Лекція 10. Криптографічні алгоритми

§1. Криптографічні алгоритми

- **Криптологією** (від грец. «криптос» - таємний і «логос» - знання) називають галузь знань, що складається з криптографії і крипто аналізу, які займаються шифруванням та розшифруванням повідомлень.
- **Криптографією** називають галузь знань про способи перетворення (шифрування) інформації з метою її захисту від незаконних користувачів.
- **Криптоаналізом** називають галузь знань про методи та способи розкриття шифрів.
- Криптографія відома давно. Вона використовує найрізноманітніші шифри як чисто інформаційні, так і механічні. Зараз найбільше практичне значення має захист інформації в комп'ютерах і різноманітних системах управління, тому в подальшому будемо розглядати тільки програмні шифри для повідомлень в алфавіті {0,1}.
- **Криптографічним алгоритмом** називають алгоритм, в якому використовують математичну функцію для зашифровування і розшифровування інформації.
- **Зашифровуванням інформації** називають процес перетворення відкритих даних на закриті (зашифровані) за допомогою певних правил, які визначені в шифрі.
- **Розшифровуванням інформації** називають процес перетворення закритих (зашифрованих) даних на відкриті за допомогою певних правил, які визначені в шифрі.

Шифром криптографічного алгоритму називають сукупність обернених перетворень множини відкритих даних на множину зашифрованих даних, заданих алгоритмом криптографічного перетворення.

Розкриттям (зламуванням) шифру називають процес перетворення закритих даних на відкриті при невідомому ключі й (або) невідомому алгоритмі.

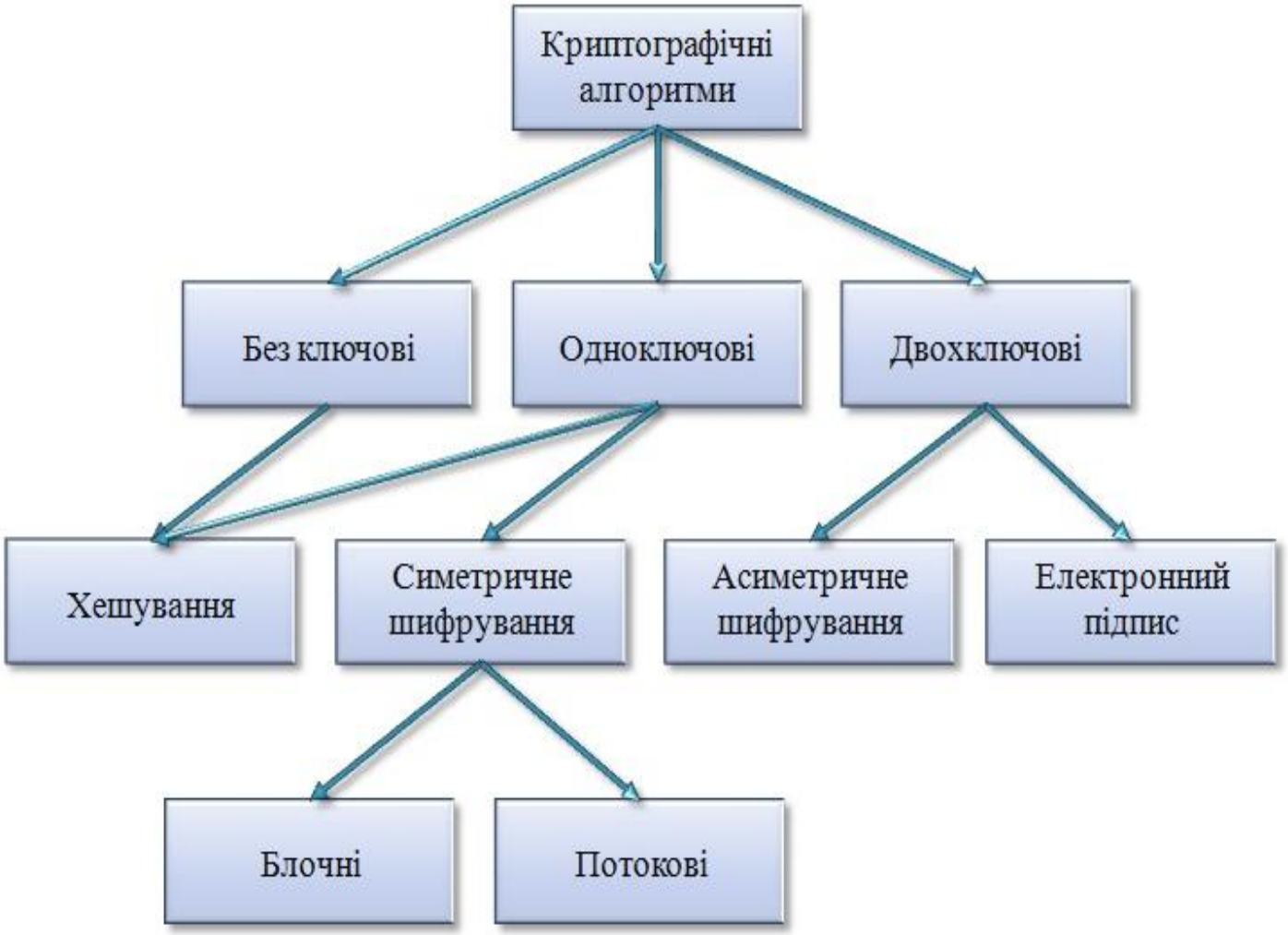
Ключем криптографічного алгоритму називають конкретний секретний стан параметрів криптографічного алгоритму, який забезпечує вибір одного варіанта перетворення із сукупності можливих.

Криптографічні алгоритми можуть бути обмежені, симетричні (із закритим ключем).

Обмежений алгоритм ґрунтується на збереженні самого алгоритму в таємниці, тому такі алгоритми мають тільки історичний інтерес і не відповідають теперішнім стандартам.

