МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»

Факультет безпеки підприємств

Кафедра організації комплексного захисту інформації

ОПОРНИЙ КОНСПЕКТ ЛЕКЦІЙ З ДИСЦИПЛІНИ

ОСНОВИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

Укладач: к.т.н., доц. Лахно В.А.

Київ – 2016 рік

Тема 1. Історія криптографії

Історія криптографії налічує близько 4 тисяч років. Як основний критерій періодизації криптографії можна взяти технологічні характеристики використовуваних методів шифрування.

До нашого часу криптографія займалася виключно забезпеченням конфіденційності повідомлень (тобто шифруванням) — перетворенням повідомлень із зрозумілої форми в незрозумілу і зворотне відновлення на стороні одержувача, роблячи його неможливим для прочитання для того, хто перехопив або підслухав без секретного знання (а саме ключа, необхідного для дешифровки повідомлення). В останні десятиліття 21 сторіччя сфера застосування криптографії розширилася і включає не лише таємну передачу повідомлень, але і методи перевірки цілісності повідомлень, ідентифікування відправника/одержувача (аутентифікація), цифрові підписи, інтерактивні підтвердження, та технології безпечного спілкування тощо.

Найперші форми тайнопису вимагали не більше ніж аналог олівця та паперу, оскільки в ті часи більшість людей не могли читати. Поширення писемності, або писемності серед ворогів, викликало потребу саме в криптографії. Основними типами класичних шифрів є перестановочні шифри, які змінюють порядок літер в повідомленні, тапідстановочні шифри, які систематично замінюють літери або групи літер іншими літерами або групами літер. Прості варіанти обох типів пропонували слабкий захист від досвідчених супротивників. Одним із ранніх підстановочних шифрів був шифр Цезаря, в якому кожна літера в повідомленні замінювалась літерою через декілька позицій із абетки. Цей шифр отримав ім'я Юлія Цезаря, який його використовував, зі зсувом в 3 позиції, для спілкування з генералами під час військових кампаній, подібно до коду EXCESS-3 в булевій алгебрі.

Шляхом застосування шифрування намагаються зберегти зміст спілкування в таємниці, подібно до шпигунів, військових лідерів, та дипломатів. Зберіглися також відомості про деякі з ранніх єврейських шифрів. Застосування криптографії радиться в Камасутрі як спосіб спілкування закоханих без ризику незручного викриття.[1] Стеганографія(тобто, приховування факту наявності повідомлення взагалі) також була розроблена в давні часи. Зокрема, Геродотприховав повідомлення — татуювання на поголеній голові раба — під новим волоссям. До сучасних прикладів стеганографії належать невидимі чорнила, мікрокрапки, цифрові водяні знаки, що застосовуються для приховування інформації.

Шифротексти, отримані від класичних шифрів (та деяких сучасних), завжди видають деяку статистичну інформацію про текст повідомлення, що може бути використано для зламу. Після відкриття частотного аналізу (можливо,арабським вченим аль-Кінді) в 9-тому столітті, майже всі такі шифри стали більш-менш легко зламними досвідченим фахівцем. Класичні шифри зберігли популярність, в основному, у вигляді головоломок (див. Криптограма).

Майже всі шифри залишались беззахисними перед криптоаналізом з використанням частотного аналізу до винаходуполіалфавітного шифру, швидше за все, Леоном-Баттіста Альберті приблизно в 1467 році (хоча, існують свідчення того, що знання про такі шифри існували серед арабських вчених). Винахід Альберті полягав в тому, щоб використовувати різні шифри (наприклад, алфавіти підстановки) для різних частин повідомлення. Йому також належить винахід того, що може вважатись першим шифрувальним приладом: колесо, що частково реалізовувало його винахід (див. Шифрувальний диск Альберті).

В поліалфавітному шифрі Віженера (англ. *Vigenère cipher*), алгоритм шифрування використовує *ключове слово*, яке керує підстановкою літер в залежності від того, яка літера ключового слова використовується. В середині 1800-тих,Чарльз Беббідж показав, що поліалфавітні шифри цього типу залишились частово беззахисними перед частотним аналізом.[2]

Хоча частотний аналіз є потужною та загальною технікою, шифрування, на практиці, часто було ефективним; багато із криптоаналітиків не знали цю техніку. Дешифрування повідомлень без частотного аналізу практично означало необхідність знання використаного шифру, спонукаючи, таким чином, до шпигунства, підкупу, крадіжок, зрад, тощо для отримання алгоритму. Згодом, в 19-тому столітті, було визнано, що збереження алгоритму шифрування в таємниці не забезпечує захист від зламу; насправді, було встановлено, що будь-яка адекватна криптографічна схема залишається у безпеці, навіть за умови доступу сторонніх. Збереження в таємниці ключа має бути достатньою умовою захисту інформації нормальним шифром. Цей фундаментальний принцип було вперше проголошено в 1883 Огюстом Керкгофсом, і загальновідомий як принцип Керкгоффза; різкіший варіант озвучив Клод Шеннон як *максиму Шеннона — ворог знає систему*.

Було створено різні механічні прилади та інструменти для допомоги в шифруванні. Одним з найперших є скітала встародавній Греції, палиця, що, як вважається, використовувалась Спартанцями в якості перестановочного шифру. В середньовіччя, було винайдено інші засоби, такі як дірочний шифр, що також використовувася для часткової стеганографії. Разом із винаходом поліалфавітних шифрів, було розроблено досконаліші засоби, такі як власний винахід Альберті шифрувальний диск, табула ректа Йогана Тритеміуса, та мультициліндр Томаса Джефферсона(повторно винайдений Базерієсом приблизно в 1900 році).

Поява цифрових комп'ютерів та електроніки після Другої світової війни зробило можливим появу складніших шифрів. Більше того, комп'ютери дозволяли шифрувати будь-які дані, які можна представити в комп'ютері у двійковому виді, на відміну від класичних шифрів, які розроблялись для шифрування письмових текстів. Це зробило непридатними для застосування лінгвістичні підходи в криптоаналізі. Багато комп'ютерних шифрів можна характеризувати за їхньою роботою з послідовностями бінарних бітів (інколи в блоках або групах), на відміну від класичних та механічних схем, які, зазвичай, працюють безпосередньо з літерами. Однак, комп'ютери також знайшли застосування у криптоаналізі, що, в певній мірі, компенсувало підвищення складності шифрів. Тим не менше, гарні сучасні шифри залшались попереду криптоаналізу; як правило, використання якісних шифрів дуже ефективне (тобто, швидке і вимагає небагато ресурсів), в той час як злам цих шифрів потребує набагато більших зусиль ніж раніше, що робить криптоаналіз настільки неефективним та непрактичним, що злам стає практично неможливим.

Тема 2. Криптоаналіз

Криптоаналіз – розділ криптології, що займається математичними методами порушення конфіденційності і цілісності інформації без знання ключа.

Термін був введений американським криптографом Вільямом Ф. Фрідманом в 1920 році.

У більшості випадків під Криптоаналізом розуміється з'ясування ключа; криптоаналіз включає також методи виявлення уразливості криптографічних алгоритмів або протоколів.

Спочатку методи криптоаналізу грунтувалися на лінгвістичних закономірностях природного тексту і реалізовувалися з використанням тільки олівця та паперу. З часом в криптоаналізі наростає роль чисто математичних методів, для реалізації яких використовуються спеціалізовані криптоаналітичні комп'ютери.

Спробу розкриття конкретного шифру із застосуванням методів криптоаналізу називають криптографічною атакою на цей шифр. Криптографічну атаку

Криптоаналіз еволюціонував разом з розвитком криптографії: нові, більш досконалі шифри приходили на зміну вже зламаним системам кодування тільки для того, щоб криптоаналітики винайшли більш витончені методи злому систем шифрування. Поняття криптографії та криптоаналізу нерозривно пов'язані один з одним: для того, щоб створити стійку до злому систему, необхідно врахувати всі можливі способи атак на неї.

Класичний криптоаналіз

Хоча поняття криптоаналізу було введено порівняно недавно, деякі методи злому були винайдені десятки століть тому. Першою відомою письмовою згадкою про криптоаналіз є «Манускрипт про дешифрування криптографічних повідомлень», написаний арабським вченим аль-Кінді ще в 9 столітті. У цій науковій праці міститься опис методу частотного аналізу.

Частотний аналіз - основний інструмент для злому більшості класичних шифрів перестановки або заміни. Даний метод грунтується на припущенні про існування нетривіального статистичного розподілу символів, а також їх послідовностей одночасно і у відкритому тексті, і в шифротексті. Причому даний розподіл буде зберігатися з точністю до заміни символів як в процесі шифрування, так і в процесі дешифрування. Варто відзначити, що за умови досить великої довжини шифрованого повідомлення моноалфавітні шифри легко піддаються частотному аналізу: якщо частота появи літери в мові та частота появи деякого присутнього в шифротексті символу приблизно рівні, то в цьому випадку з великою часткою ймовірності можна припустити, що даний символ і буде цієї самою літерою. Самим простим прикладом частотного аналізу може служити банальний підрахунок кількості кожного з зустрінутих символів, потім процедури ділення отриманого числа символів на кількість всіх символів в тексті і множення результату на сто, щоб представити остаточний відповідь у відсотках. Далі отримані процентні значення порівнюються з таблицею імовірнісного розподілу літер для передбачуваної мови оригіналу.

Брюс Шнайер виділяє 4 основних і 3 додаткових методи криптоаналізу, припускаючи знання криптоаналітиками алгоритму шифра:

Основні методи криптоаналізу:

Атака на основі шифротексту

Атака на основі відкритих текстів і відповідних шифротекстів

Атака на основі підібраного відкритого тексту (можливість вибрати текст для шифрування)

Атака на основі адаптивно підібраного відкритого тексту

Додаткові методи криптоаналізу:

Атака на основі підібраного шифротексту

Атака на основі підібраного ключа

Бандитський криптоаналіз

Атаки на основі шифротексту

Припустимо, криптоаналітик має деяке число шифротекстів, отриманих в результаті використання одного і того ж алгоритму шифрування. У цьому випадку криптоаналітик може вчинити тільки атаку на основі шифротексту. Метою криптографічної атаки в цьому випадку є знаходження якомога більшого числа відкритих текстів, відповідних наявним шифротекстам, або, що ще краще, знаходження використовуваного при шифруванні ключа.

Вхідні дані для подібного типу атак криптоаналітик може отримати в результаті простого перехоплення зашифрованих повідомлень. Якщо передача здійснюється відкритим каналом, то реалізація завдання по збору даних порівняно легка і тривіальна. Атаки на основі шифротексту є найслабшими і незручними.

Атака на основі відкритих текстів і відповідних шифротекстів[ред. • ред. код]

Основна стаття: Атака з відомим відкритим текстом

Нехай у розпорядженні криптоаналітика є не тільки шифротексти, але і відповідні їм відкриті тексти.

