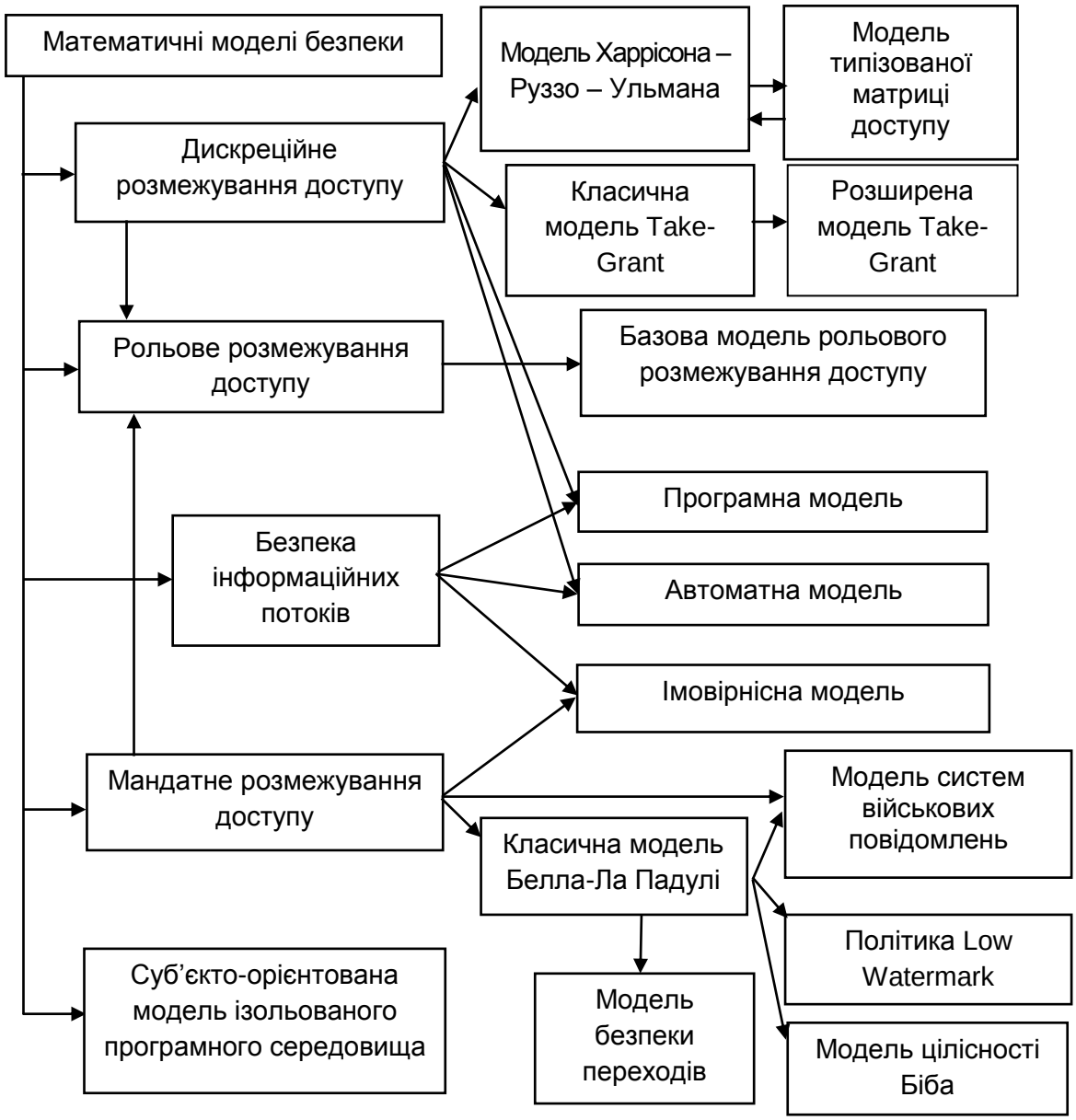


Рис. 13.1. **Схема класифікації та взаємозв'язки математичних моделей безпеки комп'ютерних систем**

Усі математичні моделі безпеки поділяються, згідно з на п’ять класів:

моделі систем дискреційного розмежування доступу;

моделі систем мандатного розмежування доступу;

моделі безпеки інформаційних потоків;

моделі рольового розмежування доступу;

суб’єктно-орієнтована модель ізольованого програмного середовища.

13.1.1. Дискреційна модель доступу

Дискреційна модель доступу будується на основі дискреційного розмежування доступом (*Discretionary Access Control*), яке визначається двома правилами:

усі суб’єкти та об’єкти системи ідентифіковано;

права доступу суб’єктів до об’єктів визначаються на основі деяких зовнішніх щодо системи правилах.

Основним елементом систем з дискреційним розмежуванням доступу є матриця доступу.

Якщо {*S*} – множина суб’єктів; {*O*} – множина об’єктів системи; {*R*} – множина типів доступу, тоді матриця доступу визначається як прямокутна матриця розмірності *S*×*O*, рядки якої відповідають суб’єктам, а стовпчики – об’єктам. При цьому кожен елемент матриці $M(s,o) \subseteq R$ визначає права доступу суб’єкта *S* на об’єкт *O*.

До ***переваг дискреційної моделі*** доступу можна віднести відносно просту реалізацію системи розмежування доступу. Цим і зумовлений той факт, що більшість розповсюджених сьогодні комп’ютерних систем реалізують саме дискреційну політику розмежування доступу. Ще однією перевагою вважають високу деталізацію доступу.

До недоліків дискреційної моделі слід віднести статичність визначених у ній правил розмежування доступу. Ця модель не враховує динаміку зміни стану комп’ютерної системи. Крім того, її складно адмініструвати.

Існує ще один недолік: оскільки правила розмежування доступу – зовнішні, система, взагалі кажучи, не може надати надійних механізмів перевірки того, чи не призведуть ті чи інші дії користувача до порушень політики безпеки. Виняток становить модель Take-Grant.

Вказані недоліки змусили шукати інші, досконаліші моделі політики безпеки.

У якості прикладу дискреційної моделі доступу розглянемо модель ***Харрісона – Руззо – Ульмана***.

Нехай {*S*} – множина суб'єктів; {*O*} – множина об'єктів системи; {*R*}={r,w,e} – множина типів доступу (r – read; w – write; e – execute). Тоді матриця доступу буде мати вигляд:

	O_1	O_2	…	O_m
S_1	r	-	…	r,w
S_2	-	e	…	r
…	…	…	…	…
S_n	r,w	r,w,e	…	r,w,e

Оскільки модель Харрісона – Руззо – Ульмана належить до дискреційних, усі переваги та недоліки останніх притаманні і їй. Зокрема, вона не надає надійних механізмів перевірки легітимності дій користувача з точки зору політики безпеки. Також вона не здатна сама слідкувати за зміною рівня безпеки інформації.

13.1.2. Мандатна модель доступу

Мандатна (повноважна) політика безпеки ґрунтується на мандатному розмежуванні доступу (Mandatory Access Control), яке повинно задовольняти чотири вимоги:

усі суб'єкти та об'єкти системи однозначно ідентифіковані;

задано рівні конфіденційності інформації;

кожному об'єкту системи привласнено рівень конфіденційності, який визначає цінність інформації, що міститься в ньому;

кожному суб'єкту привласнено певний рівень доступу (повноважень), який визначає рівень довіри до нього в комп'ютерній системі.

Основна мета мандатної моделі доступу полягає в тому, щоб перешкоджати витоку інформації від об'єктів з високим рівнем конфіденційності до об'єктів з низьким рівнем, тобто протидіяти виникненню небезпечних інформаційних потоків "з гори-вниз".

Основною ***перевагою мандатної системи*** доступу є те, що вона сама слідкує за збереженням рівня конфіденційності інформації та протидіє його зниженню. Вона також простіша в адмініструванні.

Недоліками цієї моделі доступу вважають те, що вона не розмежовує права доступу до інформації всередині одного рівня конфіденційності, а також виникнення деяких парадоксів, які зумовлені простотою формальної моделі.

Часто поняття мандатної моделі доступу описуються в термінах моделі Белла-Ла Падулі.

Модель доступу Белла-Ла Падулі. Нехай {*S*} – множина суб'єктів; {*O*} – множина об'єктів системи; {*R*}={r,w,e} – множина типів доступу (r – read; w – write; e – execute). Нехай також визначено рівні конфіденційності об'єктів L_O={*U*,*SU*,*S*,*TS*} (Unclassified (*U*) – несекретно; Sensitive but Unclassified (*SU*) – для службового користування (ДСК); Secret (*S*) – таємно; Top Secret (*TS*) – цілком таємно). Аналогічну множину можна обрати для рівнів доступу суб'єктів: L_S={*U*,*SU*,*S*,*TS*}.

Для того, щоб система протидіяла зниженню рівня конфіденційності інформації, було запропоновано два внутрішніх правила, згідно з якими надається доступ: ***Don't read up*** (заборонити читати інформацію вищого рівня конфіденційності) та ***Don't write down*** (заборонити записувати інформацію в об'єкти нижчого рівня конфіденційності). Такий підхід інтуїтивно зрозумілий: користувач не повинен читати інформацію вищого рівня конфіденційності, якщо він має нижчий (don't read up), а також не може записувати свою інформацію в об'єкт нижчого рівня, щоби користувачі з нижчим рівнем доступу не могли з нею ознайомитися (don't write down).

Доступ відбувається в такий спосіб. Коли суб'єкт "просить" систему безпеки надати йому доступ на читання до деякого об'єкта, система розмежування доступу порівнює його рівень доступу з рівнем конфіденційності об'єкта і надає доступ, якщо $L_S \geq L_O$. У разі, якщо суб'єкт просить доступ на запис, його буде надано, якщо $L_O \geq L_S$.

Звідси яскраво видно один з недоліків моделі Белла-Ла Падулі: можна записати нашу інформацію у вищій рівень конфіденційності, а прочитати її потім не можна.

Для ілюстрації розглянемо дворівневу модель Белла-Ла Падулі (рис. 13.2). Нехай є два суб'єкти $S = \{S_1, S_2\}$ і два об'єкти $O = \{O_1, O_2\}$; список типів доступу налічує усього читання та запис: *R* = {R,W}.

Рівні конфіденційності об’єктів та рівні доступу суб’єктів такі: L_O = = {1,2}; L_S= {1,2}, причому рівень 1 вважається вищим за рівень 2.

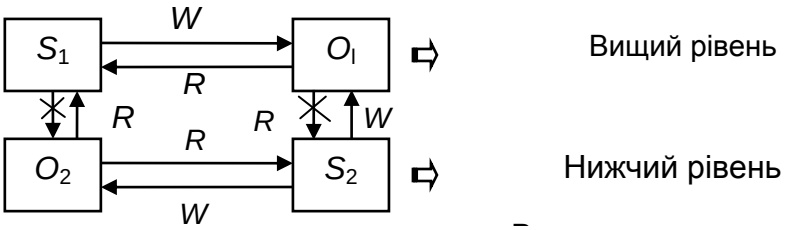


Рис. 13.2. **Дворівнева модель Белла-Ла Падулі**

Проаналізуємо отриману модель. Як можна спостерігати, всередині одного рівня доступ на запис та читання не забороняється. Однак, проблема тут полягає в іншому: як в межах моделі розмежувати права доступу до об’єктів різних користувачів, що мають один рівень доступу? В таких випадках всередині одного рівня конфіденційності використовують дискреційну модель доступу.

Що стосується різних рівнів конфіденційності, то, як бачимо, S_1 отримає доступ до об’єкта O_2 на читання, а от записати туди свою інформацію не зможе. Більш екстремальним видається інший випадок: модель дозволяє суб’єкту S_2 записати свою інформацію в об’єкт O_1, а прочитати її система розмежування доступу потім не дозволить.

Для подолання вказаних недоліків було розроблено мандатні моделі зі зміною атрибутів доступу в процесі роботи системи. Однією з таких модифікованих моделей вважається ***модель ватерлінії*** (Low Watermark Model).