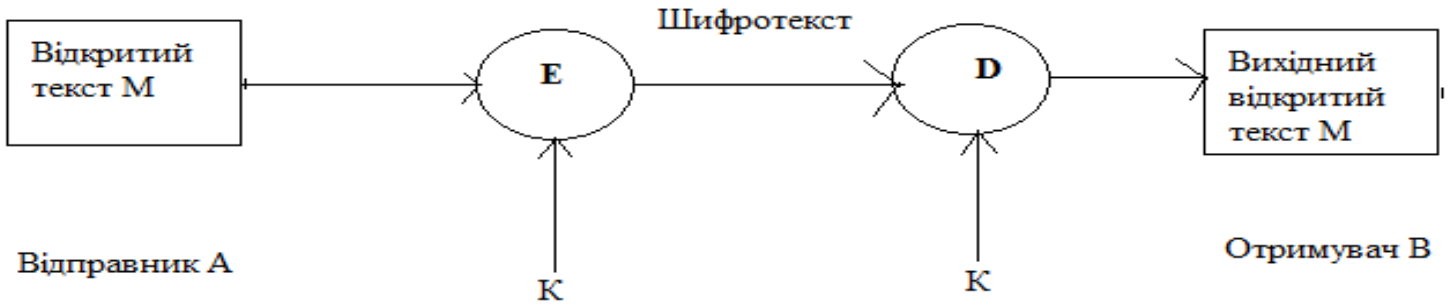
Щоб правильно задіяти криптоалгоритм (КА), тобто забезпечити надійний і адекватний захист, потрібно розуміти, які бувають КА і який тип алгоритму краще пристосований для вирішення конкретного завдання.

Основна схема класифікації криптоалгоритмів



;

Симетричним криптографічним алгоритмом називають алгоритм, в основу якого покладено зберігання в таємниці ключа зашифровування. Тобто, це алгоритм, який використовує для зашифровування і розшифровування один і той самий ключ.



Функції зашифровування і розшифровування в таких системах мають вигляд $E_k(M)=C \quad D_k(M)=M$

де M – відкритий текст повідомлення; E_k - процес зашифровування з ключем К;

D_k - процес розшифровування з ключем К.

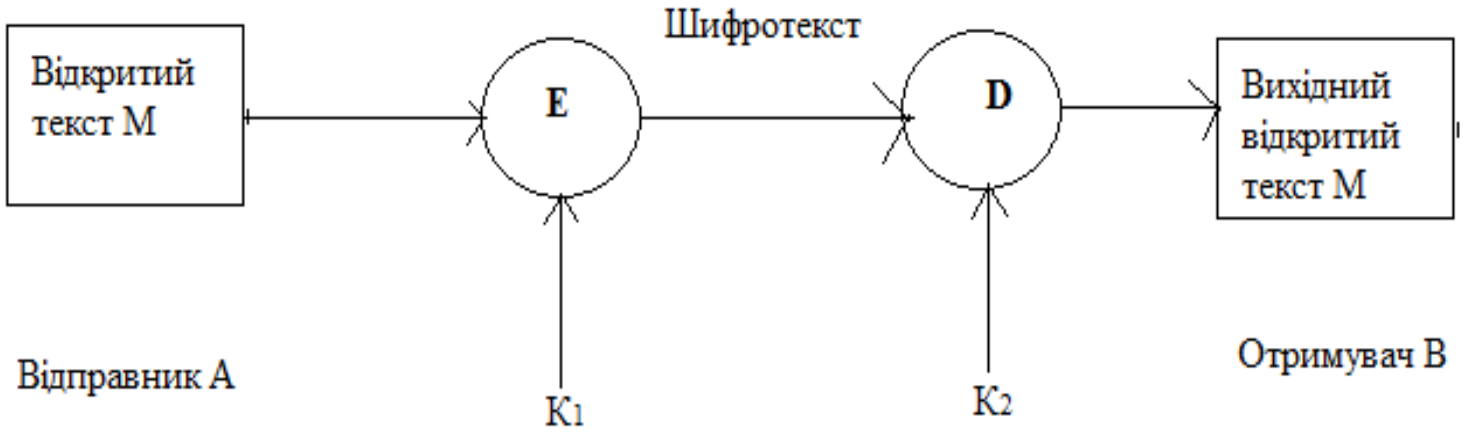
При цьому справедлива рівність $D_k(E_k(M))=M$

. Наведені раніше рівності справедливі лише для систем із симетричними криптоалгоритмами. Тобто ці алгоритми потребують, щоб перед початком передавання таємних повідомлень учасники обміну погодили, який ключ буде використовуватися.

Асиметричним криптографічним алгоритмом називають алгоритм, у якого ключ, використаний пр. зашифровуванні повідомлення, відрізняється від ключа при його розшифровуванні.

Крім того, ключ розшифровування повідомлення не може бути (принаймні в період якогось проміжку часу) розрахований за ключем розшифровування. Такі алгоритми називають алгоритмами з відкритим ключем тому, що ключ розшифровування може бути відкритим для використання, але тільки відповідний користувач з відповідним ключем розшифровування може розшифрувати повідомлення, у таких системах ключ зашифровування називають відкритим ключем, а ключ розшифровування – закритим.

Блок-схему зашифровування та розшифровування в асиметричних криптосистемах наведено на рис.



Функції зашифровування і розшифровування в таких системах мають вигляд:

$E_{k_1}(M)=C \quad D_{k2}(M)=M$, де M – відкритий текст повідомлення;

E_{k_1} - процес зашифровування з ключем K_1 D_{k_2} - процес розшифровування з ключем K_2

При цьому справедлива рівність $D_{k_2}\left(E_{k_1}(M)\right)=M$

В залежності від кількості ключів, які застосовуються у конкретному алгоритмі:

•**Безключові КА** – не використовують в обчисленнях ніяких ключів;

•**Одноключові КА** – працюють з одним додатковим ключовим параметром (якимсь таємним ключем);

•**Двухключові КА** – на різних стадіях роботи в них застосовуються два ключових параметри: секретний та відкритий ключі.

В задежності від характеру впливів, що виробляються над даними, алгоритми підрозділяються на:

•**Перестановочні** - Блоки інформації (байти, біти, більші одиниці) не змінюються самі по собі, але змінюється їх порядок проходження, що робить інформацію недоступною сторонньому спостерігачеві.

•**Підстановочні** - Самі блоки інформації змінюються за законами криптоалгоритму. Переважна більшість сучасних алгоритмів належить цій групі.