Тоді існують два варіанти постановки задачі:

Знайти ключ, використаний для перетворення відкритого тексту в шифротекст

Створити алгоритм, здатний дешифрувати будь-яке повідомлення, закодоване за допомогою цього ключа

Отримання відкритих текстів відіграє вирішальну роль у здійсненні цієї атаки. Відкриті тексти витягують з самих різних джерел. Так, наприклад, можна здогадатися про вміст файлу по його розширенню.

У разі злому листування можна зробити припущення, що лист має структуру типу:

«Привітання»

«Основний текст»

«Завершальне форма ввічливості»

«Підпис»

Отже, атака може бути організована шляхом підбору різних видів «Привітання» (наприклад, «Здрастуйте!», «Добрий день» і т. д.) та/або «Заключною форми ввічливості» (таких як «З повагою», «Щиро Ваш »і т. п.). Легко помітити, що дана атака сильніше атаки на основі одного лише шифротексту.

Атака на основі підібраного відкритого тексту

Для здійснення такого типу атаки криптоаналітику необхідно мати не тільки якусь кількість відкритих текстів та отриманих на їх основі шифротекстів. Крім іншого, в даному випадку криптоаналітик повинен мати можливість підібрати кілька відкритих текстів і отримати результат їх шифрування.

Завдання криптоаналітика повторюють завдання для атаки на основі відкритого тексту, тобто отримати ключ шифрування, або створити дешифрувальний алгоритм для даного ключа.

Отримати вхідні дані для такого виду атаки можна, наприклад, таким чином:

Створити і відправити підроблене не зашифровані повідомлення нібито від одного з користувачів, які зазвичай користуються шифруванням.

У деяких випадках можна отримати відповідь, в якому буде міститься зашифрований текст, що цитує зміст підробленого повідомлення.

При здійсненні атаки подібного типу криптоаналітик має можливість підбирати блоки відкритого тексту, що за певних умов може дозволити отримати більше інформації про ключ шифрування.

Атаки на основі адаптивно підібраного відкритого тексту

Атака такого типу є більш зручним окремим випадком атаки на основі підібраного відкритого тексту. Зручність атаки на основі адаптивно підібраного відкритого тексту полягає в тому, що крім можливості вибирати текст для шифрування , криптоаналітик може прийняти рішення про шифрування того чи іншого відкритого тексту на основі вже отриманих результатів операцій шифрування. Іншими словами, при здійсненні атаки на основі підібраного відкритого тексту криптоаналітик вибирає всього один великий блок відкритого тексту для подальшого шифрування, а потім на основі цих даних починає зламувати систему. У разі організації адаптивної атаки криптоаналітик може отримувати результати шифрування будь-яких блоків відкритого тексту, щоб зібрати дані, що його цікавлять, які будуть враховані при виборі наступних відправлених на шифрування блоків відкритого тексту і так далі. Наявність зворотного зв'язку дає атаці на основі адаптивно підібраного шифротексту перевагу перед усіма перерахованими вище типами атак.

Тема 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).

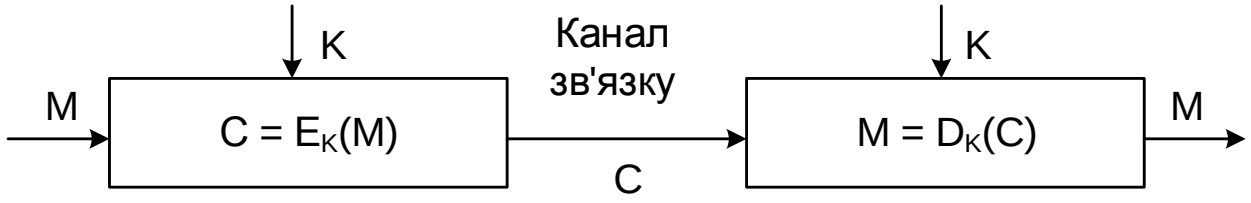


Рис. 3.1. **Загальна схема симетричного шифрування**

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.

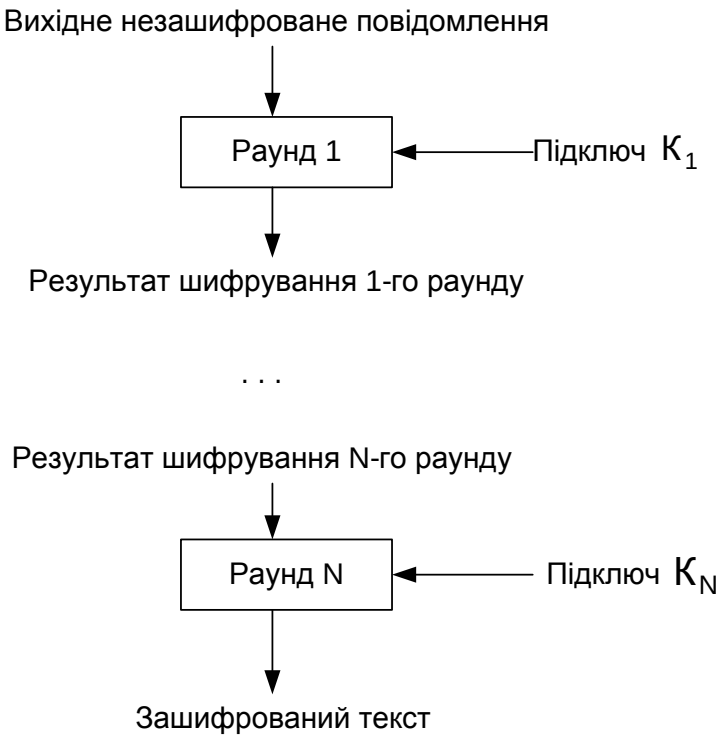


Рис. 3.2. **Структура симетричного алгоритму**

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.

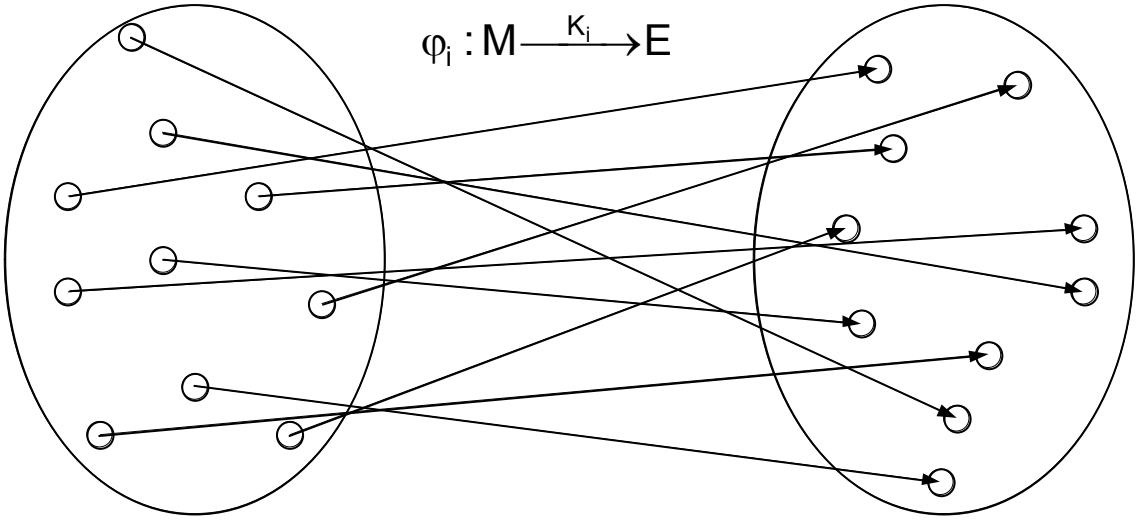


Рис. 3.3. **Відображення** $\varphi_i : M \xrightarrow{K_i} E$ **множини відкритих текстів у множину криптограм**

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:

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

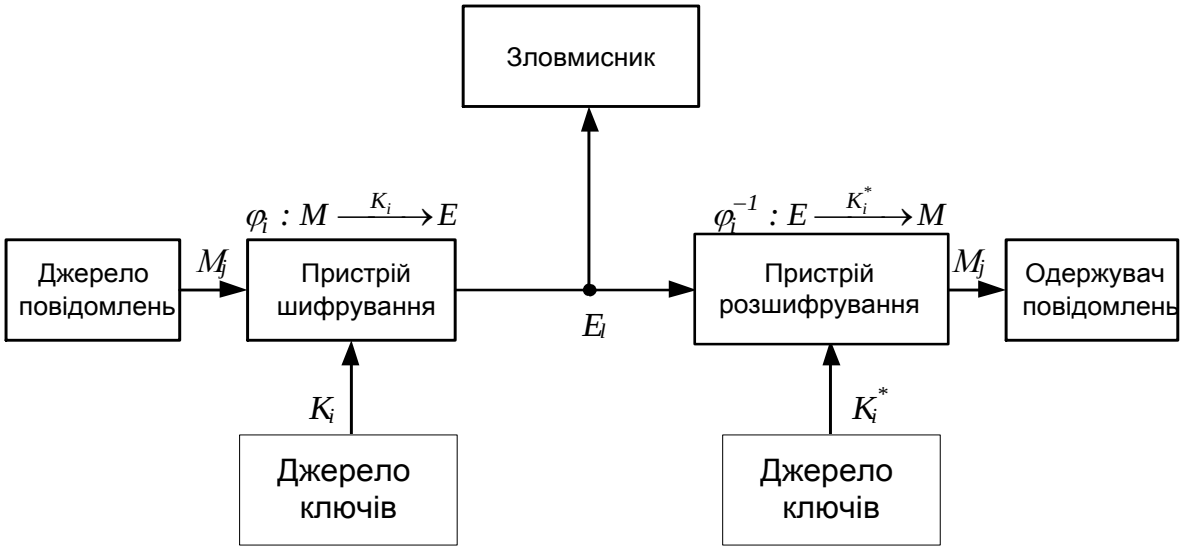


Рис. 3.4. **Структурна схема секретної системи**

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.