У цій моделі, якщо суб’єкт *S* читає об’єкт *O*, рівень конфіденційності якого $L_O > L_S$, то доступ надається (правило *don’t read up* в моделі Белла-Ла Падулі), однак рівень доступу суб’єкта "підтягується", тобто $L_S = L_O$. В разі, коли суб’єкт хоче писати в об’єкт при $L_S > L_O$, то йому також надається доступ (правило *don’t write down* в моделі Белла-Ла Падулі), але рівень конфіденційності об’єкта автоматично підвищується до рівня суб’єкта, тобто знову $L_S = L_O$.

Політика безпеки інформаційних потоків зосереджена на розмежуванні усіх інформаційних потоків між об’єктами системи на дві множини, що не перетинаються: множину корисних потоків та множину небезпечних потоків. Мета політики інформаційних потоків полягає

в тому, щоб забезпечити неможливість виникнення небезпечних потоків у комп'ютерній системі. Реалізація цієї політики досить складна, особливо, якщо треба забезпечувати неможливість виникнення небезпечних потоків по часу. Політика безпеки інформаційних потоків застосовується, як правило, разом з політикою дискреційного або мандатного розмежування доступу.

Політика рольового розмежування доступу є розвитком дискреційної моделі доступу; при цьому права доступу суб'єктів групуються з врахуванням специфіки їх застосування, утворюючи ролі.

Створення ролей робить дискреційну політику безпеки зрозумілішою для користувачів. На її основі можна реалізувати гнучкі, динамічні правила розмежування доступу, в тому числі, на основі ролевого розмежування можна реалізувати мандатну модель.

Політика ізольованого програмного середовища реалізується на основі ізолювання суб'єктів таким чином, щоби вони могли породжувати інших суб'єктів лише з дозволеного переліку. При цьому необхідно контролювати цілісність об'єктів системи, які впливають на функціональність суб'єктів, що активізуються.

Інша група моделей розглядає питання захисту обчислювальних систем від загроз цілісності інформації. Однією з таких моделей є ***модель цілісності Біба***.

У моделі існують рівні цілісності суб'єктів та об'єктів I_S та I_O. При цьому доступ на читання надається лише тоді, коли $I_S \le I_O$, а доступ на запис – якщо $I_O \le I_S$. Ці два правила, по аналогії з моделлю Белла-Ла Падулі, можуть бути описані таким чином:

don't read down (це називається простим правилом цілісності);

don't write up (це називають – правилом цілісності).

Ці правила також інтуїтивно зрозумілі. Система повинна перешкоджати зниженню рівня цілісності, а отже, суб'єкт може читати лише інформацію вищого рівня цілісності, однак змінювати її не може. Змінювати він може лише інформацію з меншим рівнем цілісності.

Хорошим прикладом можуть служити файли операційної системи. Вони повинні мати найвищий рівень цілісності. Необхідно захищати їх від зміни (запису). Водночас вони повинні бути доступними для усіх користувачів, отже, рівень конфіденційності у них – найнижчий.

Існують також моделі Біба з пониженням рівня цілісності об’єкта та з пониженням рівня цілісності суб’єкта. В першому випадку після операції запису інформації суб’єктом в об’єкт рівень цілісності об’єкта зменшується до рівня цілісності суб’єкта (так що $I_S = I_O$), в другому – після операції читання рівень цілісності суб’єкта зменшується до рівня прочитаного об’єкта (так що знов $I_S = I_O$).

Якщо потрібно одночасно захищати конфіденційність та цілісність інформації, використовують композитні моделі розмежування доступу, які об’єднують модель конфіденційності Белла-Ла Падулі та модель цілісності Біба.

13.2. Аналіз захищеності сучасних операційних систем

При оцінці ступеня захищеності операційних систем діє нормативний підхід, згідно з яким сукупність завдань, що виконується системою безпеки, повинна задовольняти певні вимоги. Їх перелік визначається загальноприйнятими стандартами, наприклад, TCSEC ("Оранжева книга") або Загальні критерії. В Україні також діють критерії оцінки захищеності комп’ютерних систем від несанкціонованого доступу (НСД). Такі стандарти складають основі ***політики безпеки системи***. Політика безпеки передбачає відповіді на такі питання: яку інформацію захищати, якого роду загрози можуть реалізовуватися в системі, які засоби планується використати для захисту від кожного типу атак.

До сучасних популярних операційних систем (ОС) сьогодні прийнято відносити два сімейства: Windowsta Linux. Обидва сімейства, як це було показано, переважно задовольняють вимоги класу С2 "Оранжевої книги", які коротко полягають в такому:

1. Кожен користувач повинен бути ідентифікований унікальним вхідним ім’ям і паролем для входу у систему. Доступ до комп’ютера надається лише після автентифікації. Повинні бути виконані запобіжні заходи проти спроби застосування фальшивої програми реєстрації.

2. Система повинна бути в змозі використовувати унікальні ідентифікатори користувачів, щоб стежити за їх діями (тобто реалізовувати управління дискреційним доступом). Власник ресурсу

(наприклад, файла) повинен мати можливість контролювати доступ до цього ресурсу.

3. Для управління довірчими відносинами необхідна підтримка наборів ролей (різних типів облікових записів). Окрім того, в системі повинні бути засоби для управління привілейованим доступом.

4. ОС повинна захищати об'єкти від повторного використання. Перед виділенням новому користувачу всі об'єкти, включаючи пам'ять і файли, повинні бути проініційовані.

5. Системний адміністратор повинен мати можливість обліку всіх подій, що належать до безпеки (це значить, що повинен бути налагоджений аудит безпеки).

6. Система повинна захищати себе від зовнішнього впливу або нав'язування, такого, як модифікація завантаженої системи або системних файлів, що зберігаються на диску.

Проаналізуємо тепер, як у рамках архітектури згаданих операційних систем забезпечується виконання вимог політики безпеки.

13.3. Підсистема захисту в ОС Windows

Вивчення структури системи захисту допомагає зрозуміти особливості її функціонування. Незважаючи на слабку документованість ОС Windows за непрямими джерелами, можна судити про особливості її функціонування.

Архітектура системи безпеки Windows XP складається з таких компонентів:

а) ***процес реєстрації користувачів*** (logon-process, winlogon). Його завдання полягає у перехопленні запитів користувача на реєстрацію та отриманні ідентифікаційних даних від користувачів;

б) ***локальна підсистема безпеки*** (Local Security Authority SubSysthem LSASS). Її завдання – слідкувати за процесом автентифікації, доступом користувачів та аудитом в системі;

в) ***менеджер облікових записів*** (Security Account Manager, SAM). Він повинен забезпечувати підтримку автентифікаційної бази даних SAM;

г) ***диспетчер звернень*** (SecurityReferenceMonitor) перехоплює звернення користувачів до об'єктів захисту і передає їх на обробку LSASS. SRM виконується в режимі ядра ОС.

Якщо розглядати детальніше, то система захисту ОС Windows складається з таких компонентів (рис. 13.3):

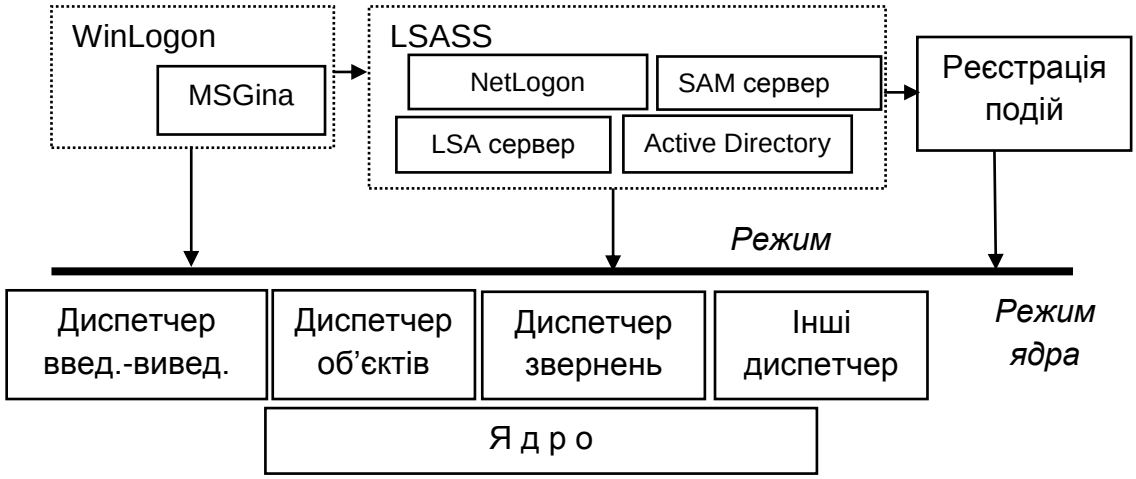


Рис. 13.3. **Система захисту ОС Windows**

Процедура реєстрації (Logon Processes), яка обробляє запити користувачів на вхід у систему. Вона запускає початкову інтерактивну процедуру діалогу із користувачем на екрані, і віддалену процедуру входу, яка дозволяє віддаленим користувачам отримати доступ з робочої станції мережі до серверних процесів Windows. Процес Winlogon реалізований у файлі Winlogon.exe і виконується як процес режиму користувача. Стандартна бібліотека автентифікації Gina реалізована у файлі Msgina.dll.

Локальний адміністратор безпеки (Local Security Authority, LSA), який гарантує, що користувач має дозвіл на доступ у систему. Цей компонент – центральний для системи захисту Windows XP. Він породжує маркери доступу, керує локальною політикою безпеки і надає користувачам автентифікаційні послуги. LSA також контролює політику аудиту і веде журнал, в якому зберігаються повідомлення, що породжуються диспетчером доступу. Основна частина функціональності реалізована в Lsasrv.dll.

Менеджер облікових записів (Security Account Manager, SAM) керує базою даних облікокористувачів. База даних містить інформацію про всіх користувачів і групи користувачів. Ця служба реалізована в Samsrv.dll і виконується в процесі LSASS.

Диспетчер звернень (Security Reference Monitor, SRM), який перевіряє, чи має користувач право на доступ до об'єкта і на виконання

тих дій, які він намагається зробити. Цей компонент забезпечує легалізацію доступу і політику аудиту, що визначаються LSA. Він надає послуги для програм супервізорного режиму і режиму користувача і гарантує, що користувачі і процеси, які здійснюють спроби доступу до об'єкта, мають необхідні права. Також він породжує повідомлення служби аудиту, коли це необхідно. Це компонент ядра системи: Ntoskrnl.exe.

Усі компоненти активно використовують базу даних LSASS, що містить параметри політики безпеки локальної системи, яка зберігається в розділі *HKLM\SECURITY* реєстру.

Як уже зазначалося, реалізація моделі дискреційного контролю доступу пов'язана з наявністю в системі одного з її найважливіших компонентів – монітора безпеки. Це особливий вид суб'єкта, який активується при кожному доступі, може відрізнити легальний доступ від нелегального і не допустити останнього. Монітор безпеки входить до складу диспетчера звернень (SRM), який, згідно з описом, забезпечує також управління ролевим і привілейованим доступом.

Автентифікація користувача. Вхід у систему. Згідно з політикою безпеки, для доступу до комп'ютера користувач повинен пройти процедуру автентифікації. Ця процедура ініціюється комбінацією клавіш "CTRL+ALT+DEL". Ця комбінація клавіш, відома як SAS (secure attention sequence), завжди перехоплюється драйвером клавіатури, який викликає при цьому справжню (а не "троянського коня") програму автентифікації. Процес користувача не може сам перехопити цю комбінацію клавіш або відмінити або скасувати її обробку драйвером. Кажучи мовою стандартів, в системі реалізована функціональність захищеного каналу (trusted path functionality). Ця особливість відповідає вимогам захисту рівня B "Оранжевої книги".

Процедурою автентифікації користувача в системі управляє програма, WinLogon, інтерактивною процедурою, яка відображає початковий діалог із користувачем на екрані. Процес WinLogon активно взаємодіє з бібліотекою GINA (Graphic Identification aNd Authentication – графічною бібліотекою ідентифікації і автентифікації). Бібліотека GINA є змінним компонентом, інтерфейс із нею добре документований, тому іноді в застосуваннях, що реалізовують захист, наявна версія GINA відрізняється від оригінальної. Розробники можуть замінити цю бібліотеку на аналог з іншою функціональністю. Одержавши ім'я і пароль

користувача від GINA, WinLogon викликає модуль LSASS для автентифікації цього користувача. В разі успішного входу в систему, Winlogon отримує з реєстру профіль користувача, визначає тип оболонки і запускає її.