Залежно від розміру блоку інформації криптоалгоритми поділяються на:

•**Потокові шифри** - Одиницею кодування є один біт. Результат кодування не залежить від минулого раніше вхідного потоку. Схема застосовується в системах передачі потоків інформації, тобто в тих випадках, коли передача інформації починається і закінчується в довільні моменти часу і може випадково перериватися. Найбільш поширеними представниками потокових шифрів являються скремблери.

•**Блочні шифри** - Одиницею кодування є блок з декількох байтів (в даний час 4-32). Результат кодування залежить від усіх вихідних байтів цього блоку. Схема застосовується при пакетній передачі інформації та кодування файлів.

У кожній конкретній ситуації вибір кпиптоалгоритму визначають такими факторами:

•Особливістю інформації, яку захищають (документи, графічні файли, вихідні тексти програм і т. інш.);

•Особливістю середовища зберігання і передачі інформації;

•Цінністю інформації, характером секретів, часом зберігання таємниці%

•Обсягом інформації, швидкістю її передавання, ступенем оперативності надання її користувачеві;

•Можливостями власників інформації, власників засобів збирання, оброблення, зберігання та передачі інформації з її захисту;

•Характером погроз, можливостями супротивника.

Криптографічним безпечним алгоритмом називають алгоритм, у якому:

•Вартість його зламування більша, ніж вартість зашифрованої інформації;

•Час зламування алгоритму більший, ніж час, протягом якого повинна зберігатися в таємності зашифрована інформація;

•Обсяг зашифрованої інформації ключем менший, ніж обсяг даних, необхідний для зламування алгоритму;

Є багато різних криптографічних алгоритмів, з яких виділимо найпоширеніші.

DES (Data Encryption Standard, стандарт шифрування даних) алгоритм – найпопулярніший комп'ютерний алгоритм шифрування, є міжнародним стандартом. Це симетричний алгоритм, у якому один і той самий ключ використовують для зашифровування і розшифровування інформації.

RSA (Rivest-Shamir-Adleman, прізвища його розробників) алгоритм – найпопулярніший асиметричний алгоритм з відкритим ключем; його використовують для шифрування і розшифровування повідомлень, а також для цифрових підписів.

Алгоритм Ель-Гамаля – асиметричний алгоритм з відкритим ключем, який використовують для шифрування і розшифровування повідомлень, а також для цифрових підписів.

Алгоритм Рабіна – асиметричний алгоритм з відкритим ключем, який використовують тільки для шифрування і розшифровування повідомлень.

Алгоритм Діффі-Хеллмана дає змогу обмінюватися секретним ключем для симетричних криптосистем з використанням каналу, захищеного від модифікацій.

Стандарт шифрування даних DES (Data Encryption Standard) був розроблений у 70-х роках фахівцями IBM і у 1976 році був прийнятий у якості федерального стандарту Сполучених Штатів для захисту комерційної та урядової інформації, не пов'язаної з національною безпекою.

DES оперує з інформацією, поданою у двійковій формі, а довжина блоку і довжина ключа вибрані рівними 64. Іншими словами, двійкове повідомлення M розбивається на блоки по 64 біти і шифрується кожен блок окремо, використовуючи один і той же двійковий ключ K довжини 64. Таким чином, повідомлення $M = M_1 M_2 M_3 \ldots$ перетворюється у криптотекст $C = C_1 C_2 C_3 \ldots$, де $C_i = E_k(M_i)$

У стандарті DES кожен блок криптотексту C, також є двійковою послідовністю довжини 64.

Алгоритм шифрування E повинен задовольняти трьом умовам:

1) можливість дешифрування: для будь-якого ключа K різним блокам повідомлення M' і M'' відповідають різні блоки криптотексту $E_k(M')$ і $E_k(M'')$,інакше кажучи, алгоритм E_k з будь-яким ключем здійснює перестановку двійкових послідовностей довжини 64;

2) ефективність: і шифрування, і дешифрування відбуваються швидко;

3) надійність: якщо ключ невідомий, то немає способу розкриття шифру.

Схема алгоритму DES працює з 64-бітовим блоком відкритого тексту. Після початкової перестановки блок розбивається на праву і ліву половини довжиною по 32 біта. Потім виконується 16 етапів однакових дій, функцією f у яких дані поєднуються з ключем. Після шістнадцятого етапу права і ліва половини поєднуються й алгоритм завершується заключною перестановкою (зворотною відносно початкової)

На кожному етапі біти ключа зсуваються і потім з 56 бітів ключа вибираються 48 бітів. Права половина даних збільшується до 48 бітів за допомогою перестановки з розширенням, поєднується за допомогою XOR з 48 бітами зміщеного і переставленого ключа, проходить через 8 S-блоків, утворюючи 32 нових біта, і переставляється знову. Ці чотири операції і виконуються функцією f (див. рисунок 5.2). Потім результат функції f поєднується з лівою половиною за допомогою іншого XOR. У підсумку цих дій з'являється нова права половина, а вихідна права половина стає новою лівою. Ці дії повторюються 16 разів, тобто здійснюється 16 етапів DES.

Якщо *Bi* - це результат *i*-ої ітерації, *Li* і *Ri* - ліва і права половини *Bi*, *Ki* - 48- бітовий ключ для етапу *i*, а *f* - це функція, що виконує усі підстановки, перестановки і XOR із ключем, то етап можна представити так: $L_i = R_{i-1} \quad R_i = L_{i-1} \oplus f\left(R_{i-1}, K_i\right)$

Початкова перестановка виконується ще до етапу 1, при цьому 58-й біт переставляється у бітову позицію 1, 50-й біт - у бітову позицію 2, 42-й біт - у бітову позицію 3, і так далі (таблиця 5.1).