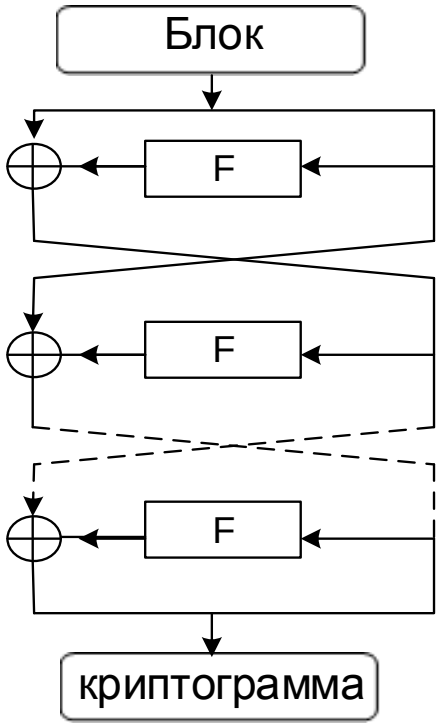


Рис. 3.5. **Структура сітки Х. Фейстеля**

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:

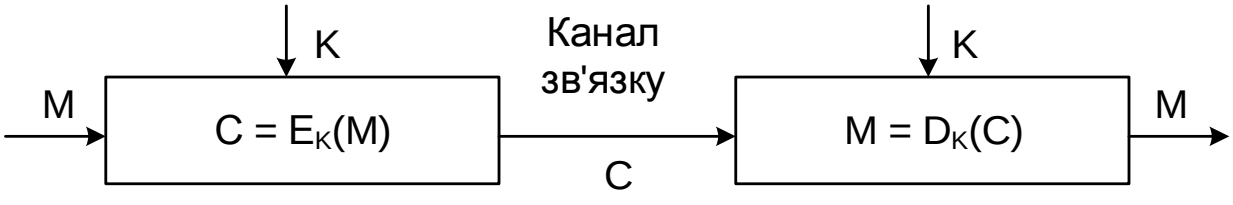


Рис. 3.6. **Схема роботи симетричної криптосистеми**

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):

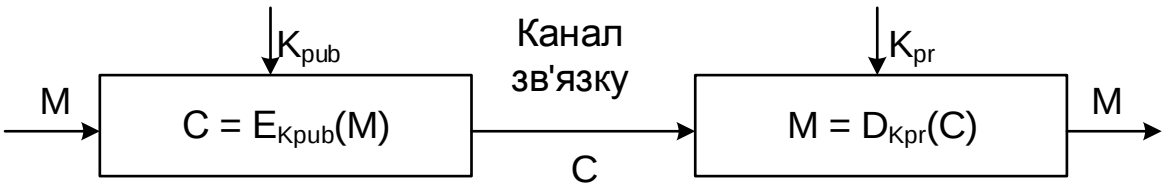


Рис. 3.7. **Схема роботи асиметричної криптосистеми**

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):

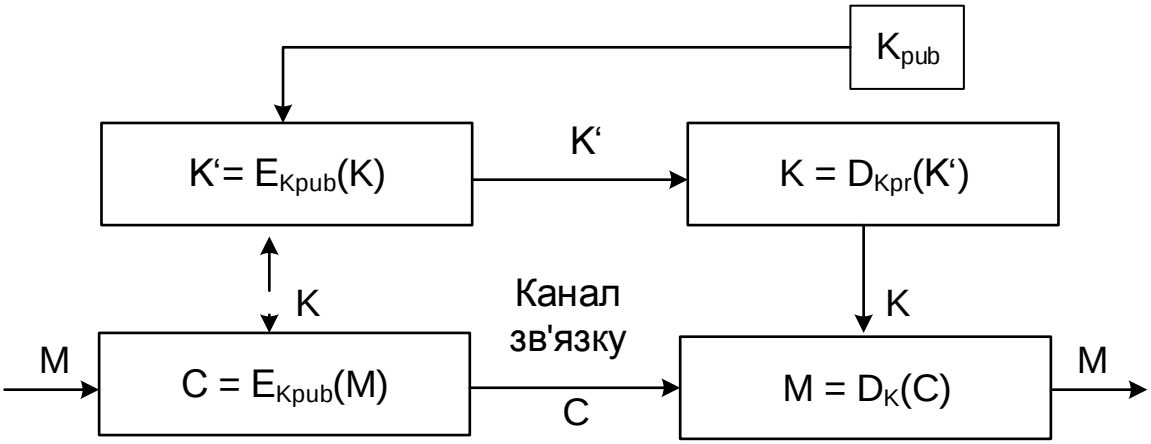


Рис. 3.8. **Схема роботи комбінованої криптосистеми**

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.

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.

Тема 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).

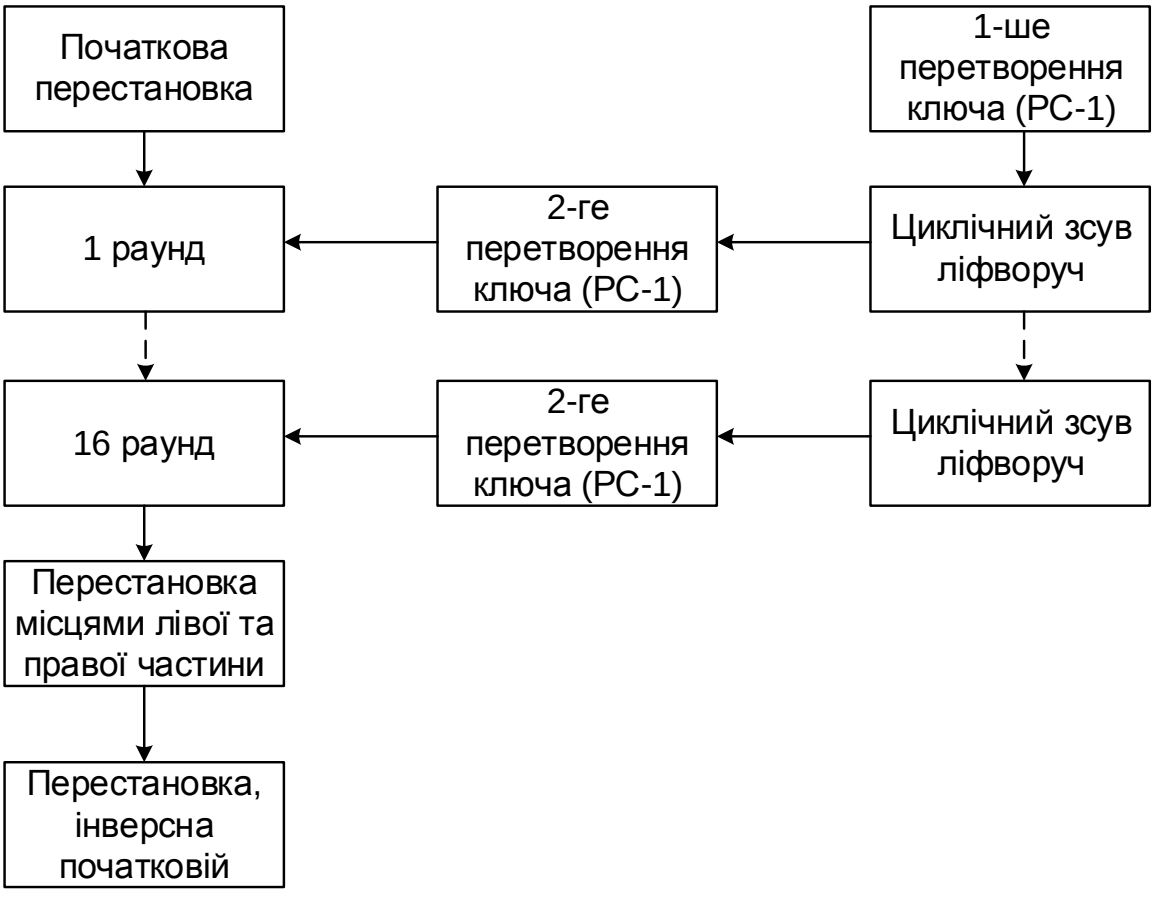


Рис. 4.1. **Загальна схема DES**

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.

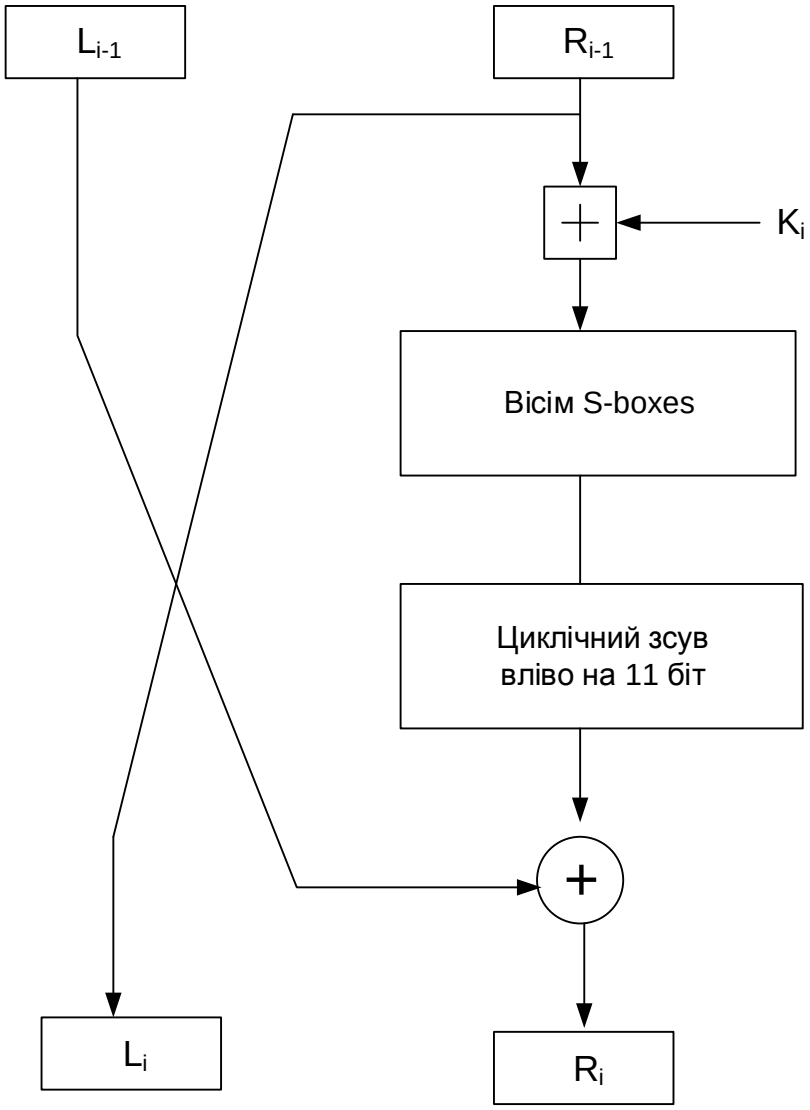


Рис. 4.2. **Один раунд DES**

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.