Комбінація SAS може бути одержана системою не тільки на етапі реєстрації користувача. Якщо користувач уже увійшов до системи, то після натиснення клавіш "*CTRL+ALT+DEL*" він отримує такі можливості: подивитися список активних процесів, ініціювати перезавантаження або вимкнення комп'ютера, змінити свій пароль і заблокувати робочу станцію. У свою чергу, якщо робоча станція заблокована, то після введення SAS користувач має можливість її розблокування. Іноді може бути здійснене примусове виведення користувача із системи з подальшим входом у неї адміністратора.

У процесі автентифікації викликається системна функція LogonUser, яка, виходячи з імені користувача, його пароля та імені робочої станції або домену, повертає вказівник на маркер доступу користувача. Маркер згодом передається всім дочірнім процесам. При формуванні маркера використовуються ключі SECURITY і SAM реєстру. Перший ключ визначає загальну політику безпеки, а другий ключ містить інформацію про захист для індивідуальних користувачів.

Маркер доступу. Контекст користувача. Як уже зазначалося, найбільш важливою характеристикою суб'єкта є маркер доступу. Зазвичай маркер створюється при інтерактивному вході користувача в систему і зберігає відомості про контекст користувача. Спочатку маркер зв'язується з процесом-оболонкою Windows Explorer, а потім усі процеси, породжені користувачем під час сеансу роботи, одержують дублікат даного маркера. Важливо розуміти, що самостійно створити маркер програмне забезпечення користувача не може, це може зробити тільки служба LSASS.

Принцип мінімальних привілеїв рекомендує виконання всіх операцій із мінімальними привілеями, необхідними для досягнення результату. Це дозволяє зменшити втрати від спроб навмисного збитку й уникнути випадкових втрат даних. Наприклад, користувачу не рекомендується реєструватися як адміністратор системи без необхідності. У крайньому випадку, можна вдатися до послуг штатної утиліти runas, яка дозволяє запускати застосування від імені іншого облікового запису.

Для безпечної роботи в дусі принципу мінімуму привілеїв ОС Windows підтримує механізми створення маркерів з обмеженими привілеями і запозичення маркера. Перший дозволяє вилучити з маркера певні привілеї, наявність яких при виконанні даної операції не потрібна, а другий дозволяє запускати застосуваннявід імені іншого облікового запису. API системи дозволяє впливати на перелік діючих привілеїв маркера, дублювати маркер, щоб його міг запозичувати інший процес, розв'язувати проблеми запозичення прав і створення обмежених маркерів.

Запозичення прав. В ОС Windows є цікаві можливості виконання коду з маркером, відмінним від маркера даного процесу. Це, наприклад, істотно для серверів, які повинні виступати від імені і з привілеями клієнтів. Одна з таких можливостей – запуск нового процесу зі вказаним маркером за допомогою функції CreateProcessAsUser. Ця функція є аналогом функції CreateProcess, за винятком параметра hToken, який указує на маркер, що визначає контекст користувача нового процесу.

Інший спосіб виконати код із запозиченим маркером – здійснити *запозичення прав (impersonation)*. Запозичення прав означає, що один з потоків процесу функціонує з маркером, відмінним від маркера поточного процесу. Зокрема, клієнтський потік може передати свій маркер доступу серверному потоку, щоб сервер міг отримати доступ до захищених файлів та інших об'єктів від імені клієнта. Згодом потік може повернутися в нормальний стан, тобто використовувати маркер процесу.

Windows не дозволяє серверам виступати в ролі клієнтів без їх відома. Щоб уникнути цього, клієнтський процес може обмежити рівень запозичення прав. Тому для участі в перевтіленні маркер повинен мати відповідний рівень перевтілення. Цей рівень визначається параметром маркера TokenImpersonationLevel і може бути отриманий функцією GetTokenInformation. Для процесу запозичення прав важливо, щоб цей параметр мав значення не нижче за SecurityImpersonation, що означає можливість імперсонації на локальній машині. Значення Security-Delegation дозволяє серверному процесу виступати від імені клієнта як на локальному, так і на віддаленому комп'ютері. Останнє значення має відношення до делегування, яке є природним розвитком запозичення прав, але працює тільки за наявності домена і функціонуванні Active-Directory. За замовчуванням встановлюється рівень Security Imper-sonation.

Маркер запозичення зазвичай створюють шляхом дублювання існуючого маркера і додавання йому потрібних прав доступу та рівня імперсонації. Це можна зробити за допомогою функції Duplicate Token Ex. Власне, запозичення прав здійснюється за допомогою функції ImpersonateLogged On User. Щоб повернутися в початковий стан, потік повинен викликати функцію RevertToSelf.

Важливо також не забувати, що коли перевтілений потік здійснює доступ до об'єкта, то решта потоків процесу, навіть не перевтілених, також має до нього доступ. Подібні ситуації вимагають ретельного аналізу, оскільки тут можуть виникати помилки, що важко відстежуються.

Контроль доступу. Маркер доступу. Ідентифікатор безпеки SID. Структура ідентифікатора безпеки. SID користувача (і групи) є унікальним внутрішнім ідентифікатором і структурою змінної довжини з коротким заголовком, за яким іде довге випадкове число. Це числове значення формується з ряду параметрів, причому стверджується, що ймовірність появи двох однакових SID практично дорівнює нулю. Зокрема, якщо видалити користувача в системі, а потім створити його під тим же ім'ям, то SID створеного користувача буде вже іншим.

Дізнатися свій ідентифікатор безпеки користувач легко може за допомогою утиліт *whoami* або *getsid* з ресурсів Windows. Наприклад, так:

>whoami/user/sid.

Система зберігає ідентифікатори безпеки в бінарній формі, проте існує і текстова форма зображення SID. Текстова форма використовується для виведення поточного значення SID, а також для інтерактивного уведення (наприклад, у реєстр).

У текстовій формі кожен ідентифікатор безпеки має певний формат. Спочатку знаходиться префікс S, за яким слідує група чисел, розділених дефісами. Наприклад, SID адміністратора системи має вигляд: S – 1 – 5 – <домен> – 500, а SID групи *Все* (everyone), в яку входять всі користувачі, включаючи анонімних і гостей, – S –1 – 1 – 0.

Ролевий доступ. Привілеї. Поняття привілею. З метою гнучкого управління системною безпекою в ОС Windows реалізовано керування довірчими відносинами (trusted facility management), яке вимагає підтримку набору ролей (різних типів облікових записів) для різних рівнів роботи в системі. Треба зазначити, що ця особливість системи відповідає вимогам захисту рівня В "Оранжевої книги", тобто

жорсткішим вимогам, ніж перераховані вище. В системі є управління привілейованим доступом, наприклад, функції адміністрування доступні тільки одній групі облікових записів – Administrators (Адміністратори).

Відповідно до своєї ролі кожен користувач має певні привілеї і права на виконання різних операцій відносно системи в цілому, наприклад, правом на зміну системного часу або правом на створення файла. Аналогічні права відносно конкретних об'єктів називаються дозволами. І права, і привілеї призначаються адміністраторами окремим користувачам або групам як частина налаштувань безпеки. Багато системних функцій (наприклад, LogonUser і InitiateSystemShutdown) вимагають, щоб програма, яка їх викликає, мала відповідні привілеї.

Кожен привілей має два текстові зображення: дружнє ім'я, що відображається в інтерфейсі користувача Windows, і програмне ім'я, яке використовується застосуваннями, а також LUID, – внутрішній номер привілею в конкретній системі. Окрім привілеїв, у Windows є близькі до них права облікових записів.

Важливо, що навіть адміністратор системи за замовчуванням володіє далеко не всіма привілеями. Це пов'язано з принципом надання мінімуму повноважень. У кожній новій версії ОС Windows, відповідно до цього принципу, проводиться ревізія переліку привілеїв, що надаються кожній групі, і загальна тенденція полягає в зменшенні їх кількості. З іншого боку, загальна кількість привілеїв у системі зростає, що дозволяє проектувати чимраз більш гнучкі сценарії доступу.

Управління привілеями. Призначення і відкликання привілеїв – прерогатива локального адміністратора безпеки LSA (Local Security Authority). Тому, щоб програмно призначати і відкликати привілеї, необхідно застосовувати функції LSA. Локальна політика безпеки системи означає наявність набору глобальних відомостей про захист, наприклад, про те, які користувачі мають право на доступ у систему, а також про те, якими вони володіють правами. Тому кажуть, що кожна система, в рамках якої діє сукупність користувачів, що володіють певними привілеями відносно даної системи, є об'єктом політики безпеки. Об'єкт політики використовується для контролю бази даних LSA. Кожна система має тільки один об'єкт політики, який створюється адміністратором LSA під час завантаження і захищений від несанкціонованого доступу з боку застосувань.

Управління привілеями користувачів включає задачі переліку, завдання видалення, виключення привілеїв і ряд інших. Отримати перелік привілеїв конкретного користувача можна, наприклад, з маркера доступу процесу, породженого даним користувачем. Там же, в маркері, можна відключити один або кілька привілеїв.

Об'єкти. Дескриптор захисту. У ОС Windows всі типи об'єктів захищені однаковим чином. Із кожним об'єктом пов'язаний *дескриптор захисту (security descriptor)*.

Зв'язок об'єкта з дескриптором відбувається в момент створення об'єкта. Наприклад, один з аргументів функції CreateFile – вказівник на структуру *SECURITY_ATTRIBUTES*, яка містить вказівник на дескриптор захисту.

Дескриптор захисту (див. рис. 13.4) містить SID власника об'єкта, SID груп для даного об'єкта і два вказівники на списки DACL (Discretionary ACL) і SACL (System ACL) контролю доступу. DACL і SACL містять дозвільні й заборонні щодо доступу списки користувачів і груп, а також списки користувачів, чиї спроби доступу до даного об'єкта підлягають аудиту.

Структура кожного ACL списку проста. Це набір записів ACE (Access Control Entry), кожен запис містить SID і перелік прав, наданих суб'єкту із цим SID.

Такий дескриптор захисту привласнюється кожному об'єкту при його створенні або зміні прав доступу. Дескриптор захисту використовується LSASS для порівняння прав суб'єкта з дозволами на доступ до об'єктів і вирішується чи надати доступ до об'єктів.

У списку ACL є записи ACE двох типів – які дозволяють або забороняють доступ. Дозвільний запис містить SID користувача або групи і бітовий масив (access mask – маску доступу), що визначає набір операцій, які процеси, що запускаються цим користувачем, можуть виконувати з даним об'єктом. Заборонний запис діє аналогічно, але в цьому випадку процес не може виконувати перераховані операції. Бітовий масив, або маска доступу, складається з 32 бітів і зазвичай формується програмним способом із певним чином визначених констант, описаних у файлах-заголовках компілятора (переважно у файлі WinNT.h). Формат маски доступу можна подивитися у відповідній літературі.