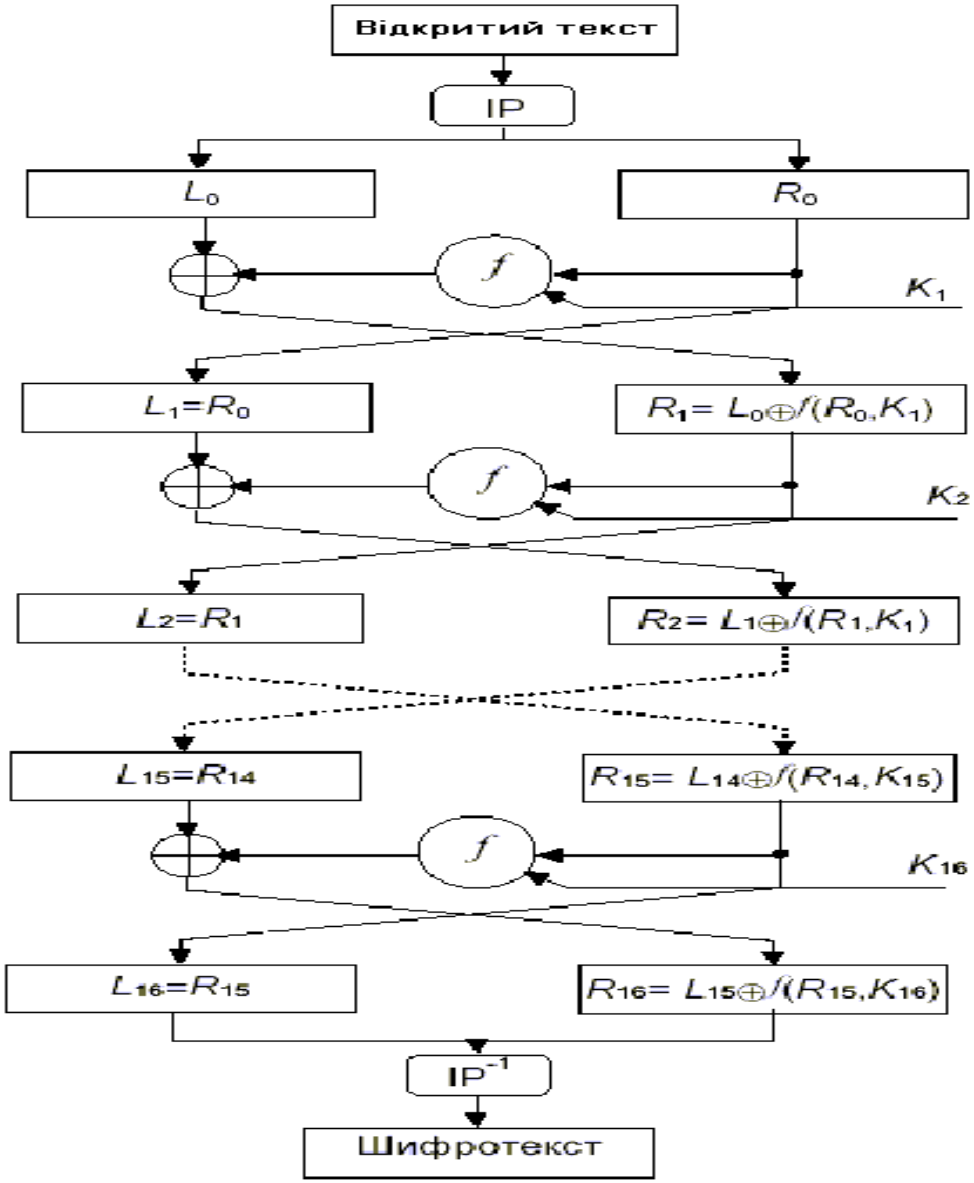


Рисунок 5.1 – Алгоритм DES

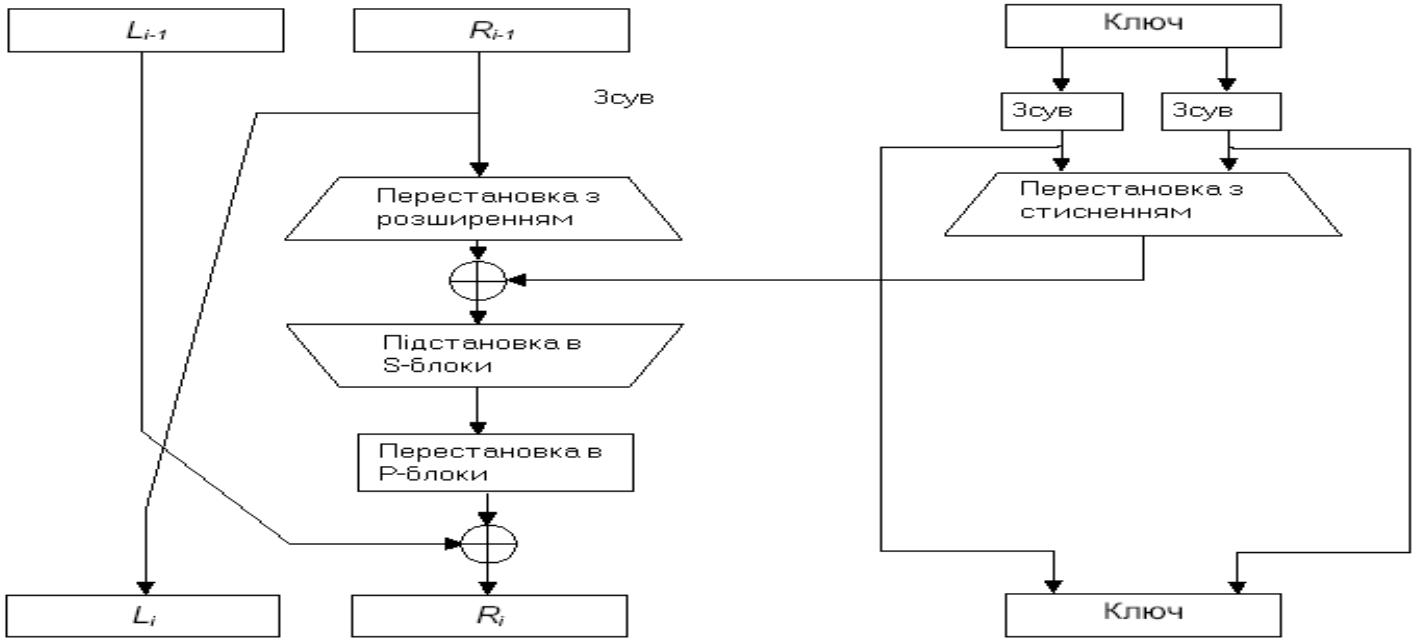


Рисунок 5.2 - Один етап DES

Початкова перестановка і відповідна завершальна перестановка не впливають на безпеку DES. Оскільки програмна реалізація цієї багатобітової перестановки нелегка (на відміну від тривіальної апаратної), у багатьох 12
програмних реалізаціях DES початкова і заключна перестановка не використовуються. Хоча такий новий алгоритм не відповідає стандарту DES.