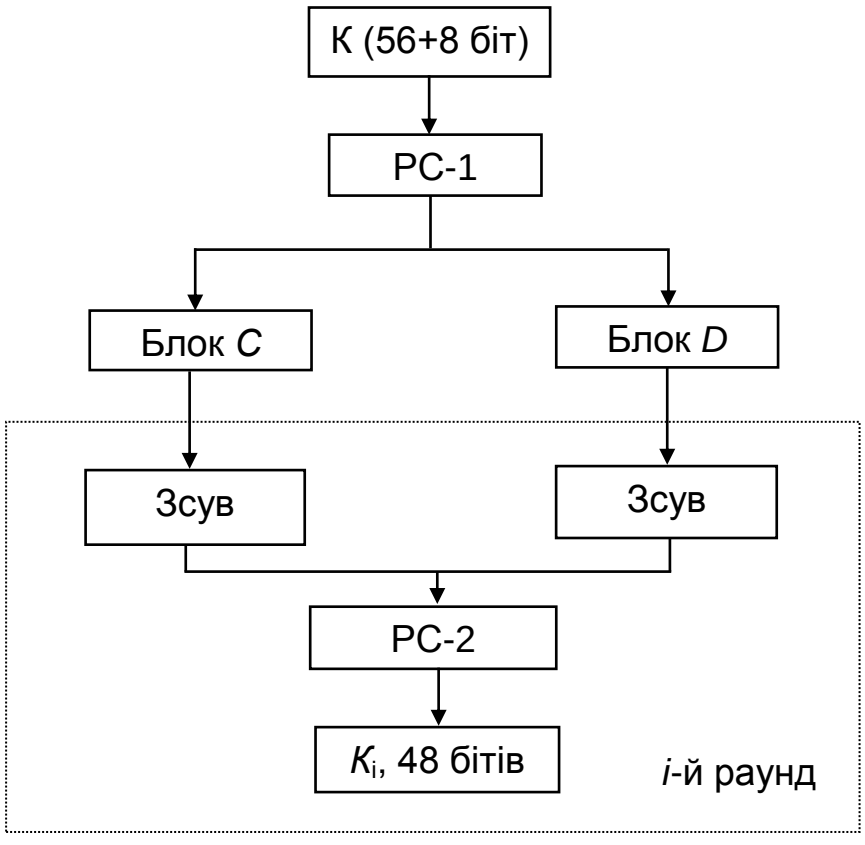


Рис. 4.3. **Блок-схема формування раундових ключів DES**

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.

4.2. 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

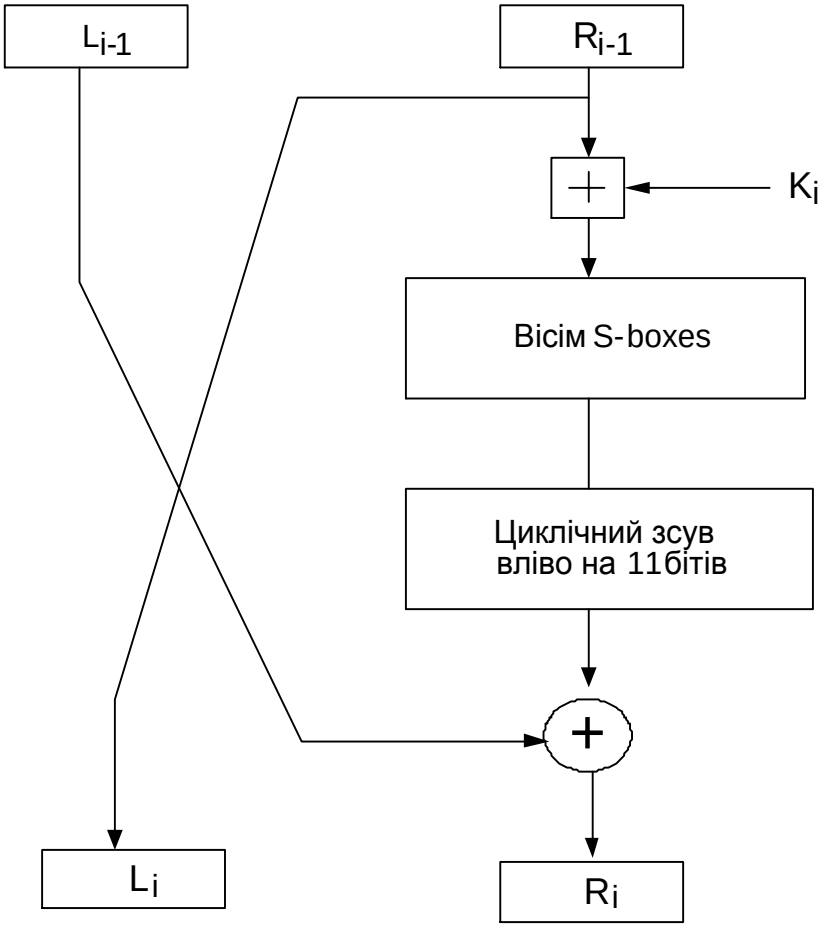


Рис. 4.6. **Один раунд алгоритму ГОСТ 28147-89**

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.

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.

Тема 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.

Тема 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 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.

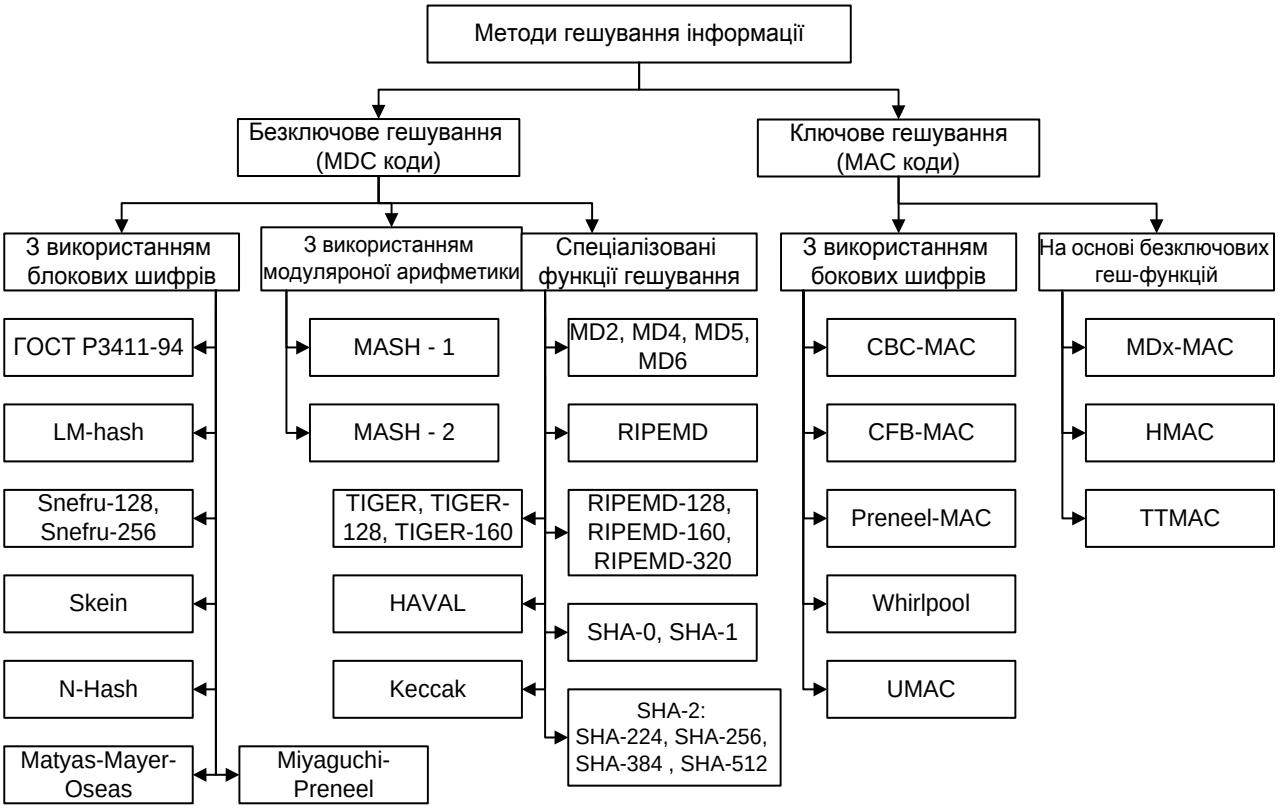


Рис. 6.1. **Класифікація геш-функцій**

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 Преніле́м (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.

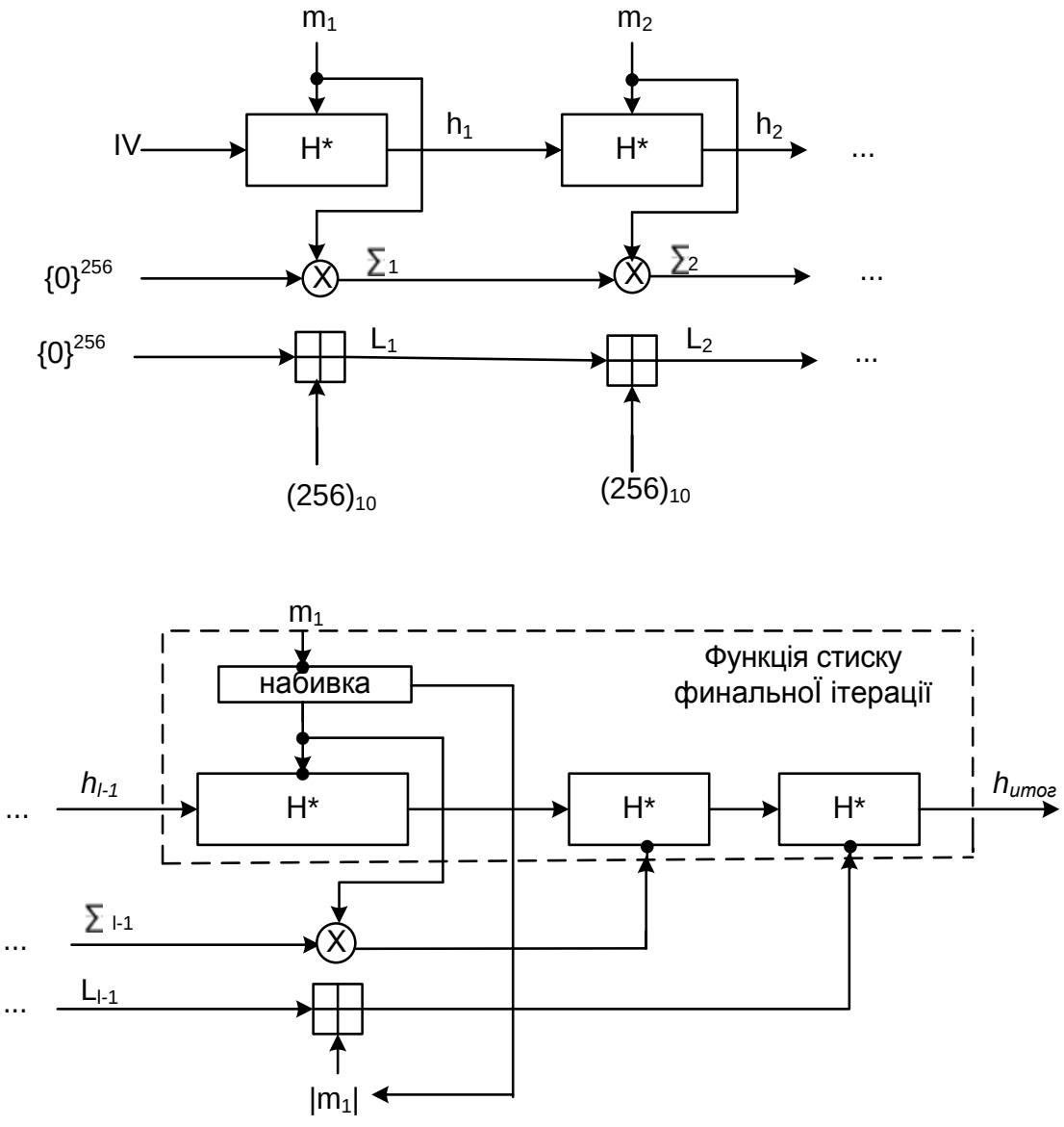


Рис. 6.2. **Геш-функція ГОСТ Р 34.11-94**

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

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.