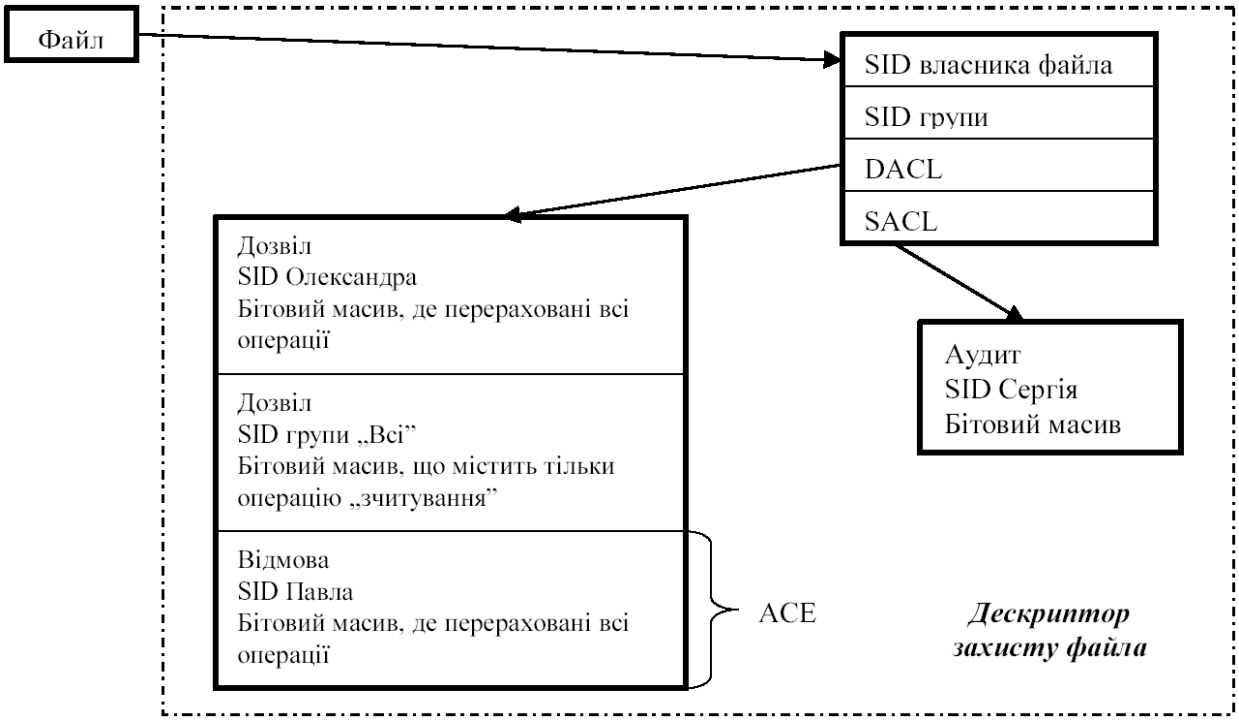


Рис. 13.4. **Структура дескриптора захисту для файла**

На прикладі наведеного на рис. 13.4 власник файла Олександр має право на всі операції з даним файлом, іншим зазвичай дається тільки право на читання, а Павлу заборонені всі операції. Таким чином, список DACL описує всі права доступу до об’єкта. Якщо цього списку немає, то всі користувачі мають всі права; якщо цей список існує, але він порожній, права має тільки його власник.

Окрім списку DACL, дескриптор захисту включає також список SASL, який має таку ж структуру, що й DACL, тобто складається з таких самих ACE-записів, тільки замість операцій, що регламентують доступ до об’єкта, в ньому перераховані операції, що підлягають аудиту. У прикладі на рис. 13.4 операції з файлом процесів, що запускаються Сергієм, описані у відповідному бітовому масиві, реєструватимуться в системному журналі.

Для установки прав доступу до файла, що знаходиться на NTFS розділі диску, потрібно вибрати вкладку "Безопасность" аплету "Свойства", який виникає в Windows Explorer при натисненні на ярлик файла правою кнопкою мишки.

Отже, дескриптор захисту має досить складну структуру і його формування виглядає непростим завданням. На щастя, в Windows є стандартний механізм, що призначає доступ до об’єктів "за замовчуванням", якщо застосування не поклопоталося створити його явно.

В таких випадках кажуть, що об’єкту призначений стандартний захист. Прикладом може служити створення файла за допомогою функції CreateFile, де параметру-покажчику на структуру *SECURITY_ATTRIBUTES* присвоєне значення NIL. Деякі об’єкти використовують тільки стандартний захист (м’ютекси, події, семафори).

Суб’єкти зберігають інформацію про стандартний захист, який буде призначено створеним об’єктам, у своєму маркері доступу. Зрозуміло, в ОС Windows є всі необхідні засоби для налаштування стандартного захисту, зокрема списку DACL "за замовчуванням", в маркері доступу суб’єкта.

Неприпустимість повторного використання об’єктів. Згідно з політикою безпеки, ОС повинна захищати об’єкти від повторного використання. Перед виділенням новому користувачу всі ресурси, включаючи пам’ять і файли, повинні бути проініційовані. Контроль повторного використання об’єкта призначений для запобігання спробам незаконного отримання конфіденційної інформації, залишки якої могли зберегтися в деяких об’єктах, що раніше використалися і звільнених іншим користувачем.

Безпека повторного застосування повинна гарантуватися для областей оперативної пам’яті (зокрема, для буферів зобразами екрана, розшифрованими паролями тощо), для дискових блоків і магнітних носіїв у цілому. Очищення повинне проводитися шляхом запису маскуючої інформації в об’єкт при його звільненні (перерозподілі).

У ОС Windows гарантується безпека повторного використання областей фізичної пам’яті. Якщо процесу користувача знадобилася вільна сторінка пам’яті, вона може бути виділена тільки зі списку проініційованих сторінок. Якщо цей список порожній, сторінка береться зі списку вільних сторінок і заповнюється нулями. Якщо і цей список порожній, диспетчер пам’яті витягує сторінку зі списку простоюючих (standby) сторінок та ініціює її. Незанулена сторінка передається тільки для відображення проектованого файлу. В цьому випадку фрейм незануленої сторінки ініціалізується даними, які буде записано.

Що стосується повторного використання вмісту файлів, добре поміркувавши, стає зрозуміло, що у звичайного зловмисника практично відсутні механізми отримання інформації, що зберігається в дискових блоках знищених файлів. Виділення нових дискових блоків здійснюється

тільки для операцій запису на диск даних із заздалегідь занулених сторінок пам'яті. Зрозуміло, за наявності адміністративних прав і фізичного доступу до комп'ютера можна здійснити злам системи захисту, але в подібних ситуаціях для отримання конфіденційної інформації існують простіші способи.

Захист від зовнішнього нав'язування. Відповідно до політики безпеки, операційна система повинна захищати себе від зовнішнього впливу або нав'язування, такого як модифікація завантаженої системи або системних файлів, що зберігаються на диску. Щоб задовольнити вимоги політики безпеки, в ОС Windows вбудовані засоби захисту файлів (Windows File Protection, WFP), які захищають системні файли навіть від змін із боку користувача з адміністративними правами. В основі захисту лежать засоби фіксації змін у системних файлах. Даний засіб, безумовно, підвищує стабільність системи.

Згідно з документацією, моніторингу і захисту підлягають усі файли, що поставляються у складі ОС, з розширеннями sys, dll, exe і ocx, а також деякі шрифти TrueType (Micros.ttf, Tahoma.ttf і Tahomabd.ttf). Якщо з'ясовується, що файл змінено, він замінюється копією з каталогу %systemroot%\system32\dllcache, на який вказує один із записів у реєстрі. Якщо розмір місця, виділеного для цього каталогу, недостатній, то в ньому зберігаються копії не всіх системних файлів. У тих випадках, коли робиться спроба видалення системного файла і при цьому його не виявляється в каталозі dllcache, система намагається відновити його з компакт-диску ОС Windows або з мережевих ресурсів.

Окрім того, адміністратор системи за допомогою утиліти Sfc.exe (system file checker) може здійснити перевірку коректності версії всіх системних файлів. Відомості про файли з некоректним номером версії заносяться в протокол. Коректність системних файлів перевіряється за допомогою механізму електронного підпису і до непідписаних файлів слід ставитися з обережністю. Для перевірки підпису і виявлення непідписаних файлів служить штатна утиліта SigVerif.exe.

Виявлення вторгнень. Аудит системи захисту. Навіть найкраща система захисту рано чи пізно буде зламана, тому виявлення спроб вторгнення – найважливіше завдання системи захисту. Основним інструментом виявлення вторгнень є запис даних аудиту. Окремі дії

користувачів протоколюються, а одержаний протокол використовується для виявлення вторгнень.

Аудит, таким чином, полягає в реєстрації спеціальних даних про різні типи подій, що відбуваютьсяв системі і так чи інакше впливають на стан безпеки комп'ютерної системи. До таких подій зазвичай зараховують такі:

вхід або вихід із системи;

операції з файлами (відкрити, закрити, перейменувати, видалити);

звернення до віддаленої системи;

зміна привілеїв або інших атрибутів безпеки (режиму доступу, рівня доступу користувача тощо).

Подія аудиту в ОС Windows може згенеруватися застосуванням користувача, диспетчером об'єктів або іншим кодом режиму ядра. Щоб ініціювати фіксацію подій, пов'язаних із доступом до об'єкта, необхідно сформувати в дескрипторі безпеки цього об'єкта список SACL, в якому перераховані користувачі, чиї спроби доступу до даного об'єкта підлягають аудиту. Це можна зробити програмно або за допомогою інструментальних засобів ОС.

Огляд підсистеми захисту WindowsXP закінчимо короткою характеристикою відмінностей нової ОС Microsoft Windows 7.

Що нового у Windows 7. По-перше, вона зберегла усі найкращі риси системи захисту XP та Vista. По-друге, система захисту значно розширила свої можливості.

Коротко охарактеризуємо нові можливості системи захисту нової ОС від Microsoft.

1. ***Вдосконалено центр забезпечення безпеки ОС***. Microsoft значно розширила його можливості та надала нову назву: центр підтримки. На відміну від попереднього, центр підтримки інформує користувача не тільки про проблеми безпеки, а й про усі інші проблеми системи, які вважаються критичними або важливими. У випадку виявлення проблем, центр не тільки повідомить користувача про них, а й виконає пошук необхідного програмного забезпечення в Інтернеті та надасть посилання користувачу для прийняття рішення. Це трохи нагадує рекомендації Microsoft Base line Security Analyzer з розширеними можливостями (рис. 13.5).

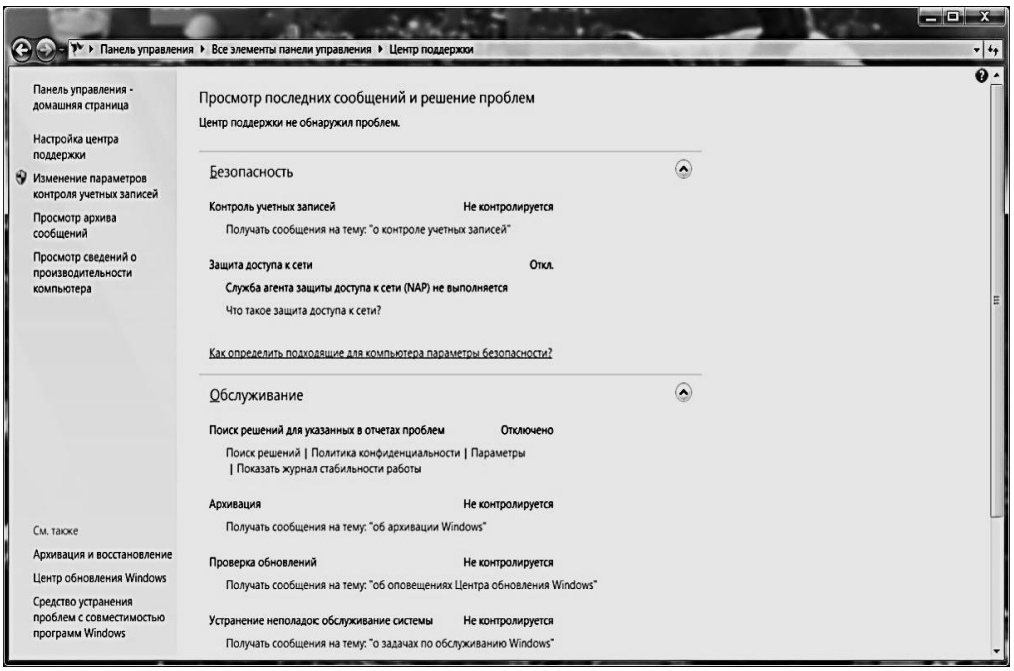


Рис. 13.5. **Зовнішній вигляд центру підтримки**

2. ***Контроль облікових записів користувачів***. Набув подальшого вдосконалення так званий UserAccountControl (Контроль облікових записів користувачів), який раніше викликав багато нарікань з боку адміністраторів. Тепер можна гнучко налаштувати повідомлення цієї служби, звівши до потрібної величини запити на дозвіл тієї чи іншої операції, яка вимагає адміністративних повноважень. Передбачено навіть відключення служби контролю, однак робити це настійливо не рекомендується (рис. 13.6).