Таблиця 5.1 - Початкова перестановка

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

64-бітовий ключ DES спочатку зменшується до 56-бітового шляхом відкидання кожного восьмого біта (таблиця 5.2). Ці біти використовуються тільки для контролю парності, дозволяючи перевіряти правильність ключа.

Таблиця 5.2 - Перестановка ключа

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

Після отримання 56-бітового ключа для кожного з 16 етапів DES генерується новий 48-бітовий підключ. Ці підключі *Ki*, визначаються в такий спосіб. Спочатку 56-бітовий ключ поділяється на дві 28-бітові половинки, які циклічно зсуваються ліворуч на один чи два біти в залежності від етапу (таблиця 5.3).

Таблиця 5.3 - Число бітів зсуву в залежності від етапу

Этап	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після зсуву вибирається 48 з 56 бітів. Оскільки при цьому не тільки вибирається підмножина бітів, але і змінюється їхній порядок, ця операція називається перестановка із стисненням (таблиця 5.4). Її результатом є набір з 48 бітів. Наприклад, біт зсунутого ключа в позиції 33 переміщається в позицію 35 результату, а 18-й біт зсунутого ключа відкидається.

Таблиця 5.4 - Перестановка зі стисненням

14,	17,	11,	2,4,	1,	5,	3,	28,	15,	6,	21,	10,
23,	19,	11,	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

Через зсув для кожного підключа використовується інша підмножина бітів ключа. Кожен біт використовується приблизно в 14 з 16 підключів. 13

Права половина даних *Ri* розширюється від 32 до 48 бітів. Оскільки при цьому не просто повторюються визначені біти, але і змінюється їхній порядок, ця операція називається перестановкою з розширенням. Іноді вона називається *E*-блоком. Для кожного 4-бітового вхідного блоку перший і четвертий біт являють собою два біти вихідного блоку, а другий і третій біти - один біт вихідного блоку. Наприклад, біт вхідного блоку в позиції 3 переміститься в позицію 4 вихідного блоку, а біт вхідного блоку з позиції 21 - у позиції 30 і 32 вихідного блоку (рисунок 5.3).

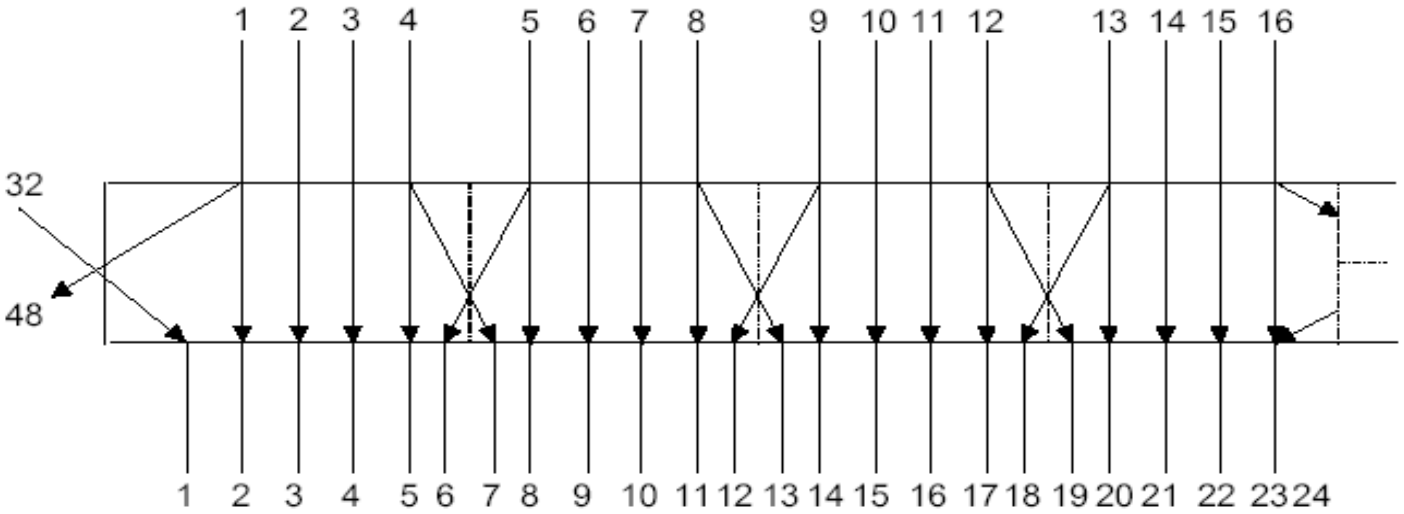


Рисунок 5.3 - Перестановка з розширенням

Хоча вихідний блок більше вхідного, кожен вхідний блок генерує унікальний вихідний блок (таблиця 5.5).

Таблиця 5.5 - Перестановка з розширенням

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

Після об'єднання стиснутого блоку з розширеним за допомогою XOR над 48-бітовим результатом виконується операція підстановки. Підстановки здійснюються у восьми блоках підстановки (S-блоках). У кожного S-блоку 6-бітовий вхід і 4-бітовий вихід. 48 бітів поділяються на вісім 6-бітових підблоків. Кожен окремий підблок обробляється окремим S-блоком: перший підблок - S-блоком 1, другий - S-блоком 2, і так далі (рисунок 5.4).