6.2. 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. 6.3 наведена схема однієї ітерації алгоритму SHA-1.

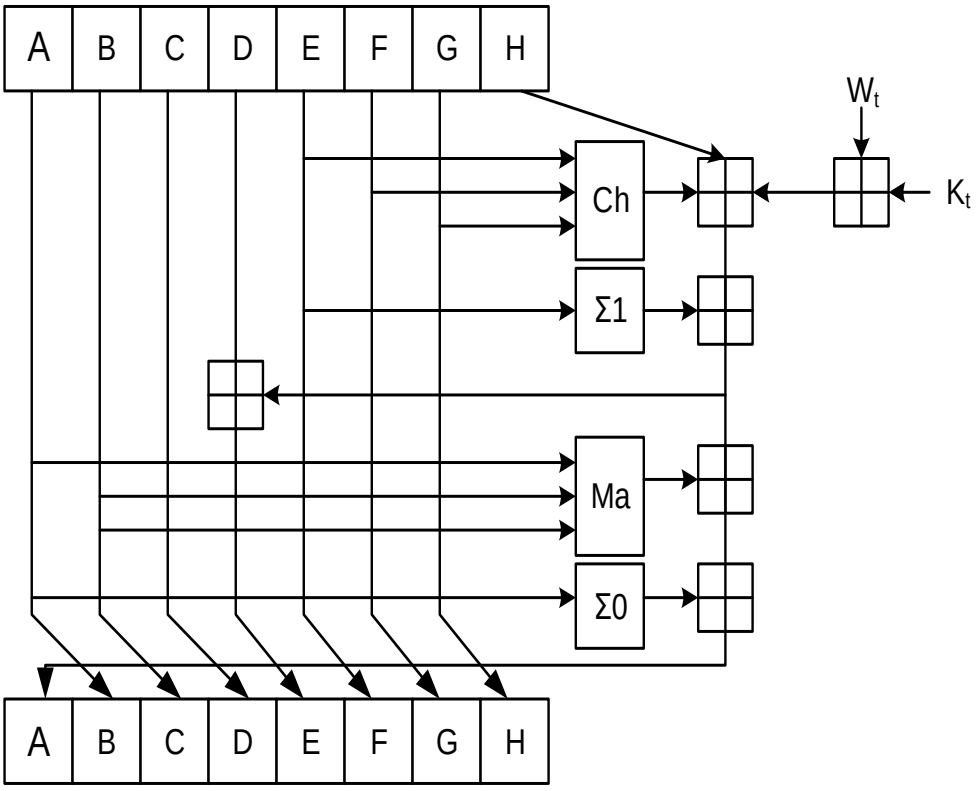


Рис. 6.3. **Одна ітерація алгоритму SHA1**

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. 6.4 наведена схема однієї ітерації алгоритму SHA-2.

У табл. 6.5 наведено деякі технічні характеристики різних варіантів SHA-2. "Внутрішній стан" означає проміжну геш-суму після обробки чергового блоку даних.

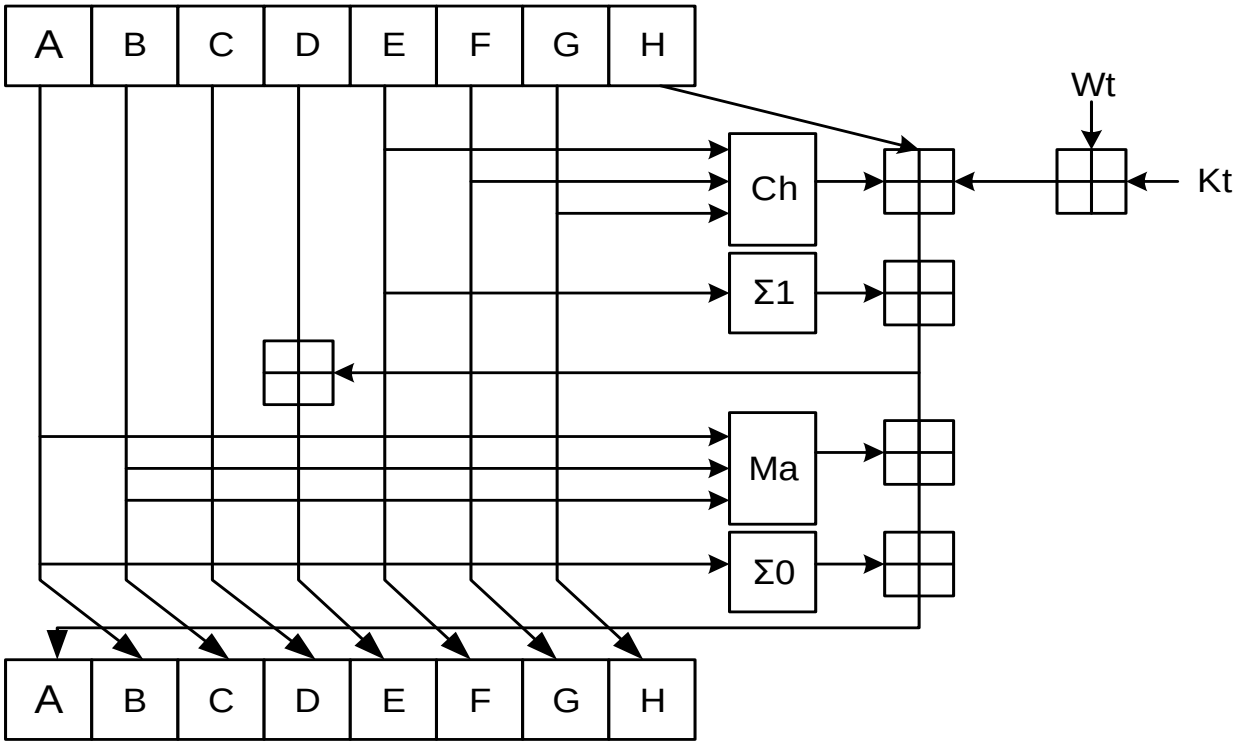


Рис. 6.4. **Схема однієї ітерації алгоритмів SHA-2**

Таблиця 6.5

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.

Тема 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 ∈ M визначає відповідний підпис s ∈ 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.

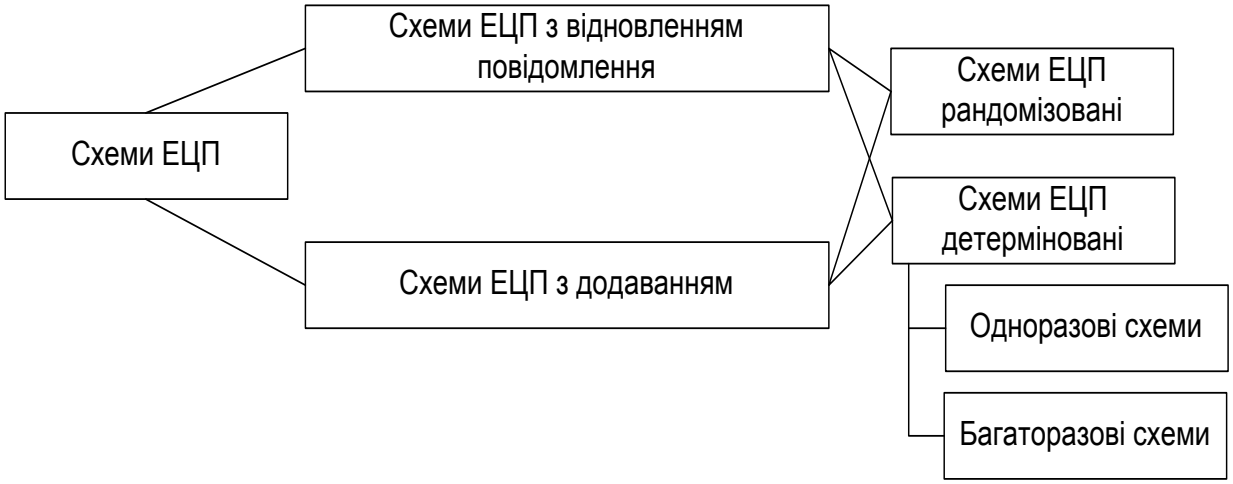


Рис. 7.1. **Класифікація схем ЕЦП**

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.

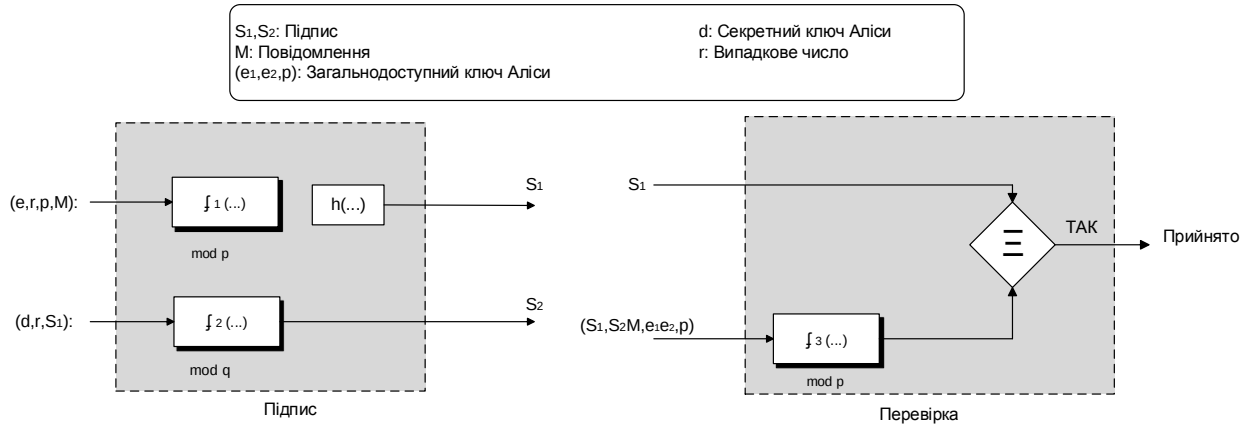


Рис. 7.2. **Схема цифрового підпису Клауса – Шнорра**

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.

