Факультет комп'ютерних наук та кібернетики

Київського національного університету імені Тараса Шевченка

О.В. Галкін, О.С. Шкільняк

Основи криптології

Навчальний посібник

Київ – 2023

УДК 003.26:004.056.55

Г16

Рецензенти:

д-р фіз.-мат. наук, проф. О.О. Марченко,

к-т фіз.-мат. наук, доц. Л.Л. Омельчук.

Рекомендовано до друку вченою радою
факультету комп'ютерних наук та кібернетики
(протокол №5 від 27 листопада 2023 року)

Галкін О.В., Шкільняк О.С.

Г16 Основи криптології: навчальний посібник / Галкін О.В., Шкільняк О.С. – Київ, 2023. – 119 с.

В посібнику викладено основні підходи та методи сучасної криптографії для розв'язання задач, що виникають в процесі обробки, збереження та передачі інформації. Подано попередні математичні відомості, розглянуто симетричні шифри, криптосистеми з відкритим ключем та атаки на них, проблему розподілу ключів, криптографічні хеш-функції і схеми цифрового підпису, наведено приклади використання криптоалгоритмів.

Для студентів, які вивчають дисципліну "Основи криптології", а також навчаються за освітніми програмами, пов'язаними з інформаційними та комп'ютерними технологіями.

УДК 003.26:004.056.55

Зміст

Частина 1. Вступ. Історичні шифри

1.1 Поняття шифрування

Алгоритм шифрування (або шифр) – це переведенння відкритого тексту в текст зашифрований (або шифротекст, шифрограму, криптограму) за допомогою секретного ключа. Цей процес називають шифруванням. Будемо писати

$$C = E_k(m),$$

де m – **відкритий текст**, E – **шифруюча** функція, k – **секретний ключ** і C – **шифротекст**.

Зворотний процес називають розшифруванням та пишуть

$$m = D_k(C).$$

Зауважимо, що алгоритми шифрування та розшифрування E та D відкриті, і секретність вихідного тексту m в даному шифротексті C залежить від секретності ключа k.

1.2 Шифр зсуву

Це один із найпростіших шифрів. Процес шифрування полягає у заміні кожної літери на іншу, зміщену від вихідної на певну кількість позицій в алфавіті залежно від значення ключа.

Так, наприклад, якщо ключ дорівнює 3, то літера «A» вихідного тексту в шифруванні зображується через «D», замість літери «B» з'явиться «E» тощо. Слово «HELLO» буде зашифроване як «KHOOR». Коли цей шифр використовуються із ключем, що дорівнює трьом, його часто називають шифром Цезаря.

1.3 Шифр заміни

Основний недолік шифру зсуву полягає в тому, що існує дуже мало можливих ключів, всього 25. З метою усунення зазначеного недоліку було винайдено шифр заміни. Розглянемо цей шифр.

Випишемо спочатку алфавіт, а безпосередньо під ним – той самий алфавіт, але з переставленими літерами:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
G O Y D S I P E L U A V C R J W X Z N H B Q F T M K
```

Шифрування полягає в заміні кожної літери у відкритому тексті на відповідну нижню літеру.

Щоб розшифрувати текст, потрібно кожну його літеру знайти у нижньому рядку таблиці та замінити її відповідною верхньою. Таким чином, криптограма слова «hello» виглядатиме як «ESVVJ».

Кількість всіх можливих ключів такого шифру збігається з числом різних перестановок 26 елементів.

1.4 Шифр Віженера

Основним недоліком шифрів зсуву є те, що кожна літера відкритого тексту при шифруванні остаточно замінюється фіксованим символом. Значить, при зламі ефективно може бути використана статистика відповідної мови щодо вживаності літер.

Один із шляхів подолання зазначеної проблеми полягає в тому, щоб брати кілька наборів символів замість стандартного алфавіту та шифрувати літери відкритого тексту, вибираючи відповідні знаки з різних наборів у певній послідовності.

Одним із шифрів, що реалізують цей прийом, є шифр Віженера. Як і у випадку шифру зсуву, ми знову перенумеруємо літери, починаючи з 0.

Секретний ключ тут – це коротка послідовність букв (тобто слово, часто назване **гаслом**), яке повторюється знову і знову, формуючи потік ключів.

Кодування полягає у додаванні літер відкритого тексту з буквами потоку ключів (які сприймаються як числа). Наприклад, якщо ключем є слово *sesame*, то шифрування виглядатиме так:

```
   t h i s i s a t e s t m e s s a g e
+
   s e s a m e s e s a m e s e s a m e
=
   L L A S U W S X W S F Q W W K A S I
```

Шифр Віженера не дуже складно зламати. Існує спосіб, за допомогою якого можна визначити довжину гасла, що генерує потік ключів. Цей спосіб називають тестом Казіскі. Його ідея ґрунтується на періодичності потоку ключів. Крім того, в природній мові існують найпоширеніші буквосполучення: біграми і триграми.

Тому при зломі шифру Віженера можна застосовувати статистичний аналіз.

1.5 Шифр «Енігма»

В певний момент виникла необхідність механізувати процес шифрування. Брався повний диск з нанесеними на нього з двох сторін контактами, що відповідають алфавітам відкритого і шифрованого текстів, причому контакти з'єднувалися по деякій підстановці, названій комутацією диску. Комутація визначала заміну літер в початковому кутовому положенні. При зміні кутового положення змінювалася і підстановка на спряжену. Звідси назва механізму – **ротор**, або **роторна машина**.

Розглянемо найпростішу версію роторної машини «Енігма», яка має три комутаційні диски. Припустимо, що комутація першого диску задає підстановку

a b c d e f g h i j k l m n o p q r s t u v w x y z

T M K G O Y D S I P E L U A V C R J W X Z N H B Q F

яка реалізується в початковому кутовому положенні.

Тоді перша літера повідомлення замінюється відповідно до цієї підстановки.

Перед шифруванням другої літери відкритого тексту комутаційний диск повертається однією позицію, і виходить інше правило заміщення:

a b c d e f g h i j k l m n o p q r s t u v w x y z

M K G O Y D S I P E L U A V C R J W X Z N H B Q F T

Для третьої літери використовується чергова підстановка:

a b c d e f g h i j k l m n o p q r s t u v w x y z

K G O Y D S I P E L U A V C R J W X Z N H B Q F T M

І так для всіх трьох дисків.

В «Енігмі» при кожному повороті першого ротора з'єднане з ним кільце потрапляє в паз другого диска і штовхає його. Аналогічно покрокові ітерації третього ротора контролюються другим ротором.

Обидва кільця рухливі, та їх положення теж формують частину простору ключів. Зрештою, для подвійної заміни літер при кожному шифруванні використовувалася штекерна панель, що збільшувало складність.

Таким чином, у формуванні простору ключів брали участь комутації дисків, порядок їх компонування, початкові кутові положення дисків та положення кілець. В результаті загальна кількість секретних ключів сягала до 2^{75}.

Для контролю відповідності операцій шифрування та розшифрування використовувався так званий рефлектор, в ролі якого виступала фіксована відкрита підстановка, задана таблицею

a b c d e f g h i j k l m n o p q r s t u v w x y z

Y R U H Q S L D P X N G O K M I E B F Z C W V J A T

Спрощена схема «Енігми» зображена на рис. 1. Червоними лініями показано шлях, яким літера «A» відкритого тексту замінюється на «D» шифротексту. Шифрування і розшифрування має здійснюватися машинами, що знаходяться в однаковому положенні.

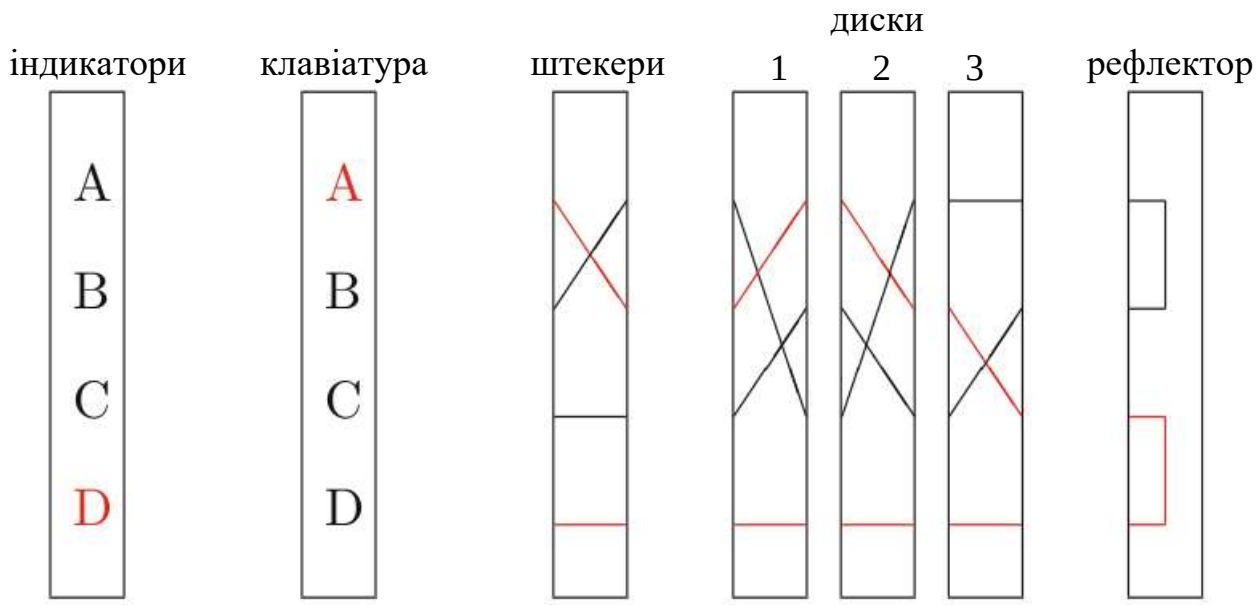


Рисунок 1. – Спрощена машина «Енігма».

Під час експлуатації «Енігми» щодня змінювали наступні установки:

- розташування штекерів,
- комутаційні диски та їх компонування,
- позиції кілець,
- початкові кутові положення дисків.

Помилою при експлуатації цієї машини було те, що з кожним повідомленням не змінювали ключ.

Для атаки на «Енігму» можна використовувати так званий **індекс збігу**:

$$IC = \sum_{i=A}^{Z} \frac{f_i \cdot (f_i - 1)}{n \cdot (n - 1)},$$

(якщо $x_1, \ldots, x_n$ – рядок літер, то $f_1, \ldots, f_{25}$ – числа появи літер у рядку).

Варто зауважити, що для випадкового набору букв цей індекс дорівнює $\mathrm{IC}(x) \approx 0{,}038$, тоді як для осмисленого тексту англійською чи німецькою мовою він дорівнює $\mathrm{IC}(x) \approx 0{,}065$.

Отже, атака на Енігму виглядає наступним чином:

1. Знайдемо порядок комутаційних дисків. Звільнимо панель від штекерів та встановимо диски у позицію «a, a, a». Перебираючи всі кутові положення дисків та їх взаємні розташування, визначимо ту комбінацію, яка дасть найвищий індекс збігу IC для перетвореного тексту. На це потрібно $60 \cdot 263 \approx 220$ операцій розшифрування.

2. Апроксимуємо початкові кутові положення дисків. Розглянемо стартові кутові позиції. На цьому етапі штекерна панель залишається порожньою, а диски знову встановлюються в позицію «a, a, a» з дотриманням порядку, знайденого на попередньому кроці. Проходимо через усі положення кожного з трьох комутаційних дисків та першого кільця, розшифровуючи повідомлення при кожній комбінації. Знову прагнемо наблизити коефіцієнт IC до максимального значення. При цьому відбувається $26^4 \approx 2^{19}$ розшифрувань.

3. Визначення початкових кутових положень. До цього моменту відомий як порядок слідування дисків, так і початкові положення першого кільця і першого

ротора. Крім того, відомі приблизні початкові положення інших дисків. Тепер перебираємо всі положення другого кільця та другого диска, повторюючи попередні дії. Це вимагатиме $26^2 = 2^9$ операцій. В результаті відомі точні положення другого кільця та другого комутаційного диска. Аналогічна процедура проводиться і для третього диска.

4. Визначення позицій штекерів. Штекери підбираємо по черзі, поки не зможемо прочитати шифротекст. Це можна зробити за допомогою індекса збігу *IC* (що вимагає дуже великого шифротексту) або статистичного тексту на основі інформації про розподіл триграм відповідної мови.

Частина 2. Математичний вступ до криптографії

2.1 Арифметика залишків

Розглянемо основні математичні поняття та факти, необхідні для розуміння сучасної криптографії.

Арифметика залишків служить основою зокрема криптосистем з відкритим кодом.

Зафіксуємо додатне натуральне число N, яке назвемо **модулем**. Якщо різниця двох цілих чисел $b - a$ ділиться на N націло, тоді пишуть $a \equiv b (\mathrm{mod}\ N)$ і кажуть, що числа a і b можна порівняти за модулем N.

Очевидно, в цьому випадку числа a та b мають однакову остачу від ділення на N. Звідси і походить поняття «арифметики залишків», або «модульної арифметики».

Далі будемо писати просто $a \equiv b$, якщо зрозуміло, за яким модулем N відбувається порівняння.

Будемо також використовувати (mod N) для позначення **оператора модуля** на множині цілих чисел. Наприклад,

$$18 \ (\mathrm{mod}\ 7) = 4,$$

$$-18 (\mathrm{mod}\ 7) = 3.$$

Всі можливі залишки від ділення чисел на N утворюють множину

$$\mathbb{Z}/N\mathbb{Z} = \{0, \dots, N-1\}.$$

Зрозуміло, $\mathbb{Z}/N\mathbb{Z}$ – множина значень оператора модуля (mod N).

Оскільки всі порівнянні між собою за модулем N цілі числа мають одну й ту саму остачу, можемо вважати, що елемент із $\mathbb{Z}/N\mathbb{Z}$ представляє цілий клас чисел вигляду $x + kN$, де $k = 0, -1, 1, -2, 2, \dots$.

На множині $\mathbb{Z}/N\mathbb{Z}$ є дві основні операції: додавання та множення.

Приклад додавання: (11 + 13) (mod 16) = 24 (mod 16) = 8.

Приклад множення: (11 · 13) (mod 16) = 143 (mod 16) = 15.

Наведемо властивості операцій над залишками.

1. Замкненість додавання:

для будь-яких a, b із $\mathbb{Z}/N\mathbb{Z}$: $a + b$ теж належить $\mathbb{Z}/N\mathbb{Z}$.

2. Асоціативність додавання:

для будь-яких a, b із $\mathbb{Z}/N\mathbb{Z}$: $(a + b) + c = a + (b + c)$.

3. Нуль є одиничним елементом (або **одиницею**) за додаванням:

для будь-якого a із $\mathbb{Z}/N\mathbb{Z}$: $a + 0 = 0 + a = a$.

4. Завжди існує **обернений** елемент за додаванням:

для будь-якого елемена a із $\mathbb{Z}/N\mathbb{Z}$: $a + (N - a) = (N - a) + a = 0$.

5. Комутативність додавання:

для будь-яких a, b із $\mathbb{Z}/N\mathbb{Z}$: $a + b = b + a$.

6. Замкнененість множення:

для будь-яких a, b із $\mathbb{Z}/N\mathbb{Z}$: $a \cdot b$ належить $\mathbb{Z}/N\mathbb{Z}$.

7. Асоціативність множення:

для будь-яких a, b, c із $\mathbb{Z}/N\mathbb{Z}$: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

8. Число 1 є одиничним елементом (одиницею) за множенням:

для будь-якого a із $\mathbb{Z}/N\mathbb{Z}$: $a \cdot 1 = 1 \cdot a = a$.

9. Множення та додавання пов'язані законом дистрибутивності:

для будь-яких a, b із $\mathbb{Z}/N\mathbb{Z}$: $(a + b) \cdot c = a \cdot c + b \cdot c$.

10. Комутативність множення:

для будь-яких a, b із $\mathbb{Z}/N\mathbb{Z}$: $a \cdot b = b \cdot a$.

2.2 Групи та кільця

Групою називається множина з операцією, яка:

- замкнена,
- має одиницю,
- асоціативна,
- щодо неї кожен елемент має обернений.

Групу з комутативною операцією називають **комутативною**, або **абелевою**. Майже всі групи, що зустрічаються в криптографії, – абелеві.

Розглянемо приклади груп:

Цілі, дійсні чи комплексні числа за додаванням. Одиницею є звичайний 0, а оберненим до x служить $-x$, оскільки $x + (-x) = 0$.

Ненульові раціональні, дійсні чи комплексні числа. Одиниця в них – це 1, і оберненим до x буде $1 / x$, оскільки $x \cdot (1 / x) = 1$

Усі дійсні квадратні матриці $N \times N$ з визначником не рівним 0 утворюють групу GL(N). Груповою операцією є операція множення, одиницею – одинична матриця, оберненим елементом – обернена матриця. Це приклад не абелевої групи.

Група називається **мультиплікативною**, якщо груповою операцією є множення елементів, тобто $f = g \cdot h$. Таку групу зазвичай позначають (G, $\cdot$).

Групу називають **адитивною**, якщо її операцією є додавання: $f = g + h$. Для адитивних груп використовують позначення (G, +).

Абелева група називається **циклічною**, якщо в ній є особливий елемент (**твірна**), що будь-який інший елемент групи отримується багатократим застосуванням до нього групової операції.

Наприклад, у групі цілих чисел за додаванням будь-який елемент можна отримати в результаті багатократного додавання 1 (або –1) до себе.

Якщо g – твірна циклічної групи G, то пишуть $G = \langle g \rangle$.

У мультиплікативному випадку кожен елемент h групи G можна записати у вигляді $h = g^x$, а в адитивному – $h = x \cdot g$, де x в обидвох випадках – деяке ціле число, яке називають дискретним логарифмом елемента h за основою g.

Кільцем назвемо множину R з двома операціями, додаванням і множенням, що традиційно позначаються «+» і «·», які мають перелічені вище властивості 1–9. Позначається кільце трійкою (R, $\cdot$, +).

Якщо також виконується властивість 10, то кільце називають **коммутативним**.

Очевидно, що множина залишків $\mathbb{Z}/N\mathbb{Z}$ з операціями додавання та множення є комутативним кільцем, яке часто називають кільцем лишків за модулем N.

2.3 Центральна задача арифметики залишків та функція Ейлера

Центральною задачею арифметики залишків є розв'зок рівняння

$$a \cdot x = b \pmod{N},$$

тобто пошук елемента x, що належить $\mathbb{Z}/N\mathbb{Z}$, який задовольняє цю рівність.

Існує простий критерій, що дозволяє визначити, чи дане рівняння не має жодного, має один чи багато розв'язків за допомогою обчислення найбільшого спільного дільника.

Якщо НСД (a, N) = 1, тоді існує єдиний розв’язок. Його можна знайти за допомогою допоміжного числа c, що задовольняє рівняння $a \cdot c = 1 \pmod{N}$, після чого невідомий шуканий x обчислюється за формулою $x = b \cdot c \pmod{N}$.

Якщо НСД (a, N) $\neq 1$ і g = НСД (a, N) ділить b, тоді рівняння матиме g розв'язків. Щоб їх знайти, потрібно розділити вихідне рівняння на g:

$$a' \cdot x' = b' (\mathrm{mod}\ N'),$$

де $a' = a/g$, $b' = b/g$ та $N' = N/g$. Якщо x' – розв’язок отриманого рівняння, то розв’язком вихідного буде число вигляду

$$x = x' + i \cdot N',$$

де $i = 0,1, \dots, g - 1$.

В інших випадках розв’язків не існує.

Говорять, що числа a та N **взаємно прості**, якщо НСД (a, N) = 1.

Кількість елементів кільця $\mathbb{Z}/N\mathbb{Z}$, взаємно простих з N, задається функцією Ейлера і дорівнює $\varphi(N)$. Значення $\varphi(N)$ можна знайти за розкладанням числа N на прості множники. Якщо

$$N = \prod_{i=1}^{n} p_i^{e_i},$$

де p_i – різні прості числа, то

$$\varphi(N) = \prod_{i=1}^{n} p_i^{e_i-1}(p_i - 1).$$

Особливий інтерес становлять такі окремі часткові випадки.

Якщо $N = p$ – просте, то $\varphi(p) = p - 1$.

Якщо p та q – прості числа і $p \neq q$, то $\varphi(p \cdot q) = (p - 1)(q - 1)$.

2.4 Мультиплікативні обернені за модулем

Розв'язуючи рівняння вигляду $ax = b \pmod N$, приходимо до питання існування мультиплікативної оберненої в числа a по модулю N. Його буде природно позначити через a^{-1}.

Особливо цікавим є випадок простого модуля $N = p$, оскільки при цьому для будь-якого ненульового елемента, що належить $\mathbb{Z}/N\mathbb{Z}$, знайдеться єдиний розв'язок рівняння $ax = 1 \pmod p$.

Отже, якщо p – просте число, то будь-який ненульовий елемент $\mathbb{Z}/N\mathbb{Z}$ має мультиплікативну обернену. Кільця з такою властивістю називаються полями.

Полем називається множина $(F, \cdot, +)$ з двома операціями, що мають додаткові властивості:

- $(F, +)$ – абелева група з одиничним елементом 0,
- $(F \setminus \{0\}, \cdot)$ – абелева група з одиничним елементом 1,
- $(F, \cdot, +)$ задовольняє закону дистрибутивності.

Інакше кажучи, поле – це комутативне кільце, у якому кожний ненульовий елемент оборотний. Приклад поля – поле множини раціональних чисел.

Визначимо множину оборотних елементів $\mathbb{Z}/N\mathbb{Z}$ як

$$(\mathbb{Z}/N\mathbb{Z})^* = \{x \in \mathbb{Z}/N\mathbb{Z} \mid \text{НСД}(x, N) = 1\}.$$

Для загального кільця A позначення A^* закріплено за найбільшою його підмножиною елементів, які утворюють групу по множенню.

Можна перевірити, що множина $(\mathbb{Z}/N\mathbb{Z})^*$ є групою по множенню з кількістю елементів $\varphi(N)$.

У спеціальному випадку, коли $N = p$ – просте число, отримуємо:

$$(\mathbb{Z}/N\mathbb{Z})^* = \{1, \dots, p-1\},$$

оскільки кожен ненульовий елемент кільця $\mathbb{Z}/p\mathbb{Z}$ взаємно простий з p і тому оборотний.

Іншими словами, $\mathbb{Z}/p\mathbb{Z}$ є скінченним полем, яке зазвичай називається **полем лишків за модулем** p і позначається символом $\mathbb{F}_p$.

Як випливає з визначення, мультиплікативна група F^* поля F збігається з множиною $F\backslash\{0\}$.

В частковому випадку поля залишків отримуємо

$$\mathbb{F}_p=\mathbb{Z}/p\mathbb{Z} = \{0, \dots, p-1\} \text{ та } = (\mathbb{Z}/p\mathbb{Z})^* = \{1, \dots, p-1\}.$$

Розглянемо важливу теорему теорії скінченних груп.

Теорема Лагранжа. Якщо $(G, \cdot)$ є групою порядку (що має кількість елементів) $n = \#G$, то кожен її елемент a із G задовольняє співвідношенню $a^n = 1$.

Таким чином, для x із $(\mathbb{Z}/N\mathbb{Z})^*$ виконується рівність

$$x^{\varphi(N)} = 1 \pmod N,$$

оскільки $\#(\mathbb{Z}/N\mathbb{Z})^* = \varphi(N)$.

Наслідком теореми Лагранжа є наступна теорема.

Мала теорема Ферма. Припустимо, що число p просте і a належить $\mathbb{F}_p^*$; тоді

$$a^p = 1 \pmod N.$$

2.5 Скінченні поля

Розглянемо інший приклад скінченного поля. Візьмемо множину многочленів від змінної x із коефіцієнтами з $\mathbb{F}_p$.

Ця множина позначається через $\mathbb{F}_p[x]$ і утворює кільце щодо природних операцій суми та множення многочленів.

Як приклад розглянемо цікавий випадок $p = 2$.

В кільці $\mathbb{F}_2[x]$ виконуються рівності:

$$(1 + x + x^2) + (x + x^3) = 1 + x^2 + x^3,$$

$$(1 + x + x^2) \cdot (x + x^3) = x + x^2 + x^4 + x^6.$$

Можна зафіксувати многочлен $f(x)$ та розглядати інші елементи кільця $\mathbb{F}_p[x]$ за модулем $f(x)$, тобто оперувати остачами від ділення многочленів на $f(x)$.

Як і натуральні числа по модулю N, можливі залишки утворюватимуть кільце. Воно позначається через

$$\mathbb{F}_p[x]/f(x)\mathbb{F}_p[x] \text{ або } \mathbb{F}_p[x]/(f(x)).$$

Наприклад, нехай $f(x) = x^4 + 1$ і $p = 2$. Тоді

$$(1 + x + x^2) \cdot (x + x^3) \pmod{x^4 + 1} = 1 + x^2,$$

оскільки

$$x + x^2 + x^4 + x^5 = (x + 1) \cdot (x^4 + 1) + (1 + x^2).$$

Слід пам’ятати, що всі коефіцієнти розглядаються за модулем 2.

Нагадаємо, що основне рівняння арифметики залишків мало вигляд

$$ax = b \pmod{N}.$$

Аналогічне рівняння можна написати і для випадку многочленів. Нехай a, b та f – многочлени з $\mathbb{F}_p[x]$. Треба знайти розв’язок рівняння

$$a\alpha = b \pmod{f}$$

відносно α.

Тут, як і в цілих числах за модулем N, відповідь залежатиме від найбільшого спільного дільника многочленів a і f та трьох випадків. Найцікавіша ситуація виникає при НСД$(a, f) = 1$ – a та f взаємно прості.

Многочлен називається **незвідним**, якщо він не має дільників, відмінних від нього самого і констант (аналог простоти цілих чисел).

Припустимо, що $p = 2$ і розглянемо два незвідні многочлени:

$$f_1(x) = x^7 + x + 1 \text{ та } f_2(x) = x^7 + x^3 + 1.$$

Таким чином, виникають два скінченні поля

$$F_1 = \mathbb{F}_2[x]/(f_1(x)) \text{ та } F_2 = \mathbb{F}_2[x]/(f_2(x)),$$

кожне з яких складаються з 2^7 двійкових многочленів зі степінню не більше 6.

Додавання в обох полях виглядає однаково, оскільки при обчисленні суми складаються коефіцієнти многочленів за модулем 2.