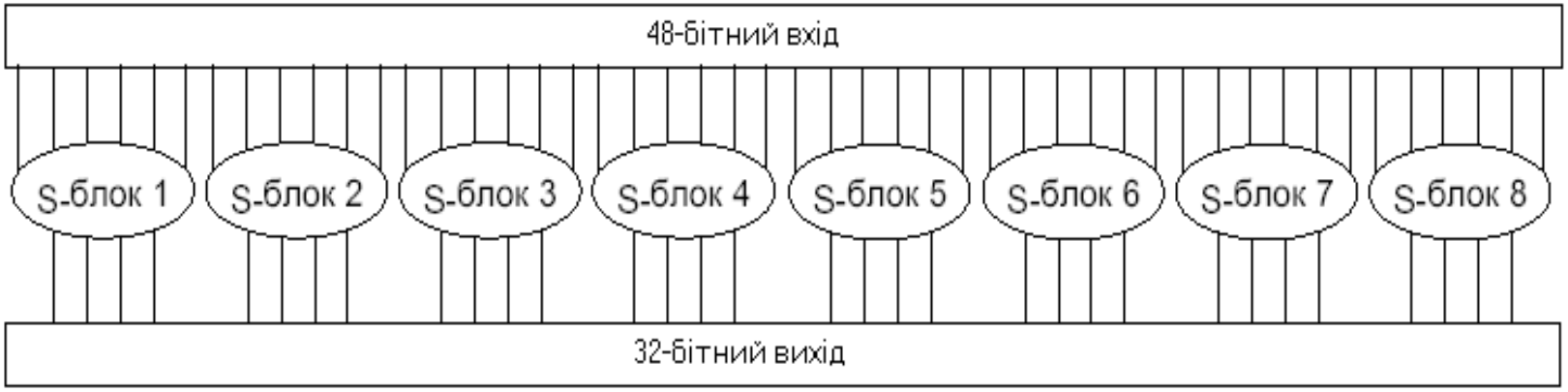


Рисунок 5.4 - Підстановка - S-блоки.

Кожен S-блок являє собою таблицю з 2 рядків і 16 стовпців. Кожен елемент у блоці є 4-бітовим числом. По 6 вхідних бітах S-блоку визначається, під якими номерами стовпців і рядків шукати вихідне значення (таблиця 5.6).

Таблиця 5.6 - S-блоки

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

S-блок 2:

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

S-блок 3:

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

S-блок 4:

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

S-блок 5:

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

S-блок 6:

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

S-блок 7:

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

S-блок 8:

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

Вхідні біти певним чином визначають елемент S-блоку. Розглянемо 6-бітовий вхід S-блоку: $b1$, $b2$, $b3$, $b4$, $b5$ і $b6$. Біти $b1$ і $b6$ поєднуються, утворюючи 2-15 бітове число від 0 до 3, що відповідає рядку таблиці. Середні 4 біти з $b2$ по $b5$ об'єднуються утворюючи 4-бітове число від 0 до 15, що відповідає стовпцю таблиці. Наприклад, нехай на вхід шостого S-блоку (тобто біти функції XOR з 31 по 36) подаються як 110011. Перший і останній біт об'єднуючись утворять 11, що відповідає рядку 3 шостого S-блоку. Середні 4 біти утворять 1001, що відповідає стовпцю 9 того ж S-блоку. Елемент S-блоку 6, що знаходиться на перетині рядка 3 і стовпця 9, - це 14. (рядки і стовпці нумеруються з 0, а не з 1.) Замість 110011 на вихід подається 1110.

Звичайно ж набагато легше реалізувати S-блоки програмно у вигляді масивів з 64 елементами.

Підстановка за допомогою S-блоків є ключовим етапом DES. Інші дії алгоритму лінійні і легко піддаються аналізу. S-блоки нелінійні і саме вони в більшому ступені ніж всі інше, забезпечують безпеку DES.

32-бітовий вихід підстановки за допомогою S-блоків (вісім 4-бітових блоків), перетасовуються відповідно до P-блоку. Ця перестановка переміщує кожен вхідний біт в іншу позицію. Жоден біт не використовується двічі, і жоден біт не ігнорується. Цей процес називається прямою чи простою перестановкою (таблиця 5.7). Наприклад, біт 21 переміщається в позицію 4, а біт 4 - у позицію 31.

Таблиця 5.7 - Перестановка за допомогою P-блоків

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

Нарешті результат перестановки за допомогою P-блоків об’єднується за допомогою XOR з лівою половиною початкового 64-бітового блоку. Потім ліва і права половини міняються місцями.

Кінцева перестановка є зворотною стосовно початкової перестановки. Ліва і права половини не міняються місцями після останнього етапу DES, замість цього об'єднаний блок R16L16 використовується як вхід заключної перестановки (таблиця 5.8). Перестановка половинок з наступним циклічним зсувом привела б до точно такого ж результату. Це зроблено для того, щоб алгоритм можна було використовувати як для шифрування, так і для дешифрування.

Таблиця 5.8 - Заключна перестановка

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

Для шифрування і дешифрування використовується той самий алгоритм. DES дозволяє використовувати для шифрування і дешифрування блоку ту саму функцію. Єдина відмінність полягає в тому, що ключі повинні використовуватися в зворотному порядку. Тобто, якщо на етапах шифрування використовувалися ключі $K_1, K_2, K_3, ..., K_{16}$ то ключами дешифрування будуть $K_{16}, K_{15}, K_{14}, ..., K_1$

Алгоритм, що створює ключ для кожного етапу також циклічний. Ключ зсувається праворуч, а число позицій зсуву дорівнює 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

Існує 4 режими роботи алгоритму DES: електронна кодова книга (ECB), зчеплення блоків шифру (CBC), зворотній зв'язок по шифртексту (CFB), зворотній зв'язок по виходу (OFB).

Криптосистема RSA

Алгоритм RSA запропонували в 1978 р. троє авторів: Р.Райвест (Rivest), А.Шамір (Shamir) і А.Адлеман (Adleman). Алгоритм одержав свою назву за першими буквами прізвищ його авторів. Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, що може працювати як у режимі шифрування даних, так і в режимі електронного цифрового підпису.

Надійність алгоритму ґрунтується на складності задач факторизації великих чисел та обчислення дискретних логарифмів.

У криптосистемі RSA відкритий ключ *K,* секретний ключ *k*, повідомлення *M* і криптограма *C* належать множині цілих чисел $Z_N = \{1,2,...N-1\}$, де $N = P \cdot Q$ - модуль (*P і Q*- випадкові великі прості числа). Для забезпечення максимальної безпеки вибирають *P* і *Q* рівної довжини і зберігають у секреті.

Відкритий ключ *K* вибирають випадковим чином так, щоб виконувалися умови:

$1 < K < \varphi(N)$ $НСД(K, \varphi(N)) = 1$ $\varphi(N) = (P-1)(Q-1)$, де

$\varphi(N)$ - функція Ейлера, *НСД* - найбільший спільний дільник.