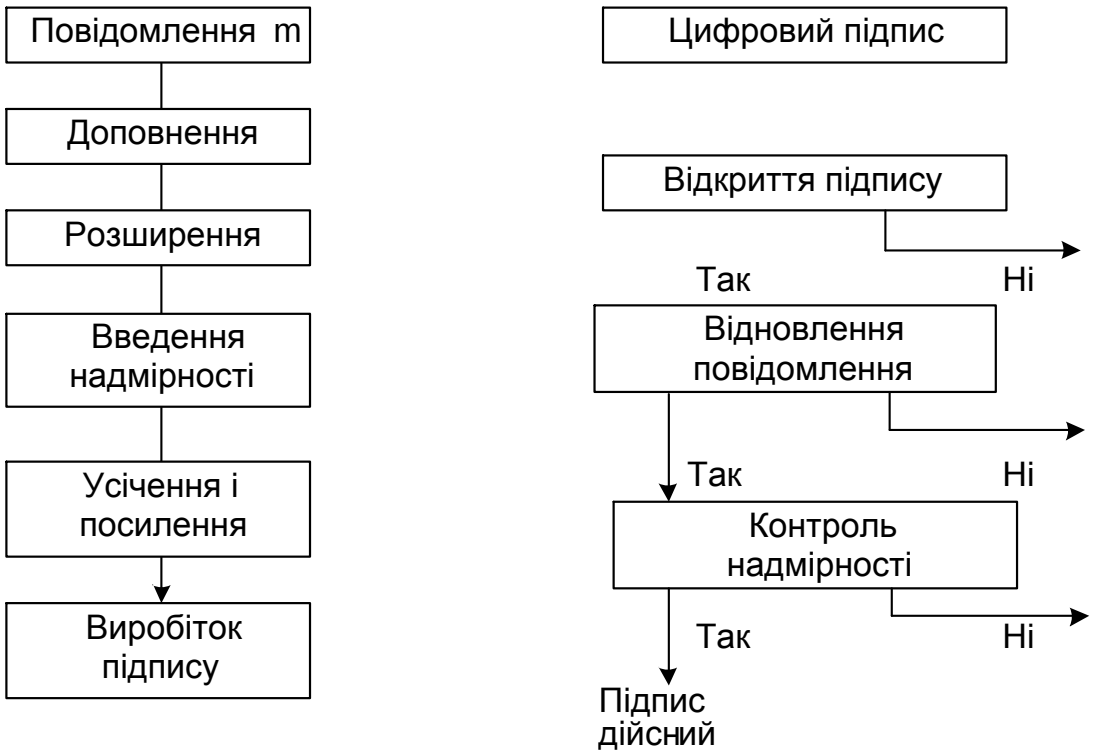
2. *Розширення повідомлення.* З доповненого повідомлення формується розширене повідомлення ME (message extension) шляхом ітеративної конкатенації ліворуч доповненого повідомлення із самим собою.

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.

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.

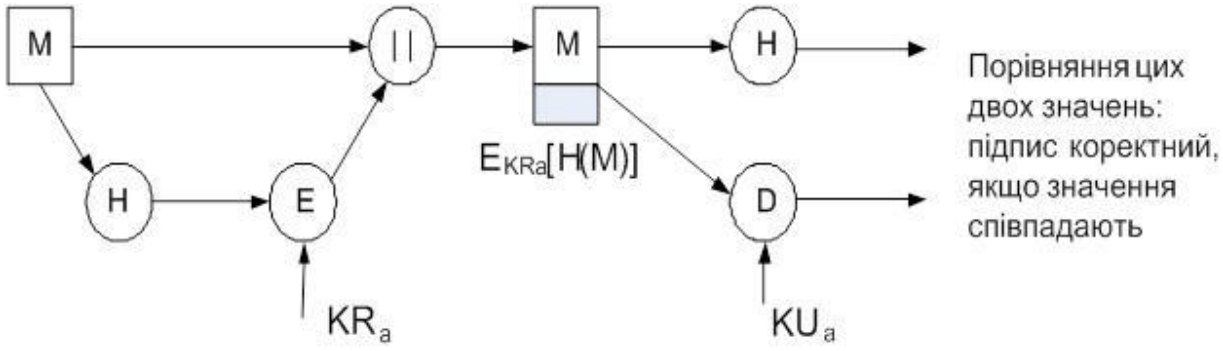


Рис. 7.4. **Створення і перевірка підпису за алгоритмом RSA**

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 $s = k^{-1}(h + xr)(mod q)$ 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 $u_1 = s^{-1}h(mod q) i u_2 = s^{-1}r(mod q)$.
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 $k = s^{-1}(h + xr)(mod q)$, 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.

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_T 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).

Тема 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.

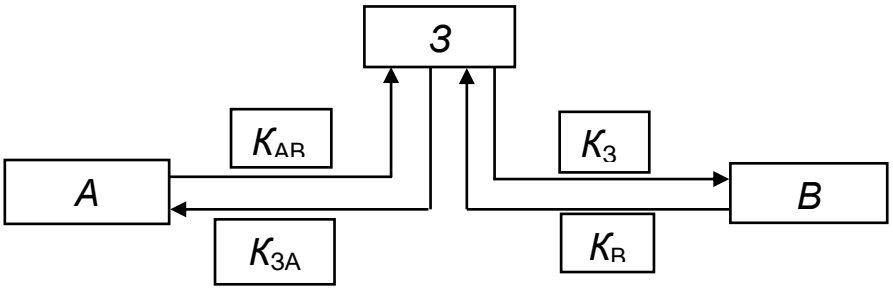


Рис. 8.1. **Схема атаки *Man-in-Middle* при обміні криптографічними ключами**

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).

Тема 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).

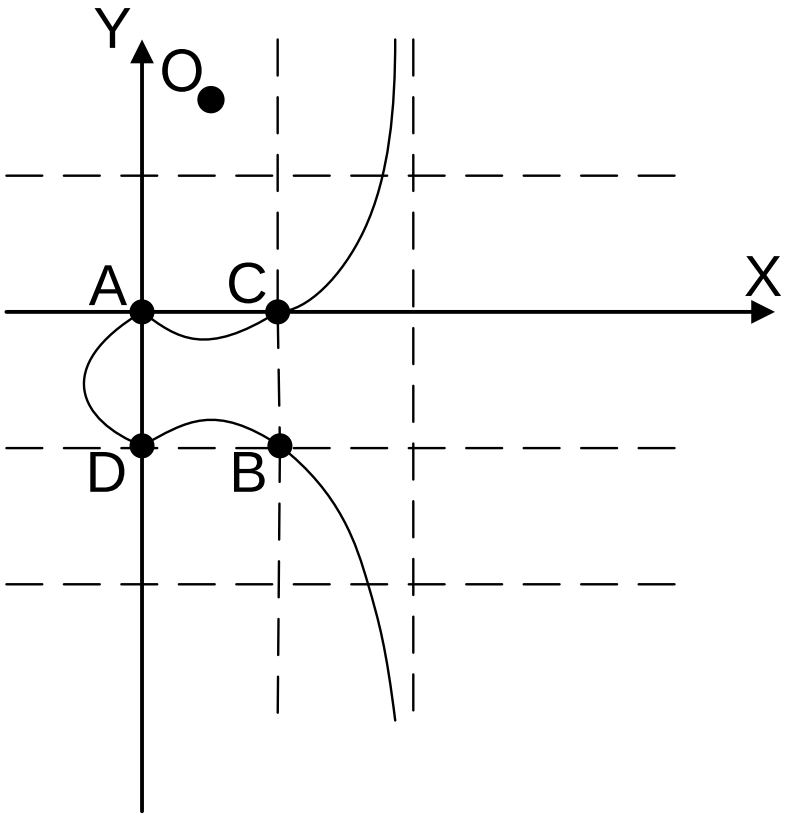


Рис. 9.1. **Приклад еліптичної кривої із чотирма точками**

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 $P_1 = – P_2$.

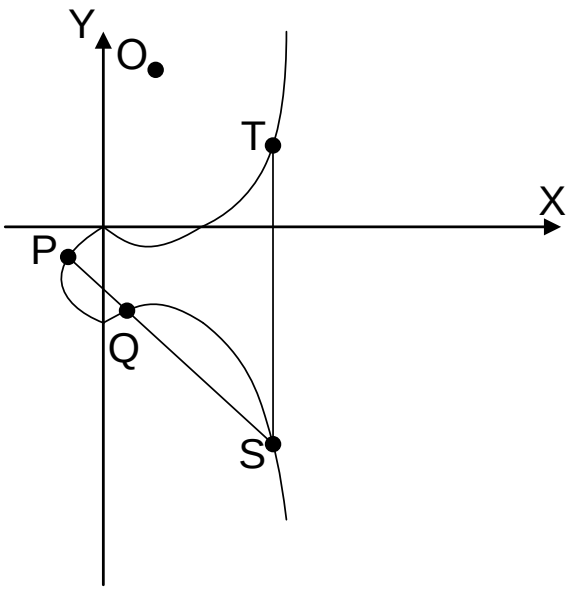


Рис. 9.2. **Додавання точок на еліптичній кривій**

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 $P \neq Q$, 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 $P \neq Q$, $\lambda = (3x_{12} + a)/2y_1$, 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 $P = k \times Q$.

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).

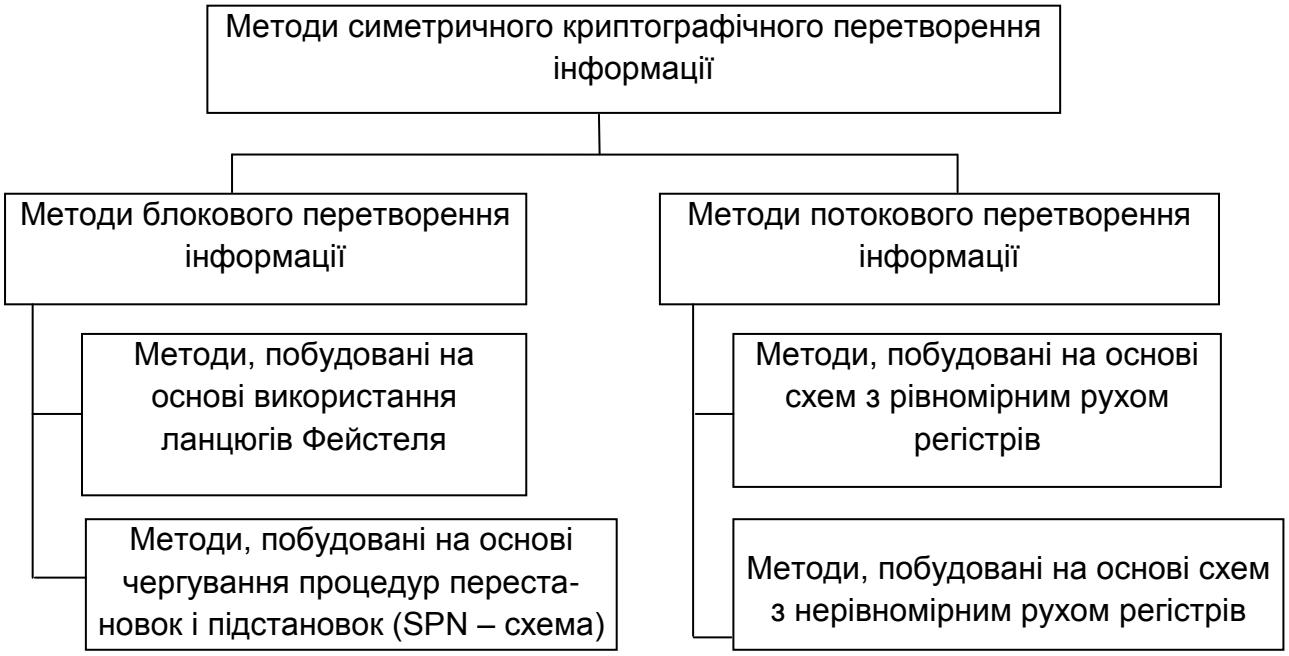


Рис. 9.3. **Класифікація методів симетричного криптографічного перетворення інформації**

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.