Але множення має відмінності:

$$(x^3 + 1) \cdot (x^4 + 1) \pmod{f_1(x)} = x^4 + x^3 + x,$$

$$(x^3 + 1) \cdot (x^4 + 1) \pmod{f_2(x)} = x^4.$$

Розглянемо питання ізоморфізму полів F_1 та F_2. Розпочнемо із визначення ізоморфізму.

Кажуть, що поля F_1 та F_2 ізоморфні, якщо існує відображення $\varphi: F_1 \to F_2$, яке називають **ізоморфізмом**, що задовольняє двом умовам:

$$\varphi(\alpha + \beta) = \varphi(\alpha) + \varphi(\beta) \text{ та } \varphi(\alpha \cdot \beta) = \varphi(\alpha) \cdot \varphi(\beta).$$

Неважко побачити, що ізоморфізм існує між двома скінченними полями з однаковою кількістю елементів. Зокрема, він існує між нашими полями F_1 та F_2.

Отже, всі скінченні поля фактично збігаються або з цілими числами за простим модулем, або з многочленами за модулем незвідного многочлена (який теж можна називати простим).

Таким чином маємо фундаментальну теорему:

Теорема. Існує єдине (з точністю до ізоморфізму) скінченне поле з кількістю елементів, що дорівнює степеню простого числа.

Позначатимемо поле з $q = p^d$ елементів символом $\mathbb{F}_q$ або GF(q). Позначення GF(q) означає поле Галуа з q елементів.

Будь-яке скінченне поле K містить в собі екземпляр поля цілих чисел за деяким простиму модулем p. Це поле називається **простим підполем** поля K.

Число p елементів простого підполя називається **характеристикою** поля і позначається через char K. Зокрема, char $\mathbb{F}_p = p$.

2.6 Основні алгоритми

Розглянемо три основних алгоритми, які далі будуть часто використовуватися.

Алгоритм Евкліда. Знаходження найбільшого спільного дільника.

Нехай треба обчислити НСД чисел $r_0 = a$ та $r_1 = b$. Для цього обчислюємо r_2, r_3, r_4,... виконуючи ділення з остачею за наступною схемою:

$$r_2 = q_1 r_1 + r_0,$$

$$r_3 = q_2 r_2 + r_1,$$

$$\ldots\ldots\ldots\ldots$$

$$r_m = q_{m-1} r_{m-1} + r_{m-2},$$

$$r_{m+1} = q_m r_m.$$

Очевидно, що якщо число d ділить як a, так і b, то воно ділить й усі r_i, починаючи з $i = 0$ і закінчуючи $i = m$. Отже,

$$\text{НСД}(a, b) = \text{НСД}(r_0, r_i) = \ldots = \text{НСД}(r_{m-1}, r_m) = r_m.$$

Розглянемо приклад обчислення НСД (21, 12):

НСД (21, 12) = НСД (21 (mod 12), 12) =

= НСД (9, 12) = НСД (12 (mod 9), 9) =

= НСД (3, 9) = НСД (9 (mod 3), 3) = НСД (0, 3) = 3.

Розширений алгоритм Евкліда. За допомогою цього алгоритму можна знайти обернений елемент $a \pmod N$.

Розглянемо алгоритм на прикладі: обчислимо обернений елемент 7 за модулем 19.

Покладемо $r_0 = 7$, $r_1 = 19$. Тоді:

$r_2 = 5 = 19 - 2 \cdot 7$,

$r_3 = 2 = 7 - 5 = 7 - (19 - 2 \cdot 7) = -19 + 3 \cdot 7$,

$r_4 = 1 = 5 - 2 \cdot 2 = (19 - 2 \cdot 7) - 2 \cdot (-19 + 3 \cdot 7) = 3 \cdot 19 - 8 \cdot 7$.

Звідси

$$1 = -8 \cdot 7 \pmod{19},$$

так що

$$7^{-1} = -8 = 11 \pmod{19}.$$

Китайська теорема про залишки стверджує, що система рівнянь

$$x = a \pmod N,$$

$$x = b \pmod M$$

має єдиний розв'язок за модулем $M \cdot N$ тоді і тільки тоді, коли M та N взаємно прості.

Окрім того, вона пропонує простий метод пошуку цього розв'язку.

Наприклад, розв’язком системи

$$x = 4 \pmod 7,$$

$$x = 3 \pmod 5$$

буде $x = 18$.

Розглянемо алгоритм розв'язку. Якщо x задовольняє кожне з рівнянь, то для деякого цілого числа u мали б виконуватися рівності:

$$x = 4 + 7u \text{ та } x = 3 \pmod 5.$$

Підставляючи перше зі співвідношень у друге, отримуємо

$$4 + 7u = 3 \pmod 5.$$

Після перетворень, прийходимо до рівняння

$$2u = 7u = 3 - 4 = 4 \pmod 5.$$

Оскільки НСД(2, 5) = НСД(7, 5) = 1, останнє рівняння можна розв'язати відносно u. Насамперед обчислимо $2^{-1} \pmod 5 = 3$, оскільки $2 \cdot 3 = 6 = 1 \pmod 5$. Потім знайдемо значення

$$u = 2^{-1} \cdot 4 \pmod 5 = 3 \cdot 4 \pmod 5 = 2 \pmod 5.$$

Тепер підставимо знайдене значення u у вираз для x та отримаємо розв’язок

$$x = 4 + 7u = 4 + 7 \cdot 2 = 18.$$

Розглянемо загальний випадок теореми. Нехай $m_1, \ldots, m_r$ та $a_1, \ldots, a_r$ – цілі числа, причому всі m_i попарно взаємно прості. Потрібно знайти такий елемент x за модулем $M = m_1 m_2 \ldots m_r$, що

$$x = a_i \pmod{m_i} \text{ для всіх } i.$$

Китайська теорема про залишки гарантує існування та єдиність розв’язку та дає відповідь:

$$x = \sum_{i=1}^{r} a_i M_i y_i \pmod M, \text{ де}$$

$$M_i = M/m_i,\ y_i = M_i^{-1} \pmod{m_i}.$$

Як приклад, знайдемо число x за модулем $M = 1001 = 7 \cdot 11 \cdot 13$ таке, що

$$x = 5 \pmod 7,$$

$$x = 3 \pmod{11},$$

$$x = 10 \pmod{13}.$$

Тоді

$$M_1 = 143,\ y_1 = 5,$$

$$M_2 = 91,\ y_2 = 4,$$

$$M_3 = 77,\ y_3 = 12,$$

отримуємо розв'язок

$$x = \sum_{i=1}^{r} a_i M_i\, y_i (\mathrm{mod}\, m) =$$

$$= 715 \cdot 5 + 364 \cdot 3 + 924 \cdot 10 \ (\mathrm{mod}\ 1001) = 894.$$

2.7 Ймовірності

Випадковою величиною називається змінна x, що приймає свої значення з деякою ймовірністю. Якщо змінна x набуває значення 5 з ймовірністю 0,01, писатимемо

$$p(X = s) = 0{,}01.$$

Припустимо, що T – випадкова величина, що відображає результат підкидання симетричної монети. Оскільки поява орла чи решки рівноймовірні, то

$$p(T = \text{орел}) = 1/2,\ p(T = \text{решка}) = 1/2.$$

Якщо X – дискретна випадкова величина, то множина ймовірностей всіх її значень називають **розподілом ймовірностей**, а функцію $p(X = x)$, що співставляє значенню змінної відповідну ймовірність – **щільністю розподілу ймовірностей**.

В загальному випадку справджуються наступні характеристики:

$$p(X = x) \geq 0,\ \sum_x p(X = x) = 1.$$

Дві випадкові величини X і Y називаються **незалежними**, якщо для всіх можливих значень x та y має місце рівність

$$p(X = x,\ Y = y) = p(X = x) \cdot p(Y = y).$$

Умовною ймовірністю $p(X = x \mid Y = y)$ випадкових величин X і Y називається ймовірність того, що змінна X набуває значення x, якщо відомо, що $Y = y$.

Теорема Байєса. Якщо $p(Y = y) > 0$, то

$$p(X = x|Y = y) = \frac{p(X = x) \cdot p(Y = y|X = x)}{p(Y = y)} = \frac{p(X = x, Y = y)}{p(Y = y)}.$$

Для незалежних випадкових величин маємо

$$p(X = x \mid Y = y) = p(X = x),$$

тобто значення величини X не залежить від того, чому дорівнює випадкова величина Y.

Частина 3. Симетричне шифрування

3.1 Симетричні шифри

Основна вимога до сучасного симетричного шифру – кількість можливих ключів має бути дуже великою. Вважають, що обчислення, що складаються з 2^{80} кроків, у найближчі кілька років будуть нездійсненні. Тому ключ, що виключає злом простим перебором, має налічувати принаймні 80 бітів

Розглянемо спрощену модель шифрування рядка бітів (рис. 2). Таку модель легко реалізувати на практиці, оскільки для її реалізації необхідна одна з найпростіших комп'ютерних операцій – виключне АБО (XOR), тобто додавання за модулем 2 (також позначають ⊕).

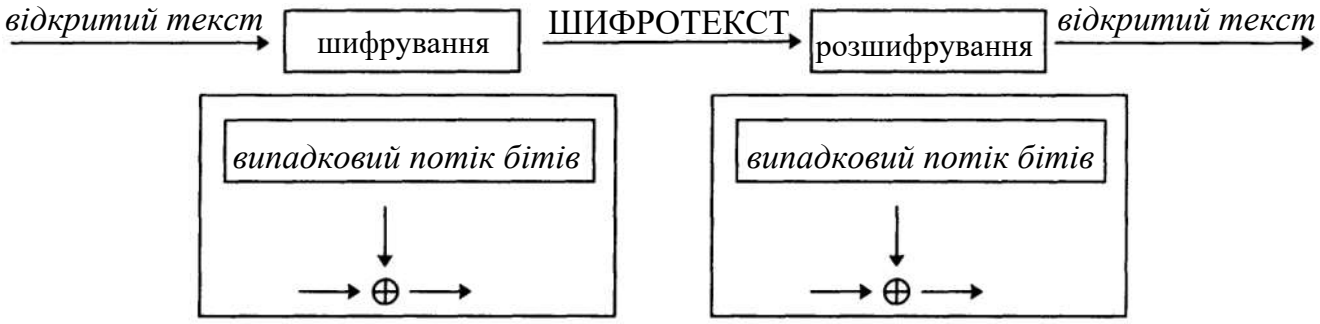


Рисунок 2. – Спрощена модель шифрування рядка бітів.

Шифруючи кожне нове повідомлення своїм ключем, довжина якого збігається із довжиною відкритого тексту, отримаємо абсолютно стійку симетричну криптосистему.

Більшість симетричних шифрів можна розділити на дві великі групи. Перша – потокові шифри, де за один раз обробляється один елемент даних (біт або літера), а друга – блокові шифри, в яких за один крок обробляється група елементів даних (наприклад, 64 біти).

3.2 Блокові шифри. Алгоритм DES

Схема симетричного блокового шифру має наступний вигляд (рис. 3).

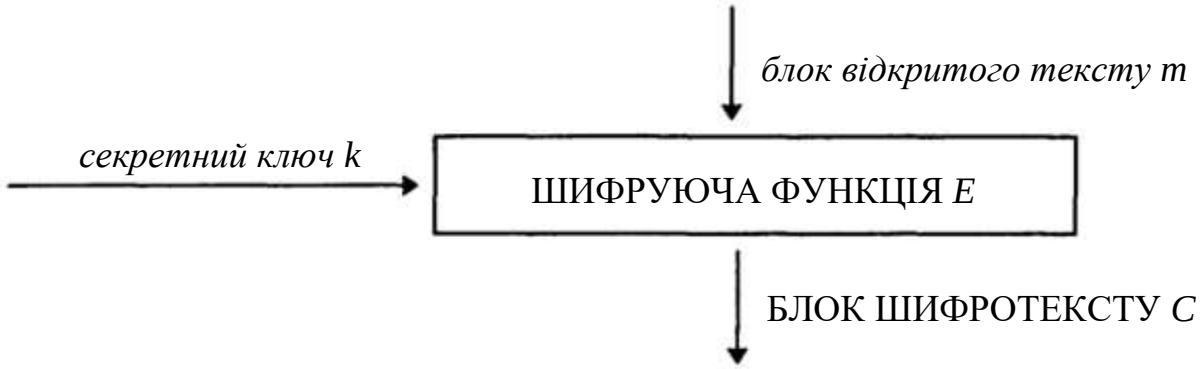


Рисунок 3. – Схема роботи блокового шифру.

Розмір блоку для шифрування зазвичай вибирають розумно великим.

Як перший приклад такого шифру розглянемо алгоритм DES (Data Encryption Standard) (був свого часу найпоширенішим симетричним шифром).

Шифр DES – варіант базового шифру Фейстеля. У стандартному варіанті DES використовується блок розміром 64 біт та ключ розміром 56 біт. Алгоритм має 16 раундів. Блоксхема алгоритму зображена на рис. 4.

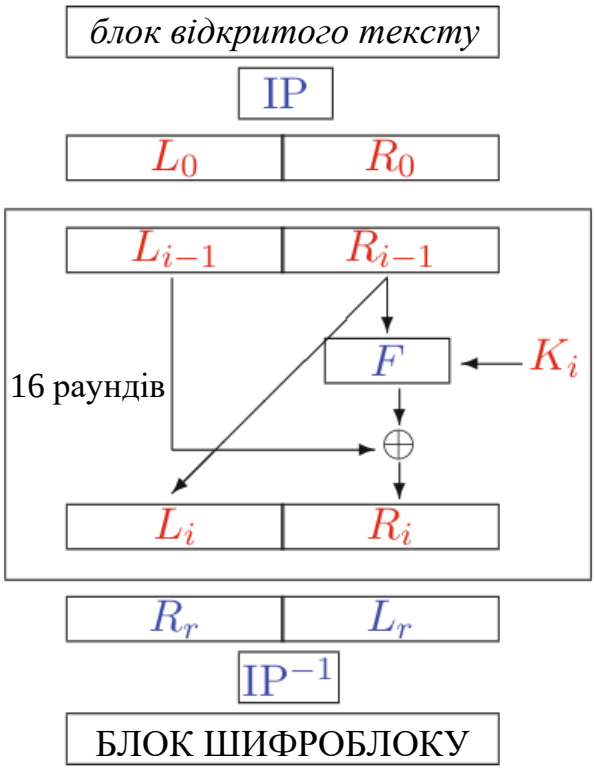


Рисунок 4. – Алгоритми DES і Фейстеля.

Тут F – раундова функція, K_i – раундові ключі, довжина кожного 48 біт.

Цікава особливість шифру в тому, що функція раунду оборотна незалежно від властивостей функції *F*. Оскільки кожен раунд шифрування здійснюється за правилом

$$L_i = R_{i-1};\ \ R_i = L_{i-1}\ \text{XOR}\ F(K_i, R_{i-1})$$

отже, розшифрування відбувається за рахунок перетворень:

$$R_{i-1} = L_i;\ \ L_{i-1} = R_i\ \text{XOR}\ F(K_i, L_i)$$

Шифр DES перетворює відкритий текст із 64 бітів наступним чином:

- здійснює початкову перестановку (IP);
- розщеплює блок на ліву та праву половини;
- здійснює 16 раундів з тим самим набором операцій;
- з'єднує половини блоку;
- здійснює скінченну перестановку.

Скінченна перестановка обернена до початкової. Це дозволяє використовувати те саме програмне забезпечення для шифрування і розшифрування.

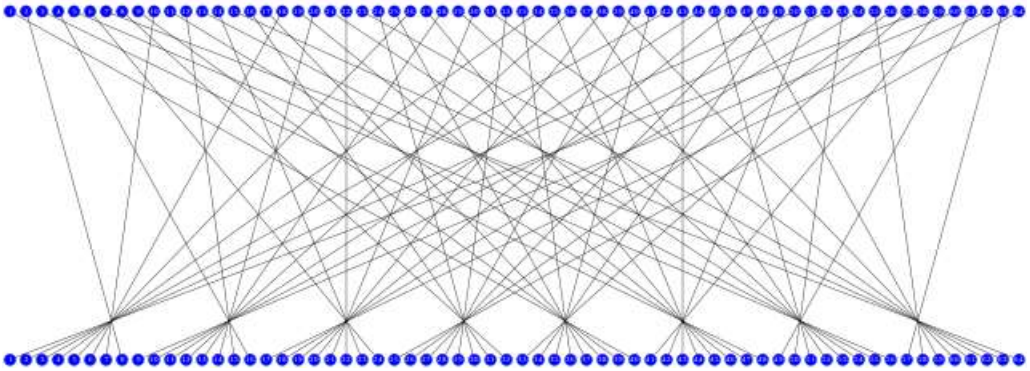
Розгортання ключа дає 16 підключів по 48 бітів кожен, виділяючи їх із 56-бітного основного ключа.

3.2.1 Основні операції алгоритму DES.

1. Початкова перестановка (initial permutation) IP. Початкова перестановка алгоритму DES визначається таблицею. (Цю та всі інші таблиці, що зображають перестановки, читають зліва направо та згори донизу.)

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

Так, число 58, розташоване в першому рядку та першому стовпці таблиці, означає, що IP переміщує 58 біт вхідних даних на перше місце.



2. Виконання 16 раундів із раундовою функцією *F* (від Feistel). Розглянемо роботу раундової функції *F*. Виконання цієї функції складається із 5 кроків.

Крок 1. Перестановка із розширенням. Права половина з 32 біт розтягується до 48 біт і перемішується.

Крок 2. Додавання до підключа. До рядка з 48 бітів, отриманому після перестановки з розширенням, і підключа (довжиною теж 48 бітів) застосовується операція виключного АБО, тобто кожна пара відповідних бітів складається за модулем 2.

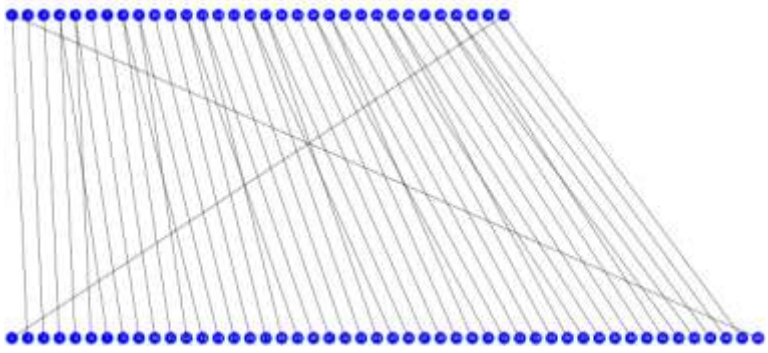
Крок 3. Розщеплення. Результат попереднього кроку розщеплюється на 8 частин по 6 бітів у кожному.

Крок 4. S-блок. Кожен 6-бітовий шматок передається в один із восьми S-блоків (блоків підстановки), де він перетворюється на набір із 4 бітів.

Крок 5. P-блок. Вісім груп 4-бітових елементів комбінуються в 32-бітовий рядок і перемішуються, формуючи вихід функції *F*.

Розглянемо деякі з кроків детальніше.

Крок 1. Перестановка із розширенням (expansion) *E* представлена таблицею:



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

Кожен її рядок відповідає бітам, що входять до відповідного S-блоку на наступному кроці.

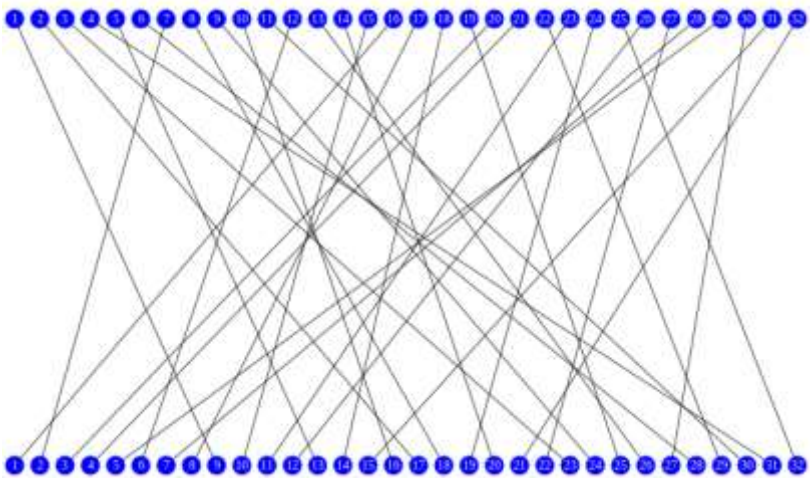
Крок 4. S-блок (від substitution). Наприклад, перший S-блок має вигляд:

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

Перший і шостий біти, що надходять на вхід S-блоку, – це номер рядка в матриці, а 2, 3, 4, 5 – номер стовпчика.

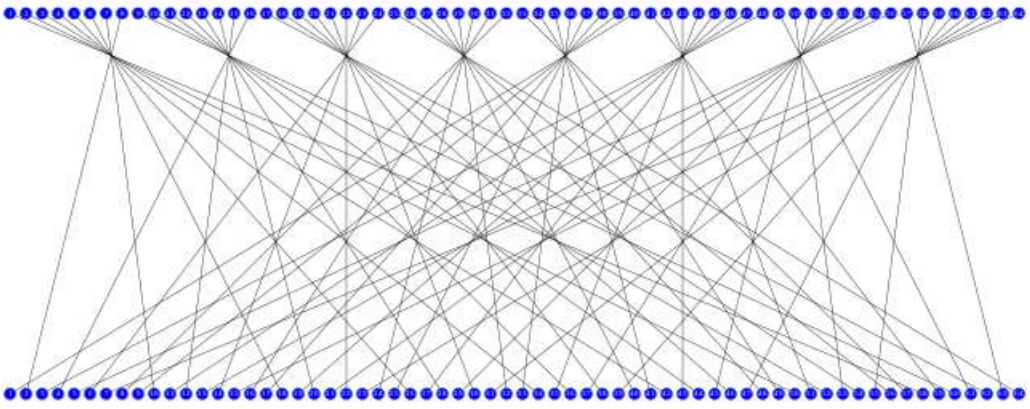
Крок 5. Перестановка (permutation) в P-блоці P. Ця перестановка перетворює 8 груп 4-бітових елементів на виході з S-блоків в 32-бітовий рядок, з'єднуючи та перемішуючи їх, згідно наступної таблиці:

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



3. Кінцева перестановка IP^{-1}. Вона здійснюється за таблицею:

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

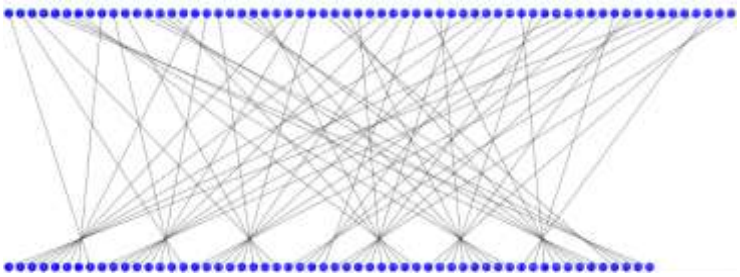


3.2.2 Розгортання ключа в DES

56 бітний ключ під час передачі доповнюється контрольними бітами до 64 біт. Кожен 8 біт – контроль парності (тобто кожен байт у ключі має складатися з непарної кількості бітів).

Алгоритм розгортання ключа.

1 крок. Робиться перестановка, яку називають PC-1 (від permuted choice), на вхід якої подається ключ з 64 бітів, на виході – ключ, що складається з 56 бітів. Перестановка проводиться згідно з таблицею нижче.



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

2 крок. Після перестановки ключ із 56 бітів ділиться на 2 частини по 28 біт кожна. Позначимо ліву частину C_0, а праву D_0.

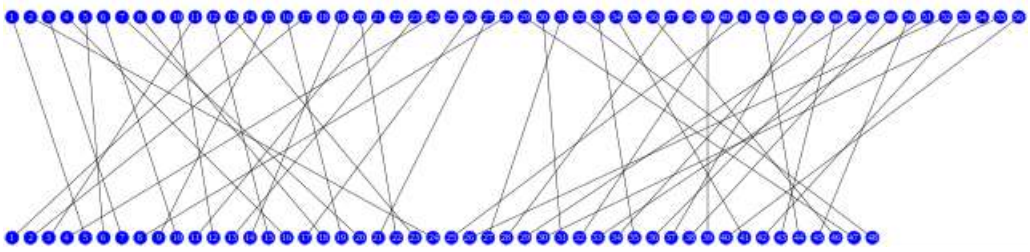
3 крок. До кожного раунду i ($i = 0…16$) обчислюється

$$C_i \leftarrow C_{i-1} \lll p_i;\ D_i \leftarrow D_{i-1} \lll p_i$$

Тут $x \lll p_i$ – циклічний зсув бітового рядка вліво на p_i позицій, де значення p_i буде $p_i = 1$ для $i = 1, 2, 9, 16$ та $p_i = 2$ для решти раундів.

4 крок. Отримані C_i та D_i об'єднуються разом.

5 крок. Робиться перестановка PC-2 згідно з таблицею:



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

3.2.3 Режими роботи алгоритму DES

1. Режим ECB (Electronic Code Book). Це найпростіший стандартний режим використання блокового шифру. У цьому режимі дані m, які необхідно зашифрувати, розбиваються на блоки по 64 біти $m_1,\ldots, m_k$. Якщо в останньому блоці менше 64 біт, він доповнюється до 64 нулями.

Далі кожен блок шифрується окремо (рис. 5): $c_i \leftarrow e_k(m_i)$.

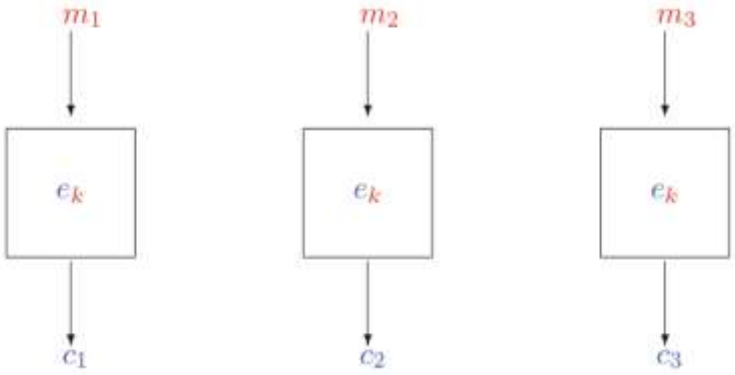


Рисунок 5. – Шифрування в режимі ECB.

Розшифрування є просто оберненою операцією.

Режим ECB не є стійким. Видалення однієї з частин криптограми можна здійснити непомітно.

2. Режим CBC (Cipher Block Chaining). Це блоковий режим. У цьому режимі весь текст розбивається на n блоків по 64 біти $m_1,\ldots, m_n$. Шифрування здійснюється за формулами (рис. 6):

$$c_1 \leftarrow e_k(m_1 \text{ XOR } IV);\ \ c_i \leftarrow e_k(m_i \text{ XOR } c_{i-1}) \text{ для } i > 1.$$

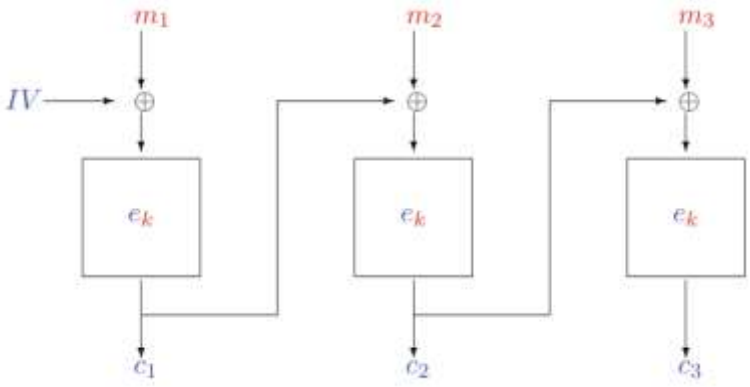


Рисунок 6. – Шифрування в режимі CBC.

Тут IV (initialization vector) – ініціалізаційний вектор, величина, яка є або унікальною (тобто ніколи не повторюється, нонс, nonce – number used once), або прив'язана до деякого значення, або випадкова; залежно від цього змінюються властивості безпеки. Загалом не вимагається, щоб це значення було секретним.

Шифротекст матиме вигляд IV || c_1 || c_2 …. Передача IV може бути відсутня, якщо приймач попередньо знатиме, яким буде це значення.

Розшифрування відбувається за формулами (рис. 7):

$$m_1 \leftarrow d_k(c_1) \text{ XOR } IV;\ m_i \leftarrow d_k(c_i) \text{ XOR } c_{i-1} \text{ для } i > 1.$$

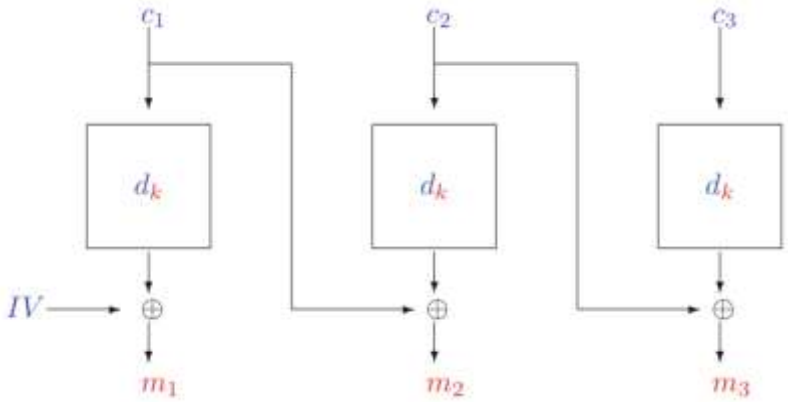


Рисунок 7. – Розшифрування в режимі CBC.

3. Режим OFB (Output FeedBack). Це потоковий режим. У цьому режимі дані розбиваються на серію блоків m_1,..., m_n. Кожен блок складається із j $(1 \le j \le n)$ бітів.

Процеси шифрування і розшифрування відбуваються у схожий спосіб (рис. 8).

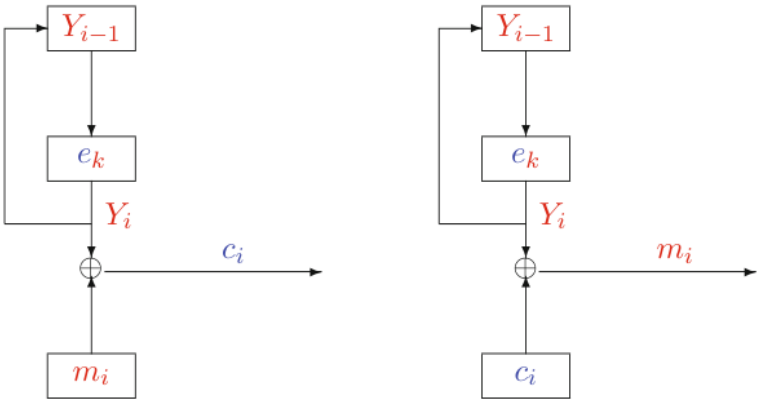


Рисунок 8. – Шифрування і розшифрування в режимі OFB.

В ході шифрування спочатку встановлюється $Y_0 \leftarrow IV$. Потім для $i = 1, 2, \ldots, n$ виконуються наступні кроки:

$$Y_i \leftarrow e_k(Y_{i-1}),$$

$$c_i \leftarrow m_i \text{ XOR } Y_i.$$

Шифротекст матиме вигляд IV || c_1 || c_2 …. Розшифрування відбувається за аналогічною схемою.

4. Режим CFB (Cipher FeedBack). Це також потоковий режим, він певною мірою схожий на попередній. Процедура шифрування наступна (рис. 9).

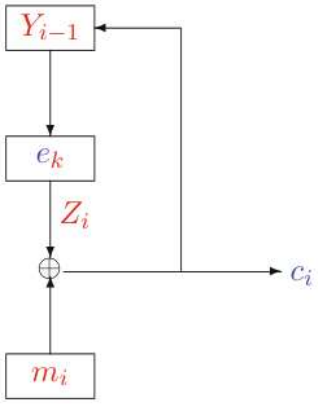


Рисунок 9. – Шифрування в режимі CFB.

Після встановлення $Y_0 \leftarrow IV$ виконуються кроки:

$$Z_i \leftarrow e_k(Y_{i-1}),$$

$$Y_i \leftarrow m_i \text{ XOR } Z_i.$$

5. Режим CTR (Counter Mode). Сучасніший метод, він комбінує багато переваг ECB, але без недоліків. Також є потоковим режимом (рис. 10).

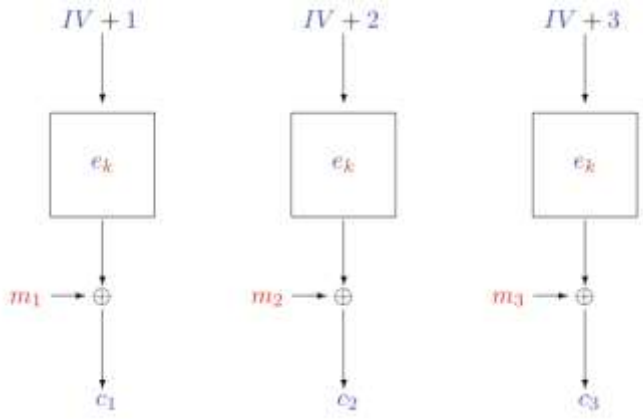


Рисунок 10. – Шифрування в режимі CTR.

Наступний блок потоку ключа породжується шифруванням послідовних значень лічильника IV:

$$c_i \leftarrow m_i \text{ XOR } e_k(IV \text{ XOR } \langle i \rangle_n),$$

де $\langle i \rangle_n$ – це n-бітне представлення числа i. Значення лічильника гарантовано не мають повторюватися в наступних шифруваннях.

Режим дозволяє розпаралелювати процеси шифрування і розшифрування.

3.3 Алгоритм 3DES

Довжина ключа в 56 біт на сьогодні є недостатньою, тому можна використати DES з трьома ключами і трьома ітераціями основного шифру (рис. 11). Таким чином досягається довжина ключа 168 біт.

Алгоритм вважається досить надійним, однак не настільки, як можна було б очікувати від такої довжини ключа. Проте низька швидкодія і незручності програмної реалізації, характерні для DES, роблять 3DES не дуже практичним для сучасних потреб.

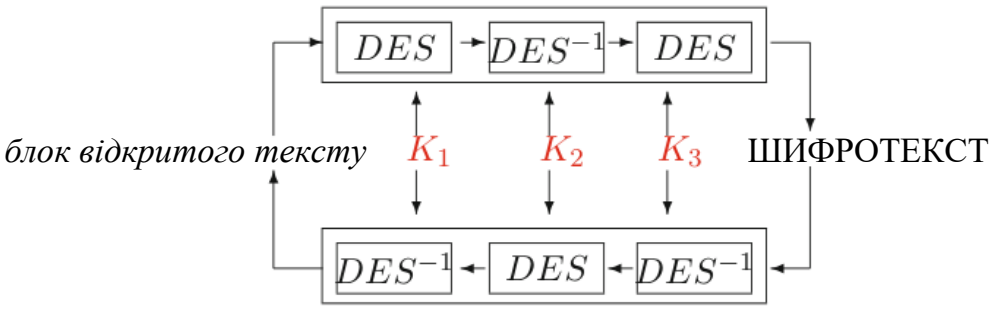


Рисунок 11. – Схема роботи 3DES.

3.4 Алгоритм Rijndael (AES)

Rijndael (рейндал) – симетричний блоковий алгоритм (розмір блоку 128/192/256 біт, ключа 128/192/256 біт), прийнятий як стандарт уряду США за результатами конкурсу AES (Advanced Encryption Standard) у версіях з розміром блоку 128 біт.

Розглянемо найпростіший варіант, у якому розмір блоку 128 біт і розмір ключа 128 біт. Кількість раундів в алгоритмі дорівнює 10.

Крок 1. Вхідний блок даних input довжиною 128 біт копіюється в масив state[r][c] = input[r + 4c], де r = 0..3, c = 0..3. При цьому кожен елемент матриці займає 8 біт.

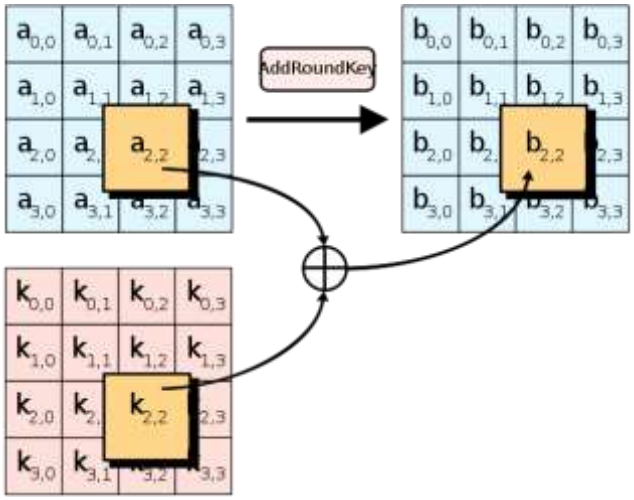
Крок 2. До матриці state[r][c] застосовується наступний алгоритм:

```
AddRoundKey(state, K[0]);
 for (i=1; i<=9; i++){
  T = SubBytes(state);
  ShiftRows(T);
  F = MixColumns(T);
  AddRoundKey(F, K[i]);
  state = F;
 }
 SubBytes(state);
 ShiftRows(state);
 AddRoundKey(state, K[10]);
```

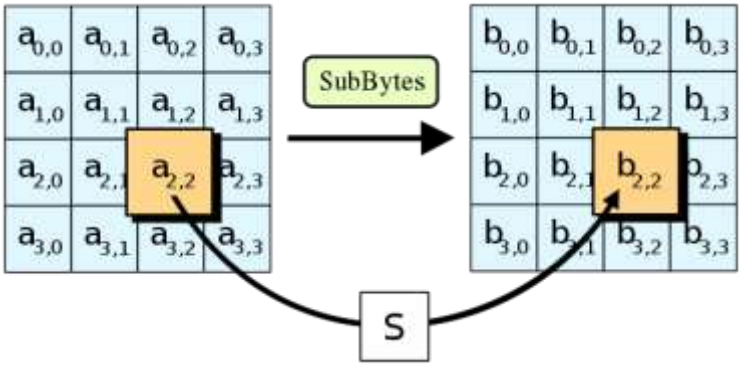
Тут K[i], i = 1..10 – раундові ключі.

Розглянемо функції, що входять в алголритм.

1. AddRoundKey(state, K[i]) – функція додавання XOR двох матриць state й K[i] (додавання здійснюється поелементно).



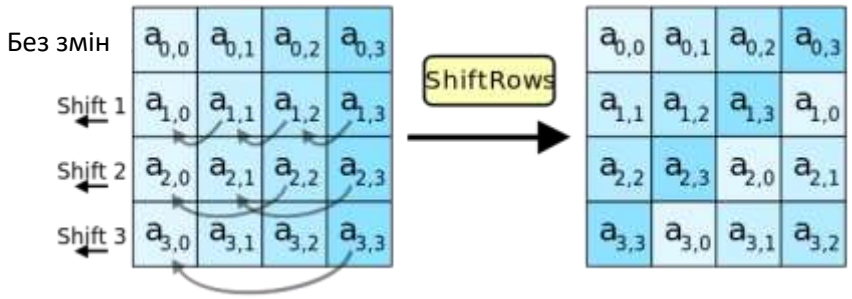
2. B = SubBytes(A). Ця функція перетворює матрицю A в B.



S-перетворення: $b_{ir} = S(a_{ir})$ (a_{ir} та b_{ir} – 8-бітові числа). Перетворення складається з 2 частин: в першій частині знаходиться мультиплікативне обернене c_{ir} до a_{ir} по mod 2^8. У другій частині робиться перетворення:

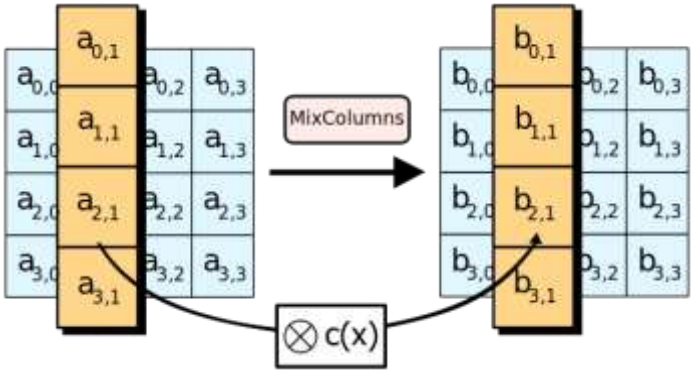
$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix}$$

3. ShiftRows(A). ShiftRows працює з рядками T. При цій трансформації рядки стану циклічно зсуваються на r байт по горизонталі залежно від номера рядка. Для нульового рядка $r = 0$, для першого рядка $r = 16$ і т.д. Тобто, маємо:



4. B = MixColumns(A).

В процедурі MixColumns чотири байти кожного стовпця A змішуються з використанням оборотної лінійної трансформації. MixColumns обробляє стани по стовпчикам, трактуючи кожен із них як поліном четвертого степеня. Над цими поліномами проводиться множення в скінченному полі $\mathbb{F}_{2^8}$ за модулем $x^4 + 1$ на фіксований многочлен $c(x) = 3x^3 + x^2 + x + 2$. Іншими словами, отримуємо:



Алгоритм розгортання ключа.

Крок 1. 128-бітний ключ ділиться на 4 слова по 32 біти K = {A[0], A[1], A[2], A[3]}.

Крок 2. i-й ключ знаходиться за алгоритмом

```
W[0] = A[0]; W[1] = A[1]; W[2] = A[2]; W[3] = A[3];
for (i=1; i<=10; i++){
T = CRot(W[4*i-l]);//циклічний зсув вліво на
// 1 байт (8 біт)
F = SBlock(T);
D = F XOR RC[i];
```

```
W[4*i] = W[4*i-4] XOR D;
W[4*i+1] = W[4*i-3] XOR W[4*i];
W[4*i+2] = W[4*i-2] XOR W[4*i+1];
W[4*i+3] = W [4*i-1] XOR W [4*i+2]; }
```

Функція SBlock(B) застосовує до 8 бітового числа S-перетворення.

RC[i] – константа раунду, яка має вигляд RC[i] = $\{x^{i-1},00,00,00\}$, де $x = \{02\}$, а x^{i-1} – степінь x в полі $\mathbb{F}_{2^8}$.

Таким чином, раундовим ключем буде

$$K[i]=\{W[4i], W[4i+1], W[4i+2], W[4i+3]\},\ i = 1,\ldots 10.$$

3.5 Потокові шифри. Алгоритм RC4

RC – скорочення від Ron's Code або Rivest Cipher (код Рона, шифр Рівеста). Алгоритми з серії RC розроблені Роном Рівестом (Ron Rivest) з MIT, загалом вони між собою не пов’язані. Шифр є надзвичайно простим і ефективним, однак через нещодавно виявлені вразливості алгоритм наразі не рекомендовано використовувати.

RC4 належить до потокових шифрів. Ядро алгоритму складається із функції генерації потоку ключа. Ця функція генерує послідовність бітів k_i, яка потім поєднується з відкритим текстом m_i за допомогою побайтового сумування по модулю два. Так отримується шифрограма c_i:

$$c_i = m_i \text{ XOR } k_i.$$

Розшифрування полягає в регенеруванні потоку ключа k_i та сумуванні його та шифрограми c_i за модулем два. Завдяки властивостям операції XOR, на виході отримаємо початковий незашифрований текст m_i:

$$m_i = c_i \text{ XOR } k_i.$$

RC4 – фактично клас алгоритмів, що визначаються за розміром їх блоку. Параметр n є розміром слова алгоритму і визначає довжину блоку. Зазвичай $n = 8$, тому розмір робочого масиву (S-блоку) буде $2^8 = 256$.

Алгоритм RC4 складається з двох частин: ініціалізації та генерації потоку ключа.

При ініціалізації (key-scheduling algorithm, KSA) спочатку відбувається початкове заповнення масиву:

```
for (i=0; i<256; i++)
{S[i] = i;}
```

Далі робиться скремблювання (перемішування) шляхом перестановок, визначених ключем:

```
j = 0;
for (i=0; i<256; i++){
 j = (j + S[i] + Key[i mod L]) mod 256;
 swap(S[i], S[j]);
}
```

Тут довжина ключа складає L байт.

Генерація потоку ключа після початкових присвоєнь $i = 0$ та $j = 0$ відбувається в наступному циклі відоповідно до довжини вхідного тексту:

```
i = (i+1) mod 256;
j = (j + S[i]) mod 256;
swap(S[i], S[j]);
t = (S[i] + S[j]) mod 256;
K = S[t];
```

Далі до згенерованого псевдовипадкового слова K та відповідного байту тексту застосовується операція XOR, породжуючи таким чином шифрограму.

Частина 4. Криптосистеми з відкритим ключем

4.1 Шифрування з відкритим ключем

Симетричні шифри вимагають, щоб кожній зі сторін був відомий той самий секретний ключ. Тоді як в асиметричних шифрах використовуються два ключі: відкритий (public) та секретний (private). Шифрування відбувається за допомогою відкритого ключа, дешифрування за допомогою секретного ключа. Інакше кажучи, шифрування здійснюється за схемою

$$C = E_{k_1}(m),$$

де k_1 – відкритий ключ, m – вихідне повідомлення, C – шифротекст.

Дешифрування здійснюється за схемою

$$m = D_{k_2}(C),$$

де k_2 – секретний ключ.

Якщо відкритий ключ опубліковано разом з іменем користувача, то кожен може за допомогою його зашифрувати свого листа, наприклад, Алісі. Але прочитати такого листа зможе тільки Аліса, бо лише вона має відповідний секретний ключ.

Подібні криптосистеми працюють завдяки однобічному математичному зв'язку, що існує між двома ключами, при якому інформація про відкритий ключ ніяк не допомагає відновити секретний, але володіння секретним ключем забезпечує можливість розшифровувати повідомлення, зашифровані відкритим ключем. Реалізувати це можна за допомогою деякого перетворення, яке легко здійснити, але важко (з точки зору теорії складності) зробити зворотне (обернути) без знання деяких секретних даних: **односторонньої функції з секретом** (trapdoor one-way function). Тому кажуть, що асиметричні алгоритми шифрування ґрунтуються на так званих складних завданнях.

4.2 Задача факторизації цілих чисел на прості множники.

Задача такого роду є, мабуть, найважливішою односторонньою функцією, що використовується в криптографії з відкритим ключем. Під **розкладанням на**

множники (або **факторизацією**) цілого числа розуміють його подання у вигляді добутку простих дільників, наприклад,

$10 = 2 \cdot 5$,

$60 = 22 \cdot 3 \cdot 5$,

$2^{113} - 1 = 3\,391 \cdot 23\,279 \cdot 65\,993 \cdot 1\,868\,569 \cdot 1\,066\,818\,132\,868\,207$.

Визначення множників є надзвичайно трудомісткою обчислювальною операцією. Для оцінки складності алгоритму, який розкладає ціле число N на прості множники, часто використовують функцію

$$L_N(\alpha, \beta) = \exp((\beta + o(1))(\ln N)^{\alpha}(\ln \ln N)^{1-\alpha}).$$

Варто зауважити, що якщо алгоритм, що розкладає на множники ціле число, має складність $O(L_N(0, \beta))$, то він працюватиме за поліноміальний час (розмір входу задачі – $\ln N$). Однак при складності алгоритму $O(L_N(1, \beta))$, йому потрібен вже експоненціальний час.

Таким чином, швидкість зростання функції $L_N(\alpha, \beta)$ при $0 < \alpha < 1$ перебуватиме між поліноміальною та експоненційною. Тому про алгоритм зі складністю $O(L_N(\alpha, \beta))$ при $0 < \alpha < 1$ говорять, що він працює за суб-експоненціальний час. В той же час обернена до факторизації операція – множення – дуже проста і виконується швидше, ніж за $O(L_N(0,2))$.

Існує декілька методів факторизації числа $N = p \cdot q$ з простими p та q. Найбільш відомі з них:

1. **Пробне ділення.** В цьому алгоритмі для всіх простих чисел p, що не перевищують $\sqrt{N}$, перевіряється умова $N/p \in \mathbb{Z}$. Такий підхід близький до перебору і має експоненціальну складність $L_N(1,1)$.

2. **Метод еліптичної кривої** добре працює, якщо один з простих множників p не перевищує 2^{50}. Його складність суб-експоненціальна і оцінюється як $L_p(1/2, c)$.

3. **Квадратичне решето** – найшвидший спосіб розкладання чисел, що лежать між 10^{80} і 10^{100}. Його складність $L_N(1/2, 1)$.

4. **Квадратичне решето в числовому полі**. Найкращий метод для чисел, що мають 100 і більше десяткових знаків. З його допомогою можна розкладати на множники числа до $10^{155} \approx 2^{512}$ за оцінки складності $L_N(1/2, 1)$.

Існує низка інших складних задач, пов'язаних із факторизацією, які використовуються для розробки криптосистем із відкритим ключем. Припустимо, дано число $N = p \cdot q$, але не відомий жоден із його простих дільників. Виникають чотири основні задачі:

- Факторизація: знайти дільники p та q числа $N = p \cdot q$.
- Задача RSA: дано числа C та E, останнє з яких задовольняє співвідношення

$$\text{НСД}\big(E, (p-1)(q-1)\big) = 1.$$

  Потрібно знайти таке число m, що

$$m^E = C \pmod{N}.$$

- Тест на квадратичне відрахування: визначити, чи є задане число A повним квадратом за модулем N.
- Добування квадратних коренів: дано таке число A, що $A = x^2 \pmod{N}$, потрібно обчислити x.

  Між цими задачами є зв'язок.

Лема 1. Задача розкладання на множники та добування квадратного кореня поліноміально еквівалентні.

Доведення. Спочатку зведемо задачу добування квадратного кореня до задачі розкладання на множники. Припустимо, ми маємо алгоритм, який розкладає число на множники, і необхідно з його допомогою видобути квадратний корінь за модулем складеного числа N. Розглянемо рівняння

$$Z = x^2 \pmod{N}$$

відносно змінної x. Знайдемо прості дільники P_i числа N та обчислимо

$$S_i = \sqrt{Z} \pmod{P_i},$$

що можна зробити за поліноміальний час за допомогою алгоритму Шенкса (метод «крок немовляти, крок велетня»). Після цього, використовуючи китайську теорему про остачі, знайдемо x як розв’язок системи рівнянь

$$x = S_i \pmod{P_i}.$$

Отже, обчислення квадратного кореня за модулем N не складніше за задачу факторизації.

Тепер покажемо, як розкладання на множники може бути зведено до задачі добування квадратного кореня.

Використаємо алгоритм добування коренів для визначення дільників числа N. Тобто, нам дано число $N = p \cdot q$ і потрібно знайти p. Для цього виберемо випадкове число x, що належить $(\mathbb{Z}/N\mathbb{Z})^*$ і обчислимо

$$z = x^2 \pmod{N}.$$

Використовуючи алгоритм добування коренів, знайдемо

$$y = \sqrt{z} \pmod{N}.$$

Зауважимо, що існує рівно чотири квадратних корені, оскільки N – добуток двох простих чисел. З ймовірністю 50% можна стверджувати, що

$$y \neq \pm x \pmod{N}.$$

Якщо цей корінь не підійде, можна вибрати інший, тобто в середньому після двох проходів алгоритму буде отримано потрібну нерівність. Тепер, оскільки

$$x^2 = y^2 \pmod{N},$$

то N має ділити число

$$x^2 - y^2 = (x - y)(x + y).$$

Але N не може ділити ні $x - y$, ані $x + y$, бо $y \neq \pm x \pmod{N}$. Отже, дільники числа N повинні ділити і суму, і різницю x та y. Тому один із нетривіальних дільників числа N може бути знайдений як $\text{НСД}(x - y, N)$.

Очевидно, зведення задач однієї до одної можна здійснити за поліноміальний час, що і доводить поліноміальну еквівалентність задач факторизації та добування кореня. ■

Лема 2. Задача RSA не складніша за проблему факторизації.

Доведення. Застосовуючи алгоритм факторизації, розкладемо число N на прості множники, обчислимо значення функції Ейлера $\Phi = \phi(N)$ і знайдемо

$$D = 1/E \pmod{\Phi}.$$

Знаючи число D, легко відновити m, оскільки

$$C^D = m^{ED} = m^{1 \pmod{\Phi}} = m \pmod{N}.$$

Звідси випливає, що задача RSA не складніша за проблему факторизації, що й потрібно було довести. ■

4.3 Алгоритм RSA

Розглянемо алгоритм RSA (Rivest – Shamir – Adleman). Це перший з алгоритмів з відкритим ключем, що витримав випробування часом. В основі алгоритму лежить одноіменна задача.

Нехай А хоче зв'язатися з В використанням RSA. Тоді А робить такі дії.

1. Підбирає два великих простих числа p та q.

2. Знаходить їх добуток (**модуль** алгоритму) $N = p \cdot q$ і публікує його (відкрито повідомляє В).

3. А вибирає шифруючу експоненту E, яка задовольняє умову

$$\text{НСД}\ (E, (p-1) \cdot (q-1)) = 1,$$

і теж публікує її (повідомляє В використовуючи відкриті канали).

Таким чином, пара (N, E) є відкритим ключем А. Значення Е зазвичай вибирають рівним 3, 17 чи 65537.

4. А визначає свою розшифровуючу експоненту d з рівняння

$$E \cdot d = 1 \pmod{(p-1)(q-1)}.$$

Звідси секретним ключем є трійка чисел (d, p, q).

Якщо В хоче відправити А зашифроване повідомлення m, то виконує наступні дії. Він формує шифротекст

$$C = m^E \pmod{N}$$

та надсилає його А. Отримавши шифротекст, А розшифровує його:

$$m = C^d \pmod{N}.$$

Залишаємо як вправу доведення того, що А справді відтворить в результаті вихідне повідомлення m.

На сьогодні стійким вважається ключ, якщо N має довжину 1024 біт, а у випадку довгострокових ключів 2048 біт.

Розглянемо приклад використання RSA. Нехай $p = 7$ та $q = 11$.

Тоді $N = p \cdot q = 77$ та $(p - 1)(q - 1) = 6 \cdot 10 = 60$.

Оберемо як відкриту шифруючу експоненту $E = 37$, бо НСД $(37, 60) = 1$. Розшифровуючу експоненту знайдемо зі співвідношення $37 \cdot d \pmod{60} = 1$, звідки $d = 13$.

Припустимо, числова форма повідомлення, яке треба зашифрувати, $m = 2$. Отримуємо:

$$C = m^E \pmod{N} = 2^{37} \pmod{77} = 51.$$

Тобто, $C = 51$. Тепер здійснимо розшифрування:

$$m = C^d \pmod{N} = 51^{13} \pmod{77} = 2.$$

Лема 3. Якщо задача RSA є важкорозв'язною, то криптосистема RSA обчислювально захищена від атак із вибором відкритого тексту в тому розумінні, що зловмисник не здатний коректно відновити відкритий текст, маючи лише шифротекст.

Доведення. Розробимо алгоритм розв'язання задачі RSA, ґрунтуючись на алгоритмі злому однойменної криптосистеми. Нагадаємо, в задачі RSA дано число N, що має два невідомих простих дільники p та q, елементи E, Y, що належать $(\mathbb{Z}/N\mathbb{Z})^*$, і потрібно знайти такий елемент x, що $x^E \pmod{N} = Y$.

Дешифруємо повідомлення $C = Y$ і отримаємо відповідний відкритий текст, що задовольняє, за визначенням, співвідношення

$$m^E \pmod{N} = C = Y.$$

Звідси можна побачити, що зламавши криптосистему, зможемо розв'язати задачу RSA, що й потрібно було довести. ■

4.4 Секретна експонента та проблема факторизації.

Покажемо, що злом криптосистеми RSA, в сенсі обернення однойменної функції, еквівалентний проблемі факторизації, розв'язок якої дозволяє знайти секретний ключ d за відкритою інформацією про N та E.

Лема 4. Якщо відома розшифровуюча експонента d алгоритму RSA, що відповідає відкритому ключу (N, E), то число N можна ефективно розкласти на множники.

Доведення. Нехай задана секретна експонента d, а також відомі E та N.

Розглянемо алгоритм Лас-Вегаса. Це ймовірнісний алгоритм, який з певною ймовірністю повертає правильний результат. Можна алгоритмічно перевірити, чи отриманий результат коректний, інакше повторно його запустити. Алгоритм Лас-Вегаса гарантує, що правильний результат колись буде отримано. Подібний підхід має сенс, якщо кількість можливих розв'язків відносно обмежена і перевірка правильності потенційного результату набагато простіша за безпосереднє обчислення розв'язку. Використаємо цю ідею.

Відомо, що для деякого цілого s виконується рівність

$$Ed - 1 = s(p-1)(q-1).$$

Візьмемо довільне ціле ненульове число X. Тоді

$$X^{Ed-1} = 1 \pmod{N}.$$

Ця рівність випливає з теореми Лагранжа, оскільки X належить $(\mathbb{Z}/N\mathbb{Z})^*$. Порядок групи $(\mathbb{Z}/N\mathbb{Z})^*$ дорівнює $\varphi(N) = (p-1)(q-1)$, тоді за теоремою Лагранжа $X^{\varphi(N)} = 1 \pmod{N}$.

Обчислюємо квадратний корінь Y_1 за модулем N:

$$Y_1 = \sqrt{X^{Ed-1}} = X^{\frac{Ed-1}{2}} \pmod{N},$$

що можна зробити, оскільки $Ed-1$ відоме і парне (бо парне $(p-1)(q-1)$). Приходимо до тотожності

$$Y_1^2 - 1 = 0 \pmod{N}$$

яку можна використати для визначення дільників числа N шляхом обчислення НСД $(Y_1 - 1, N)$ за умови $Y_1 \neq \pm 1 \pmod{N}$.

Припустимо, умова не виконується. Тоді, якщо $Y_1 = -1 \pmod{N}$, повернемося на початок і виберемо інше X. У випадку $Y_1 = 1 \pmod{N}$ можна взяти ще один квадратний корінь

$$Y_2 = \sqrt{Y_1} = X^{\frac{Ed-1}{4}} \pmod{N}.$$

Знову отримуємо

$$Y_2^2 - 1 = Y_1 - 1 = 0 \pmod{N},$$

звідки НСД $(Y_2 - 1, N)$ – дільник числа N. Тут знову можемо отримати, що $Y_2 = \pm 1 \pmod{N}$, і доведеться повторювати все знову.

Алгоритм слід повторювати доти, поки N не буде розкладено на множники або не отримаємо число $(Ed - 1)/2^t$, яке вже не буде ділитися на 2, і тоді треба вибрати інше X. ■

Розглянемо приклад. Нехай N = 1 441 499, E = 17 та d = 507 905. Тоді

$$T_1 = (Ed - 1)/2 = 4\ 317\ 192,$$

$$X = 2.$$

Для обчислення Y_1 здійснимо перетворення:

$$Y_1 = X^{(Ed-1)/2} = 2^{T_1} = 1 \pmod{N}.$$

Оскільки Y_1 дорівнює 1, слід взяти

$$T_2 = T_1/2 = (Ed - 1)/4 = 2\ 158\ 596 \text{ та } Y_2 = 2^{T_2}.$$

Тепер $Y_2 = X^{(Ed-1)/4} = 2^{T_2} = 1 \pmod{N}$. Після повторення попередніх кроків отримуємо

$$T_3 = (Ed - 1)/8 = 1\ 079\ 298,$$

$$Y_3 = X^{(Ed-1)/8} = 2^{T_3} = 119\ 533 \pmod{N}.$$

Звідси

$$Y_3^2 - 1 = (Y_3 - 1)(Y_3 + 1) = 0 \pmod{N},$$

тоді простий дільник числа N можна знайти, обчисливши

$$\text{НСД } (Y_3 - 1, N) = 1\ 423.$$

4.5 Значення функції Ейлера та проблема факторизації

Як ми побачили, інформація про секретну експоненту d дозволяє знайти простий дільник числа N. Знання функції Ейлера $\Phi = \varphi(N)$ допоможе розв'язати ту саму задачу.

Лема 5. Значення $\Phi = \varphi(N)$ дозволяє ефективно розкласти число N на множники.

Доведення. Маємо:

$$\Phi = (p-1)(q-1) = N - (p+q) + 1.$$

Отже, поклавши $S = N + 1 - \Phi$, отримаємо $S = p + q$. Необхідно визначити числа p та q, знаючи їх суму S і добуток N.

Тоді отримаємо квадратне рівняння

$$q^2 - Sq + N = 0,$$

звідки

$$q = \frac{S + \sqrt{S^2 - 4N}}{2}, \qquad p = \frac{S - \sqrt{S^2 - 4N}}{2},$$

що й потрібно було довести. ■

Розглянемо приклад. Нехай N = 18 923, а $\Phi = \varphi(N)$ = 18 648.

Тоді S = 276 та

$$q = \frac{276 + \sqrt{276^2 - 4 \cdot 18923}}{2} = 149, \;\; p = 127.$$

4.6 Проблема спільного модуля

Оскільки арифметика залишків – дороге задоволення з погляду обчислень, було б досить привабливо розробити систему шифрування, у якій користувачі поділяють загальний модуль N, але використовують різні шифруючі та розшифровуючі експоненти (E_i, d_i). Це дозволило б прискорити алгоритми шифрування та розшифрування для апаратної реалізації. Однак в цьому випадку алгоритм втрачає стійкість.

Припустимо, що зловмисником є один із законних клієнтів криптосистеми, приміром, Користувач 1. Він може знайти значення розшифровуючої експоненти d_2, яку зберігає в таємниці Користувач 2.

Спочатку він обчислює p та q за допомогою алгоритму з доведення леми 4, він може це зробити, знаючи свою розшифровуючу експоненту d_1.

Потім він знаходить $\varphi(N) = (p-1)(q-1)$ і, нарешті, розкриває значення d_2 за формулою

$$d_2 = \frac{1}{E_2} \left(\mathrm{mod}\ \varphi(N)\right).$$

Тепер розглянемо ситуацію, коли зловмисник не є користувачем системи. Припустимо, що Користувач 1 надсилає однакове повідомлення m Користувачам 2 і 3 в зашифрованому вигляді, тобто C_2 і C_3, де

$$C_2 = m^{E_2} \ (\mathrm{mod}\ N), \qquad C_3 = m^{E_3} \ (\mathrm{mod}\ N).$$

Якщо зловмисник перехоплює ці повідомлення, він може обчислити

$$T_2 = E_2^{-1} \ (\mathrm{mod}\ E_3), \qquad T_3 = (T_2 E_2 - 1)/E_3.$$

та відновити вихідне повідомлення за схемою

$$C_2^{T_2} C_3^{-T_3} = m^{E_2 T_2} m^{-E_3 T_3} = m^{1+E_3 T_3} m^{-E_3 T_3} = m^{1+E_3 T_3 - E_3 T_3} = m^1 = m.$$

4.7 Проблема використання малих шифруючих експонент

Часом в криптосистем RSA з метою економії витрат на шифрування використовуються невеликі шифруючі експоненти E. Це також може призвести до проблем.

Припустимо, що є три користувача з різними модулями шифрування N_1, N_2 і N_3 та однією шифруючою експонентою $E = 3$. Нехай дехто надсилає їм те саме повідомлення m, зашифроване трьома різними відкритими ключами. Зловмисник бачить три криптограми:

$$C_1 = m^3 \ (\mathrm{mod}\ N_1),$$
$$C_2 = m^3 \ (\mathrm{mod}\ N_2),$$
$$C_3 = m^3 \ (\mathrm{mod}\ N_3),$$

і за допомогою китайської теореми про залишки знаходить розв’язок системи

$$\{X = C_i \ (\mathrm{mod}\ N_i) |\ i = 1, 2, 3\}$$

у вигляді

$$X = m^3 \ (\mathrm{mod}\ N_1 N_2 N_3).$$

Оскільки $m^3 < N_1 N_2 N_3$, то цілі числа X та m^3 повинні збігатися. Тому, обчислюючи кубічний корінь X, розкривається вихідне повідомлення.

Нехай, наприклад, $N_1 = 323$, $N_2 = 299$, $N_3 = 341$ і хтось перехоплює повідомлення $C_1 = 50$, $C_2 = 299$ та $C_3 = 1$.

Щоб відновити вихідне повідомлення m, необхідно знайти розв'язати систему

$$X = 50 \pmod{323},$$
$$X = 299 \pmod{299},$$
$$X = 1 \pmod{341}.$$

Розв'язоком цієї системи буде $X = 300\,763 \pmod{N_1N_2N_3}$, після добування кубічного кореня отримаємо

$$m = X^{1/3} = 67.$$

4.8 Криптосистема Ель-Гамаля

В основі криптосистеми Ель-Гамаля (схеми ElGamal) лежить інша трудомістка задача – дискретне логарифмування. Розглянемо клас задач, пов'язаних з дискретним логарифмуванням.

4.8.1 Задача дискретного логарифмування

Нехай $(G, \cdot)$ – скінченна абелева група, наприклад, мультиплікативна група скінченного поля або еліптична крива над скінченним полем. Проблема обчислення дискретних логарифмів (ПДЛ) полягає у визначенні цілого числа x, яке при заданих A, B, що належать G задовольняє співвідношення

$$A^x = B.$$

Неформально, $x = \log_A B$, звідси походить назва.

Для деяких груп G подібна задача досить проста. Наприклад, якщо G – група цілих чисел за модулем N за додаванням, то в ПДЛ необхідно за відомими A, B з $\mathbb{Z}/N\mathbb{Z}$ знайти розв'язок рівняння $x \cdot A = B$.

Для інших груп завдання буде складнішим. Наприклад, в мультиплікативній групі скінченного поля найкращий з відомих алгоритмів, що розв'язує проблему дискретного логарифмування, – це метод квадратичного решета в числовому полі. В такому разі складність обчислення дискретних логарифмів оцінюється як

$L_N(1/3, c)$. Для таких груп, як еліптичні криві, задача дискретного логарифмування ще складніша.

З проблемою дискретного логарифмування також пов'язано декілька близьких задач. Щоб їх сформулювати, припустимо, що задана скінченна абелева група $(G, \cdot)$ та її елемент A.

- ПДЛ – задача дискретного логарифмування: за заданими A, B з G знайти такий x, що $B = A^x$.
- ЗДГ – задача Діффі-Геллмана: задані елементи A, $B = A^x$ та $C = A^y$, потрібно обчислити $D = A^{xy}$.
- ПВДГ – проблема вибору Діффі-Геллмана: дано A, $B = A^x$, $C = A^y$ та $D = A^z$, треба визначити, чи є z добутком $z = x \cdot y$.

Розглянемо, як пов'язані ці задачі.

Лема 6. В будь-якій скінченній абелевій групі G задача Діффі-Геллмана не складніша за проблему дискретного логарифмування.

Доведення. Припустимо, що існує оракул $\mathcal{O}_{\text{лог}}$, який розв’язує задачу дискретного логарифмування, тобто за елементами A та $B = A^x$ знаходить показник x. Для пошуку відповіді для задачі Діффі-Геллмана з заданими $B = A^x$ і $C = A^y$ обчислюємо

$$Z = \mathcal{O}_{\text{лог}}(B) \text{ та } D = C^Z.$$

Отримане значення D – шукана відповідь до задачі Діффі-Геллмана. Очевидно, таке зведення має поліноміальну складність і видає розв’язок задачі Діффі-Геллмана в припущенні, що алгоритм $\mathcal{O}_{\text{лог}}$ коректно обчислює логарифми.

Отже, задача Діффі-Геллмана є не складнішою за дискретне логарифмування. ■

Лема 7. Проблема вибору Діффі-Геллмана в будь-якій скінченній абелевій групі G не складніша за задачу Діффі-Геллмана.

Доведення. Нехай $\mathcal{O}_{\text{ДГ}}$ – алгоритм, що розв’язує задачу Діффі-Геллмана, який за елементами A^x і A^y обчислює елемент A^{xy}.

Розглянемо елементи $B = A^x$, $C = A^y$ і $D = A^z$ та обчислимо $E = \mathcal{O}_{\text{ДГ}}(B, C)$. Якщо $E = D$, то проблема вибору розв'язана позитивно, інакше – негативно. Таким чином, зведення однієї задачі до іншої відбувається за поліноміальний час. ■

4.8.2 Алгоритм Ель-Гамаля

Розглянемо, яким чином **генеруються ключі**.

1. Генерується випадкове просте число p довжини n.
2. Вибирається породжуючий елемент – довільне ціле число g таке, що

$$g^{\varphi(p)} = 1 \pmod{p}$$

та $g^l \neq 1 \pmod{p}$ при $1 \leq l \leq \varphi(p) - 1$, де $\varphi(p)$ – функція Ейлера.

3. Вибирається випадкове число x з інтервалу $(1 \ldots p)$, взаємно просте з $p - 1$.
4. Обчислюється $h = g^x \bmod p$.

Таким чином, відкритим ключем є трійка (p, g, h), закритим ключем – число x.

Розглянемо тепер процедуру **шифрування**. Нехай потрібно зашифрувати повідомлення m:

1. Обирається випадкове секретне число k, взаємно просте з $p - 1$.
2. Обчислюються $a = g^k \bmod p$, $b = h^k m \bmod p$.
3. Шифротекстом буде пара чисел (a, b).

Поглянемо на **розшифрування**. Знаючи закритий ключ x, вихідне повідомлення можна відновити із шифротексту (a, b) за формулою:

$$m = b(a^x)^{-1} \pmod{p}.$$

Неважко перевірити, що $a^x = g^{kx} \pmod{p}$, а також

$$\frac{b}{a^x} = \frac{y^k m}{a^x} = \frac{g^{xk} m}{g^{xk}} = m \pmod{p}.$$

Розглянемо приклад. Виберемо $p = 809$ та $g = 3$. Очевидно, що

$$3^{808} = 1 \pmod{809},$$

і при жодних менших степенях співвідношення не виконується.

Виберемо довільне число $x = 68$, тоді $h = 3^{68} \pmod{809} = 65$. Тобто. відкритим ключем є (809, 3, 65), секретний ключ $x = 68$.

Припустимо, треба зашифрувати повідомлення $m = 100$. Виберемо випадкове число $k = 89$, воно взаємно просте з 809 – 1 = 808.

Знаходимо $a = 3^{89} \pmod{809} = 345$, знаходимо $b = 65^{89} \cdot 100 \pmod{809} = 517$. Тобто шифротекстом буде (345, 517).

Розшифрування робиться так: $m = 517 \cdot (345^{68})^{-1} \pmod{809} = 100$.

Лема 8. Якщо задача Діффі-Геллмана важкорозв'язна, то система Ель-Гамаля захищена проти атак з вибором відкритого тексту, де захищеність означає, що зловмисник не може відновити відкритий текст за перехопленою шифрограмою за розумний час.

Доведення. Вважаємо, що у нас є оракул $\mathcal{O}$, що розкриває шифр Ель-Гамаля. На вхід оракула подаються відкритий ключ (p, g, h) та шифротекст (a, b), а вихідними даними є дешифрований відкритий текст.

Покажемо тепер, як з допомогою оракула $\mathcal{O}$ розв'язується завдання Діффі-Геллмана, тобто задані g^x і gy, а потрібно знайти g^{xy}.

Виберемо відкритий ключ у системі Ель-Гамаля відповідно до задачі, тобто покладемо $h = g^x \pmod{p}$. Виберемо шифротекст у вигляді (a, b), де $a = g^y \pmod{p}$, а b – випадкове число. Застосуємо оракул і отримаємо відкритий текст

$$m = \mathcal{O}\big(h, (a, b)\big).$$

Тепер обчислимо

$$\frac{b}{m} = \frac{b}{b \cdot ((g^y)^x)^{-1}} = g^{xy},$$

що і треба було довести. ■

Криптосистема Ель-Гамаля в поданому вигляді не є стійкою проти атак з вибором шифротексту. Розглянемо таку атаку.

Нехай зловмисник бачить шифротекст:

$$(a, b) = (g^k \bmod p, h^k m \bmod p).$$

Тоді він може створити задовільний шифротекст, що відповідає повідомленню $2m$:

$$(a, 2b) = (g^k \bmod p, 2h^k m \bmod p).$$

Імунізація алгоритму полягає у додаванні до шифротексту додаткової інформації, а саме зашифрованого хеш-значення від відкритого тексту (схеми Ченга-Себеррі).

4.9 Криптосистема Рабіна

Криптосистема Рабіна заснована на проблемі факторизації великих цілих чисел. Якщо точніше, вона пов'язана з трудомісткістю добування квадратного кореня по модулю складеного числа $N = p \cdot q$. Вважається, що ця криптосистема є стійкішою за RSA.

Розглянемо криптосистему Рабіна. Виберемо різні прості числа p і q, які задовольняють умову:

$$p \pmod 4 = q \pmod 4 = 3.$$

Такий спеціальний вигляд простих чисел сильно прискорює процедуру видобування коренів за модулем p та q. Секретним ключем системи є пара (p, q).

Для визначення відповідного відкритого ключа беруть добуток $N = p \cdot q$ і генерують ціле випадкове число B з множини $\{0, \ldots, N-1\}$. Відкритий ключ – це пара (B, N).

Для шифрування повідомлення m в алгоритмі Рабіна обчислюють

$$C = m(m + B) \pmod N.$$

Тобто шифрування використовує операції додавання та множення за модулем N, що забезпечує більш високу швидкість шифрування, ніж RSA.

Розшифрування є набагато складнішою процедурою і вимагає обчислення

$$m = \sqrt{\frac{B^2}{4} + C} - \frac{B}{2} \pmod N.$$

Очевидно, для добування коренів за модулем N корисно знати розкладання N на прості дільники.

Оскільки N – добуток двох простих чисел, існує чотири можливі квадратні корені з числа за модулем N. Тому при розшифруванні отримуємо чотири

можливих відкритих тексти. Щоб вибір був більш визначеним, варто до відкритого тексту додавати певну надлишкову інформацію.

При розшифруванні буде отримано вихідне повідомлення з таких міркувань (вважаємо, що обрали правильний корінь з можливих):

$$\sqrt{\frac{B^2}{4}+C}-\frac{B}{2}=\sqrt{\frac{B^2+4m(m+B)}{4}}-\frac{B}{2}=\sqrt{\frac{4m^2+4Bm+B^2}{4}}-\frac{B}{2}=$$

$$=\sqrt{\frac{(2m+B)^2}{4}}-\frac{B}{2}=\frac{2m+B}{2}-\frac{B}{2}=m.$$

Розглянемо приклад. Нехай відкриті ключі $p = 127$ і $q = 131$, а секретні ключі $N = 16\,637$ та $B = 12\,345$. Для шифрування повідомлення $m = 4410$ обчислюємо

$$C = m(m+B) \pmod N = 4633.$$

При оберненій операції спочатку знаходимо

$$T = B^2/4 + C \pmod N = 1500,$$

а потім добуваємо квадратні корені з T по модулю p і q:

$$\sqrt{T} \pmod p = \pm 22,\ \sqrt{T} \pmod q = \pm 37.$$

Після цього, застосовуючи китайську теорему про залишки до пари $\pm 22 \pmod p$ та $\pm 37 \pmod q$, знаходимо корінь з T за модулем N:

$$s = \sqrt{T} \pmod N = \pm 3\,705 \text{ або } \pm 14\,373.$$

Чотири варіанти розшифрування 4 410, 5 851, 15 078, 16 519 отримуються зі співвідношення

$$s - \frac{B}{2} = s - \frac{12\,345}{2}.$$

Частина 5. Хеш-функції

5.1 Криптографічні хеш-функції

Криптографічна **хеш-функція** h – це функція, визначена на бітових рядках довільної довжини зі значеннями (**хеш-кодами** чи **хеш-значеннями**) в рядках бітів фіксованої довжини. Найважливіша властивість криптографічних хеш-функцій – їх **односторонність**: повинно бути неможливо в обчислювальному відношенні по елементу Y з множини значень хеш-функції підібрати таке x з області визначення, при якому $h(x) = Y$.

Але односторонності недостатньо. Хеш-функція h має бути **захищеною від повторень**: має бути обчислювально неможливо знайти два такі різні значення x та x', при яких $h(x) = h(x')$. Захищені від повторень хеш-функції будуються складніше, ніж односторонні.

Щоб знайти повтори у значеннях хеш-функції f, необхідно запам'ятовувати результати обчислення $f(x_1)$, $f(x_2)$, $f(x_3)$, …, поки не станеться повтор. Якщо значення цієї функції n-бітової довжини, очікуваний збіг з'явиться через $O(2^{n/2})$ ітерацій. При цьому, число кроків, необхідних для обчислення прообразу, має порядок $O(2^n)$ для коректно заданої хеш-функції. Отже, для досягнення необхідного рівня стійкості захищена від повторень функція, визначена на 80-бітових рядках, має приймати значення в 160-бітових рядках.

Навіть більше, криптографічна хеш-функція повинна мати властивість **захищеності від других прообразів**, тобто по заданому M має бути практично неможливо відшукати таке $M' \neq M$, для якого $h(M) = h(M')$.

Підсумовуючи, криптографічній хеш-функції необхідні наступні властивості:

- захищеність від відновлення прообразів: обчислювально неможливо знайти повідомлення із заданим значенням хеш-функції;
- захищеність від повторень: обчислювально неможливо знайти два повідомлення з однаковим значенням хеш-функції;

– захищеність від других прообразів: по заданому повідомленню неможливо визначити інше повідомлення з тим самим значенням хеш-функції.

Лема 9. Властивість захищеності від відновлення прообразів сильніша за захищеність від повторень та других прообразів.

Лема 10. Захищеність від других прообразів сильніша за захищеність від повторень.

Основний принцип проєктування хеш-функції полягає в тому, що її значення мають давати лавинний ефект: невелика зміна аргументу хеш-функції має сильно вплинути на її значення.

5.2 Сімейство MD4

Деякі хеш функції знайшли широке застосування в криптографії, серед них MD5, RIPEMD-160, SHA-1 та SHA-2. Всі ці хеш-функції походять від одного простішого алгоритму MD4 і за своєю природою ітераційні. В ході років було знайдено ряд вразливостей в майже усіх ранніх хеш-функціях сімейства, зокрема, MD4, MD5 та SHA-1, тому рекомендовано використовувати алгоритми SHA-2 або SHA-3 (концептуально інший алгоритм на основі криптографічної губки).

Сім основних алгоритмів сімейства MD4 мають наступні параметри:

– MD4 складається з 3 раундів по 16 кроків у кожному зі 128-бітовим вихідним рядком.
– MD5 складається з 4 раундів по 16 кроків у кожному зі 128-бітовим вихідним рядком.
– SHA-1 включає 4 раунди по 20 кроків у кожному, довжина його вихідних даних – 160 бітів.
– RIPEMD-160 складається з 5 раундів по 16 кроків у кожному і має довжину вихідного рядка 160 бітів.
– SHA-256 має 64 однокрокових раунди, а довжина його значення – 256 бітів.
– SHA-512 складається з 80 однокрокових раундів і видає рядки в 512 бітів.
– SHA-384 практично ідентичний SHA-512, його вихід урізаний до 384 бітів.

5.2.1 Алгоритм MD4

Розглянемо алгоритм MD4 як найпростіший у сімействі. В ньому беруть участь три порозрядні функції від трьох 32-бітових змінних

$$f(u,v,w) = (u \wedge v) \vee ((\neg u) \wedge w),$$

$$g(u,v,w) = (u \wedge v) \vee (u \wedge w) \vee (v \wedge w),$$

$$h(u,v,w) = u \oplus v \oplus w.$$

На протязі роботи алгоритму відслідковується поточний хеш-стан (H_1, H_2, H_3, H_4), початкові значення 32-бітових змінних якого дорівнюють

H_1 = 67452301, H_3 = 98BADCFE,

H_2 = EFCDAB89, H_4 = 10325476.

Для кожного раунду існують свої власні фіксовані константи ($y[i]$, $z[i]$, $w[i]$). При цьому

y[0..15] = 0;

y[16..31] = 5A827999,

y[32.. 47] = 6ED9EBA1.

Значення z та w мають вигляд

z[0..15] = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],

z[16..31] = [0,4,8,12,1,5,9,13,2,6,10,14,3,7,11,15],

z[32..47] = [0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15],

w[0..15] = [3,7,11,19,3,7,11,19,3,7,11,19,3,7,11,19],

w[16..31] = [3,5,9,13,3,5,9,13,3,5,9,13,3,5,9,13],

w[32..47] = [3,9,11,15,3,9,11,15,3,9,11,15,3,9,11, 15].

Потік даних складається з 16 слів, що одночасно завантажуються до масиву $X[j]$ ($0 \le j < 16$). Далі виконуються наступні перетворення (⋘ – циклічний зсув):

(A,B,C,D) = (H_1,H_2,H_3,H_4).

Виконати перший раунд:

```
for (int j=0;j<=15;j++){
  t = A + f(B,C,D) + x[z[j]] + y[j];
  (A,B,C,D) = (D,t<<<w[j],B,C)
}
```