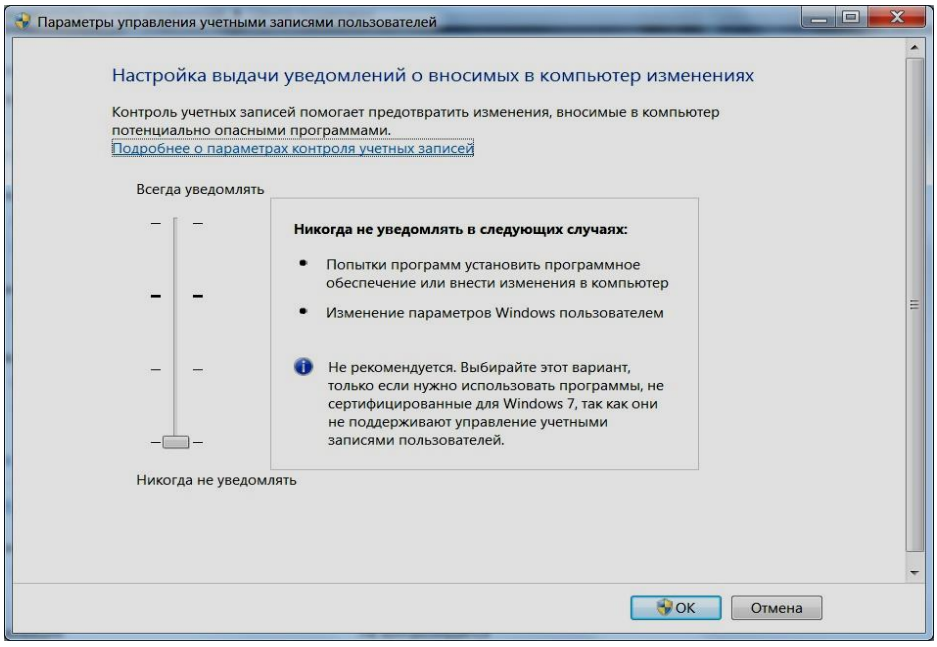


Рис. 13.6. **Форма контролю облікових записів користувачів**

3. ***Шифрування за допомогою BitLocker***. У WindowsVista дебютував механізм суцільного шифрування жорстких дисків BitLocker. З деякими вдосконаленнями він мігрував і на Windows 7 (Enterprise та Ultimate – версії). До вдосконалень відноситься таке: а) тепер підготовка дисків для шифрування відбувається при встановленні системи, причому вона сама резервує для потреб шифрування певну ділянку активного диску (рис. 13.7):

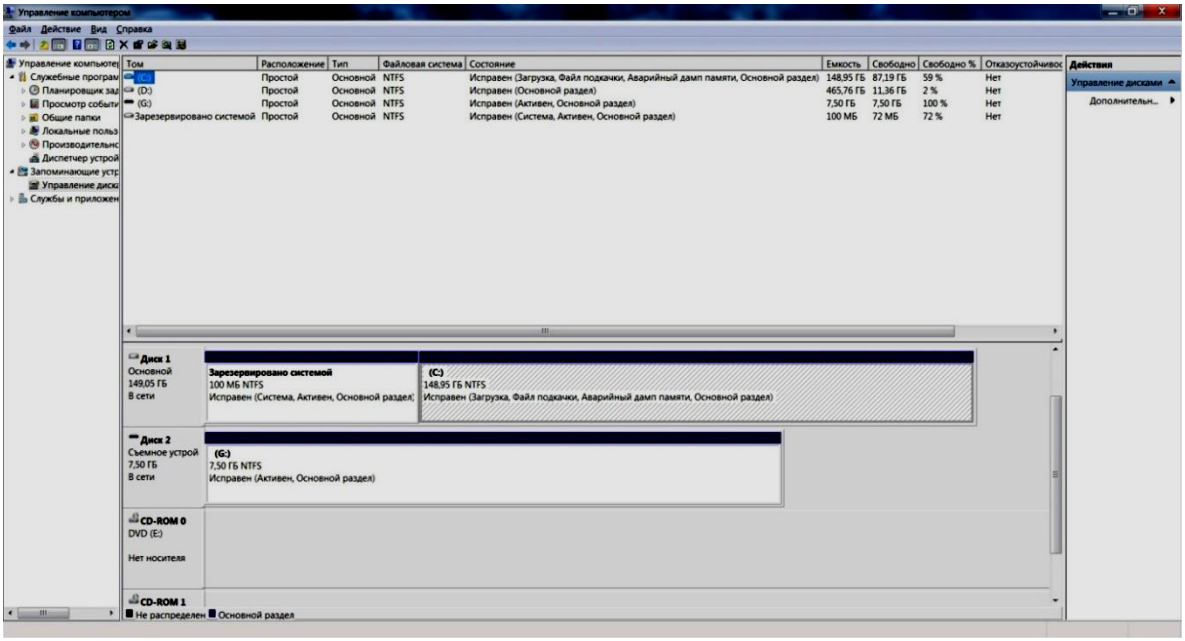


Рис. 13.7. **Керування дисками у Windows 7**

б) з’явилася можливість шифрування не тільки системного, а й усіх інших дисків системи, відформатованих під NTFS.

На відміну від шифрованої файлової системи з попередніх версій Windows (EncryptedFileSystem, EFS), яка дозволяла шифрувати окремі файли, BitLocker виконує суцільний криптографічний захист усієї інформації на диску, що значно підвищує степінь її захисту при фізичному доступі до комп’ютера або диску.

4. ***Шифрування інформації на знімних носіях***. Windows 7 за допомогою нового механізму BitLockerToGo дозволяє шифрувати інформацію на знімних носіях не тільки з файловою системою NTFS, але й FAT32 та FAT. При цьому захищені носії можуть читатися і в попередніх версіях Windows (XP та Vista), однак режим запису в цьому випадку недоступний. Однак існує можливість (у груповій політиці безпеки) створити таке шифрування знімних дисків, яке буде підтримуватися і в попередніх версіях Windows (XP, Vista).

5. ***Заборона використання USB-дисків***. Використовуючи групову політику безпеки (gpedit.msc з командного рядка), адміністратор може виконати такі корисні з точки зору безпеки дії: а) заборонити використання усіх типів знімних носіїв на цьому комп'ютері (знімних дисків, CD-ROM); б) заборонити використання усіх типів знімних носіїв за винятком строго визначених (досягається за допомогою ідентифікатора пристрою – його можна подивитися у диспетчері пристроїв);

в) створення та підтримка "чорного списку" знімних пристроїв; г) створення під час шифрування BitLocker унікальних ідентифікаторів для знімних дисків організації з подальшим відбором їх за цією ознакою і забороною використання інших дисків (рис. 13.8).

6. ***Технологія AppLocker для контролю ПЗ, що використовується на комп'ютері.*** У Windows 7 запропоновано нову технологію створення замкненого програмного середовища. У WindowsXP є політика обмеженого використання програм (групова політика безпеки), яка дозволяє контролювати використання ПЗ користувача, однак вона досить незручна у використанні. Microsoft вдосконалила цю політику і під новою назвою впровадила механізм у Windows 7. AppLocker простіше у використанні, він має майстра генерування правил, а його нові можливості зменшують витрати часу на адміністрування, дозволяють вести аудит програм, що запускаються користувачами, гнучко маніпулювати правилами доступу до певних застосувань та файлів, використовуючи різні правила аж до цифрових підписів продуктів.

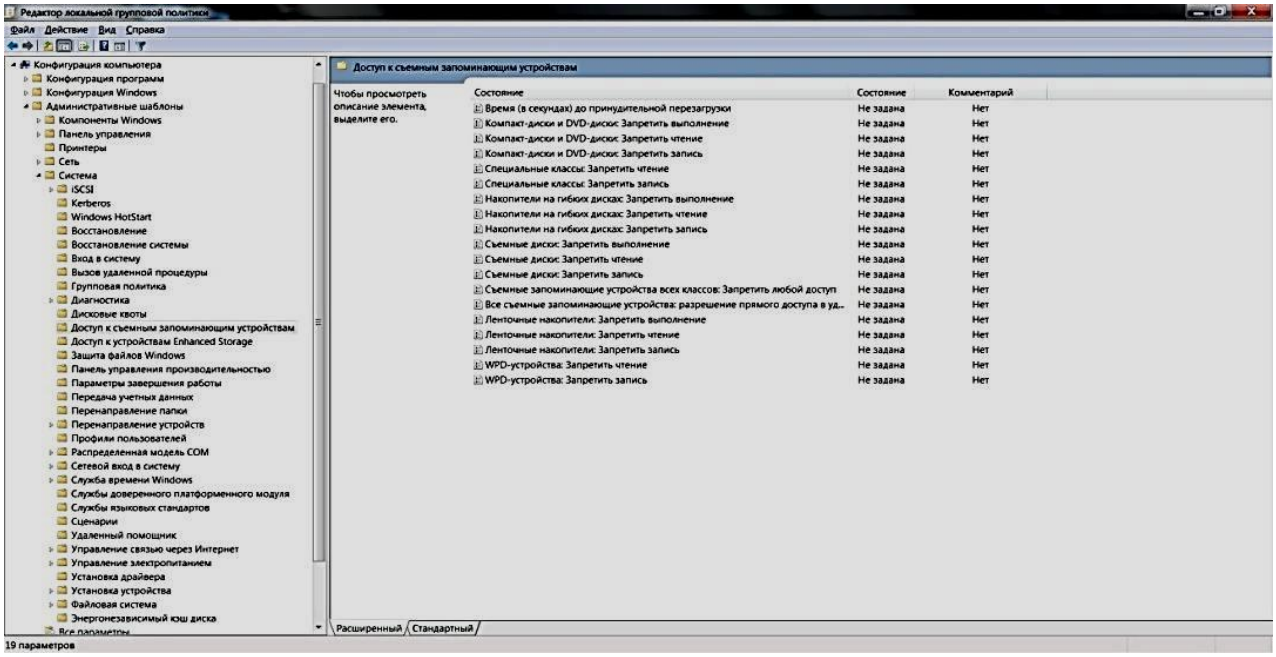


Рис. 13.8. **Редагування доступу до знімних дисків у груповій політиці безпеки**

AppLocker налаштовується за допомогою групової політики безпеки або на локальному комп'ютері як оснащення локальної політики безпеки (рис. 13.9).

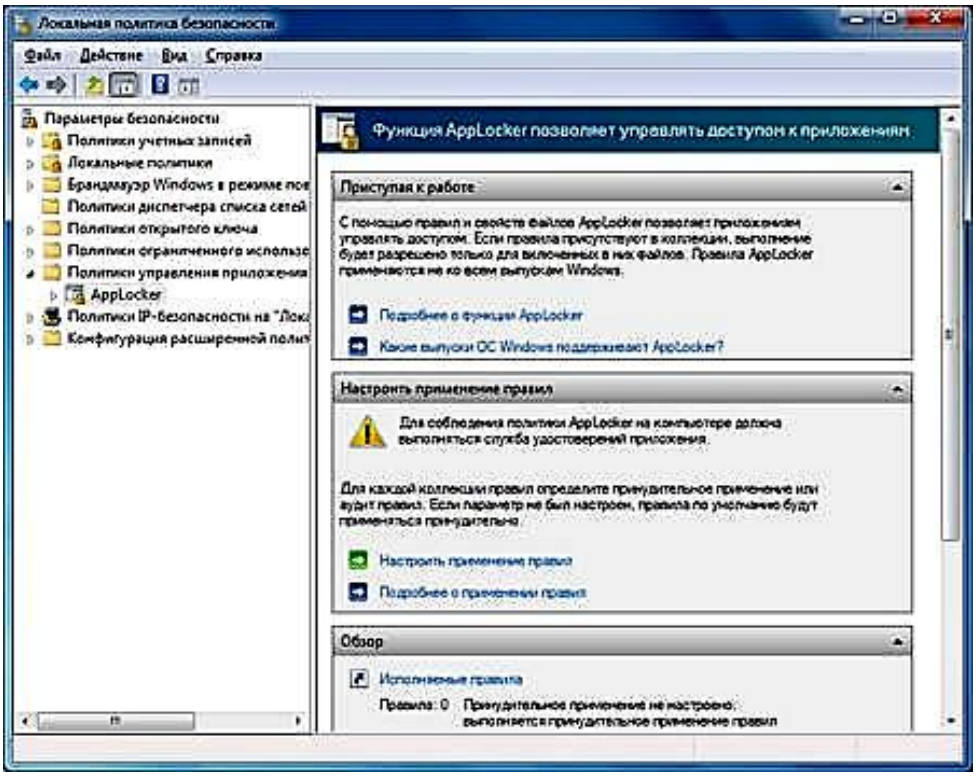


Рис. 13.9. **Налаштування правил AppLocker у локальній політиці безпеки**

7. ***Блокування мережевих загроз***. Від мережевих загроз Windows 7 захищає брандмауер. Він існував і у Windows XP, але там він контролював лише вхідний трафік. У Windows 7 брандмауер контролює вже і висхідний трафік. Крім цього, знято обмеження на один активний профіль, як це було у WindowsVista, отже можна використовувати одночасно кілька активних профілів, по одному на кожний адаптер. Переваги очевидні: можна, наприклад, сидячи в громадському місці, де є безпровідна точка доступу, підключатися одночасно через VPN до корпоративної мережі, і при цьому бути впевненим, що брандмауер застосує загальний профіль до Wi-Fi адаптера, а профіль домену активує для VPN-тунелю.