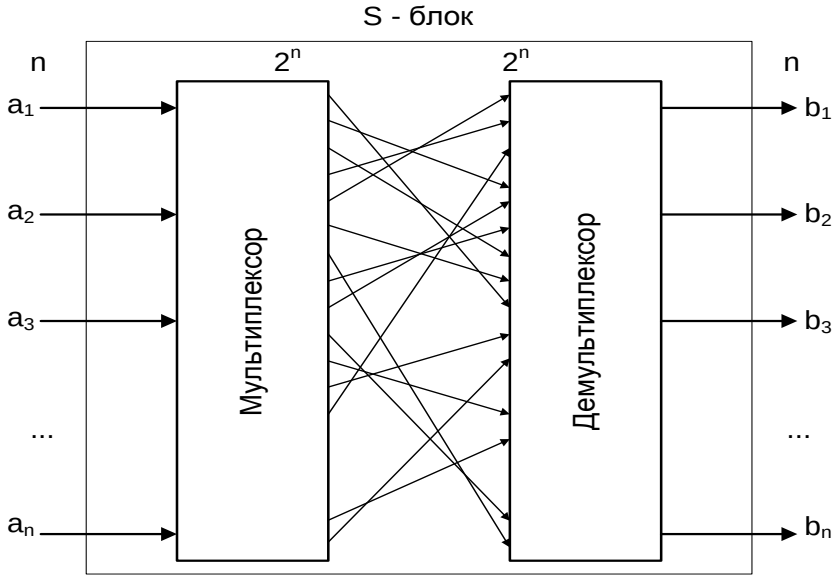


Рис. 9.4. **Схема перетворення даних у нелінійному вузлі заміни**

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.

Тема 10. Механізми та протоколи керування ключами в (PKI) інформаційної системи

10.1. О сновні 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).

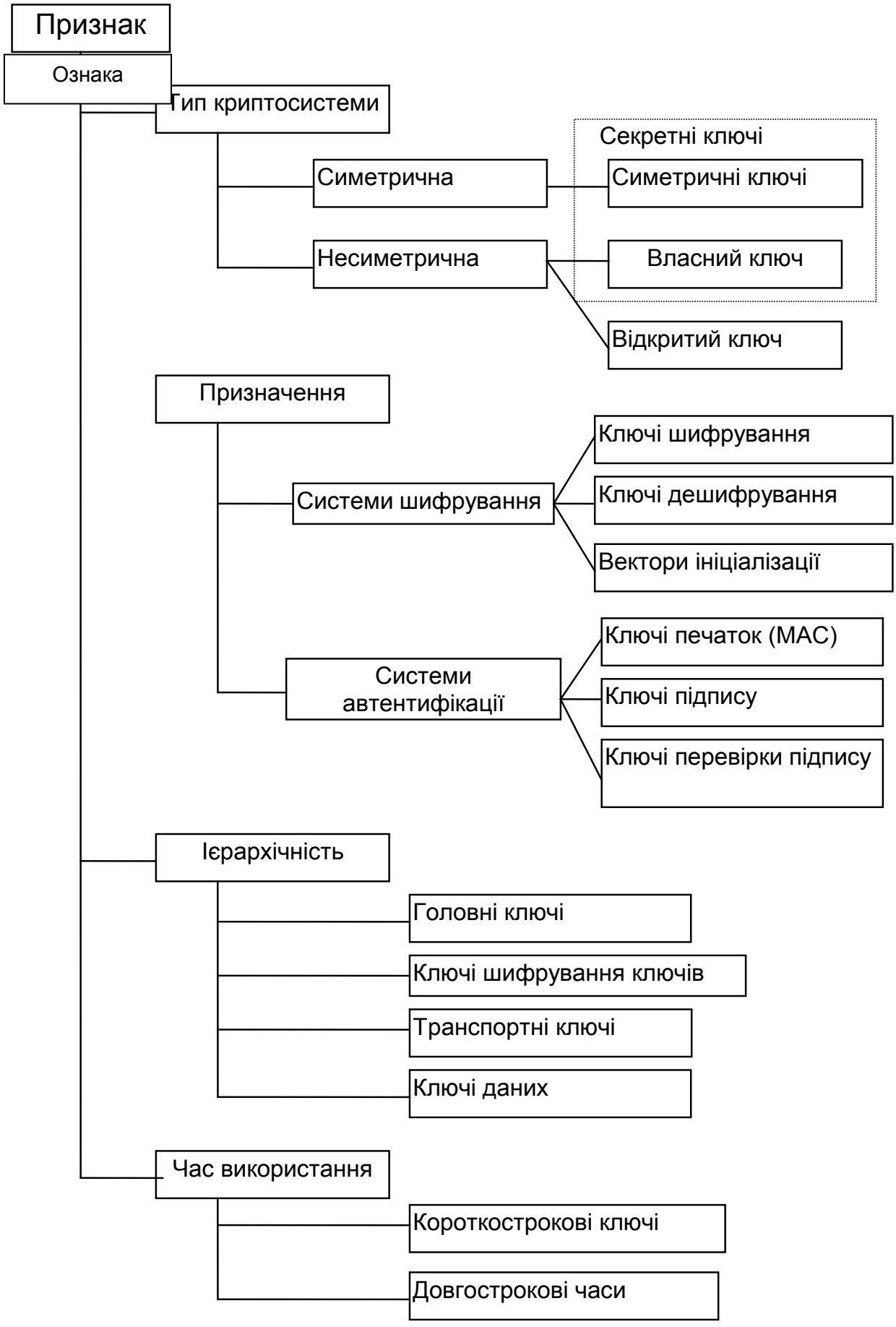


Рис. 10.1. **Класифікація ключів**

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.

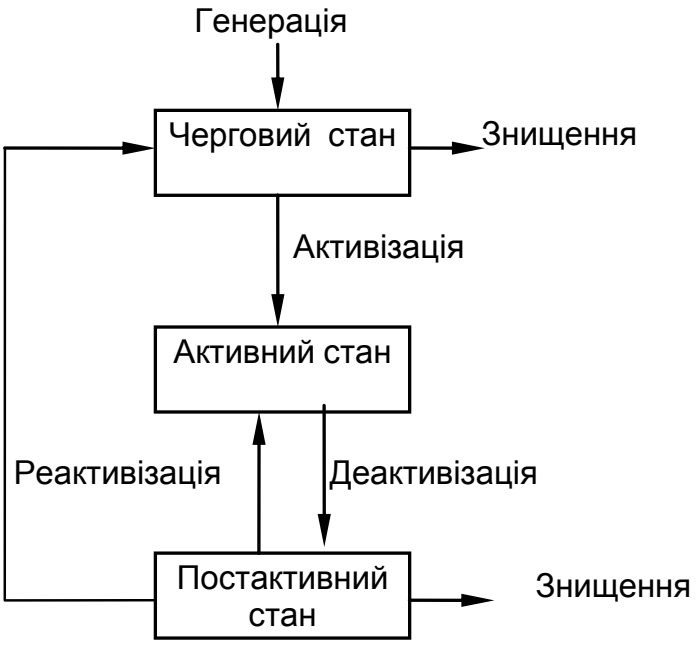


Рис. 10.2. **Взаємозв'язок основних і перехідних станів**

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.

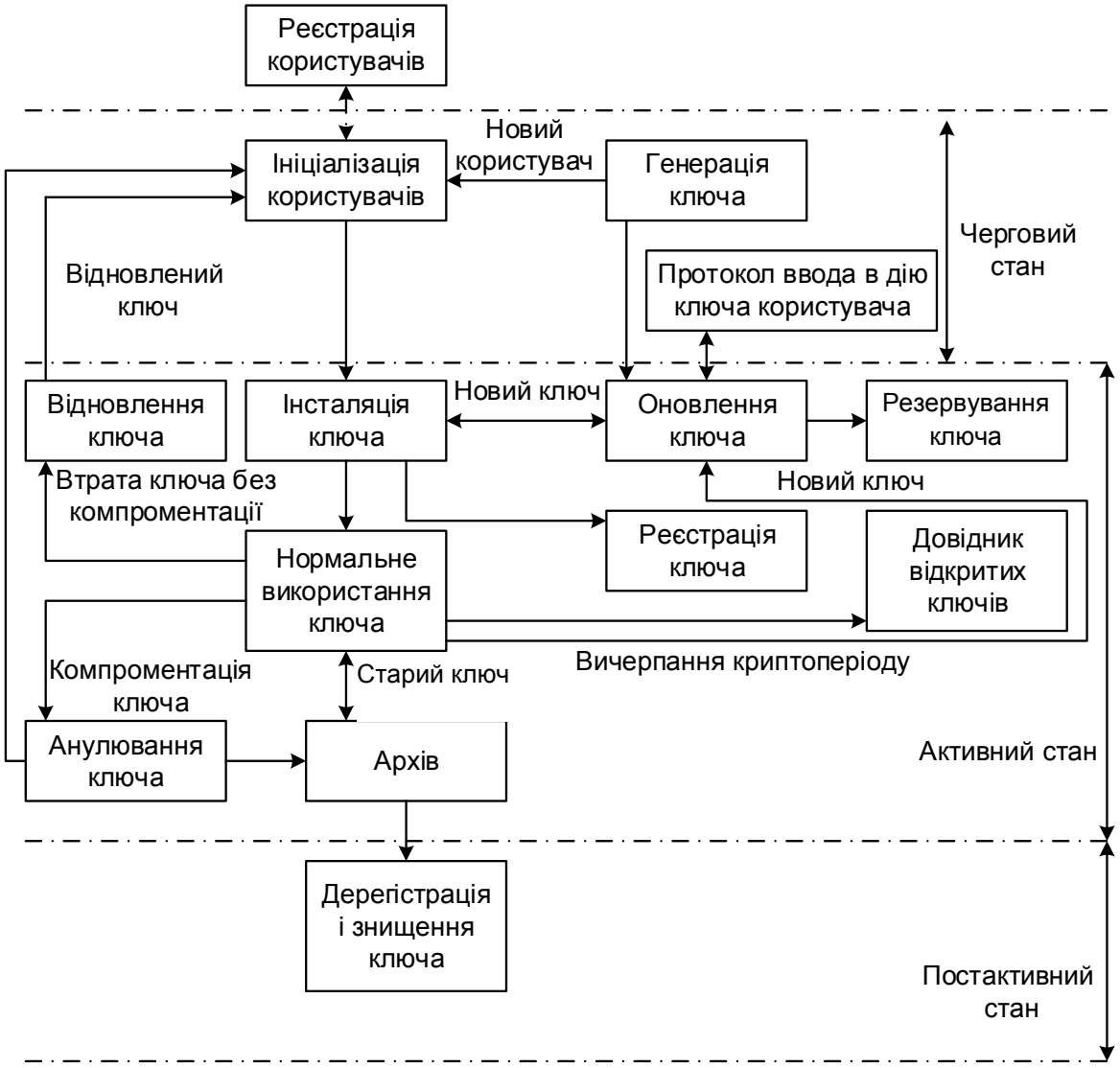


Рис. 10.3. **Життєвий цикл керування ключами**

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 АС. 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 $К_{AB}$ 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 $К_{AB}$.