Функція Ейлера $\varphi(N)$ вказує кількість додатних цілих чисел в інтервалі від 1до *N*, що взаємно прості з *N*. Друга із зазначених вище умов означає, що відкритий ключ *K* і функція Ейлера $\varphi(N)$повинні бути взаємно простими.

Далі, використовуючи розширений алгоритм Евкліда, обчислюють секретний ключ *k*, такий, що $k \cdot K \equiv 1 (\mathrm{mod}\, \varphi(N))$ або $k = K^{-1} (\mathrm{mod}(P-1)(Q-1))$

Це нескладно здійснити, оскільки відома пара простих чисел (*P*,*Q*) і можна легко знайти $\varphi(N)$

Слід зазначити, що *k* і *N* повинні бути взаємно простими.

Відкритий ключ *K* використовують для шифрування даних, а секретний ключ *k* - для дешифрування.

Процедура шифрування визначає криптограму *C* через пару (відкритий ключ *K*, повідомлення *M*) у відповідності з наступною формулою:

$$C = E_K(M) = M^K \pmod N$$

Як алгоритм швидкого обчислення значення C використовують ряд послідовних піднесень до квадрату цілого *M* і множень на *M* приведенням за модулем *N*.

Зворотня операція, тобто визначення значення *M* за відомим значенням C, *K* і *N*, практично не здійсненна при $N \approx 2^{512}$

Однак обернену задачу, тобто задачу дешифрування криптограми C, можна вирішити, використовуючи пару (секретний ключ *k*, криптограма C) за наступною формулою:

$$M = D_k(C) = C^k \pmod N$$

Таким чином, якщо криптограму

$$C = M^k \pmod N$$

піднести до степеня *k*, то в результаті відновлюється вихідний відкритий текст *M*, тому

$$(M^K)^k = M^{Kk} = M^{n-\varphi(N)+1} \equiv M \pmod N$$

Таким чином, одержувач, що створює криптосистему, захищає два параметри: секретний ключ *k* і пару чисел (*P, Q*), добуток яких дає значення модуля *N*. З іншого боку, одержувач відкриває значення модуля *N* і відкритий ключ *K*.

Супротивнику відомі лише значення *K* і *N*. Якби він зміг розкласти число *N* на множники *P* і *Q* (задача факторизації), то він довідався б "таємний хід" - трійку чисел $\{P, Q, K\}$ обчислив значення функції Ейлера $\varphi(N) = (P-1)(Q-1)$

і визначив значення секретного ключа *k*.

Однак, як уже відзначалося, розкладання дуже великого *N* на множники не здійсненно за реальний час (за умови, що довжини обраних *P* і *Q* складають не менш 100 десяткових знаків).

Процедура шифрування і дешифрування в криптосистемі RSA.

Припустимо, що користувач *A* хоче передати користувачу *B* повідомлення в зашифрованому виді, використовуючи криптосистему RSA.

У такому випадку користувач *A* виступає в ролі відправника повідомлення, а користувач *B* - у ролі одержувача. Як відзначалося вище, криптосистему RSA повинен сформувати одержувач повідомлення, тобто користувач *B*.

Розглянемо послідовність дій користувача *B* і користувача А.

1. Користувач *B* вибирає два довільних великих простих числа *P* і *Q*.
2. Користувач *B* обчислює значення модуля $N = P \cdot Q$
3. Користувач *B* обчислює функцію Ейлера $\varphi(N) = (P-1)(Q-1)$ і вибирає випадковим чином значення відкритого ключа *K* з урахуванням виконання умов: $1 < K < \varphi(N)$ $НСД(K, \varphi(N)) = 1$ $\varphi(N) = (P-1)(Q-1)$
4. Користувач *B* обчислює значення секретного ключа k, використовуючи розширений алгоритм Евкліда $k = K^{-1} (\mathrm{mod}(P-1)(Q-1))$
5. Користувач *B* пересилає користувачу *A* пари чисел (*N,K*) по незахищеному каналу.

Якщо користувач *A* хоче передати користувачу *B* повідомлення *M,* він виконує наступні кроки.

6. Користувач *A* розбиває вихідний відкритий текст *M* на блоки, кожний з яких може бути представлений у вигляді числа $M_i = 1,2,...,N-1$
7. Користувач *A* шифрує текст, представлений у вигляді послідовності чисел *M,* за формулою $C_i = M_i^K (\mathrm{mod}\, N)$ і відправляє криптограму $C_1, C_2, ..., C_i, ...$ користувачеві *B*.
8. Користувач *B* розшифровує прийняту криптограму, використовуючи секретний ключ *k,* за формулою $M_i = C_i^k (\mathrm{mod}\, N)$

У результаті буде отримана послідовність чисел *M*, що формують собою вихідне повідомлення *M.* Щоб алгоритм RSA мав практичну цінність, необхідно мати можливість без істотних витрат генерувати великі прості числа, вміти оперативно обчислювати значення ключів *K* і *k.*

Приклад 10.1. Використовуючи алгоритм RSA , виконайте шифрування і розшифрування повідомлення «*Буду завтра*»

Розв'язання. Використовуючи крок 1 алгоритму, вибираємо 2 (для прикладу невеликі) прості числа *P*=59,*Q*=61. Користуючись кроком 2 алгоритму, обчислюємо значення модуля $N = P \cdot Q$=59 x 61=3599. Згідно з кроком 3 алгоритму обчислюємо функцію Ейлера $\varphi(3599) = 3480$ і, з урахуванням виконання умов $1 < K < \varphi(N)$ $НСД(K, \varphi(N)) = 1$

вибираємо відкритий ключ *K*=53, як взаємно просте число. Користуючись кроком 4 алгоритму, розраховуємо таємний ключ алгоритму *k* , який дорівнює 197. Використовуючи крок 5 алгоритму, користувач *B* пересилає незахищеним каналом користувачеві *A* перу чисел (3599, 53).