8. ***Захищений доступ до ресурсів корпоративної мережі***. Microsoft пропонує в новій ОС ще одну нову технологію – DirectAccess, яка забезпечує захищене з'єднання з корпоративною мережою для віддалених користувачів. Відмінність цієї технології від VPN полягає

в тому, що захищений тунель встановлюється у фоновому режимі без втручання користувача. Однак, це можливо лише тоді, коли на комп'ютері користувача встановлено Enterprise (Ultimate) версію ОС, а на серверах компанії – WindowsServer 2008 R2.

9. ***Захисник Windows***. Для захисту ОС від шпигунського програмного забезпечення Microsoft пропонує свою систему WindowsDefender (захисник Windows), яка раніше випускалася окремо. Він працює за схемою, схожою до антивірусних програм, підтримує зв'язок з мережевою спільнотою MicrosoftSpyNet, автоматично поновлює свої бази даних шпигунського програмного забезпечення з Інтернету та запитує у спільноти правильні способи реагування на ту чи іншу загрозу.

10. ***Вбудований антивірус***. Microsoft пропонує у складі своєї нової операційної системи свій безкоштовний антивірус – MicrosoftSecurity-Essentials, який досить непогано зарекомендував себе у багатьох тестах антивірусів. Зрозуміло, що це дуже зручно, особливо для індивідуальних користувачів.

З деталями архітектури та властивостями операційних систем сімейства Windows можна ознайомитися у роботі [10].

Не дивлячись на усі переваги ОС Windows, ця операційна система, як відомо більшості користувачів, досить вразлива для вірусів та троянських програм, на відміну від іншої популярної ОС, Linux. Розглянемо відмінності архітектури цих операційних систем, які призводять до такої значної відмінності.

13.4. Порівняння архітектури Windows та Linux

Віруси, троянські коні та інші деструктивні програми вражають настільні комп'ютери з Windows внаслідок цілої низки причин, властивих Windows та не властивих Linux:

1. Windows лише недавно еволюціонувала від однокористувацької моделі до багатокористувацької.
2. Windows за своєю архітектурою є монолітною, а не модульною системою.
3. У Windows надто широко використовується RPC-механізм.
4. Windows фокусується на відомому графічному інтерфейсі для настільних комп'ютерів.

Розглянемо вказані причини детальніше.

Windows лише недавно еволюціонувала від однокористувацької моделі до багатокористувацької. Система Windows з самого початку була розроблена, щоб забезпечити користувачам та програмам вільний доступ до усієї системи, а це означає, що хто завгодно міг скомпрометувати критичну системну програму або файл. Це також значить, що віруси та інші деструктивні програми могли зробити те ж саме, тому що Windows не ізолювала користувачів та прикладне програмне забезпечення від критичних ділянок операційної системи.

Операційна система Windows XP стала першою версією Windows, де з'явилися суттєві результати спроб ізолювання користувачів від системи, так що кожен користувач має свої власні особисті файли та обмежені системні повноваження. Платою за це стало, що програмні продукти, розроблені для попередніх версій перестали працювати. Саме тому в Windows XP передбачено режим сумісності, тобто режим, який дозволяє програмам працювати так, ніби вони функціонують в однокористувацькому середовищі. Windows XP – це прогрес, однак і Windows XP не можна назвати дійсно багатокористувацькою системою.

Windows Server 2003 – це наступний крок до істинно багатокористувацької системи, але навіть у Windows Server 2003 не вдалося ліквідувати усі "дірки" в системі захисту. Саме тому в Windows Server 2003 довелося відключити використання "за замовчуванням" деяких функцій web-навігатора (наприклад, ActiveX, написання сценаріїв тощо). Якщо б Microsoft переписала ці функції для роботи у безпечному режимі в істинно багатокористувацькому середовищі, вони не створювали би таких серйозних загроз, перед якими беззахисна Windows.

Windows за своєю архітектурою є монолітною, а не модульною системою. Монолітна система – це система, де більшість функцій інтегровано в єдиний модуль. Протилежністю такої системи є така, де функції розподілено за кількома рівнями, причому кожен рівень має обмежений доступ до інших.

Взаємозалежності такого типу мають два поганих каскадних побічних ефекти. По-перше, в монолітній системі кожна "дірка" в одній частині системи впливає на усі сервіси та застосування, які залежать від цієї частини системи. Інтегрувавши Інтернет Explorer в операційну систему, Microsoft створила систему, де будь-яка "дірка" в Інтернет Explorer створює загрози для комп'ютера з Windows в цілому.

Така архітектурна модель впливає набагато глибше, ніж здається. Так, у монолітній системі вразливості в захисті виявляються критичнішими, ніж можна було чекати.

Проілюструвати це зможе проста аналогія. Уявимо собі ідеальну ОС, яка складається з трьох сфер: одна в центрі; друга, більшого діаметру, охоплює першу; а третя сфера охоплює дві перших. Користувач бачить лише третю сферу. Це рівень, де він запускає застосування, наприклад, текстові процесори. Вони використовують необхідні функції, що надаються другою сферою, наприклад, засоби візуалізації графічних зображень або форматування текстів. Ця друга сфера (спеціалісти називають її "користувацькими процесами") не має прямого доступу до критичних частин системи. Щоб виконати свою роботу, вона має спитати дозволу внутрішньої сфери. Внутрішня сфера виконує найважливіші функції, тому, що має безпосередній доступ до усіх критичних частин системи. Вона керує пам’яттю, дисками та рештою важливих частин системи. Ця сфера називається "ядром" і є серцем ОС.

У такій архітектурі "дірка" в програмі графічного відображення не може нанести глобальних збитків комп’ютеру, оскільки функції візуалізації не мають прямого доступу до найкритичніших частин системи. Навіть якщо користувач завантажить у текстовий процесор зображення з втіленим вірусом, цей вірус не зможе пошкодити нічого окрім власних файлів користувача, оскільки функція графічного відображення не має доступу до жодної критичної частини системи.

Проблема Windows полягає в тому, що в ній не дотримуються розумних конструкторських принципів розділення функцій по відповідних описаних рівнях. Windows вкладає занадто багато функцій у ядро, центральну сферу, де можна заподіяти найбільшого збитку. Наприклад, якщо інтегрувати графічні функції в ядро, ці функції зможуть нашкодити усій системі. Таким чином, як тільки виявиться "дірка" в алгоритмі графічного відображення, надмірно інтегрована архітектура Windows полегшить використання цієї "дірки" для отримання повного контролю над системою, для руйнування всієї системи.

Монолітна система нестабільна за самою своєю природою. Коли в системі так багато взаємозв’язків, зміна однієї з її частин породжує багато загроз. Одна зміна в системі може вплинути (і таки впливає) на усі сервіси та застосування, які залежать від цієї частини системи. Саме

тому оновлення, які виправляють одну частину Windows, часто порушують роботу інших сервісів та застосувань. Для прикладу можна навести такий факт: для пакета оновлень Windows XP Service Pack 2 було створено список випадків (він, до речі, постійно оновлюється), коли його встановлення призвело до виходу з ладу програм сторонніх виробників. Таке явище в монолітній системі звичайна справа.

У Windows занадто широко використовується RPC-механізм. Абревіатура RPC означає "віддалений виклик процедур" (Remote Procedure Call). RPC – це те, що відбувається, коли одна програма відправляє через мережу вказівку іншій програмі виконати якусь дію. Віддаленим викликом процедури цей механізм називається тому, що не має значення, функціонують ці програми на тому же комп'ютері, чи десь в Інтернеті.

RPC-механізми – це потенціальна загроза безпеці, оскільки їх призначення – дозволити комп'ютерам, які знаходяться десь у мережі, віддавати даному комп'ютеру вказівки виконати ті, чи інші дії. Як тільки виявляється вразливість в програмі, що використовує RPC-механізм, будь-хто з мережі може використати її, щоб змусити вражений комп'ютер виконати якісь дії. На жаль, користувачі Windows не можуть заблокувати RPC-механізм, оскільки Windows використовує його, навіть якщо комп'ютер не підключений до мережі. Дивно, але деякі з найбільш серйозних вразливостей у Windows Server 2003 – наслідок "дірок" у самих RPC-функціях Windows, а не в програмних продуктах, що їх використовують.

Важливо зазначити, що RPC-механізми не завжди необхідні, і не зрозуміло, чому Microsoft так широко їх використовує.

Особливості архітектури Linux. За даними статистичних опитувань Evans Data Linux Developers Survey, 92 % респондентів ніколи не зіштовхувалися з випадками зараження ОС Linux вірусами, троянськими та іншими деструктивними програмами.

Той факт, що віруси, троянські та інші деструктивні програми дуже рідко можуть (якщо взагалі можуть) заразити Linux-системи, частково можна пояснити такими причинами:

1. Linux має довгу історію використання ретельно проробленої багатокористувацької архітектури.

2. За своєю архітектурою Linux є, в основному, модульною системою.

3. Функціонування Linux не залежить від RPC-механізму, а сервіси зазвичай за замовчуванням налаштовані не використовувати RPC-механізми.

4. Сервери Linux ідеально підходять для віддаленого адміністрування.

Розглянемо ці причини детальніше.

Linux має довгу історію використання ретельно проробленої багатокористувацької архітектури. Linux ніколи не був однокористувацькою системою. Тому в ній з самого початку закладено принцип ізолювання користувачів від застосувань, файлів та каталогів, що впливають на ОС в цілому. Кожному користувачу надається користувацький каталог, де зберігаються усі його файли даних, конфігураційні файли, що належать цьому користувачу. Коли користувач запускає якесь застосування (наприклад, текстовий процесор), воно запускається з обмеженими повноваженнями. Це застосування має право на запис лише у власний каталог цього користувача. Воно не може нічого записати у системні файли, навіть у каталог іншого користувача, якщо тільки адміністратор явним чином не надасть цьому користувачеві таке право.

Ще важливіше, що Linux надає практично усі функціональні можливості (наприклад, візуалізацію зображень JPEG) у вигляді модульних бібліотек. Тому, коли текстовий процесор відображає JPEG-зображення, відповідні функції запускаються з тими ж обмеженими повноваженнями, що й сам текстовий процесор. Якщо в програмах візуалізації JPEG-зображень є "дірка", зловмисник зможе використати її тільки для отримання таких самих повноважень, як у цього користувача, що значно обмежує масштаби можливих збитків. У цьому переваги модульних систем, вони ближчі до ідеалу описаному ОС.

Навіть сервіси, наприклад, web-сервери, звичайно запускаються як користувачі з обмеженими повноваженнями. Так, Debian GNU/Linux запускає web-сервер Apache як користувача "www-data", що належить до такої самої групи. Якщо зловмисник на комп’ютері з Debian отримає повний контроль над web-сервером Apache, він зможе впливати лише на файли, які належать користувачу "www-data", тобто на web-сторінки. В свою чергу, MySQL, запускається з повноваженнями користувача

"mysql". Навіть, якщо Apache та MySQL разом обслуговують web-сторінки, зловмисник, отримавши контроль над Apache, не буде мати повноважень, які дозволяють використати цю вразливість для отримання контролю над сервером баз даних, тому що він "належить" іншому користувачу. Крім того, такі облікові записи конфігуруються без можливості доступу до командного рядка, а отже, не зможуть подати довільну команду серверу Linux.