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 $К_{AB}$ 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.

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).

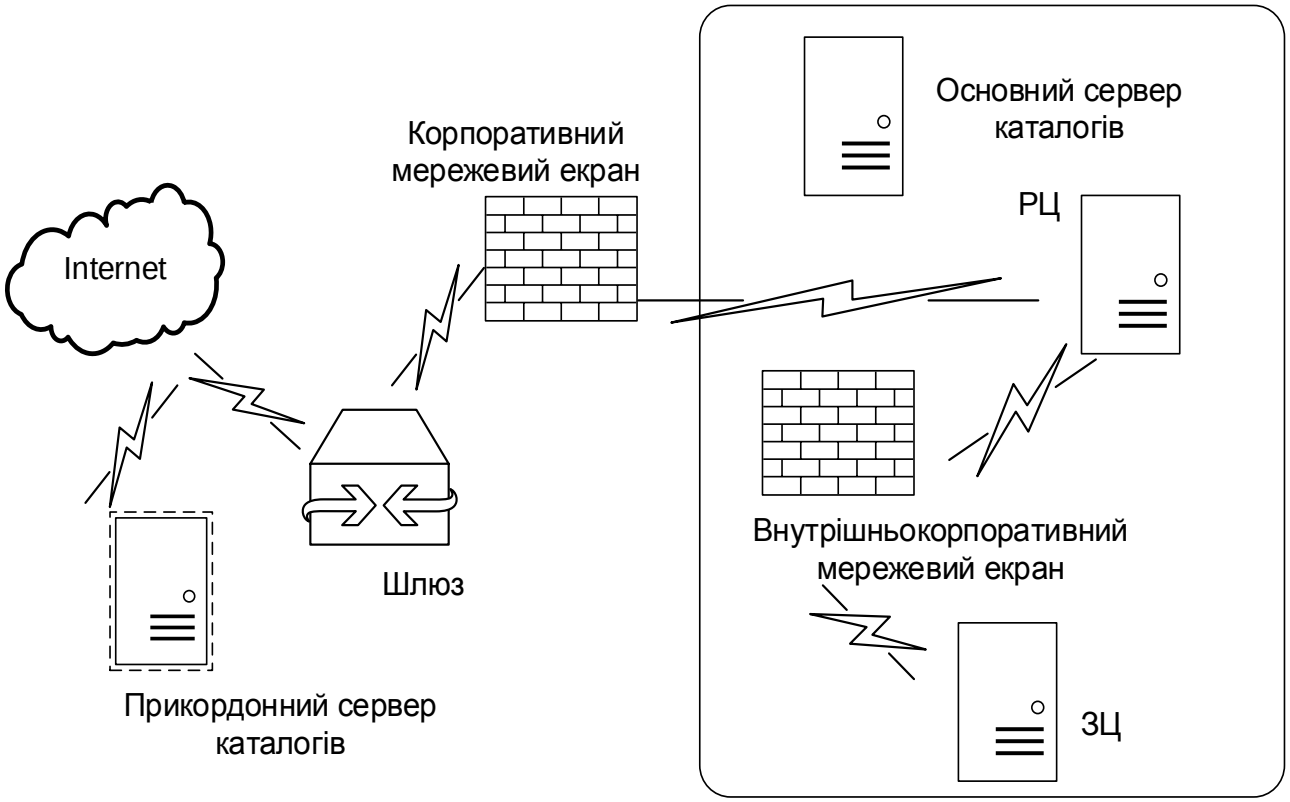


Рис. 10.5. **Фізична топологія PKI**

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.

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. 10.6. 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.

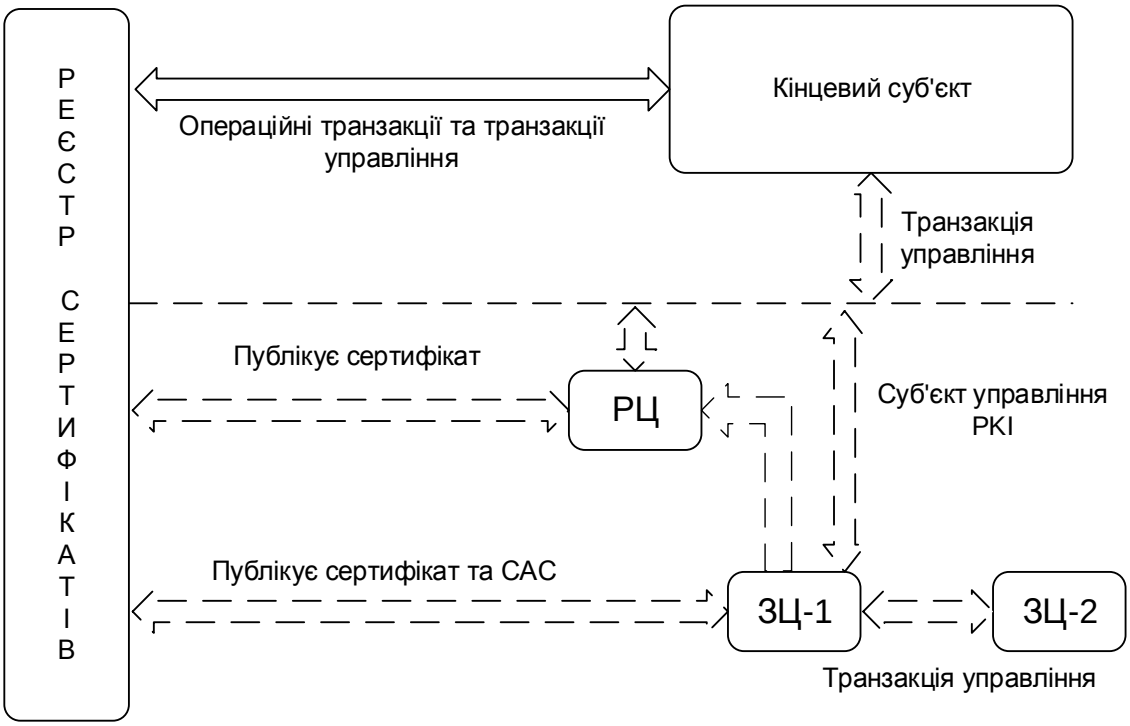


Рис. 10.6. **Схема функціонування PKI**

На рис. 10.6 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 серти-фіката 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 САС з використанням протоколів LDAP, HTTP і FTP. Наприклад, пошук САС 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.

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. 10.7 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. 10.7. **Життєвий цикл сертифіката**

Зразкові сценарії управління життєвим циклом сертифіка-тів 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. 10.8 секретний ключ використовується для підписання ділових контрактів.

Рис. 10.8. **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. 10.9 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.

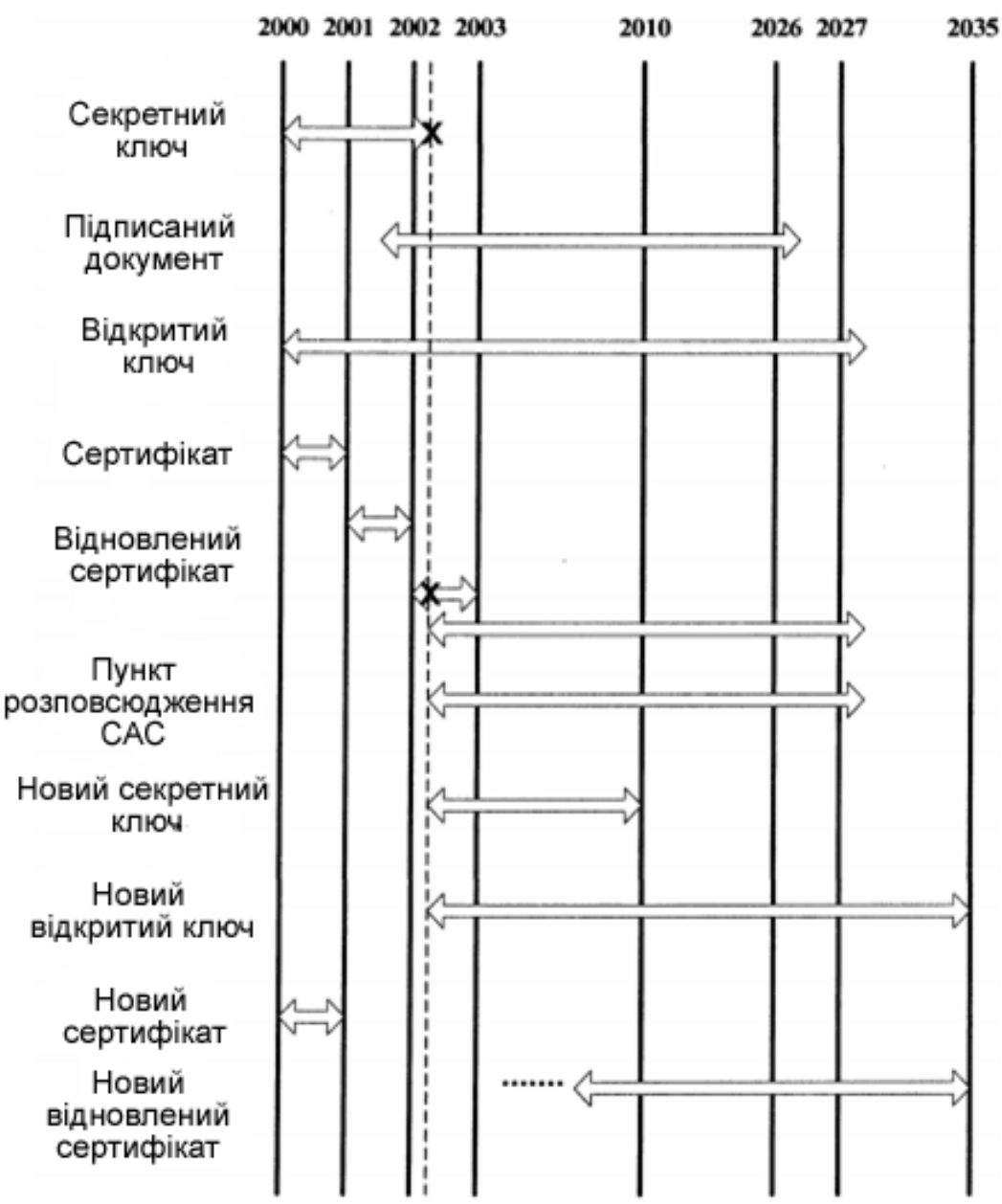


Рис. 10.9. **Компрометація особистого ключа підпису**

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.

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.doc-stoc.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/.