```
Виконати другий раунд:
    for (int j=16;j<=31;j++){
      t = A + g(B,C,D) + x[z[j]] + y[j];
      (A,B,C,D) = (D,t<<<w[j],B,C)
    }
Виконати третій раунд:
    for (int j=32;j<=47;j++){
      t = A + h(B,C,D) + x[z[j]] + y[j];
      (A,B,C,D) = (D,t<<<w[j],B,C)
    }
(H1,H2,H3,H4) = (H1 + A, H2 + B, H3 + C, H4 + D).
```

Вихідні дані є конкатенацією остаточних значень змінних H_1, H_2, H_3, H_4.

5.3 Хеш-функції на основі блокових шифрів

Хеш-функції можна будувати за допомогою n-бітового блокового шифру E_k. Для цього повідомлення розбивається на блоки x_0, x_1, …, x_k. Вибір розміру блоку залежить від хеш-функції.

Значення хеш-функції збігаються з остаточним значенням H_k, отриманим за допомогою ітерацій

$$H_0 = IV, \qquad H_i = f(x_i, H_{i-1}).$$

Функція f може мати вигляд, зокрема

- хеш-функції Девіса-Мейєра $f(x_i, H_{i-1}) = E_{x_i}(H_{i-1}) \oplus H_{i-1}$,
- хеш-функції Матіса-Мейєра-Осеаса $f(x_i, H_{i-1}) = E_{g(H_{i-1})}(x_i) \oplus x_i$,
- хеш-функції Міагучі-Пренеля $f(x_i, H_{i-1}) = E_{g(H_{i-1})}(x_i) \oplus H_{i-1} \oplus x_i$.

Частина 6. Схеми цифрового підпису

6.1 Цифрові підписи

Загальна схема для електронних підписів має вигляд:

ПОВІДОМЛЕННЯ + секретний ключ А = ПІДПИС,

ПОВІДОМЛЕННЯ + ПІДПИС + ВІДКРИТИЙ КЛЮЧ А = ТАК/НІ.

Ця схема називається **схемою підпису з доповненням** (appendix), оскільки підпис додається в кінець повідомлення перед його відправкою. Отримане повідомлення подається на вхід процедури перевірки підпису.

Інший варіант – **схема підпису з відновленням повідомлення** (message recovery), коли повідомлення відновлюється на виході процедури перевірки підпису:

ПОВІДОМЛЕННЯ + секретний ключ А = ПІДПИС,

ПІДПИС + ВІДКРИТИЙ КЛЮЧ А = ТАК/НІ + ПОВІДОМЛЕННЯ.

Головна проблема цифрових підписів полягає в тому, яким відкритим ключам можна довіряти, по суті – перевірці їх справжності.

Більш формально, схема цифрового підпису складається з двох перетворень:

- секретне перетворення підпису s,
- відкрите перетворення перевірки V.

Вважатимемо, що використовується схема підпису з відновленням повідомлення (різниця зі схемою підпису з доповненням буде невеликою).

Абонент, надсилаючи повідомлення m, обчислює $S = s(m)$ і передає результат S, де S – цифровий підпис на повідомленні m. При цьому початкове повідомлення можна додатково зашифрувати, якщо необхідна секретність.

Отримувач підпису S застосовує відкрите перетворення перевірки V до S та отримує на виході процедури повідомлення m та деякий біт v, який відповідає за результат перевірки підпису. Якщо перевірка пройшла позитивно, то адресат отримує впевненість у цілісності повідомлення (воно не було змінене під час

передачі), в його оригінальності (повідомлення надіслане саме відправником) та у відсутності ренегатства (відправник не зможе заперечувати надсилання).

Розглянемо різноманітні схеми цифрових підписів.

6.2 Алгоритм DSA

Алгоритм DSA **(**Digital Signature Algorithm) – це алгоритм із використанням відкритого ключа для створення електронного підпису, але не для шифрування. В основі його – проблема обчислення дискретного логарифму. Найновіша версія стандарту DSS (Digital Signature Standard) 2023 року забороняє використовувати основний алгоритм DSA для генерації цифрових підписів, проте забезпечує перевірку вже існуючих підписів. Розглянемо цей алгоритм.

Вибір параметрів проходить так.

1. Обирається хеш-функція $H(x)$. В оригінальній версії використовувалася SHA-1, згодом стандартом було рекомендовано хеш-функцію SHA-2.
2. Вибирається велике просте число q, розмірність якого у бітах збігається з виходом хеш-функції.
3. Обирається просте число p таке, що $p - 1$ ділиться на q, тобто $(p-1)/q$ – ціле число. Розмірність p визначає криптостійкість системи. Зараз для систем, які повинні бути стійкими до 2030 року, рекомендується довжина 2048 (чи 3072) біт.
4. Вибирається довільне число h і число g такі, що $g = h^{(p-1)/q} \bmod p$, причому $g \neq 1$; якщо останнє не виконується, то береться інше h. Часто беруть $h = 2$.

Таким чином, трійка чисел (p, q, g) є параметрами домену та поширюється між користувачами домену (тобто не є секретною).

Генерація відкритого та закритого ключа: закритий ключ є випадковим числом x в інтервалі $(0, q)$, відкритий ключ обчислюється за формулою $y = g^x \bmod p$.

Для підписання повідомлення m виконаємо наступне.

1. Вибирається випадкове число k з інтервалу $(0, q)$.
2. Обчислюється $r = (g^k \bmod p) \bmod q$.

3. Обчислюється $s = (k^{-1} \cdot (H(m) + x \cdot r)) \bmod q$. У випадку $r = 0$ чи $s = 0$ треба буде обрати інше k.

Підписом виступає пара чисел (r, s)

Перевірка відбувається за алгоритмом.

1. Обчислюється $w = s^{-1} \bmod q$.

2. Обчислюється $u = (H(m) \cdot w) \bmod q$.

3. Обчислюється $z = (r \cdot w) \bmod q$.

4. Обчислюється $v = ((g^u \cdot y^z) \bmod p) \bmod q$. Якщо в результаті $v = r$, то підпис вірний.

Розглянута схема цифрового підпису є коректною в тому сенсі, що кожен бажаючий може перевірити її коректність.

Оскільки $s = (k^{-1} \cdot (H(m) + x \cdot r)) \bmod q$, звідси

$$k = H(m) \cdot s^{-1} + x \cdot r \cdot s^{-1} = H(m) \cdot w + x \cdot r \cdot w,$$

бо відомо, що $w = s^{-1}$. Тоді отримуємо

$$g^k = g^{H(m)w} \cdot g^{xrw} = g^{H(m)w} \cdot y^{rw} = g^u \cdot y^{rw} = g^u \cdot y^z,$$

оскільки $y = g^x$, $u = H(m) \cdot w$ та $z = rw$. З викладеного вище видно, що

$$r = (g^k \bmod p) \bmod q = (g^u \cdot y^z \bmod p) \bmod q = v.$$

Розглянемо приклад. Нехай $q = 13$, $p = 53$ (очевидно, що $(p-1)/q$ – ціле) і $g = 16$. Нехай секретний ключ $x = 3$. Тоді відкритий ключ має вигляд $y = g^x \bmod p = 15$.

Припустимо, необхідно підписати повідомлення із хеш-значенням $H(m) = 5$.

Виберемо довільне число $k = 2$, тоді

$$r = (g^k \bmod p) \bmod q = 5,$$

$$s = (k^{-1} \cdot (H(m) + x \cdot r)) \bmod q = 10, \text{ де } k^{-1} = 7 \pmod{13}.$$

Таким чином, підпис має вигляд (5, 10). Перевіримо підпис:

$$u = \mathrm{H} \cdot s^{-1} \pmod q = 7,$$

$$z = r \cdot s^{-1} \pmod q = 7,$$

$$v = ((g^u \cdot y^z) \bmod p) \bmod q = 5.$$

Отримали $r = v = 5$, тобто підпис коректний.

6.3 Підпис Шнорра

Схема Шнорра – ще один протокол ідентифікації, надійність якого заснована на складності дискретного логарифмування. Цей алгоритм дозволяє проводити попередні обчислення, що зручно при малих обчислювальних ресурсах.

Процедура вибору параметрів системи.

1. Вибираються просте p і просте q, такі що $p \bmod q = 1$.

2. Обирається елемент b, для якого виконується умова $b^q \pmod p = 1$ та $b \neq 1$.

3. Параметри (p, q, b) вільно публікуються.

4. Вибирається параметр безпеки t, $2^t < q$.

Генерація відкритого та закритого ключа: секретний ключ a обирається в інтервалі $0 < a < q$, відкритий ключ v обчислюється по формулі $v = b^{-a} \pmod p$.

Розглянемо послідовність подій.

1. А вибирає випадкове число r ($1 \le r \le q - 1$) і обчислює $x = b^r \pmod p$. Ці дії можуть бути пророблені заздалегідь.

2. Значення x надсилається В.

3. В надсилає випадкове e із діапазону $1 \le e \le 2^t - 1$.

4. А обчислює $y = (a \cdot e + r) \pmod q$.

5. А відсилає В отримане значення y.

6. В обчислює $z = b^y \cdot v^e \pmod p$.

7. В перевіряє $z = x$.

Безпека алгоритму залежить від параметра безпеки t: складність розкриття приблизно дорівнюватиме 2^t.

Розглянемо приклад. Виберемо прості $p = 48\,731$ та $q = 443$, візьмемо $b = 11\,444$ ($b^q \pmod p = 1$).

Тоді системними параметрами будуть (48 731, 443, 11 444), а $t = 8$.

Нехай А вибирає закритий ключ $a = 357$ і потім обчислює відкритий ключ $v = b^{-a} \pmod p = 7\,355$.

А випадково обирає $r = 274$ та надсилає В значення $x = b^r \pmod p = 37\ 123$. У відповідь В відсилає $e = 129$.

А відсилає В значення $y = (a \cdot e + r) \pmod q = 255$.

В обчислює $z = b^y \cdot v^e \pmod p = 37\ 123$ та ідентифікує А, оскільки $z = x$.

Алгоритм Шнорра можна модифікувати для використання в якості цифрового підпису.

Нехай А хоче надіслати В повідомлення m; причому В має переконатися, що повідомлення надійшло саме від А.

Генерація підпису. А вибирає випадкове число r $(1 \le r \le q - 1)$ і обчислює $x = b^r \pmod p$. Нехай обрано хеш-функцію H. А поєднує повідомленя m з x та хешує результат: $E = H(m||x)$ і далі обчислює $y = (a \cdot E + r) \pmod q$. Підписом є пара (E, y).

Перевірка підпису. В обчислює $z = b^y \cdot v^E \pmod p$. Потім z та отримане повідомлення m' пропускаються через хеш-функцію: $E' = H(m'||z)$. Якщо $E = E'$, то підпис вважається справжнім.

6.4 Протокол Фіата-Шаміра

Протокол Фіата-Шаміра – це відомий протокол ідентифікації з нульовим розголошенням. «Нульове розголошення» означає, що користувач доводить, що знає свій закритий ключ («секрет»), не розкриваючи його, тим самим підтверджуючи свою ідентичність. Розглянемо цей протокол.

Попередні дії. Довірений центр T обирає та публікує модуль $n = p \cdot q$, де p та q – великі прості числа, що тримаються в таємниці.

Генерація відкритого та закритого ключа. Претендент вибирає число s, взаємно просте з n з інтервалу $[1, n - 1]$ – секретний ключ, обчислює $v = s^2 \pmod n$ та реєструє v в T як свій відкритий ключ.

Претендент доводить свою ідентичність верифікатору протягом t раундів, кожен з яких виглядає наступним чином.

1. Претендент А вибирає випадкове r, що належить інтервалу $[1, n - 1]$.

2. А обчислює $x = r^2 \pmod n$.

3. А відсилає значення x верифікатору.

4. Верифікатор В випадково обирає біт e ($e = 0$ або $e = 1$) і надсилає його претенденту.

5. А обчислює $y = r \cdot s^e \pmod n$. Якщо $e = 0$, то $y = r$, інакше $y = r \cdot s \pmod n$.

6. Претендент відправляє значення y верифікатору.

7. Якщо $y = 0$, В відкидає підпис. Іншими словами, А не вдалося довести знання s.

8. Інакше верифікатор перевіряє, чи дійсно $y^2 = x \cdot v^e \pmod n$. Якщо це так, то відбувається перехід до наступного раунду протоколу.

Припустимо, що претендент хоче обдурити, тоді А вибирає випадкове r і надсилає $x = r^2 \pmod n$. Тоді якщо $e = 0$, то А вдало повертає верифікатору $y = r$, в разі $e = 1$, претендент не зможе правильно відповісти, бо не знає s, а видобути квадратний корінь з v за модулем n складно.

Розглянемо приклад. Припустимо, довірений центр вибрав прості $p = 683$ та $q = 811$ і опублікував $n = 683 \cdot 811 = 553\,913$. А вибирає $s = 43\,215$ та обчислює $v = 43\,215^2 \pmod{553\,913} = 1\,867\,536\,225 \pmod{553\,913} = 295\,502$.

А вибирає випадкове $r = 38\,177$ і рахує:

$$x = 38\,177^2 \pmod{553\,913} = 1\,457\,483\,329 \pmod{553\,913} = 138\,226.$$

Якщо В відправив $e = 0$, А повертає $y = r = 38\,177$. Інакше, А повертає

$$y = 38\,177 \cdot 43\,215 \pmod{553\,913} = 1\,649\,819\,055 \pmod{553\,913} = 266\,141.$$

Якщо e дорівнювало 0, то

$y^2 = 38\,177^2 \pmod{553\,913} = 1\,457\,483\,329 \pmod{553\,913} = 138266$. Підтверджено.

Інакше, $y^2 = 266\,141^2 \pmod{553\,913} = 70\,831\,031\,881 \pmod{553\,913} = 514832$ та $x \cdot v = 138\,226 \cdot 295\,502 \pmod{553\,913} = 40\,846\,059\,452 \pmod{553\,913} = 514\,832$. Підтверджено.

6.5 Підпис Ніберга-Руппеля

Схема Ніберга-Руппеля – це приклад підпису на основі дискретного логарифмування з відновленням повідомлення.

Всі схеми підпису з відновленням повідомлення використовують відкриту функцію надлишковості f. Вона перетворює фактичне повідомлення на дані, які потім підписуються. Однак, на відміну від хеш-функції, функція f має обернену. Можливий вигляд такої функції: $f(m) = m||m$.

Будемо вважати, що множина значень функції f – цілі числа, що належать групі $A = \mathbb{F}_P^* = \{1, \dots, P-1\}$. Нехай число Q ділить націло $P-1$, тобто $(P-1)/Q$ належить множині цілих чисел, а G – твірна підгрупи групи A порядку Q.

Вибір відкритого та секретного ключа: секретний ключ – випадкове значення x з інтервалу (0, …, Q), відкритий ключ – значення $Y = G^x \pmod P$.

Алгоритм підпису має вигляд:

1. Вибирається випадкове число k.
2. Обчислюється $R = G^k \pmod P$.
3. Знаходиться $E = f(m) \cdot R \pmod P$, де m – вихідне повідомлення.
4. Визначається $S = x \cdot E + k \pmod Q$.

Таким чином, підпис має вигляд (E, S).

Процедура перевірки підпису:

1. Обчислюють $U_1 = G^S Y^{-E} \pmod P$.
2. Обчислюють $U_2 = EU_1^{-1} \pmod P$.

Якщо U_2 має вигляд $f(m)$, то підпис підтверджується і повідомлення відновлюється за формулою $m = f^{-1}(U_2)$, інакше підпис відкидається.

Розглянемо приклад. Нехай $Q = 101$, $P = 607$ та $G = 601$. Виберемо ключову пару у вигляді: $x = 3$, $Y = G^x \pmod P = 391$.

Нехай потрібно підписати повідомлення $m = 12$. Тоді: вибираємо довільне число $k = 45$ і обчислюємо $R = G^k \pmod P = 143$.

Припустимо, що $f(m) = m + 2^4 m$. Тоді обчислюємо $f(m) = 204$,

$$E = f(m) \cdot R \pmod P = 36,$$

$$S = x \cdot E + k \pmod Q = 52.$$

Таким чином, підпис має вигляд (36, 52).

Тепер перевіримо підпис. Обчислюємо:

$$U_1 = G^S Y^{-E} \pmod{P} = 143,$$

$$U_2 = E U_1^{-1} \pmod{P} = 204.$$

Необхідно переконатися в тому, що знайдене число U_2 можна представити у вигляді $m + 2^4 m$, тобто розв'язати рівняння $m + 2^4 m = 204$. Очевидно, тут $m = 12$, значить підпис достовірний і підписувалося повідомлення $m = 12$.

Частина 7. Розподіл ключів для симетричних алгоритмів

7.1 Проблема розподілу симетричних ключів

Для використання симетричних криптоалгоритмів потрібна система розподілу секретних ключів. Існує кілька типів симетричних ключів:

1. Статичний (довгостроковий) ключ. Час життя цього ключа може досягати кількох років.

2. Ефемерний чи сеансовий (короткочасний) ключ. Час життя такого ключа від кількох секунд до одного дня.

Можна виділити декілька розв'язків проблеми розподілу ключів.

- Фізичний розподіл. Такий розподіл можуть здійснювати кур'єри.
- Розподіл за допомогою протоколів із секретним ключем. В цьому випадку розподіл здійснюється за допомогою так званих довірених центрів.
- Розподіл за допомогою протоколів з відкритим ключем. Тут застосовуються асиметричні алгоритми.

Насамперед розглянемо питання, пов'язані зі зберіганням довгострокових секретних ключів, тобто проблему розділення секрету. Існує два основних способи розв'язання цієї проблеми.

1 спосіб. Ключ поділяється на N частин $k = \{k_1, \ldots, k_n\}$ між n хранителями. Тоді для злому такого люча необхідно мати всі N частин.

2 спосіб. Схема порогового поділу (схема Шаміра). У цій схемі ключ також ділиться на N частин, проте для повного відновлення ключа необхідно лише $T < N$ частин.

Нехай ключ K розділяється на N частин. Виберемо просте число $P > N + 1$. Вважатимемо, що ключ належить полю $\mathbb{F}_p$. Виберемо довільні $N - 1$ значень X_i, $i = 1, .., N$, що належать полю $\mathbb{F}_p$. Значення X_i повідомляються всім хранителям.

Вибирається $T - 1$ елемент $\{a_1, \ldots, a_{T-1}\}$, що належить полю $\mathbb{F}_p$, потім будується многочлен

$$F(X) = K + \sum_{j=1}^{T-1} a_j X^j.$$

Після цього знаходяться значення $y_i = F(X_i)$, які роздаються кожному хранителю (по одному кожному).

Якщо L хранителів хочуть відновити ключ, вони обмінюються значеннями y_i, а потім розв'язують систему рівнянь

$$y_1 = K + a_1 X_1 + \cdots + a_{T-1} X_1^{T-1}$$

.....

$$y_L = K + a_1 X_L + \cdots + a_{T-1} X_L^{T-1}$$

Система матиме єдиний розв'язок, якщо $L \geq T$.

7.2 Протокол широкоротої жаби

Якщо існує n користувачів, які бажають обмінюватися секретною інформацією, для цього необхідно $n(n-1)/2$ ключів.

Протокол широкоротої жаби (Wide-Mouthed Frog) було запропоновано Майклом Барроузом. Нехай є два суб'єкти A (Alice, Аліса) та B (Bob, Боб), охочих домовитися про спільний секретний ключ K_{ab}.

Для реалізації цього протоколу необхідна синхронізація годинників A та B. Нехай S – довірена особа (тобто особа, якій довіряють A та B, trusted third party – TTP). Тоді протокол складається з обміну двома повідомленнями (рис. 12):

1. $A \rightarrow S: A, \{T_a, B, K_{ab}\}_{K_{as}}$,

2. $S \rightarrow B: \{T_s, A, K_{ab}\}_{K_{bs}}$.

Отримавши перше повідомлення, довірена особа S розшифровує останню частину повідомлення від A і перевіряє, що часова мітка T_a близька до поточного часу. Тоді шифрує ключ K_{ab} разом зі згенерованою власною часовою міткою T_s та надсилає утворену шифрограму B. При отриманні повідомлення від S, B у свою чергу його розшифровує, переконується в свіжості часової мітки і може прочитати ключ K_{ab} та ім'я A особи, яка хоче надіслати секретну інформацію.

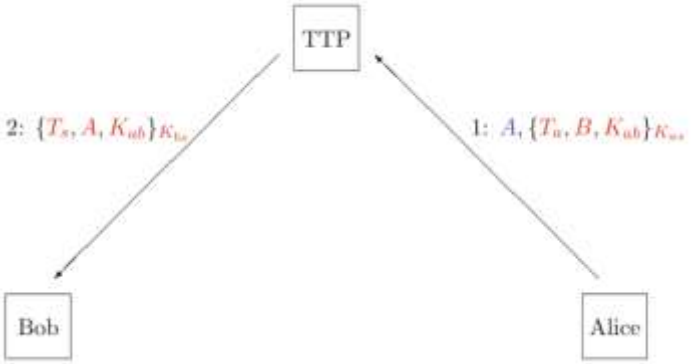


Рисунок 12. – Протокол широкоротої жаби.

Коректна тимчасова мітка означає час надсилання ключа. Однак користувач А міг згенерувати цей ключ давно і зберігати його на своєму жорсткому диску, і з цього ключа могли зняти копію. Варто зазначити, що цей протокол працюватиме коректно лише за синхронізованого годинника його учасників.

7.3 Протокол Нідгема-Шредера

Це ще один протокол розподілу секретних ключів. В протоколі Нідгема-Шредера присутні 3 суб'єкти: A, B та S (тут S довірена особа для A та B). Обмін повідомленнями відбувається за наступною схемою (рис. 13):

$A \to S: A, B, N_a,$

$S \to A: \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}},$

$A \to B: \{K_{ab}, A\}_{K_{bs}},$

$B \to A: \{N_b\}_{K_{ab}},$

$A \to B: \{N_b - 1\}_{K_{ab}}.$

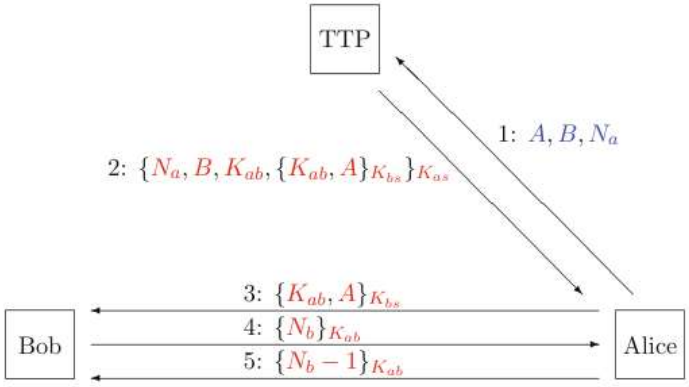


Рисунок 13. – Протокол Нідгема-Шредера.

1. A надсилає у відкритому вигляді повідомлення до S, в якому міститься її ім’я (тобто ініціатора обміну), ім'я B (тобто ім'я того, з ким хоче зв'язатися A) та N_a – деяка числова мітка-нонс.

2. S після отримання повідомлення від A, генерує ключ K_{ab} і надсилає назад до A. Туди також включена мітка від A N_a й зашифроване повідомлення з ключем сесії K_{ab} для надсилання B.

3. A, отримавши повідомлення від S, розшифровує його та виділяє з нього частину з зашифрованим ключем і в незмінному вигляді пересилає B.

4. B надсилає A згенеровану мітку N_b, зашифровану ключем K_{ab}, щоб засвідчити, що у нього є ключ.

5. A, отримавши повідомлення, генерує повідомлення, що залежить від значення N_b, шифрує його ключем K_{ab} і пересилає, таким чином підтверджуючи, що вона досі на зв’язку.

Основний недолік протоколу Нідхейма-Шредера полягає в тому, що в результаті його роботи у користувача немає підстав вважати, що отриманий ключ є новим. Зловмисник, знайшовши повідомлення та ключ попередніх сеансів, може використати старі листи замість останніх трьох повідомлень, у яких згадується B. Таким чином зловмисник може обдурити B, змушуючи його прийняти свій ключ, в той час, як B думає, що спілкується з A.

7.4 Протокол Цербер (Kerberos)

Протокол Цербер був розроблений у Массачусетському технологічному інституті в 1987 році. В його основі лежать ідеї протоколу Нідгема-Шредера. Сучасна версія Kerberos використовується в ОС Windows та багатьох інших системах.

Передбачається, що комп'ютерна мережа складається з клієнтів та сервера, причому клієнтами можуть бути користувачі, програми чи служби. Цербер зберігає центральну базу даних, що включає як клієнтів, так і їх секретні ключі. Метою Kerberos є ідентифікація клієнтів та генерування для них сеансових

ключів. Крім того, Цербер може служити системою надання доступу до різноманітних послуг та ресурсів, для цього використовуються сервер автентифікації (Authentication Server, AS) та сервер видачі квитків (Ticket-granting Server, TGS), який видає сертифікати доступу до ресурсів.

Обмін повідомленнями цього протоколу здійснюється за схемою (рис. 14):

$A \to S: A, B,$

$S \to A: \{T_S, L, K_{ab}, B, \{T_S, L, K_{ab}, A\}_{K_{bs}}\}_{K_{as}},$

$A \to B: \{T_S, L, K_{ab}, A\}_{K_{bs}}, \{A, T_A\}_{K_{ab}},$

$B \to A: \{T_A + 1\}_{K_{ab}}.$

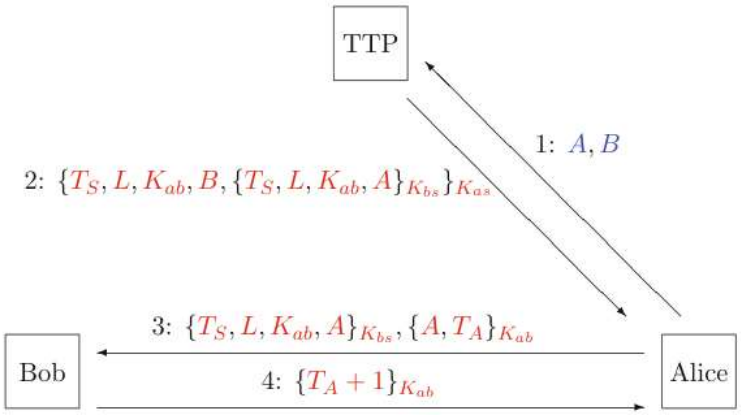


Рисунок 14. – Цербер.

Припустимо, A хоче скористатися ресурсами B. Вона звертається до довіреної особи S з відповідним повідомленням. S створює сертифікат для B, зашифрований ключем K_{bs}, і надсилає його A для передачі B, а також власну копію K_{ab}. Сертифікат містить ключ K_{ab}, час його життя L і часову мітку T_S. Виданий сертифікат використовується для підтвердження особи A при зверненні до B шляхом надсилання зашифрованої часової мітки. Таким чином, усуваються недоліки, притаманні протоколу Нідгема-Шредера, але за вимоги синхронізації годинників.

Розглянемо протокол Kerberos докладніше. Роль довіреної особи в протоколі грає Центр розподілу ключів (Key Distribution Center, KDC). KDC є службою, що працює на фізично захищеному сервері. Він підтримує базу даних з

інформацією про облікові записи всіх головних абонентів безпеки своєї області (домену, хоча цього терміну немає у Windows ще починаючи з Windows 2000).

Разом з інформацією про кожного абонента безпеки в базі даних KDC зберігається криптографічний ключ, відомий лише цьому абоненту та службі KDC. Цей ключ, який називають **довгостроковим**, використовується для зв'язку користувача системи безпеки із центром розподілу ключів. В більшості практичних реалізацій протоколу Цербер довгострокові ключі створюються на основі пароля користувача.

Отже, відбуваються наступні кроки.

1. Якщо клієнту потрібно звернутися до сервера, він передусім надсилає запит до центру KDC.

2. У відповідь на запит клієнта, який має намір підключитися до сервера, KDC направляє обидві копії сеансового ключа (ключ сесії, session key) клієнту. Повідомлення, призначене клієнту, шифрується за допомогою довгострокового ключа клієнта, а сеансовий ключ для сервера разом з інформацією клієнта вкладається в блок даних, який отримав назву сеансового квитка (session ticket). Також сеансовий квиток повністю шифрується за допомогою довгострокового ключа сервера, який знають лише служба KDC та даний сервер. Після цього вся відповідальність за обробку квитка, що містить зашифрований сеансовий ключ, покладається на клієнта, який має доставити його на сервер.

3. Отримавши відповідь KDC, клієнт видобуває з нього сеансовий квиток і свою копію сеансового ключа, які поміщає в безпечне сховище (воно розташоване не на диску, а в оперативній пам'яті). Коли виникає необхідність зв'язатися з сервером, клієнт надсилає йому повідомлення, що складається з квитка, зашифрованого із застосуванням довгострокового ключа цього сервера, та власного автентифікатора, зашифрованого за допомогою сеансового ключа. Цей квиток у комбінації з автентифікатором якраз і складає «посвідчення», за яким сервер визначає «особистість» клієнта.

4. Сервер, отримавши «посвідчення особи» клієнта, за допомогою свого секретного ключа розшифровує сеансовий квиток і бере з нього сеансовий ключ,

який використовує для розшифрування автентифікатора клієнта. Якщо все відбувається нормально, робиться висновок, що посвідчення клієнта видано довіреним посередником, тобто службою KDC.

5. Клієнт може вимагати у сервера проведення взаємної автентифікації. У цьому випадку сервер за допомогою своєї копії сеансового ключа шифрує мітку часу з автентифікатора клієнта і в такому вигляді пересилає її йому як власний автентифікатор.

Перевагою сеансових квитків є те, що серверу не потрібно зберігати сеансові ключі для зв'язку з клієнтами. Вони зберігаються в кеші облікових даних (credentials cache) клієнта, який направляє квиток на сервер щоразу, коли хоче зв'язатися з ним. Сервер, зі свого боку, отримавши від клієнта квиток, розшифровує його та виймає сеансовий ключ. Коли потреба в цьому ключі зникає, сервер може просто стерти його зі своєї пам'яті.

7.4.1 Квитки на видачу квитків

Довгостроковий ключ користувача створюється на основі його пароля. Коли комп'ютер проходить реєстрацію, клієнт Kerberos, встановлений на робочій станції, пропускає вказаний користувачем пароль через хеш-функцію (всі реалізації протоколу Kerberos 5 повинні обов'язково підтримувати алгоритм DES-CBC-MD5, хоча можуть застосовуватися інші алгоритми). В результаті формується криптографічний ключ.

У центрі KDC довгострокові ключі клієнтів зберігаються у базі даних з обліковими записами користувачів. Отримавши запит від клієнта Kerberos, встановленого на робочій станції, KDC звертається до своєї бази даних, знаходить у ній обліковий запис потрібного користувача та видобуває з відповідного поля довгостроковий ключ користувача. Процес обчислення однієї копії ключа по паролю і вилучення іншої його копії з бази даних виконується одноразово за сеанс, коли користувач заходить до мережі вперше. Відразу після отримання пароля користувача та обчислення довгострокового ключа клієнт Kerberos робочої станції запитує сеансовий квиток і сеансовий ключ, які використовуються в усіх наступних транзакціях з KDC протягом поточного сеансу роботи в мережі.

На окремий запит користувача KDC відповідає спеціальним сеансовим квитком для самого себе, так званим квитком на видачу квитків (ticket-granting ticket, квиток TGT). Як і звичайний сеансовий квиток, TGT містить копію сеансового ключа зв'язку служби (в нашому випадку KDC) з клієнтом. Квиток TGT шифрується за допомогою довгострокового ключа служби KDC, а клієнтська копія сеансового ключа – за допомогою довгострокового ключа користувача.

Отримавши відповідь KDC на свій початковий запит, клієнт розшифровує свою копію сеансового ключа, використовуючи для цього копію довгострокового ключа користувача зі своєї кеш-пам'яті. Після цього довгостроковий ключ, отриманий з пароля користувача, видаляється з пам'яті, оскільки подальший зв'язок з KDC шифруватиметься за допомогою сеансового ключа. Як і всі інші сеансові ключі, він має тимчасовий характер і діє до закінчення терміну дії квитка TGT, або ж до виходу користувача з системи, тому подібний ключ називають сеансовим ключем реєстрації (logon session key).

7.4.2 Автентифікація за межами домену

Функції центру KDC можна поділити на дві категорії: службу автентифікації, до якої входить генерація квитків TGT, та службу видачі квитків, що створює сеансові квитки. Такий розподіл дозволяє застосовувати протокол Kerberos також за межами рідного домену. Клієнт, який отримав квиток TGT зі служби автентифікації одного домену, може скористатися ним для отримання сеансових квитків у службах видачі квитків інших доменів.

Розглянемо приклад міждоменної автентифікації. Припустимо, в нас є два домени A та B. Тоді достатньо домовитися про єдиний міждоменний ключ (в ОС Windows такий ключ генерується автоматично, коли між доменами встановлюються довірчі відносини). Як тільки ключ створено, служба видачі квитків кожного домену реєструється в центрі KDC іншого домену як головний абонент безпеки.

В результаті служба видачі квитків кожного домену починає розглядати службу видачі квитків іншого домену як ще одну свою службу. Завдяки цьому

клієнт, який пройшов автентифікацію та зареєструвався в системі, може запитувати та отримувати сеансові квитки для іншого домену.

Нехай користувач із домену A запитує доступ до сервера з домену B.

1. Клієнт Kerberos, встановлений на робочій станції користувача домену A, надсилає запит до служби видачі квитків свого домену, в якому просить видати сеансовий квиток для доступу на потрібний сервер.

2. Служба видачі квитків домену A перевіряє список своїх абонентів безпеки та переконується, що такого сервера серед них немає. Тому вона направляє клієнту так званий квиток переадресації (referral ticket), який є TGT, зашифрований за допомогою міждоменного ключа, спільного для служб KDC доменів A і B.

3. Отримавши квиток переадресації, клієнт використовує його для підготовки іншого запиту на сеансовий ключ. Потім запит пересилається до служби видачі квитків того домену, де знаходиться обліковий запис потрібного сервера, тобто домену B.

4. Його служба видачі квитків намагається розшифрувати квиток переадресації за допомогою власної копії міждоменного ключа. В разі вдалої спроби, центр KDC надсилає клієнту сеансовий квиток на доступ до відповідного сервера свого домену.

7.4.3 Підпротоколи Kerberos

Протокол Kerberos містить у собі три підпротоколи.

- Authentication Service Exchange (обмін зі службою автентифікації, AS Exchange) використовується службою KDC для передачі клієнту сеансового ключа реєстрації та квитка TGT.
- Ticket-Granting Service Exchange (обмін зі службою видачі квитків, TGS Exchange) служить для розсилки службових сеансових ключів та сеансових ключів самої служби KDC.
- Client/Server Exchange (клієнт-серверний обмін, CS Exchange) використовується клієнтом для пересилання сеансового квитка доступу до служб.

Розглянемо схему роботи підпротоколу **AS Exchange**.

1. Користувач входить в мережу, ввівши своє ім'я користувача і пароль. Встановлений на робочій станції клієнт Kerberos перетворює пароль на криптографічний ключ та зберігає отриманий результат в кеш облікових даних. Після цього клієнт надсилає службі автентифікації центру KDC запит **KRB_AS_REQ** (Kerberos Authentication Service Request – запит до служби автентифікації Kerberos). Перша частина повідомлення містить ідентифікатор користувача, а також ім'я служби, якій потрібно посвідчення. У другій частині вказуються дані попередньої автентифікації (pre-authentication data), які підтверджують, що користувач знає пароль. Зазвичай це мітка часу, зашифрована з довгостроковим ключем користувача.

2. Отримавши запит **KRB_AS_REQ**, служба KDC звертається до своєї бази даних і знаходить у ній довгостроковий ключ користувача, після чого розшифровує дані попередньої автентифікації та оцінює мітку часу, що міститься в них. Якщо перевірка пройшла успішно, KDC робить висновок, що дані попередньої автентифікації були зашифровані з довгостроковим ключем користувача і, отже, надійшли від клієнта, ім'я якого міститься в першій частині повідомлення.

3. Після завершення перевірки особи користувача KDC генерує посвідчення, що підтверджує, що клієнт Kerberos на його робочій станції має право звернутися до служби видачі квитків. Цей процес має два етапи.

Етап 1. KDC створює сеансовий ключ реєстрації та шифрує його копію за допомогою довгострокового ключа користувача.

Етап 2. Служба включає ще одну копію сеансового ключа реєстрації у квиток TGT. Туди ж додається й інша інформація про користувача, приміром його дані авторизації. Потім KDC шифрує квиток TGT із власним довгостроковим ключем і, нарешті, включає зашифрований сеансовий ключ реєстрації разом з квитком TGT у пакет **KRB_AS_REP** (Kerberos Authentication Service Reply – відповідь служби автентифікації Kerberos), який надсилає клієнту.

4. Отримавши таке повідомлення, клієнт розшифровує сеансовий ключ реєстрації та зберігає його в кеші облікових даних. Після цього з повідомлення видобувається квиток TGT, який також поміщається в цю кеш-пам'ять.

Тепер перейдемо до розгляду схеми роботи підпротоколу **TGS Exchange.**

1. Клієнт Kerberos, встановлений на робочій станції користувача, просить посвідчення на доступ до служби певного сервера, для чого надсилає до служби KDC повідомлення **KRB_TGS_REQ** (Kerberos Ticket-Granting Service Request – запит до служби видачі квитків Kerberos). До нього включається ім'я користувача, автентифікатор, зашифрований за допомогою сеансового ключа реєстрації користувача, квиток TGT, отриманий за допомогою підпротоколу AS Exchange, а також ім'я служби, на доступ до якої потрібен квиток

2. Отримавши запит **KRB_TGS_REQ**, служба KDC за допомогою власного секретного ключа розшифровує квиток TGT і видобуває з нього сеансовий ключ реєстрації користувача, який використовується для розшифровки автентифікатора. Якщо вміст автентифікатора витримує перевірку, служба KDC дістає з квитка TGT реєстраційні дані користувача та генерує сеансовий ключ, спільний для клієнта та сервера.

3. Одну копію цього ключа KDC шифрує за допомогою сеансового ключа реєстрації користувача, а іншу разом із даними авторизації користувача поміщає у квиток, який шифрує за допомогою довгострокового ключа сервера. Після цього посвідчення користувача включається до пакету **KRB_TGS_REP** (Kerberos Ticket-Granting Service Reply – відповідь служби видачі квитків Kerberos) та відправляється на його робочу станцію.

4. Отримавши таке повідомлення, клієнт за допомогою сеансового ключа реєстрації користувача розшифровує сеансовий ключ доступу до служби та поміщає його до кешу облікових даних. Після цього клієнт отримує квиток на доступ до служби і зберігає його сюди ж.

Нарешті, розглянемо схему роботи підпротоколу **CS Exchange**