За своєю архітектурою Linux є модульною, а не монолітною системою. Linux – це ОС, сконструйована, в основному, за модульним принципом, від ядра до застосувань. У Linux практично немає нероздільних зв'язків між компонентами. Немає й єдиного web-навігатора, який використовується системою або програмами електронної пошти. Отже, "дірка" у web-навігаторі не обов'язково небезпечна для інших застосувань.

Ядро Linux підтримує модульні драйвери, але значною мірою є монолітним ядром, бо сервіси в цьому ядрі взаємозалежні. Усі негативні наслідки монолітності мінімізуються тим, що ядро Linux, наскільки це можливо, розроблено як найменша частина системи. Розробники Linux фанатично притримуються такого принципу: "Якщо задача може бути вирішена за межами ядра, вона має бути виконана поза ним". Це значить, що в Linux майже кожна корисна функція ("корисна" – значить "для кінцевого користувача") не має доступу до вразливих частин Linux.

Сервери Linux ідеально підходять для віддаленого адміністрування. Сервер Linux можна, а часто і треба, інсталювати без монітора та адмініструвати віддалено, оскільки при такому стилі адміністрування він не піддається таким загрозам як при локальному адмініструванні.

Можливо, це одна з найголовніших відмінностей Linux від Windows, тому, що цей фактор зводить нанівець багато критичних вразливостей, загальних для Linux та Windows, наприклад, вразливості web-навігаторів Mozilla та Інтернет Explorer.

Розглянемо детальніше методи забезпечення безпеки у Linux.

***Забезпечення безпеки в Linux*.** Якщо коротко, то структурна схема системи безпеки Linux має такий вигляд у взаємодії з іншими підсистемами (SELinux, рис. 13.10).

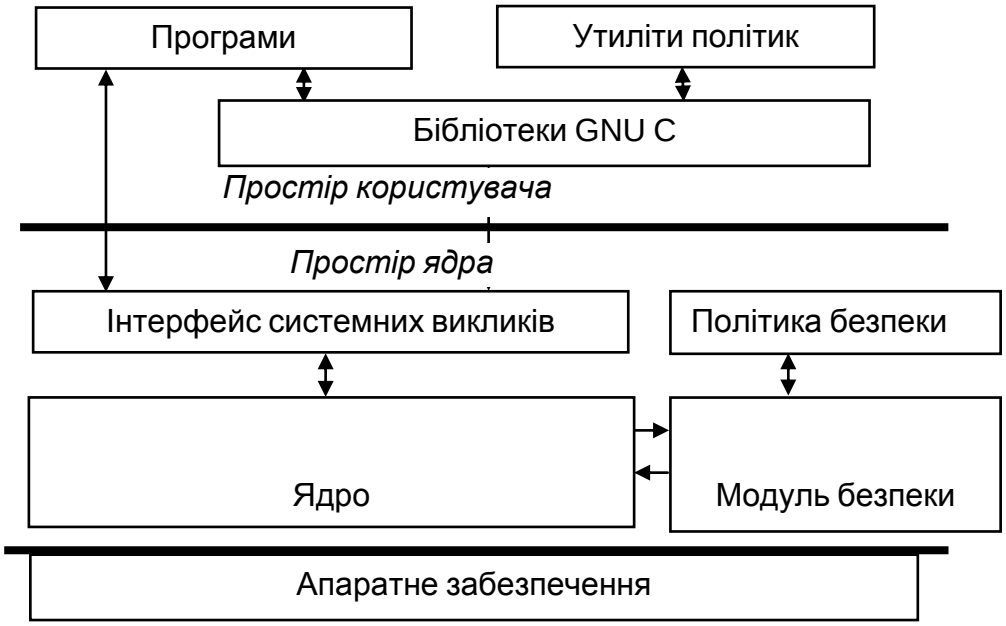


Рис. 13.10. **Структурна схема системи безпеки Linux**

Як бачимо, система, завдяки модульній структурі, дозволяє змінювати модулі безпеки та утиліти політик безпеки, що притаманне усім версіям Linux. Вони зосереджені, більшою частиною, окремо від мікроядра, що позитивно впливає на безпеку системи в цілому, а також на можливість вдосконалення цієї системи без втручання в роботу ядра.

ОС SELinux є на сьогодні однією з найбільш захищених середовищ, але, безумовно, не єдиною.

Бібліотеки AppArmor. AppArmor є альтернативою SELinux, причому в тут також використовується інфраструктура модуля LSM (LinuxSecurityModule). Причиною створення AppArmor стало те, що SELinux виявився надто складним для пересічного користувача. AppArmor має повністю налаштовувані модулі безпеки та режим навчання, який допомагає налаштувати систему безпеки. Ще однією перевагою AppArmor є те, що вона не залежить від типу файлової системи, тоді як SELinux вимагає підтримки ФС додаткових атрибутів.

Solaris 10 (TrustedSolaris). ОС Solaris 10 забезпечує мандатне керування доступом, що значно покращує систему захисту. Суттєвою перевагою є те, що завдяки цьому керуванню обмежуються повноваження користувача root, наявність якого було завжди основним недоліком Linux.

Основні принципи організації доступу в Linux

Розглянемо основні принципи організації доступу в Linux. Як відомо, основні версії Linux підтримують дискреційну модель доступу, яка реалізується файловою системою extX (X = 2, 3, 4). Основні параметри доступу Linux зводяться до наступного.

Кожен користувач має унікальний ідентифікатор користувача, UID, а група, до якої він належить – ідентифікатор групи, GID. Для автентифікації користувачів використовуються дві утиліти: getty, яка приймає login користувача, і утиліта login, яка приймає пароль і виконує автентифікацію. Паролі в Linux також зберігаються у вигляді геш-образів. У перших версіях гешування виконувалося алгоритмом DES, зараз – MD5. Облікові записи зберігаються в папці /etc/passwd/.

Структура записів бази даних така:

Ім’я користувача: геш_пароля:sid:gid:дод._інф.:home_dir:shell

Приклади:

ivanov:1QRxtta36BD:340:120:Іванов І.І.:/home/Ivanov:/bin/bash

root::1iDYwrOmhmEBU: 0:0: root:: /root: /bin/bash

або:

ivanov:x:340:120:Іванов І.І.:/home/Ivanov:/bin/bash

root:x: 0:0: root: /root: /bin/bash

Символи "*x*" на місці пароля означають, що у системі застосований більш сучасний метод зберігання паролів: вони зберігаються у файлі тіньових паролів (/etc/shadow).

Власником файла /etc/shadow є користувач root і тільки він має право читати інформацію з нього.

Формат записів у цьому файлі має такий вигляд:

root:1iDYwrOmhmEBU:10792:0:: 7:7::

bin:*:10547:0::7:7::

daemon:*:10547:0::7:7::

adm:*:10547:0::7:7::

lp:*:10547:0::7:7::

sync:*:10547:0::7:7::

shutdown:U:10811:0:-1:7:7:-1:134531940

Призначення першого поля файла shadow таке ж, як і у першого поля файла passwd. Друге поле містить геш-образ пароля. Реалізація тіньових паролів дозволяє збільшити довжину паролів від 13 до 24

символів. Символи, які використовуються у паролях, беруться з набору з 52 літер алфавіту, цифр та (/). Разом виходить 64 символи.

З третього поля починається інформація про час життя паролю. Це – кількість днів з 1 січня 1970 року до дня зміни цього пароля.

Четверте поле вказує на кількість днів, яка повинна пройти, перш ніж можна буде змінювати пароль. Поки з дня останньої зміни пароля не пройде стільки днів, скільки вказано у цьому полі, знову змінювати пароль не можна.

П’яте поле задає максимальну кількість днів, протягом яких можна використовувати пароль, після чого він має бути зміненим. Якщо тут стоїть додатна величина, то при спробі користувача зайти до системи після цього терміну призведе до того, що команду password буде запущено у режимі обов’язкової зміни пароля.

Значення з шостого поля визначає, за скільки днів до закінчення терміну дії пароля слід попереджати користувача про це.

Сьоме поле задає число днів, починаючи з дня обов’язкової зміни пароля, коли цей обліковий запис блокується. Іншими словами, якщо після цієї кількості днів користувач не зайде до системи і не змінить свій пароль, то його обліковий запис буде заблоковано.

У передостанньому полі вказано дату блокування облікового запису. Останнє поле зарезервовано і поки що не використовується.

У файлі etc/groups описано групи користувачів. Структура цього файла подібна до файла паролів. Далі наведено приклади записів:

```
root::0:
wheel::10:
bin::1:bin,daemon
daemon::2:bin,daemon
sys::3:bin,adm
adm::4:adm,daemon
```

Перше поле – ідентифікатор групи. Він повинен бути унікальним.

Друге поле – пароль. Як правило, паролі для груп не використовуються, отже, це поле практично завжди порожнє. Однак можна увести паролі і для групи.

Третє поле – ідентифікатор групи, gid. Він також має бути унікальним, хоча це і необов’язково.

Четверте поле – список користувачів, що входять до цієї групи. Імена користувачів пишуться через кому без пробілів.

Кожен користувач має створену за замовчуванням групу, яка створюється при реєструванні користувача, так звану групу входу. Ім’я цієї групи співпадає з іменем користувача. Якщо користувач або адміністратор не вкаже інакше, система призначає цю групу групою-власником усіх його файлів. Це дозволяє автоматично обмежити доступ інших користувачів до його інформації, оскільки вони належать до інших груп користувачів.

Файли та права доступу. Linux, в принципі, підтримує багато файлових систем. Однак основними є ФС типу extX (ext 2, 3, 4). Дві останні – журнальні ФС, які дозволяють відновлювати втрачену інформацію, до певної межі, звісно.

В останніх версіях Linux використовується ext 4. Особливості цих ФС полягають в тому, що файли, каталоги та пристрої вважаються файлами. Кожному пристрою відповідає свій файл. Файли та пристрої для використання треба задіяти операцією монтування. Точка монтування визначається при поданні команди. Вона може бути довільною, а керує цим ядро системи. Коли користувач хоче отримати доступ до пристрою, ядро визначає, чи має він відповідні права. При цьому виконується аналіз ідентифікатора користувача, ідентифікатори усіх груп, до яких він належить. На основі цього аналізу і виноситься рішення про надання доступу.

Linux підтримує багато різних типів файлів.

Лівий стовпчик вказує на тип файла: d – directory (каталог); c – символьний пристрій (аналог COM-порта в Windows); b – блоковий пристрій (жорсткий диск 0 – перший жорсткий диск системи); s – сокет; p – іменований канал; l – символьне посилання; "–" – простий файл користувача.

Наступні літери (до числа) – дозволи для цього файла.

Число – кількість жорстких посилань на файл.

Наступні два стовпчика – ім’я власника файла та назва його групи.

Далі – розмір файла; дата та час його створення.

В останньому стовпчику – повна назва файла.

Якщо подати команду ls – i, то отримаємо щось таке:

```
20512 -rwxr-xr-x  3 root   root     49280 Jul 27 19:37 gunzip
20512 -rwxr-xr-x  3 root   root     49280 Jul 27 19:37 gzip
20512 -rwxr-xr-x  3 root   root     49280 Jul 27 19:37 zcat
```

Зліва з’явилося число, яке вказує номер індексного дескриптора файла, в якому записано всю важливу інформацію про файл, в тому числі (в останніх версіях) підтримуються списки контролю доступу, чого раніше в Linux не було.

Списки контролю доступу розширюють можливість тріад доступу. Їх використовують, коли деяким користувачам треба надати доступ до файлу без зміни групових налаштувань.

Раніше в Linux підтримувався контроль доступу на рівні тріад.

Як можна спостерігати, після індексного дескриптора йде мітка звичайного файла, потім тріади дозволів: 1-а тріада – дозволи для власника файла; друга – для його групи; третя – для усіх решти. Значення "r" – дозвіл на читання; "w" – на запис; "x" – на виконання; "–" – дозволу на цю операцію немає.

Дозволи для каталогів трактуються інакше, ніж для файлів: дозвіл на читання – дозволяється продивлятися вміст каталогу, тобто отримати список файлів, що містяться у цьому каталозі, однак самі файли можуть бути недоступними для читання; дозвіл на запис для каталогу значить, що туди можна записувати файли або створювати нові, однак зміна існуючих файлів може виявитися недоступною; наявність права на виконання дозволяє зайти в цей каталог за допомогою команди cd. Можна читати список файлів з каталогу або записувати туди нові файли, але зайти в нього без дозволу на виконання не можна.