1. Клієнт Kerberos, встановлений на робочій станції користувача, звертається до служби сервера, для чого надсилає їй запит **KRB_AP_REQ**

(Kerberos Application Request – запит додатку Kerberos). Це повідомлення містить автентифікатор користувача, зашифрований за допомогою сеансового ключа для служби сервера, квиток, отриманий за допомогою протоколу TGS Exchange, а також прапорець, що вказує на бажання клієнта провести взаємну автентифікацію (причому наявність або відсутність цього прапорця визначається конфігурацією Kerberos; він встановлюється без запиту користувача).

2. Отримавши повідомлення **KRB_AP_REQ**, служба сервера розшифровує квиток, дістає з нього дані авторизації користувача та сеансовий ключ, за допомогою якого відразу розшифровує автентифікатор користувача. Якщо позначка часу, закладена в нього, витримує перевірку, сервер шукає в запиті прапорець взаємної автентифікації. Знайшовши його, сервер шифрує мітку часу з автентифікатора користувача сеансовим ключем, включає отриманий результат до пакету **KRB_AP_REP** (Kerberos Application Reply – відповідь додатку Kerberos) і надсилає його на робочу станцію користувача.

3. Після отримання пакету клієнт робочої станції користувача розшифровує автентифікатор сервера, використовуючи для цього сеансовий ключ і порівнює отриману мітку часу з вихідною. Їх збіг означає, що зв'язок встановлений з потрібною службою і можна розпочати обмін інформацією.

7.4.4 Структура квитка

Структура квитка та різних повідомлень Kerberos докладно описана в документі RFC 4120.

Розглянемо основні поля, що містяться в квитку. Перші три поля не шифруються. Інформація, що міститься в них, пересилається відкрито, що дозволяє клієнту використовувати її для управління квитками, що зберігаються в кеш-пам'яті. Всі інші належать до частини зашифрованої інформації **enc-part**.

1) **tkt-vno**: номер версії формату квитка. Для Kerberos 5 вказується цифра 5.

2) **realm**: ім'я області (домена), де згенеровано квиток. Служба KDC може створювати квитки тільки для серверів власного домену, тому тут по суті вказується ім'я області, де розташований сервер.

3) **sname**: ім'я сервера. Інші поля шифруються за допомогою секретного ключа сервера.

4) **flags**: прапорці квитка.

5) **key**: сеансовий ключ.

6) **crealm**: ім'я області (домен) клієнта.

7) **cname**: ім'я клієнта

8) **transited**: список областей Kerberos, які брали участь в автентифікації клієнта, якому видано цей квиток.

9) **authtime**: час первинної автентифікації клієнта. Служба KDC заповнює це поле у момент генерації квитка TGT. При генерації квитків на основі квитка TGT тимчасова мітка з поля authtime квитка TGT копіюється в поле authtime квитка, що генерується.

10) **starttime**: час набуття квитком сили.

11) **endtime**: час закінчення терміну дії квитка.

12) **renew-till**: найбільше значення поля endtime, яке може бути задано за допомогою прапора RENEWABLE (поле необов'язкове).

13) **caddr**: одна або декілька адрес, з яких може використовуватися цей квиток. Поле необов'язкове. Якщо воно не заповнене, квитком можна скористатися з будь-якої адреси.

14) **authorization-data**: атрибути привілеїв клієнта. Поле необов'язкове. Його вміст Kerberos не обробляє – він інтерпретується службою.

Поле flags має довжину 32 розряди і адресується побітово, увімкнення та вимкнення прапорців проводиться зміною значення (0 або 1) відповідного біта. Найважливіші з них:

- FORWARDABLE: вказує, що на підставі цього квитка TGT служба видачі квитків може генерувати новий квиток TGT з іншою мережевою адресою (поле є лише у квитках TGT);
- FORWARDED: вказує на те, що квиток TGT був переадресований або згенерований на основі іншого квитка TGT, який пройшов переадресацію;

- PROXIABLE: вказує, що служба видачі квитків може генерувати квитки, мережна адреса яких відрізнятиметься від наведеного в даному квитку TGT (поле є лише у квитках TGT);
- PROXY: вказує на те, що мережна адреса в даному квитку відрізняється від адреси, наведеної в квитку TGT, на підставі якого він виданий;
- RENEWABLE: використовується у поєднанні з полями endtime та renew-till, дозволяючи періодичне оновлення службою KDC квитків з підвищеним терміном дії;
- INITIAL: вказує, що цей квиток є квитком видачі квитків (поле є лише у квитках TGT).

7.5 Розподіл ключів Діффі-Геллмана

Розглянемо схему розподілу ключів, яка використовує асиметричні алгоритми шифрування (тобто з відкритим ключем).

Припустимо, A та B хочуть домовитися про секретний ключ K_{ab} по відкритому каналу зв'язку без попередньої зустрічі. Протокол Діффі-Геллмана дозволяє це зробити.

Нехай A і B відомі два числа g та p (ці числа не зберігаються в таємниці). Тоді їм слід зробити такі кроки:

1. A і B генерують великі випадкові числа a та b відповідно.
2. A обчислює значення $AR = g^a \bmod p$.
3. A надсилає значення AR B.
4. B обчислює значення $BR = g^b \bmod p$.
5. B надсилає значення BR A.
6. A обчислює секретний ключ $K_{ab} = BR^a \bmod p$.
7. B обчислює секретний ключ $K_{ab} = AR^b \bmod p$.

Розглянемо приклад. Нехай p = 2 147 483 659 і g = 2. Нехай також a = 12 345 та b = 654 323. Тоді

$$AR = g^a \bmod p = 428\ 647\ 416,$$
$$BR = g^b \bmod p = 450\ 904\ 856.$$

Загальний секретний ключ матиме вигляд

$$K_{ab} = AR^b \bmod p = BR^a \bmod p = 1\ 333\ 327\ 162.$$

Ця схема не стійка проти атаки «людина посередині». Захист проти такої атаки – цифрові підписи.

7.5.1 Протокол MQV

Як ми переконалися, електронні підписи вирішують проблему «людина посередині» в протоколі розподілу ключів Діффі-Геллмана. Однак, застосування електронних підписів збільшує накладні витрати в протоколі, тобто обсяг службової інформації, що передається. Для зменшення обсягу повідомлення Менезес, К'ю та Ванстоун розробили протокол MQV, який є узагальненням протоколу Діффі-Геллмана.

MQV дає значну економію в розмірі даних, що передаються. Він базується на задачі дискретного логарифмування в групі A, породженій елементом G. Розглянемо деталі протоколу MQV.

Нехай А і В хочуть домовитися про секретний ключ K_{ab}. Схема алгоритму наступна:

1. А вибирає випадкове число a.

2. А обчислює число $AR = G^a \bmod P$, P – просте число. Тоді (a, AR) – довгострокова ключова пара.

3. А вибирає довільне число ag.

4. А обчислює значення $AGR = G^{ag} \pmod P$. Пара (ag, AGR) – сеансовий ключ.

Аналогічним чином В вибирає свої довгострокову ключову пару (b, BR) і сеансовий ключ (bg, BGR).

5. А надсилає значення AGR

6. В надсилає значення BGR.

Далі кожен знаходить загальний таємний ключа K_{ab}.

А знає a, AR, ag, BR, AGR, BGR і знаходить секретний ключ так:

- перетворює AGR на ціле число I,

– обчислює $Sa = (I \pmod{2^L}) + 2^L \pmod{P}$, де L – половина бітового розміру порядку групи A (наприклад, якщо порядок групи $Q = 2^{160}$, то $L = 80$),
– перетворює BGR на ціле число J,
– знаходить $Ta = (J \pmod{2^L}) + 2^L \pmod{P}$,
– обчислює $Ha = (ag + Sa \cdot a) \pmod{P}$,
– знаходить $K_{ab} = (BGR \cdot BR^{Ta})^{Ha} \pmod{P}$.

У свою чергу, В знає b, BR, bg, AR, AGR, BGR і шукає секретний ключ так:

– перетворює BGR на ціле число I,
– обчислює $Sb = (I \pmod{2^L}) + 2^L \pmod{P}$,
– перетворює BGR на ціле число J,
– знаходить $Tb = (J \pmod{2^L}) + 2^L \pmod{P}$,
– обчислює $Hb = (bg + Sb \cdot b) \pmod{P}$,
– знаходить $K_{ab} = (AGR \cdot AR^{Tb})^{Hb} \pmod{P}$.

Для алгоритму MQV цифровий підпис не потрібний, бо фактично він вже там є.

Частина 8. Розподіл відкритих ключів

8.1 Прив'язка ключів та цифрові сертифікати

Основна проблема розподілу ключів при використанні асиметричних криптоалгоритмів – це прив'язка відкритого ключа до фізичної особи.

Один із способів прив'язки, типовий для багатьох ситуацій, в яких власник ключа повинен бути присутнім особисто, полягає у передачі фізичного об'єкта, наприклад, інтелектуальної картки. Володіння таким символом та знання PIN-коду (паролю) вважається достатнім для посвідчення особи.

Однак, більшість власників ключів не є людьми: це комп'ютери. Крім того, багато протоколів з відкритим ключем виконуються по мережі, де фізичну присутність власника (якщо це людина) перевірити неможливо.

Таким чином, необхідно мати декілька форм прив'язки, які можна було б застосовувати у різних ситуаціях. Основний інструмент прив'язки, що на сьогодні використовується, називається **цифровим сертифікатом**. При цьому за допомогою звертаються до особливого посередника, який називається **центром сертифікатів** (ЦС), що відповідає за автентичність (справжність) відкритого ключа.

Принцип роботи ЦС полягає в наступному.

1. Кожен із користувачів має надійну копію відкритого ключа ЦС. Наприклад, він може бути «зашитий» в комп'ютер при покупці (точніше, копія відкритого ключа ЦС зашивається в браузер).

2. ЦС ставить цифровий підпис на рядок даних, наприклад:

(Аліса, відкритий ключ Аліси)

Такий рядок даних разом із цифровим підписом і називається цифровим сертифікатом. Центр сертифікатів підпише рядок даних лише в тому випадку, коли справді вірить, що цей відкритий ключ дійсно належить Алісі.

3. Якщо Аліса надішле Бобу свій відкритий ключ, що міститься в цифровому сертифікаті, то у Боба не буде причин сумніватися в його справжності, оскільки він впевнений у коректності роботи ЦС.

Сертифікати відкритих ключів зазвичай (хоч і не завжди) зберігаються в довгостроковій пам'яті комп'ютера і видаються в міру потреби. Наприклад, більшість браузерів зберігає список сертифікатів, з якими вони колись зустрічалися.

Сертифікати зазвичай містять такі типові реквізити:

- ім'я користувача;
- відкритий ключ користувача;
- тип ключа: шифруючий або підписуючий;
- назва ЦС;
- серійний номер сертифікату;
- термін дії сертифікату.

Як правило, існуватиме кілька ЦС. Тому звичною практикою є підписування одним центром сертифікатів відкритого ключа іншого ЦС і навпаки – перехресна (взаємна) сертифікація. Користувач може не мати достовірної копії відкритого ключа певного ЦС, потрібної, щоб перевірити справжність відкритого ключа, що міститься у виданому сертифікаті.

Ця проблема якраз вирішується перехресною сертифікацією (рис. 15), коли відкритий ключ центру сертифікатів підписується іншим ЦС. Користувач спочатку перевіряє відкритий ключ відповідного ЦС, а потім ключ клієнта, підписаний цим ЦС. Якщо центрів сертифікатів досить багато, можна отримувати достатньо довгі ланцюжки, за допомогою яких перевіряється справжність того чи іншого відкритого ключа.

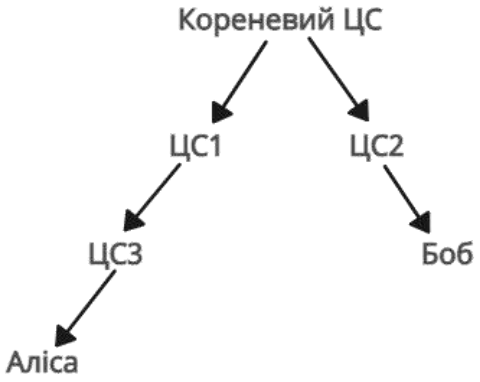


Рисунок 15. – Перехресна сертифікація.

Зазвичай функції центру сертифікатів пов’язані з двома напрямками: засвідчення особи користувача та підписування відкритих ключів. Остання задача здійснюється власне ЦС, тоді як перша функція передається центру реєстрації (ЦР).

Основна проблема системи центрів сертифікатів виникає, коли відкритий ключ користувача компрометується або стає недостовірним з інших причин. Наприклад, третій стороні стало відомо про приватний ключ користувача, або службовець з таким ключем звільняється з компанії. Оскільки скомпрометованому відкритому ключу більше не можна довіряти, всі підписані за його допомогою сертифікати стають недійсними і мають бути анульовані.

Таким чином, ЦС має певним чином інформувати користувачів про те, що всі сертифікати, що містять цей відкритий ключ, не дійсні. Такий процес називається **відкликанням сертифікатів**. Один із можливих шляхів поширення інформації про недійсні сертифікати – список відкликаних сертифікатів: це повідомлення, підписане ЦС, що містить серійні номери всіх сертифікатів, які були відкликані цим центром, але чий термін дії ще не минув. Очевидно, що сертифікати зі строком дії, що минув, до цього списку включати не обов'язково.

Система, що складається з центрів сертифікації та сертифікатів, зазвичай називається **інфраструктурою відкритих ключів**.

8.2 Система PGP

Програмний засіб PGP (Pretty Good Privacy) є прикладом системи, яка дозволяє виконувати операції шифрування і цифрового підпису повідомлень, файлів та іншої інформації, поданої в електронному вигляді, у тому числі прозоре шифрування даних на запам'ятовуючих пристроях.

Керування відкритими ключами здійснюється самими користувачами за принципом «перерозподілу довіри» знизу догори. Кожен користувач перебирає функції ЦС і підписує відкриті ключі інших користувачів.

Так, користувач А може підписати відкритий ключ користувача В, а потім передати цей підписаний «сертифікат» користувачеві С. Виходить, що А виступає

в ролі ЦС для В. Якщо користувач С довіряє думці А про достовірність ключів окремих людей, він буде впевнений, що підписаний нею ключ дійсно належить В. Оскільки користувачі постійно роблять такі перехресні засвідчення, мережа достовірних відкритих ключів зростає знизу вгору.

Сама PGP як програма використовує шифрування RSA даних невеликого розміру, як наприклад, сеансовий ключ. Для передачі великих обсягів закритої інформації використовується блоковий шифр IDEA.

Ключі, яким довіряє окремий користувач, об'єднуються в так зване кільце ключів. Це означає, що користувачі самостійно контролюють свій локальний запас відкритих ключів. Такий спосіб не відкидає централізованого зберігання відкритих ключів, але це необов'язково.

Відкликання ключів залишається невирішеною проблемою як системи PGP, так й інших подібних систем. Так, PGP пропонує при компрометації ключа повідомити всіх друзів, що слід викинути цей ключ з їхніх кілець ключів і передати цю інформацію далі.

Існує ряд реалізацій системи PGP (GnuPG, FileCrypt та інші), всі вони мають слідувати стандарту OpenPGP (RFC 4880).

8.3 Протокол захищених сокетів SSL

Криптографічний протокол SSL (Secure Sockets Layer) служить для встановлення безпечного з’єднання між клієнтом і сервером. Він дозволяє автентифікацію та безпечну передачу даних в мережі з використанням криптографічних засобів. На його основі розроблено сучасний протокол TLS (Transport Layer Security).

SSL працює поверх TCP\IP та нижче протоколів вищого рівня, таких як HTTP або IMAP. Протокол дозволяє двом машинам встановити зашифроване з'єднання.

Функції SSL-автентифікації.

1. SSL-автентифікація дозволяє користувачеві підтвердити автентичність сервера. Програмне забезпечення SSL-клієнта дає можливість використовувати

стандартні методи криптографії з відкритим ключем для перевірки сертифікатів сервера та відкритих ID: чи є вони дійсними і чи були видані за сертифікацією ЦС.

2. SSL-автентифікація клієнта дозволяє серверу підтверджувати справжність користувача. Використовуючи ті самі методи, що й при автентифікації сервера, SSL може перевірити, що сертифікат клієнта є дійсним і виданим ЦС.

3. Вся інформація, що передається між клієнтом та сервером через SSL-з'єднання шифрується. Крім того, всі дані, що передаються через шифроване з'єднання SSL, захищені механізмом виявлення (наприклад, CRC).

Протокол SSL складається з двох підпротоколів: протокол SSL-запису та рукостискання.

Протокол запису визначає формат, який використовується при передачі даних. Протокол SSL запускає рукостискання з використанням протоколу SSL-запису для обміну серіями повідомлень між сервером та клієнтом під час встановлення першого з'єднання.

Цей обмін повідомленнями призначений для забезпечення наступної послідовності дій:

- автентифікація сервера до клієнта,
- дозвіл клієнту та серверу вибрати криптографічні алгоритми або шифри, які вони підтримують,
- додаткова автентифікація клієнта на сервері,
- використання протоколу розподілу для створення загального секрету,
- створення шифрованого SSL-з'єднання.

8.3.1 Процедура рукостискання

Під час рукостискання клієнт та сервер домовляються про різні параметри, які будуть використані для забезпечення безпеки з'єднання.

1. Рукостискання розпочинається, коли клієнт підключається до SSL-сервера (порт 443 у випадку HTTPS). Запит безпечного з'єднання містить список підтримуваних шифрів та хеш-функцій.

2. З цього списку сервер обирає найсильніший шифр та хеш-функцію, яку він також підтримує, та повідомляє клієнтів про свій вибір.

3. Сервер надсилає цю відповідь у вигляді цифрового сертифікату (X.509). Сертифікат зазвичай містить інформацію про ім'я сервера, довірений центр сертифікації і відкритий ключ шифрування сервера. Перш ніж продовжити, клієнт може зв'язатися з сервером, який видав сертифікат (довіреного центру сертифікації) та переконатися в його справжності.

4. Для створення ключів сеансу використовується безпечне з'єднання. Клієнт шифрує випадкове число за допомогою відкритого ключа сервера та надсилає туди результат. Сервер може розшифрувати його за допомогою свого закритого ключа.

5. З випадкового числа обидві сторони створюють ключові дані для шифрування та розшифрування.

Далі сервер та клієнт обмінюються шифрограмами.

Розглянемо, що відбувається при рукостисканні докладніше.

1. Клієнт надсилає повідомлення ClientHello на відповідний порт, вказуючи найостаннішу останню версію SSL-протоколу, що підтримується, випадкове число і список підтримуваних методів шифрування та стиснення.

2. Сервер відповідає повідомленням ServerHello, що містить обрану сервером версію протоколу, випадкове число, надіслане клієнтом, відповідні алгоритми шифрування та стиснення з наданого клієнтом списку.

3. Сервер надсилає повідомлення Certificate, яке містить цифровий сертифікат сервера (залежно від алгоритму шифрування, цей етап може бути пропущений).

4. Сервер може запитати сертифікат у клієнта, в такому разі з'єднання буде взаємоавтентифіковано.

5. Сервер надсилає повідомлення ServerHelloDone, що ідентифікує закінчення рукостискання.

6. Клієнт відповідає повідомленням ClientKeyExchange, яке містить відкритий ключ PreMasterSecret або нічого (залежить від алгоритму шифрування).

7. Клієнт та сервер, використовуючи ключ PreMasterSecret та випадково згенеровані числа, обчислюють загальний секретний ключ. Решта інформації про ключ буде отримана із загального секретного ключа (і згенерованих клієнтом та сервером випадкових значень).

Останні два пункти фактично є протоколом Діффі-Геллмана.

8. Клієнт надсилає повідомлення ChangeCipherSpec, яке вказує на те, що вся наступна інформація буде зашифрована встановленим в процесі рукостискання алгоритмом, використовуючи загальний секретний ключ.

9. Клієнт надсилає повідомлення Finished, яке містить хеш-код і MAC-код автентифікації повідомлення, згенеровані на основі попередніх повідомлень рукостискань.

10. Сервер намагається розшифрувати Finished-повідомлення клієнта та перевірити хеш-код та MAC. Якщо процес розшифровки або перевірки не вдається, рукостискання вважається невдалим і з'єднання має бути обірвано.

11. Сервер надсилає ChangeCipherSpec та зашифроване Finished-повідомлення, в свою чергу клієнт теж виконує розшифровку та перевірку.

8.4 Сертифікат X.509

Формат сертифіката відкритого ключа визначено в документі стандарту RFC 5280. Сертифікат відкритого ключа підпису або шифрування є структурованим двійковим записом у форматі абстрактної синтаксичної нотації ASN.1. Сертифікат містить елементи даних, які супроводжуються цифровим підписом видавця сертифіката

Сертифікаті містить десять основних полів: шість обов'язкових та чотири опціональних. Більшість інформації, що вказується в сертифікаті, не є обов'язковою, а зміст обов'язкових полів сертифіката може змінюватись. До обов'язкових полів відносяться:

- серійний номер сертифікату (Certificate Serial Number);
- ідентифікатор алгоритму підпису (Signature Algorithm Identifier);
- ім’я видавця (Issuer Name);

- період дії (Validity (Not Before/After));
- відкритий ключ суб'єкта (Subject Public Key Information);
- ім'я суб'єкта сертифікату (Subject Name).

Під суб'єктом сертифікату розуміють сторону, яка контролює секретний ключ, що відповідає даному відкритому ключу.

Наявність необов'язкових полів властива сертифікатам версій 2 і 3, туди входять номер версії, два унікальні ідентифікатори та доповнення.

Видавець сертифікатів надає кожному випущеному сертифікату серійний номер Certificate Serial Number, який має бути унікальним. Комбінація імені видавця та серійного номера однозначно ідентифікує кожен сертифікат.

У полі Signature Algorithm Identifier вказується ідентифікатор алгоритму цифрового підпису, який використовувався видавцем сертифікату.

Доповнення дозволяють включати в сертифікат інформацію, яка відсутня в основному змісті, визначати валідність сертифіката та наявність у його власника прав доступу до тієї чи іншої системи.

Крім того, доповнення містять технологічну інформацію, що дозволяє легко перевірити справжність сертифіката. Кожна організація може використовувати свої приватні доповнення, що задовольняють конкретні вимоги ведення бізнесу.

Опціональне поле Extensions (доповнення) з'являється у сертифікатах третьої версії. Кожне доповнення складається з ідентифікатора типу доповнення (Extension Identifier), ознаки критичності (Criticality Flag) і значення доповнення (Extension Value).

Ідентифікатор типу доповнення визначає формат і семантику значення доповнення. Ознака критичності повідомляє додатку, що використовує цей сертифікат, чи є суттєвою інформація про призначення сертифіката і чи може програма ігнорувати цей тип доповнення.

Всі доповнення можна розділити на дві категорії: обмежувальні та інформаційні. Перші обмежують сферу застосування ключа, визначеного сертифікатом, або самого сертифіката. Другі містять нову інформацію, яка може

бути використана в прикладному програмному забезпеченні користувача сертифіката.

До обмежуючих доповнення належать:

- основні обмеження (Basic Constraints);
- призначення ключа (Key Usage);
- розширене призначення ключа (Extended Key Usage);
- політики застосування сертифікату (Certificates Policies, Policy Mappings, Policy Constraints);
- обмеження на імена (Name Constraints).

До інформаційних доповнень відносяться:

- альтернативні імена (Subject Alternative Name, Issuer Alternative Name);
- пункт розповсюдження списку анульованих сертифікатів (CRL Distribution Point, Issuing Distribution Point);
- метод доступу до інформації (Authority Access Info).

Доповнення призначення ключа (Key Usage) відображає сферу застосування секретного ключа, що відповідає зазначеному в сертифікаті відкритому ключу.

Доповнення альтернативної назви суб'єкта (Subject Alternative Name) дозволяє розширити межі ідентифікації власника сертифіката за допомогою альтернативних імен, таких як DNS-імена, IP-адреси, URI-адреси або адреси електронної пошти.

Доповнення пункту поширення списку анульованих сертифікатів (CRL Distribution Point) визначає уніфікований ідентифікатор ресурсу (Uniform Resource Identifier, URI) для вказівки місцезнаходження списку анульованих сертифікатів, тобто визначає пункт поширення списку анульованих сертифікатів.

Доповнення політик застосування сертифікату (Certificate Policies) може супроводжуватись специфікатором для вказівки в кожному сертифікаті додаткової інформації, незалежної від політики застосування сертифікатів. Стандарт X.509 жорстко не регламентує призначення та синтаксис цього поля.

8.5 Проста інфраструктура відкритих ключів SPKI

Щоб усунути деякі недоліки, притаманні стандарту X.509, було запропоновано інший тип сертифікатів, SPKI (Simple Public Key Infrastructure). Ця система покликана розв’язувати проблеми авторизації та ідентифікації. Крім того, у ній передбачено можливість делегування повноважень.

SPKI не передбачає наявності глобальної ієрархічної структури центрів сертифікатів, як в стандарті X.509. Натомість використовується принцип «перерозподілу довіри» знизу догори, схожий на PGP.

Розроблено два типи SPKI-сертіфікатів: для автентифікації ключів та для їх авторизації. Вони представлені у вигляді кортежів з 4 або 5 об'єктів.

Кортеж SPKI з чотирьох об'єктів.

Для сертифіката автентифікації та прив'язки ключа до імені, як це робить X.509, SPKI використовує 4-об'єктну структуру. Це абстрактна назва, що відображає чотири складові сертифікату:

(Видавець, Ім'я, Суб'єкт, Період дії).

На практиці сертифікат складається фактично з п'яти полів:

- відкритий ключ видавця;
- ім'я суб'єкта;
- відкритий ключ суб'єкта;
- період дії;
- підпис автора сертифікату на трійці (Ім'я, Суб'єкт, Період дії).

Кортеж SPKI із п'яти об'єктів.

Така система використовується для авторизації ключів. Ця абстакція описує 5 складників сертифікату:

(Видавець, Суб'єкт, Делегування, Авторизація, Період дії).

На практиці кожен сертифікат містить шість полів:

- відкритий ключ видавця;
- відкритий ключ суб'єкта;

– делегування, тобто ознака Так/Ні, яка повідомляє, чи може суб'єкт делегувати повноваження;
– авторизація: що, власне, може делегувати суб'єкт;
– період дії авторизації;
– підпис автора сертифікату на четвірці (Суб'єкт, Делегування, Авторизація, Період дії).

Можливо комбінувати сертифікат авторизації із сертифікатом автентифікації. У цьому може виникнути потреба, оскільки сертифікат авторизації лише дозволяє робити будь-які дії з ключем, але не повідомляє, хто має ключ. Для прив'язки ключа до імені слід використовувати сертифікат автентифікації.

Комбінування сертифікатів здійснюється за правилом редукції:

$$(I_1, S_1, D_1, A_1, V_1) + (I_2, S_2, D_2, A_2, V_2) = (I_1, S_2, D_2, A_1 \cap A_2, V_1 \cap V_2),$$

де I_i – видавець, S_i – суб'єкт, D_i – делегування, A_i – авторизація, V_i – період дії.

Розглянемо приклад. Сертифікат, яким Аліса доручає Бобу зняти з її рахунку 100 гривень і дозволяє делегувати цю дію ще комусь на розсуд Боба:

I_1 = Аліса;

S_1 = Боб;

D_1 = Так;

A_1 = зняти 100 гривень з рахунку Аліси;

V_1 = необмежено.

Розглянемо інший сертифікат. Боб доручає Чарлі зняти суму від 50 до 200 гривень з рахунку Аліси, причому зробити це до ранку наступного дня:

I_2 = Боб;

S_2 = Чарлі;

D_2 = Ні;

A_2 = зняти суму від 50 до 200 гривень з рахунку Аліси;

V_2 = до завтрашнього ранку.

Зчеплення цих двох сертифікатів за правилом редукції матиме вигляд:

I_3 = Аліса;

S_3 = Чарлі;

D_3 = Ні;

A_3 = зняти 100 гривень з рахунку Аліси;

V_3 = до завтрашнього ранку.

Оскільки Аліса дозволила Бобу делегувати повноваження, вийшло, що вона фактично доручила Чарлі зняти гроші з її рахунку до завтрашнього ранку.

8.6 Неявні сертифікати

Як відомо, кожен сертифікат повинен містити як мінімум відкритий ключ користувача і підпис центру сертифікатів на цьому ключі. Це може призвести до великих розмірів сертифікатів.

Неявні сертифікати дозволяють зменшити розмір службової інформації, що передається. Вони виглядають як $X|Y$, де X – дані, пов'язані з відкритим ключем, а Y – неявний сертифікат для X. На основі Y треба вміти відновлювати відкритий ключ, асоційований з даними X, і доводити, що сертифікат було видано певним центром сертифікатів.

Опишемо цей сертифікат.

Центр сертифікатів вибирає відкриту групу порядку Q (наприклад, групу залишків за модулем N) і елемент цієї групи x, який в секреті не зберігається. Також центр сертифікатів обирає свій секретний ключ c та обчислює відкритий ключ $Y = x^c \pmod N$.

Запит на сертифікат. Якщо А хоче отримати сертифікат та відкритий ключ, пов'язаний з її ідентифікаційною інформацією ID (наприклад, ім'я чи номер паспорта), вона вибирає секретний сеансовий ключ t і відповідний відкритий ключ $R = x^t \pmod N$, а потім надсилає до центру сертифікатів R та ID.

Обробка запиту. Центр сертифікатів перевіряє, що отримав ID саме від А. Потім він вибирає інший секретний сеансовий ключ k і обчислює

$$G = x^k R \pmod N = x^k x^t \pmod N = x^{k+t} \pmod N,$$

далі підраховує $S = c \cdot h(ID||G) + k \pmod Q$ та надсилає А пару (G, S). Таким чином, сертифікатом буде пара (ID, G).

Дії А. Вона має наступну інформацію: *t, S, R, x*. Користувач А обчислює довгостроковий секретний ключ: $a = t + S \pmod{Q}$. Тоді довгостроковий відкритий ключ буде $AR = x^a \pmod{N}$.

Дії іншого користувача. Оскільки інформація про *S* та *R* відкрита, В може дізнатися відкритий ключ А, обчисливши $R \cdot x^s \pmod{N}$. Однак, щоб встановити зв'язок відкритого ключа з *ID* та центру сертифікатів підрахувати

$$Y^{h(ID||G)}G \pmod{N} = x^{c \cdot h(ID||G)}x^{k+t} = x^{s+t} = x^a = AR.$$

Як тільки В бачить застосування ключа А, наприклад, під час перевірки підпису, він абсолютно впевнений, що цей ключ мав бути створений центром сертифікатів, оскільки в іншому випадку підпис А не витримав би перевірки.

Розглянута система має низку проблем, через що не може широко застосовуватися (хіба що традиційні сертифікати недоступні через малу пропускну здатність обладнання). Наприклад, у випадку компрометації ключа центру сертифікатів, слід було б вибрати новий ключ і створити нові сертифікати. Але тут це неможливо зробити, оскільки відкриті ключі користувачів вибираються в діалоговому режимі в процесі створення сертифіката. Неявні сертифікати вимагають, щоб і центр сертифікатів, і користувач працювали одному рівні секретності, що зазвичай є не найкращою вимогою з точки зору безпеки: як правило, центр сертифікатів повинен мати вищий рівень таємності.

8.7 Криптографія ідентифікаційної інформації

Інший спосіб засвідчення відкритих ключів без участі сертифікатів полягає у використанні системи, за допомогою якої ключ користувача отримується з його ідентифікатора (наприклад, імені). Така система називається **схемою особистісного шифрування** чи **схемою підпису на основі ідентифікаційної інформації** (identity-based encryption, ID-based encryption).

Такі системи все ще потребують довіреної третьої особи для початкового встановлення справжності користувача, але вже не треба зберігати і передавати сертифікати.

Розглянемо оригінальну схему протоколу. Вона базується на проблемі RSA.

Спочатку третя довірена сторона TTP обчислює модуль RSA, тобто число N, зберігаючи в таємниці два його прості дільники. TTP публікує відкриту експоненту E, не розголошуючи відповідну розшифровуючу експоненту d. Крім того, фіксується відображення

$$I: \{0,1\}^* \to (\mathbb{Z}/N\mathbb{Z})^*,$$

яке переводить рядок бітів в елемент $(\mathbb{Z}/N\mathbb{Z})^*$. Відображення I може бути реалізоване хеш-функцією.

Припустимо, A хоче отримати секретний ключ g, який відповідає її імені «Alice». Тоді TTP обчислює цей ключ за виразом

$$g = I(\text{Alice})^d \pmod N.$$

Щоб підписати повідомлення m, A генерує пару (T, S) за допомогою рівностей

$$T = r^E \pmod N, \quad S = g \cdot r^{h(m||T)} \pmod N,$$

де r – випадкове ціле число, а h – хеш-функція.

Для перевірки підпису (T, S) на повідомленні m, іншому користувачеві необхідна відкрита інформація про TTP та ідентифікатор A. Якщо ці дані в нього є, він переконується в коректності наступних рівностей за модулем N:

$$I(\text{Alice}) \cdot T^{h(m||T)} = g^E \cdot r^{E \cdot h(m||T)} = \left(g \cdot r^{h(m||T)}\right)^E = S^E.$$

Частина 9. Атаки на схеми з відкритим ключем

9.1 Атака Вінера на RSA

В деяких застосуваннях криптосистеми RSA необхідно прискорити процеси розшифровування. Тому обирають невеликі розшифровуючі експоненти. Зрозуміло, тоді отримується велике значення відкритої експоненти E. Проте значення d не повинно виявитися занадто малим.

Припустимо, $d < \frac{1}{3} \cdot N^{1/4}$ (тут $N = p \cdot q$, прості $q < p < 2q$, N – модуль RSA), тоді можна запропонувати ефективний алгоритм знаходження розшифровуючої екпоненти d. Він носить назву атаки Вінера.

Атака Вінера використовує теорію ланцюгових дробів. Розглянемо поняття ланцюгового дробу.

Нехай дано деяке дійсне число a, тоді визначимо наступну послідовність:

$d_0 = a,\ p_0 = q_0 = 1,\ p_1 = a_0 \cdot a_1 + 1,\ q_1 = a_1,$

$a_i = \lfloor d_i \rfloor,\ d_{i+1} = 1/(d_i - a_i),$

$p_i = a_i \cdot p_{i-1} + p_{i-2}$ при $i \geq 2,$

$q_i = a_i \cdot q_{i-1} + q_{i-2}$ при $i \geq 2.$

Послідовність цілих чисел $[a_0; a_1, a_2, ...]$ називається **ланцюговим** (або **неперервним**) **дробом**, що представляє a. Будь-яке дійсне число можна представити ланцюговим дробом. Розклад буде скінченним у випадку раціонального числа.

Раціональне число $p_n/q_n = [a_0; a_1, \dots, a_n]$ називається n-м **наближеним дробом** для ланцюгового дробу $a = [a_0; a_1, a_2, ...]$. Кожен такий дріб нескоротний, а швидкість зростання їх знаменників порівняна з показниковою.

Одним із важливих результатів теорії ланцюгових дробів є те, що якщо нескоротний дріб p/q задовольняє нерівності