Детальніше з властивостями операційної системи Linux можна ознайомитися, наприклад, у [62].

Контрольні запитання

1. Формальні моделі доступу до інформації. Дискреційний та мандатний доступ до даних в інформаційних системах.

2. Основи захищеності сучасних операційних систем.

3. Підсистема захисту в ОС Windows. Основні послуги та механізми захисту.

4. Підсистема захисту в ОС Linux. Основні переваги і недоліки.

5. Порівняння архітектури Windows та Linux. Основні можливості підсистем захисту ОС.

Використана література

1. Информационная безопасность RUNNet / А. В. Аграновский, А. К. Скуратов // Тр. XI Всеросс. научн.-методич. конф. Телематика'04. – Т. 1. – СПб., 2004. − С. 66–68.

2. Баричев С. Г. Основы современной криптографии / С. Г. Баричев, Р. Е. Серов. – М. : Горячая линия-Телеком, 2002. – 152 с.

3. Введение в криптографию / под редакцией В. В. Ященко. – СПб. : Питер, 2001. – 288 с.

4. Галицкий А. В. Защита информации в сети – анализ технологий и синтез решений / А. В. Галицкий, С. Д. Рябко, В. Ф. Шаньгин. – М. : ДМК Пресс, 2004. – 616 с.

5. Герасименко В. А. Основы теории защиты информации в автоматизированных системах обработки данных / В. А. Герасименко. – М. : Деп. в ВИНИТИ, 1991. – 410 с.

6. ГОСТ Р 34.10–2001. Государственный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. – М. : Госстандарт России, 2001. – 24 с.

7. Гортинская Л. В. Реализация протоколов коллективной подписи на основе стандартов ГОСТ 34.310−95 и ДСТУ 4145-2002 / Л. В. Гортинская, Н. А. Молдовян, Г. Л. Козина // Правове, нормативне та метрологичне забезпечення системи захисту інформації в Україні. – К. : НТУУ "КПІ". – 2008. – № 1. – С. 21–25.

8. Глушков В. М. Кибернетика, вычислительная техника, информатика. Избранные тр. в трех томах. Т. 1. Математические вопросы кибернетики. Т. 2. ЭВМ – техническая база кибернетики. Т. 3. Кибернетика и ее применение в народном хозяйстве. / В. М. Глушков. – К. : Наукова думка. – 1990. – Т. 1. – 264 с. ; Т. 2. – 267 с. ; Т. 3. – 222 с.

9. Грайворонський М. В. Безпека інформаційно-комунікаційних систем / М. В. Грайворонський, О. М. Новіков. – К. : Видавнича група BHV, 2009. – 608 с.

10. Дорошенко А. Н. Информационная безопасность. Методы и средства защиты информации в компьютерных системах : учебн. пособ. / А. Н. Дорошенко, Л. Л. Ткачев. – М. : МГУПИ, 2006. – 143 с.

11. ДСТУ 4145–2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. – К. : Держстандарт України, 2002. – 40 с.

12. ДСТУ 3396.2-97. Захист інформації. Технічний захист інформації. Терміни та визначення. – Введ. 01.01.98. – К. : Держстандарт України, 1997. – 11 с.

13. Ємець В. Сучасна криптографія. Основні поняття / В. Ємець, А. Мельник, Р. Попович. – Львів : Бак, 2003. – 144 с.

14. Жельников В. Криптография от папируса до компьютера / В. Жельников. – М. : ABF, 1994. – 324 с.

15. Жуков А. Е. Криптоанализ по побочным каналам (Side Channel Attacks) / А. Е. Жуков // Материалы конференции РусКрипто. – 2006. – 10 с.

16. Зубов А. Ю. Криптографические методы защиты информации. Совершенные шифры : учебн. пособ. / А. Ю. Зубов. – М. : Гелиос АРВ, 2005. – 192 с.

17. Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях / М. А. Иванов. – М. : Кудиц – Образ, 2001. – 368 с.

18. Основи інформаційної безпеки / С. В. Кавун, О. А. Смірнов, В. Ф. Столбов – Кіровоград : Вид. КНТУ, 2012. – 414 с.

19. Кан Д. Взломщики кодов / Д. Кан. – М. : Центрполиграф, 2000. – 452 с.

20. Казарин О. В. Безопасность программного обеспечения компьютерных систем : монография / О. В. Казарин. – М. : МГУЛ, 2003. – 212 с.

21. Коробейников А. Г. Математические основы криптологии : учебн. пособ. / А. Г. Коробейников, Ю. А. Гатчин. – СПб. : СПб ГУ ИТМО, 2004. – 106 с.

22. Кузнецов О. О. Захист інформації в інформаційних системах. Методи традиційної криптографії : навч. посібн. / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2010. – 316 с.

23. Кузнецов О. О. Захист інформації в інформаційних системах / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2011. – 512 с.

24. Лукацкий А. Обнаружение атак / А. Лукацкий. – СПб. : БХВ-Петербург, 2001. – 624 с.

25. Масленников М. Е. Практическая криптография / М. Е. Масленников. – СПб. : BHV, 2003. – 458 с.

26. Милославская Н. Р. Интрасети: доступ в Интернет, защита / Н. Р. Милославская, А. И. Толстой. – М. : Юнити-Дана, 2000. – 527 с.

27. Молдовян А. А. Криптография / А. А. Молдовян, Н. А. Молдовян, Б. Я. Советов / Серия "Учебники для вузов. Специальная литература". – СПб. : Лань, 2000. – 224 с.

28. НД ТЗІ 1.1-003-99: Термінологія в області захисту інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБУ № 22 від 28.04.1999. ДСТСЗІ СБУ. – К., 1999. – 34 с.

29. НД ТЗІ 2.5-004-99: Критерії оцінки захищеності інформації у комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБУ № 22 від 28.04.1999. ДСТСЗІ СБУ. – К., 1999. – 34 с.

30. НД ТЗІ 2.5-005-99: Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБУ № 22 від 28.04.1999. ДСТСЗІ СБУ. – К., 1999. – 34 с.

31. НД ТЗІ 3.7-001-99: Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі. Затверджено наказом ДСТСЗІ СБУ № 22 від 28.04.1999 р. ДСТСЗІ СБУ. – К., 1999. – 34 с.

32. Основы информационной безопасности : учебн. пособ. для вузов / Е. Б. Белов, В. П. Лось, Р. В. Мещеряков и др. – М. : Горячая линия – Телеком, 2006. – 544 с.

33. Остапов С. Е. Основи криптографії / С. Е. Остапов, Л. О. Валь. – Чернівці : Книги ХХІ, 2008. – 188 с.

34. Поповский В. В. Защита информации в телекоммуникационных системах : учебник / В. В. Поповский, А. В. Персиков. – Х. : ООО "Компания СМИТ", 2006. – Т. 1. – 292 с.

35. Поповский В. В. Защита информации в телекоммуникационных системах : учебник / В. В. Поповский, А. В. Персиков. – Х. : ООО "Компания СМИТ", 2006. – Т. 2. – 292 с.

36. Потий А. В. Стандартизация и сертификация в сфере защиты информации. Стандарты механизмов безопасности : учебн. пособ. / А. В. Потий. – Х. : ХНУРЕ, 2002. – 80 с.

37. Про державну таємницю : Закон України від 17.08.1995. – К. : Урядовий кур’єр. – 1995. – № 123 – 124.

38. Про захист інформації в інформаційно-телекомунікаційних системах : Закон України від 18.04.2006, – К. : Урядовий кур'єр. – 2006. № 73 – 74.

39. Про інформацію : Закон України від 03.04.1997. – К. : Урядовий кур’єр. – 1997. – № 62.

40. Ростовцев А. Г. Методы криптоанализа классических шифров / А. Г. Ростовцев, Н. В. Михайлова. – М : Наука, 2005. – 208 с.

41. Сингх С. Книга шифров / С. Сингх. – М. : АСТ: Астрель, 2007. – 447 с.

42. Основи захисту інформації : навч. посібн. / О. А. Смірнов, Л. Г. Віхрова, С. І. Осадчий та ін. – Кіровоград, 2010. – 322 с.

43. Столингс В. Криптография и защита сетей / В. Столингс. – М. : Вильямс, 2004. – 848 с.

44. Трубачев А. П. Оценка безопасности информационных технологий / А. П. Трубачев ; под общ. ред. В. А. Галатенко. – М. : СИП РИА, 2001. – 356 с.

45. Хорошко В. А. Методы и средства зашиты информации / В. А. Хорошко, А. А. Чекатков. – К. : Юниор, 2003. – 504 с.

46. Чмора А. Л. Современная прикладная криптография / А. Л. Чмора. – М. : Гелиос АРВ, 2001. – 256 с.

47. Шеннон К. Э. Теория связи в секретных системах. В кн. : Работы по теории информации и кибернетике / К. Э. Шеннон. – М. : ИЛ, 1963. – С. 333–402.

48. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер ; пер. с англ. – М. : Изд. ТРИУМФ, 2002. – 816 с.

49. Щеглов А. Ю. Защита компьютерной информации от несанкционированного доступа / А. Ю. Щеглов. – СПб. : Наука и Техника, 2004. – 384 с.

50. Biham E. (1999). Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. / E. Biham, A. Biryukov, A. Shamir // EUROCRYPT. – 1999. – Pp. 12–23.

51. Millan W. Boolean function design using hill climbing methods / W. Millan, A. Clark, E. Dawson // In 4th Australasian Conference on Information, Security and Privacy. – 1999. – Num. 1587. – P. 1–11.

52. Загальні критерії [Електронний ресурс]. – Режим доступу : www.isofts.kiev.ua/c/.../get_file.

53. Законодательная и нормативная база Украины в области ТЗИ и КЗИ [Электронный ресурс]. – Режим доступа : http://www.bezpeka.com/ru/lib /lawua.html.

54. Концепція технічного захисту інформації в Україні. – [Електронний ресурс]. – Режим доступу : http://zakon1.rada.gov.ua/cgibin/laws /main.cgi?nreg=1126-97-%EF.

55. Положення про проведення відкритого конкурсу криптографічних алгоритмів [Електронний ресурс] // Інститут кібернетики ім. В. М. Глушкова НАНУ; ДСТСЗІ [Електронний ресурс]. – Режим доступу : http://www.dstszi.gov.ua /dstszi/ control /ru/publish/article;.

56. Украинский ресурс по безопасности [Електронний ресурс]. – Режим доступа : http://kiev-security.org.ua.

57. Daemen J. AES Proposal: Rijndael, AES Algorithm Submission [Electronic resource] / J. Daemen, V. Rijmen. – Access mode : http://www.docstoc.com/docs/14641406/AES-Implementation-and-Performance-Evaluation-on-8-bit-Microcontrollers.

58. Department of Defense Trusted Computer System Evaluation Criteria [Electronic resource]. – Access mode : http://www.dynamoo.com /orange/fulltext.htm.

59. D.VAM.1 Performance Benchmarks. Revision 1.1 / R. Avanzi, B. Chevallier-Mames etc. // In: ECRYPT Research report IST-2002-507932. European Network of Excellence in Cryptology / M. Joye ed. – August, 3, 2005. – 87 p.

60. ISO/IEC 7498-2:1989 – Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture [Electronic resource]. – Access mode : http://www.iso.org/iso /catalogue_detail.htm?csnumber=14256.

61. NESSIE consortium "NESSIE Security report." Deliverable report D20 – NESSIE, 2002. – NES/DOC/ENS/WP5/D20 [Electronic resource]. – Access mode : http://www.cryptonessie.org/.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Остапов Сергій Едуардович
Євсеєв Сергій Петрович
Король Ольга Григорівна

ТЕХНОЛОГІЇ ЗАХИСТУ ІНФОРМАЦІЇ

Навчальний посібник

Відповідальний за випуск **Пономаренко В. С.**

Відповідальний редактор **Сєдова Л. М.**

Редактор **Бутенко В. О.**

Коректор **Мартовицька-Максимова В. А.**

План 2013 р. Поз. № 55-П.
Підп. до друку Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 29,75. Обл.-вид. арк. 37,19. Тираж прим. Зам. №

Видавець і виготівник – видавництво ХНЕУ, 61166, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ***Дк № 481 від 13.06.2001 р.***