$$\left| a - \frac{p}{q} \right| \leq \frac{1}{2q^2},$$

то p/q є одним з наближених дробів у розкладанні a в неперервний дріб.

Розглянемо, як цей результат використовується в атаці Вінера.

Отже, нехай є модуль $N = p \cdot q$, причому $q < p < 2q$, і розшифровуюча експонента d, що задовольняє $d < \frac{1}{3} \cdot N^{1/4}$. Крім того, шифруюча експонента E має властивість $E \cdot d = 1 \pmod{\varphi}$, де $\varphi(N) = (p-1)(q-1)$. Також вважатимемо, що $E < \varphi$. Тоді існує деяке натуральне k, що

$$E \cdot d - k \cdot \varphi = 1.$$

Поділимо на $d \cdot \varphi$ і отримаємо:

$$\left|\frac{E}{\varphi} - \frac{k}{d}\right| = \frac{1}{d \cdot \varphi}.$$

Оскільки $\varphi = N - p - q + 1 \approx N$, звідси

$$|N - \varphi| = |p + q - 1| < |2q + q - 1| = |3q - 1| < |3q| = 3 \cdot |q| < 3 \cdot \sqrt{N}.$$

Звідси можна отримати

$$\left|\frac{E}{N} - \frac{k}{d}\right| = \left|\frac{E \cdot d - k \cdot N}{d \cdot N}\right| = \left|\frac{E \cdot d - k \cdot \varphi + k \cdot \varphi - k \cdot N}{d \cdot N}\right| =$$

$$= \left|\frac{1 - k \cdot (N - \varphi)}{d \cdot N}\right| \leq \left|\frac{3 \cdot k \cdot \sqrt{N} - 1}{d \cdot N}\right| < \left|\frac{3 \cdot k \cdot \sqrt{N}}{d \cdot N}\right| = \frac{3 \cdot k}{d \cdot \sqrt{N}}.$$

Оскільки $E < \varphi$, то $k < d$. Крім того, за припущенням $d < \frac{1}{3} \cdot N^{1/4}$, тоді $d^2 < \frac{1}{9} \cdot N^{1/2}$. Тому

$$\frac{3 \cdot k}{d \cdot \sqrt{N}} < \frac{3 \cdot k}{9 \cdot d \cdot d^2} < \frac{3 \cdot d}{9 \cdot d \cdot d^2} = \frac{1}{3 \cdot d^2},$$

тобто отримуємо

$$\left|\frac{E}{N} - \frac{k}{d}\right| < \frac{1}{3 \cdot d^2}.$$

Отже, k/d – наближений дріб у розкладанні E/N.

Таким чином, розкладаючи число E/N в ланцюговий дріб, можна дізнатися розшифровуючу експоненту, по черзі підставляючи знаменники відповідних дробів у вираз $(m^E)^d = m \pmod{N}$, де m – випадково вибране число.

Розглянемо приклад. Нехай N має вигляд N = 9 449 868 410 449.

Нехай відкритий ключ криптосистеми E = 6 792 605 526 025, а секретний ключ задовольняє нерівності $d < \frac{1}{3} \cdot N^{1/4} \approx 584$.

Розкладемо число $a = E/N$ в неперервний дріб і перевіримо знаменник.

Наближеними дробами розкладання a є:

1, 2/3, 3/4, 5/7, 18/25, 23/32, 409/569, ….

Почергово перевіряючи знаменники, можна переконатися, що $d = 569$ – шуканий секретний ключ.

9.2 Атака Хостада

Раніше була розглянута атака на алгоритм RSA: нехай дано три відкритих ключі (N_i, E_i) з тією самою шифруючою експонентою $E_i = 3$. Якщо користувач надсилає одне повідомлення, зашифроване цими ключами, то зловмисник легко дешифрує повідомлення, застосувавши китайську теорему про залишки. Проти цієї атаки можна захиститись, якщо до повідомлення m перед шифруванням додати деякі специфічні для користувача дані.

Нехай шифротекст, що відправляється i-му користувачеві (усього користувачів k), має вигляд $C_i = (i \cdot 2^H + m)^e \pmod N$.

Хостад показав, що у цьому випадку повідомлення можна дешифрувати. Для цього утворюємо поліноми

$$g_i(x) = (i \cdot 2^H + x)^e - C_i.$$

При цьому відомо, що $g_i(m) = 0 \pmod N$. Треба знайти m.

Якщо покласти $N = N_1 N_2 \ldots N_K$, то, використовуючи китайську теорему про залишки, можна знайти такі T_i, що многочлен

$$g(x) = \sum_{i=1}^{K} T_i g_i(x)$$

задовольнятиме співвідношення $g(m) = 0 \pmod N$.

Використовуючи теорему Копперсміта, можна знайти корінь за поліноміальний час.

Теорема Копперсміта. Нехай f – нормований (тобто старший коефіцієнт дорівнює 1) многочлен степені D. Якщо в многочлена f за модулем N існує корінь x_0, що задовольняє нерівності $|x_0| \le X$, де $X = N^{1/(D-e)}$, тоді цей корінь можна знайти за поліноміальний час при фіксованому значенні D.

9.3 Атака Франкліна-Рейтера

Нехай A хоче відправити B два шифровані повідомлення m_1 та m_2, пов'язаних за допомогою наступного відкритого многочлена:

$$m_2 = f(m_1) \pmod N.$$

Тоді можна знайти вихідні повідомлення, якщо відомі шифротексти C_1 і C_2 при невеликій шифруючій експоненті E.

Розглянемо простий випадок $f(x) = ax + b$ і $E = 3$. Передбачається, що нападник знає a та b. Очевидно, зловмисник знає, що m_2 є коренем за модулем N многочленів

$$g_1(x) = x^3 - C_2 \text{ та } g_2(x) = (f(x))^3 - C_1.$$

Тоді $x - m^2$ є найбільшим спільним дільником многочленів $g_1(x)$ та $g_2(x)$. Застосовуючи алгоритм Евкліда знаходження НСД(g_1, g_2), неважко знайти m_2, а слідом і m_1 (оскільки $f(x)$ – лінійна функція).

9.4 Часткове розкриття секретної експоненти в RSA

У найзагальнішому випадку використання системи RSA з малою шифруючою експонентою E можна тривіальним чином розкрити половину вищих значущих бітів секретної експоненти.

Відомо, що існує таке натуральне число k, що $0 < k < E$ та

$$Ed - k(N - (p + q) + 1) = 1.$$

Припустимо, що для кожного можливого значення i ($0 < i \le E$) зловмисник обчислює

$$D_i = \left\lfloor \frac{iN + 1}{E} \right\rfloor.$$

Тоді

$$|D_k - d| \le \frac{k(p + q)}{E} \le \frac{3k\sqrt{N}}{E} < 3\sqrt{N}\,.$$

Отже, D_k – добре наближення до справжнього значення розшифровуючої експоненти d. Якщо $E = 3$, то $k = 2$, і це означає, що D_2 містить половину

значущих бітів експоненти d. Проте, навіть знаючи їх, не існує способу розкрити решту бітів.

9.5 Часткове розкриття простих множників RSA

Припустимо, що n-бітовий модуль N алгоритму RSA має розкладання на прості множники $N = pq$, причому $p \approx q$, і зловмисник дізнався $n/4$ найменших значущих бітів числа p (загальна кількість знаків в числі p близька до $n/2$).

Тоді запишемо

$$p = x_0 2^{n/4} + p_0, \quad q = y_0 2^{n/4} + q_0 ,$$

де P_0 – відомий набір значень. Тоді

$$N = P_0 q_0 \pmod{2^{n/4}}$$

і можна визначити величину $q_0 = Q_0$.

Розглянемо многочлен

$$p(x, y) = \left(P_0 + 2^{n/4} x\right)\left(Q_0 + 2^{n/4} y\right) = P_0 Q_0 + 2^{n/4}(P_0 y + Q_0 x) + 2^{n/2} xy.$$

Многочлен $p(x, y)$ залежить від двох змінних і має корінь за модулем N, а саме (x_0, y_0), де $x_0 > 0$ та $y_0 \leq 2^{n/4} \approx N^{1/4}$.

Застосовуючи узагальнення теореми Копперсміта, можна знайти (x_0, y_0) за поліноміальний час й ефективно розкласти N на прості множники.

Частина 10. Використання криптоалгоритмів

10.1 Протокол IPSec

IP Security – це комплект протоколів, що служать для шифрування, автентифікації та забезпечення захисту під час транспортування IP-пакетів. Архітектура IPSec зображена на рис. 16.

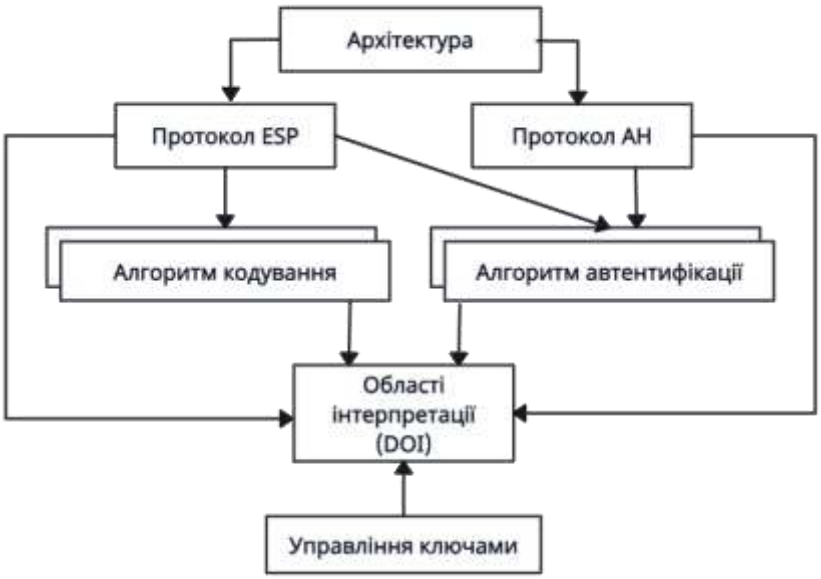


Рисунок 16. – Архітектура IPSec.

Протокол **Authentication Header** (AH) є звичайним опціональним заголовком і зазвичай розташовується між основним заголовком пакета IP та полем даних. Єдиною метою AH є забезпечення захисту від атак, пов'язаних з несанкціонованою зміною вмісту пакета, в тому числі заміною вихідної адреси мережевого рівня.

Формат AH досить простий:

Наступний заголовок (8 біт)	**Довжина навантаження (8 біт)**	**Зарезервовано (16 біт)**
Індекс параметрів безпеки (32 біти)		
Поле послідовного номера (32 біти)		
Дані автентифікації (змінної довжини, слова по 32 біти)		

Наступний заголовок вказує на наступний заголовок, **довжина навантаження** представляє довжину пакета, **індекс параметрів безпеки** є вказівником на контекст безпек, а **поле послідовного номера** містить

послідовний номер пакета. Послідовний номер пакета формується відправником і служить для захисту від атак, пов'язаних із повторним використанням даних процесу автентифікації.

Механізми забезпечення цілісності даних у відкритих телекомунікаційних мережах повинні мати засоби захисту від внесення цілеспрямованих змін. Одним з таких механізмів є спеціальне застосування алгоритму MD5: у процесі формування AH послідовно обчислюється хеш-функція від об'єднання самого пакета та деякого попередньо узгодженого ключа, а потім від об'єднання отриманого результату та перетвореного ключа.

У разі використання інкапсуляції зашифрованих даних заголовок **Encapsulating Security Payload** (ESP) є останнім у ряді опціональних заголовків, «видимих» у пакеті. Оскільки основною метою ESP є забезпечення конфіденційності даних, різні види інформації можуть вимагати застосування суттєво різних алгоритмів шифрування. Формат ESP може зазнавати значних змін залежно від криптографічних алгоритмів, що використовуються.

Можна виділити такі обов'язкові поля: **індекс параметрів безпеки** вказує на контекст безпеки, **послідовний номер** містить послідовний номер пакета, **контрольна сума** не є обов'язковою в заголовку ESP. Одержувач пакету ESP розшифровує ESP-заголовок і використовує параметри та дані алгоритму шифрування, що застосовується для декодування інформації транспортного рівня.

Отже, формат пакету ESP наступний:

<table>
<tr><td colspan="3">Індекс параметрів безпеки SPI (32 біти)</td></tr>
<tr><td colspan="3">Послідовний номер (32 біти)</td></tr>
<tr><td colspan="3">Дані навантаження (змінної довжини, ціла кількість байт)</td></tr>
<tr><td>Доповнення (0..255 октетів)</td><td>Довжина доповнення (8 біт)</td><td>Наступний заголовок (8 біт)</td></tr>
<tr><td colspan="3">Контрольна сума (слова по 32 біти)</td></tr>
</table>

Розрізняють два режими застосування ESP та AH (а також їх комбінації) – транспортний та тунельний.

Транспортний режим. Він використовується для шифрування поля даних IP-пакету. Прикладом застосування транспортного режиму є передача електронної пошти. Всі проміжні вузли на маршруті пакета від відправника до одержувача використовують лише відкриту інформацію мережевого рівня та, можливо, деякі опціональні заголовки пакета.

Недоліком транспортного режиму є відсутність механізмів приховування конкретних відправників та одержувачів пакету, а також можливість проведення аналізу трафіку.

Тунельний режим. Він передбачає шифрування усього пакета, включаючи заголовок мережевого рівня. Тунельний режим застосовується у разі потреби приховування інформаційного обміну організації із зовнішнім світом. При цьому адресні поля заголовка мережевого рівня пакета, що використовує тунельний режим, заповнюються міжмережевим екраном організації і не містять інформації про конкретного відправника пакета. При передачі інформації із зовнішнього світу до локальної мережі організації в якості адреси призначення використовується мережева адреса міжмережевого екрану. Після розшифровки міжмережевим екраном початкового заголовка мережевого рівня пакет надсилається одержувачу.

10.2 VPN-тунелі

VPN, або Virtual Private Network (Віртуальна приватна мережа) – це криптосистема, що дозволяє захистити дані під час передачі їх через незахищену мережу, таку як інтернет. Метою VPN є прозорий доступ до ресурсів мережі, де користувач може робити все те, що робить зазвичай незалежно від того, наскільки він віддалений в реальності.

З'єднання VPN завжди складається з каналу типу **точка-точка**, також відомого як **тунель**. Тунель створюється в незахищеній мережі. З'єднання точка-точка означає, що воно завжди встановлюється між двома комп'ютерами, які називаються вузлами (peers). Кожен вузол відповідає за шифрування даних до

того, як вони потраплять в тунель, та розшифровці цих даних після того, як вони тунель покинуть.

Хоча тунель VPN завжди встановлюється між двома точками, кожен peer може встановлювати додаткові тунелі з іншими вузлами (рис. 17).

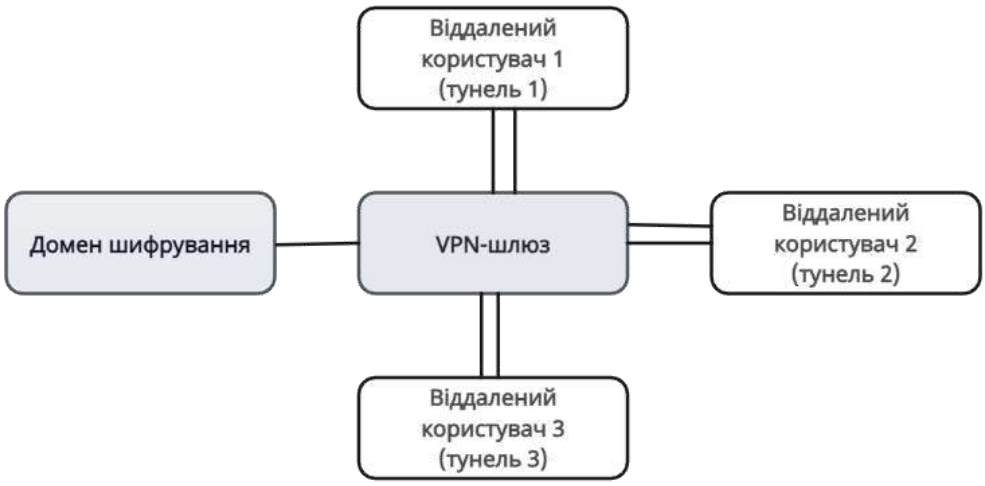


Рисунок 17. – VPN-шлюз.

Наприклад, коли трьом віддаленим станціям необхідно зв'язатися з тим самим офісом, буде створено три окремі VPN-тунелі до цього офісу. При цьому для всіх тунелів peer на стороні офісу може бути тим самим. Це можливо завдяки тому, що вузол вміє шифрувати та розшифровувати дані від імені усієї мережі, тоді VPN-вузол називається VPN-шлюзом (gateway), а його мережа – доменом шифрування (encryption domain). Зауважимо, що всередині домену шифрування самого шифрування не відбувається.

10.3 Кодування цифрового телебачення CONAX

Кодування CONAX дозволяє кодувати передачі цифрового телебачення. Для прийому та декодування передач використовуються супутникові (або кабельні) приймачі з CAM-модулем для кодування CONAX. Для декодування також потрібна карта доступу з мікроконтролером Philips.

Кодування CONAX дозволяє розділяти користувачів на групи і для кожної групи визначати пакет телепрограм, які користувачі цієї групи мають право переглядати.

Алгоритм кодування CONAX наступний.

1. Кожні 5-10 секунд у відеопотоці надсилається пакет ECW.

2. Карта доступу, використовуючи операційний ключ (operation key), декодує пакет ECW в декодоване контрольне слово (control word), або пакет DCW.

3. Пакет DCW передається в CAM-модуль, де за допомогою ключа DCW відбувається декодування відеопотоку за допомогою алгоритму CSA (Common Scrambling Algorithm, зазвичай це застосування операції XOR з ключем).

4. Операційний ключ надсилається також з відеопотоком у вигляді EMM-пакету (закодованого за допомогою RSA). Крім того, EMM містить можливо унікальну адресу карти доступу (її, як правило, називають UA або SA). Операційний ключ зазвичай змінюється 2-3 рази на добу (сам пакет надсилається набагато частіше). Операційний ключ зберігається в карті як key 20 чи key 21.

5. Операційний ключ декодується за допомогою Master Key (key 10) у карті доступу.

Процес отримання DCW виглядає так:

- CAM-модуль видобуває кодований пакет ECW з відеопотоку і відправляє його в карту доступу у вигляді ECM-повідомлення (Entitlement Control Message),
- карта, використовуючи ключ key 20 або key 21, декодує ECW-пакет в DCW-пакет,
- DCW-пакет повертається до CAM-модуля.

Процес оновлення операційного ключа:

- CAM-модуль виділяє з відеопотоку EMM-пакет та відправляє його до карти доступу,
- в картці доступу цей пакет декодується за допомогою Master Key і залишається в ній. При цьому декодування відбудеться, якщо адреса картки збігається з адресою пакета.

Частина 11. Додаткові розділи

11.1 Схема зобов'язань

Схема зобов'язань – це метод, що дозволяє користувачеві підтверджувати деяке значення, яке не розголошується, тобто у разі розголошення цього значення завдяки цій схемі буде відомо, що користувач знав його на момент видачі зобов'язання і що воно не змінилося.

Розглянемо приклад. Припустимо, Аліса хоче пограти з Бобом по телефону у гру «камінь-ножиці-папір». Ідея гри полягає в тому, що Аліса з Бобом одночасно вибирають по одному предмету з множини {камінь, ножиці, папір}. Результат вибору, тобто переможець, визначається за наступними правилами.

- Папір може обгорнути камінь. Тож якщо Аліса вибирає папір, а Боб – камінь, то перемагає Аліса.
- Камінь тупить ножиці. Значить Аліса, обравши камінь, знову переможе, якщо Боб при цьому візьме ножиці.
- Ножиці ріжуть папір. Отже, якщо Аліса назве ножиці, а Боб – папір, то й тут переможе Аліса.
- Якщо обидва гравці обрали однакові предмети, раунд закінчується з нічийним результатом.

Зрозуміло, якщо проводити гру по телефону, то неможливо домогтися одночасного оголошення предметів. Хтось має першим озвучити свій вибір, фактично надаючи цим повну перемогу другому гравцю.

Один зі способів подолання проблеми полягає у використанні «схеми зобов'язань», тобто той, хто починає гру, передає свій предмет у вигляді «затемненого зобов'язання», яке суперник не зможе прочитати до певного моменту. Після того, як Боб назве свою річ, Аліса відкриває зобов'язання.

При цьому важливо, щоб інший гравець міг переконатися у незмінності зобов'язання в період між передачею та відкриттям. Найпростіший спосіб це реалізувати – використати хеш-функцію за схемою:

$A \to B$: $H_a(r_a || \text{папір})$,

$B \to A$: ножиці,

$A \to B$: r_a, папір.

Розглянемо властивості такої схеми зобов'язань. Нехай спочатку Аліса передає свій вибір «папір», посилаючи Бобу хеш-значення H_a. При цьому Боб не може визначити прихований зміст зобов'язання, бо не знає випадкового значення r_a. Також він не може обернути хеш-функцію.

Властивість схеми зобов'язань, що забезпечує неможливість прочитання інформації, що міститься в повідомленні, називається **приховуванням**.

Далі Боб посилає Алісі слово «ножиці». Аліса, отримавши його повідомлення, навіть знаючи про свою поразку, не зможе зшахраювати, оскільки не має можливості замінити r_a на якесь інше r_a', що задовольняє умову

$$h(r_a||\text{папір}) = h(r_a'||\text{камінь}).$$

Тут достатньо вимагати, щоб хеш-функція була захищена від других прообразів.

Властивість схеми зобов'язань, завдяки якій Боб не здатний змінити вміст переданого повідомлення, називається **зв'язуючою**.

Кажуть, що схема зобов'язань

- **теоретико-інформаційно зв'язуюча**, якщо той, хто передає зобов'язання, не може змінити його значення, навіть маючи необмежені обчислювальні можливості;
- **обчислювально зв'язуюча**, якщо відправник може змінити значення зобов'язання, але в результаті величезної кількості обчислень;
- **теоретико-інформаційно приховуюча**, якщо одержувач не здатний визначити зміст зобов'язання до стадії розкриття незалежно від його обчислювальних можливостей;
- **обчислювально приховуюча**, якщо одержувач може дізнатися вміст зобов'язання до його розкриття, але внаслідок залучення невиправдано великої, хоч і скінченної кількості обчислювальних ресурсів.

З визначення випливає наступна лема.

Лема 11. Схема зобов'язань $h(r||c)$ з випадковим значенням r, зобов'язанням c та криптографічною хеш-функцією h у кращому випадку є

- обчислювально зв’язуючою,
- обчислювально приховуючою.

Доведення. Всі криптографічні хеш-функції, які обговорювалися раніше, лише обчислювально захищені від відновлення прообразів та других прообразів. Зв'язуюча властивість гарантована лише захищеністю хеш-функції від других прообразів. Отже, ця властивість захищена лише в обчислювальному сенсі.

Аналогічно, оскільки приховуюча властивість забезпечується захищеністю хеш-функції від відновлення прообразів, а воно захищене лише обчислювально. Що і потрібно було довести. ■

Можна запропонувати дві таких схеми, що приховуюча або зв'язуюча властивість в них матиме теоретико-інформаційну стійкість.

Нехай A – скінченна абелева група простого порядку Q і твірною G, B – елемент цієї групи, чий дискретний логарифм за основою G не відомий жодному з користувачів системи.

Останню властивість досить просто забезпечити. Розглянемо, наприклад, мультиплікативну групу скінченного поля $\mathbb{F}_P^*$, виберемо Q серед дільників числа $P - 1$ і побудуємо G методом, аналогічним якому можна відшукати й B:

1. Візьмемо довільне ціле R.

2. Обчислимо $F = H(R)$ (при цьому F належить $\mathbb{F}_P^*$) для деякої криптографічної хеш-функції H,

3. Покладемо $B = F^{(P-1)/Q} \pmod P$. Якщо $B = 1$ то повернемося до першого кроку. Якщо $B \neq 1$, то виведемо пару (R, B).

Таким чином отримано випадковий елемент групи $\mathbb{F}_P^*$ порядку Q, причому ця випадковість гарантується випадковим вибором R.

Тепер за парою (G, B) побудуємо дві схеми: $D(x)$ та $D_a(x)$.

$$D(x) = G^x, \quad D_a(x) = B^x G^a.$$

де a – випадкове ціле число за модулем Q. Величина a називається **затемняючою**, оскільки вона приховує передане число x навіть від нападника з необмеженими обчислювальними можливостями.

Щоб розкрити зобов'язання, користувач відкриває значення x у першій схемі та пару (a, x) – в другій.

Лема 12. Схема зобов'язань $D(x)$ є не більше ніж теоретико-інформаційно зв'язуючою і обчислювально приховуючою.

Доведення. Нехай Аліса опублікувала $C = D(x) = G^x$ і намагається змінити переданий цим зобов'язанням елемент із $\mathbb{Z}/Q\mathbb{Z}$. Очевидно, що незалежно від обчислювальних можливостей Аліси, в $\mathbb{Z}/Q\mathbb{Z}$ існує лише один елемент, що є дискретним логарифмом від C за основою G, а саме x. Таким чином, ця система теоретико-інформаційно зв'язуюча.

Тепер припустимо, що одержувач зобов'язання бажає визначити прихований елемент. Все, що потрібно для цього зробити, – це знайти дискретний логарифм за основою G. Тому наведена схема приховує лише в обчислювальному сенсі. Що й треба було довести. ■

Лема 13. Схема зобов'язань $D_a(x)$ – обчислювально зв'язуюча (якщо відправник не знає дискретного логарифму B за основою G), але теоретико-інформаційно приховуюча.

Доведення. Припустимо, що Аліса, передавши зобов'язання $M = D_a(x) = B^x G^a$, хоче змінити x на y. Для цього їй достатньо знайти $F = M/B^y$, після чого вона може обчислити дискретний логарифм $a' = \log_G F$ і стверджувати, що надіслала пару (a', y), а не (a, x). Звідси випливає, що схема обчислювально зв'язуюча.

Припустимо, одержувач збирається відновити переданий йому x, не чекаючи стадії розкриття. Оскільки для даного значення M та кожного x існує єдине a, при якому $M = B^x G^a$, то навіть маючи необмежені обчислювальні можливості, одержувач неспроможний відновити a з M. Що й потрібно було довести. ■

Варто зауважити, що обидві розглянуті схеми зобов'язань, засновані на дискретному логарифмуванні, мають властивість гомоморфності:

$$D(x_1) \cdot D(x_2) = G^{x_1} \cdot G^{x_2} = G^{x_1+x_2} = D(x_1 + x_2),$$

$$D_{a_1}(x_1) \cdot D_{a_2}(x_2) = B^{x_1}G^{a_1} \cdot B^{x_2}G^{x_2} = B^{x_1+x_2} \cdot G^{x_1+x_2} = D_{a_1+a_2}(x_1 + x_2).$$

11.2 Доведення з нульовим розголошенням

Доведення з нульовим розголошенням – це інтерактивний протокол, що дозволяє одній зі сторін (верифікатору) переконатися в достовірності будь-якого твердження (зазвичай математичного), не отримавши від іншої сторони жодної іншої інформації крім тієї, що її твердження достовірне.

Доведення з нульовим розголошенням повинно мати три властивості:

1. Повнота: якщо твердження справді вірне, то інша сторона переконає в цьому верифікатора.

2. Коректність: якщо твердження хибне, то навіть нечесні дії іншої сторони не зможуть переконати перевіряючого за винятком нехтовно малої ймовірності.

3. Нульове розголошення: якщо твердження вірне, то будь-який (навіть нечесний) верифікатор не дізнається нічого, окрім самого факту достовірності твердження.

Процес доведення продовжується певну кількість раундів. Кожен раунд доказу складається із трьох етапів. Схематично їх можна зобразити так:

A $\rightarrow$ B: доказ

B $\rightarrow$ A: виклик

A $\rightarrow$ B: відповідь

Спочатку A вибирає із заздалегідь визначеної множини деякий елемент, який стає її секретом (закритий ключ). На основі цього елемента обчислюється, а потім публікується відкритий ключ. Факт знання секрету визначає множина питань, на які A завжди зможе дати правильні відповіді, при цьому не розголошуючи секретний ключ.

Потім А вибирає випадковий елемент з множини за певними правилами, що залежать від конкретного алгоритму, обчислює доказ і надсилає його В. Після цього В вибирає з усієї множини питань одне і просить А відповісти на нього (виклик).

А посилає В відповідь на запитання. Отриманої інформації для В достатньо, щоб перевірити, чи справді А володіє секретом.

Раунди можна повторювати скільки завгодно раз, поки ймовірність того, що А вгадує відповіді не стане достатньо низькою. Подібна техніка називається також «розрізати і вибрати».

Розглянемо приклад. Допустимо Аліса знає гамільтонів цикл у великому графі *G*. Бобу відомий граф *G*, але він не знає гамільтонового циклу в ньому. Аліса хоче довести Бобу, що вона знає гамільтонів цикл, не видаючи при цьому ні самого циклу, ні жодної інформації про нього.

Для цього Аліса та Боб спільно виконують кілька раундів протоколу:

Спочатку Аліса створює граф *H*, ізоморфний *G*. Перетворення гамільтонового циклу між ізоморфними графами – тривіальна задача, тому якщо Алісі відомий гамільтонів цикл у *G*, вона також знає гамільтонів цикл в *H*. Аліса передає граф *H* Бобу.

Боб вибирає випадковий біт $b \to \{0, 1\}$.

Якщо $b = 0$, Боб просить Алісу довести ізоморфізм *G* та *H*, тобто показати відповідність вершин цих двох графів. Боб може перевірити, чи дійсно *G* і *H* ізоморфні.

Якщо $b = 1$, Боб просить Алісу показати гамільтонів цикл в *H* (для задачі ізоморфізму графів наразі не доведена ні її приналежність класу P, ані її NP-повнота, тому вважатимемо, що неможливо з гамільтонового циклу *H* вирахувати гамільтонів цикл в *G*).

В кожному раунді Боб вибирає новий випадковий біт, який невідомий Алісі, тому щоб вона могла відповісти на обидва запитання, треба, щоб *H* був дійсно ізоморфний *G* й Аліса повинна знати гамільтонів цикл в *H* (а отже, і в *G*). Тому,

після достатньої кількості раундів, Боб може бути впевнений у тому, що Аліса справді знає гамільтонів цикл в G.

З іншого боку, Аліса таким чином не розкриває жодної інформації про гамільтоновий цикл G. Більше того, Бобу складно буде довести комусь іще, що він сам або Аліса знає гамільтонів цикл у G.

Припустимо, що в Аліси немає гамільтонового циклу в G, але вона хоче надурити Боба. Тоді Алісі потрібен неізоморфний G граф G', в якому вона все ж знає гамільтонів цикл. В кожному раунді Аліса може передавати Бобу або граф H' – ізоморфний G', чи граф H – ізоморфний G. Якщо Боб попросить довести ізоморфізм, а було передано H, то обман не розкриється. Аналогічно, якщо він попросить показати гамільтонів цикл і було передано H'. В такому разі ймовірність того, що Аліса все-таки обдурить Боба після n раундів, становить $1/2^n$, що може бути меншим за будь-яку заздалегідь задану величину при достатній кількості раундів.

Припустимо, що Боб не знайшов гамільтонів цикл, але хоче довести Чарлі, що Аліса його знає. Навіть якщо Боб, наприклад, зняв на відео всі раунди протоколу, Чарлі навряд-чи повірить йому. Чарлі може припустити, що Боб з Алісою змовилися, і в кожному раунді Боб заздалегідь повідомляв Алісі свій вибір випадкового біту, щоб вона могла передавати йому H для перевірок ізоморфізму і H' для перевірок гамільтонового циклу. Таким чином без участі Аліси переконати, що вона знає гамільтонів цикл, можна лише довівши, що в усіх раундах протоколу вибиралися справді випадкові біти.

11.3 Протокол з нульовим розголошенням для схем голосування

Розглянемо протокол доведення з нульовим розголошенням, який можна використовувати у схемах голосування.

Візьмемо схему зобов'язань $D_a(x) = B^x G^a$, де $A = \{G\}$ – скінченна абелева група простого порядку Q, B – елемент A, чий дискретний логарифм за основою G невідомий, x – зобов'язання, a – випадкове унікальне число. Цікавим є випадок, коли зобов'язання x може набувати значень плюс або мінус одиниця. При

реалізації цієї схеми в протоколі голосування буде важливо довести, що передане значення дійсно належить зазначеній множині, не розкриваючи його справжнього значення.

Протокол виглядатиме наступним чином.

1. Разом з публікацією зобов'язання $D_a(x)$, той, хто доводить, вибирає випадкові числа d, r та w за модулем Q і публікує A_1 і A_2, де

$$A_1 = \begin{cases} G^r (D_a(x)B)^{-d}, & \text{якщо } x = 1, \\ G^w, & \text{якщо } x = -1; \end{cases}$$

$$A_2 = \begin{cases} G^w, & \text{якщо } x = 1, \\ G^r (D_a(x)B^{-1})^{-d}, & \text{якщо } x = -1. \end{cases}$$

2. Перевіряючий надсилає запит c.

3. Сторона, що доводить, відповідає

$$(D_1, D_2, R_1, R_2) = \begin{cases} (d, d', r, r'), & x = 1, \\ (d', d, r', r), & x = -1, \end{cases}$$

де $d' = c - d$, $r' = w + a \cdot r'$.

4. Верифікатор здійснює перевірку

$$c = D_1 = D_2,$$

$$G^{R_1} = A_1 (D_a(x)B)^{D_1},$$

$$G^{R_2} = A_2 (D_a(x)B^{-1})^{D_1}.$$

11.4 Система електронного голосування

Припустимо, що в голосуванні беруть участь m осіб з правом голосу та n лічильних комісій. Використання великої кількості лічильних комісій забезпечує анонімність голосуючих та запобігає можливості фальсифікації результатів голосування. Будемо також вважати, що виборці можуть віддати свій голос лише за одну з двох партій.

Система голосування має наступні властивості:

- голосувати мають право лише уповноважені виборці;
- жоден з тих, хто голосує, не може віддати більше одного голосу;
- жоден з учасників процесу не може дізнатися, як проголосував хтось інший;

– ніхто не може продублювати голос якогось іншого учасника кампанії;
– кінцевий результат буде коректно підрахований;
– кожен з учасників здатний переконатися, що результат підрахований правильно;
– протокол працюватиме і у випадку, коли деякі з учасників нечесні.

Установки системи. Кожна з лічильних комісій має шифруючу функцію з відкритим ключем E_i. Припустимо, що зафіксовано скінченну абелеву групу A простого порядку Q, в якій обрано пару елементів B і G, причому ніхто (включаючи лічильні комісії) не знає розв'язку рівняння $B = G^x$. Кожен з голосуючих має алгоритм підпису з відкритим ключем.

Заповнення бюлетеня. Кожен з m виборців обирає голос v_j рівним або –1, або 1 та випадкове затемнювальне число a_j з $\mathbb{Z}/Q\mathbb{Z}$ і публікує своє розв'язок

$$D_j = D_{aj}(v_j)$$

використовуючи схему зобов'язань. Це розв'язок стає відомим усім сторонам голосування: як лічильним комісіям, так і решті виборців. Голос разом із доказом підписується за допомогою схеми підпису, що належить виборцю.

Розподіл бюлетенів. Для підбиття підсумків необхідно передати віддані голоси лічильним комісіям. Щоб передати a_j та v_j лічильним комісіям, кожен виборець j застосовує схему Шаміра розділення секрету. З цією метою він обирає два випадкових многочлени за модулем Q степеня $T < n$

$$R_j(X) = v_j + r_{1j}X + \cdots + r_{Tj}X^T,$$
$$S_j(X) = a_j + s_{1j}X + \cdots + s_{Tj}X^T$$

та обчислює

$$\left(u_{ij}, w_{ij}\right) = \left(R_j(i), S_j(i)\right), \ 1 \le i \le n.$$

Голосуючий шифрує пари $\left(u_{ij}, w_{ij}\right)$, використовуючи відкритий ключ E_i i-ї лічильної комісії та надсилає їй отриманий результат. Після цього виборець відкрито реєструє

$$D_{ij} = D_{a(l,j)}(r_{lj}),$$

де $a(l,j) = s_{lj}$ при $1 \le l \le T$.

Перевірка достовірності інформації. Кожна з лічильних комісій повинна перевірити, що пара (u_{ij}, w_{ij}) справді отримана від виборця з номером j та узгоджується з переданим зобов'язанням. Це досягається в результаті перевірки наступних рівностей:

$$D_j \prod_{l=1}^{T} D_{lj}^{i'} = D_{a_j}(v_j) \prod_{l=1}^{T} D_{s_{lj}}(r_{lj})^{i'} = B^{v_j} G^{a_j} \prod_{l=1}^{T} (B^{r_{lj}} G^{s_{lj}})^{i'} =$$

$$= B^{(v_j + \sum_{l=1}^{T} r_{lj} i')} \cdot G^{(a_j + \sum_{l=1}^{T} s_{lj} i')} = B^{u_{ij}} \cdot G^{w_{ij}} .$$

Підрахунок голосів. Кожна з n лічильних комісій підраховує бюлетені та публікує результати:

$$U_i = \sum_{j=1}^{m} u_{ij}.$$

Крім того, вона оприлюднить суму затемнюючих факторів:

$$W_i = \sum_{j=1}^{m} w_{ij}.$$

Будь-який інший учасник процедури голосування може переконатися в коректності опублікованих сум, перевіряючи рівність

$$\prod_{j=1}^{m} \left(D_j \prod_{l=1}^{T} D_{lj}^{j'} \right) = \prod_{j=1}^{m} B^{u_{ij}} \cdot G^{w_{ij}} = B^{U_i} \cdot G^{W_i}.$$

Таким чином, якщо $\sum_{j=1}^{n} U_j > 0$, то більшість проголосувала за партію, закодовану як 1, якщо ж ця сума менше 0, то більше голосів у партії, закодованої як –1.

Література

1. Горбенко І. Д., Горбенко Ю. І. Прикладна криптологія. Теорія. Практика. Застосування. – Харків: Форт, 2012. – 870 с.

2. Вербіцький О. В. Вступ до криптології. – Львів: Науково-технічна література, 1998. – 249 с.

3. Остапов С. Е., Валь Л. О. Основи криптографії. Навчальний посібник. – Чернівці: Книги – XXI, 2008. – 188 с.

4. Smart N. P. Cryptography Made Simple. – Springer, 2016. – 493 p.

5. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. – CRC Press, 2001. – 780 p.

6. Paar C., Pelzl J. Understanding Cryptography: A Textbook for Students and Practitioners. – Springer, 2010. – 390 p.

7. Katz J., Lindell Y. Introduction to Modern Cryptography, 3rd edition. – CRC Press, 2021. – 648 p.

8. Mao W. Modern Cryptography: Theory and Practice. – Prentice Hall, 2003. – 707 p.

9. Klima R. E., Sigmon N. P. Cryptology: Classical and Modern, 2nd edition. – CRC Press, 2018. – 496 p.

10. Stinson D. R., Paterson M. Cryptography: Theory and Practice, 4th edition. – CRC Press, 2018. – 598 p.

11. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C, 20th Anniversary Edition. – Wiley, 2015. – 792 p.

12. Wong D. Real-World Cryptography. – Manning, 2021. – 400 p.

13. Stallings W. Cryptography and Network Security: Principles and Practice, 8th edition. Global edition. – Pearson, 2023. – 832 p.

Навчальне видання

Галкін Олександр Володимирович

Шкільняк Оксана Степанівна

ОСНОВИ КРИПТОЛОГІЇ

Навчальний посібник