Користувач *A*, готуючи повідомлення «*Буду завтра*» до передачі, подає його як послідовність цілих чисел у діапазоні від 1 до 32. Нехай *А*-01, *Б*-02, *В*-03,…, *Я*-32. Тоді повідомлення вихідного відкритого тексту М матиме вигляд набору цілих чисел 02 20 05 20 08 01 03 19 17 01. Відповідно до кроку 6 алгоритму, розбиваємо це повідомлення на блоки по чотири цифри 0220 0520 0801 0319 1701 і, виконуючи крок 7 алгоритму, шифруємо кожний блок: $C_1 = 220^{53} \bmod 3599 = 0099$ $C_2 = 520^{53} \bmod 3599 = 3273$ $C_3 = 801^{53} \bmod 3599 = 2050$ $C_4 = 319^{53} \bmod 3599 = 0719$ $C_5 = 1701^{53} \bmod 3599 = 1664$

Внаслідок кодування отримано цифри 0099 3273 2050 0719 1664, який згідно з кроком 7 алгоритму відправляється користувачеві *B*. Користувач *B* для відновлення вихідного тексту *M*, використовуючи крок 8 алгоритму, обчислює модульну експоненту, підвівши зашифроване значення C_i до ступеня *k* за модулем *N*:

$M_1 = 99^{197} \bmod 3599 = 0220$ $M_2 = 3273^{197} \bmod 3599 = 0520$ $M_3 = 2050^{197} \bmod 3599 = 0801$

$M_4 = 719^{197} \bmod 3599 = 0319$ $M_5 = 1664^{197} \bmod 3599 = 1701$

Таким чином, отримали відновлене передане повідомлення 0220 0520 0801 0319 170, яке відповідає набору цілих чисел 02 20 05 20 08 01 03 19 17 01 і переданому повідомленню відкритого тексту «*Буду завтра*».

Алгоритм криптосистеми *RSA* реалізується програмним і апаратним шляхом.

Проте і апаратна, і програмна реалізації алгоритму *RSA* в декілька сотень разів повільніші від реалізації симетричних криптосистем. Мала швидкодія криптосистем з алгоритмами *RSA* дещо обмежує сферу її застосування, але не перекреслює її цінності.

Алгоритм Ель-Гамаля

Алгоритм Ель-Гамаля, який був розроблений у 1985 році, може бути використаний як для шифрування і розшифровування повідомлень, так і для цифрових підписів, і є асиметричним алгоритмом. Безпека алгоритму Ель-Гамаля зумовлена складністю обчислення дискретних алгоритмів у кінцевому полі. Для генерування пари ключів (відкритого ключа *K* і таємного *k*) спочатку вибирають велике просте число *P* і велике ціле число *G*, ступені якого за модулем *P* породжують велику кількість елементів множини Z_P, причому $G < P$. Числа *G* і *P* можуть бути поширені серед групи користувачів.

Потім вибирають випадкове число *k*, причому $k < P$. Число *k* є таємним ключем і повинне зберігатися в секреті. Потім обчислюють відкритий ключ *K*, який дорівнює $K = G^k \pmod P$

Для шифрування повідомлення *M* вибирають випадкове ціле число T, яке повинно задовольняти таким умовам: $1 < T < P-1$ і *НСД* $(T,(P-1)) = 1$. Потім обчислюють пару чисел (a,b) шифротексту $a = G^T \pmod P$ і $b = K^T \cdot M \pmod P$

Пари чисел (a,b) є шифротекстом. Довжина шифротексту вдвічі більша від довжини відкритого тексту *M*. Для розшифровування шифротексту (a,b) обчислюють $M = \frac{b}{a^k} \pmod P$

Кроки алгоритму Ель-Гамаля

1. Вибрати деяке велике просте число *P* і велике ціле число *G*, степені якого за модулем P породжують велику кількість елементів довжини Z_P, причому $G < P$;

2. Вибрати випадкове ціле число *k* таке, що $k < P$. Число *k* є таємним ключем і повинне зберігатися в секреті.

3. Обчислити відкритий ключ за виразом $K = G^k \pmod P$

4. Для шифрування повідомлення M вибрати випадкове ціле число *T*, яке повинно задовольняти таким умовам: $1 < T < P-1$ і *НСД* $(T,(P-1)) = 1$.

5. Обчислити числа (a,b) шифротексту, використовуючи вирази $a = G^T \pmod P$ $b = K^T \cdot M \pmod P$

6. Для розшифровування шифротексту (a,b) обчислюють $M = \frac{b}{a^k} \pmod P$

Приклад 10.2. Виконайте шифрування повідомлення $M = \{D\} = \{5\}$,використовуючи алгоритм Ель-Гамаля.

Розв'язання. Використовуючи кроки 1 та 2 алгоритму, виберемо P = 17 і G = 5, а також таємний ключ k = 2. Згідно з кроком 3 алгоритму, обчислюємо відкритий ключ $K = G^k \pmod P = 5^2 \bmod 17 = 8$. Для шифрування повідомлення $M = \{5\}$ виберемо деяке випадкове число T=3. Перевіримо його на відповідність умов $1 < T < P-1$ і $НСД\ (T,(P-1)) = 1$. Дійсно НСД(3,16)= 1

Використовуючи крок 5 алгоритму, обчислюємо пари чисел a і b отримуємо:

$a = G^T \pmod P = 5^3 \bmod 17 = 125 \equiv 6 \bmod 17 \quad b = K^T \cdot M \pmod P = 8^3 \bmod 17 = 2560 = 10 \bmod 17$

Таким чином, шифротекстом для літери D є пара чисел (6,10). Згідно з кроком 6 алгоритму розшифруємо цей шифротекст. Для цього обчислюємо повідомлення M, використовуючи таємний ключ k, $M = \frac{b}{a^k} \pmod P = \frac{10}{6^2} \bmod 17$. Цей вираз подамо у вигляді $M = \frac{10}{6^2} \bmod 17$ або $M \cdot 6^2 = 10 \bmod 17$. Розв'язавши рівняння, знаходимо M = 5.

У прикладі за модуль P взято мале число для демонстрації наглядності застосування алгоритму, але в реальних системах шифрування необхідно використовувати за модуль P велике ціле просте число, наприклад від 512 до 1024 бітів. В алгоритмі Ель-Гамаля при модулі P в 150 знаків досягається той самий ступінь захисту, що і в алгоритмі RSA з модулем в 200 знаків. Це дозволяє в 5-7 разів збільшити швидкість оброблення інформації, але в такому варіанті відкритого шифрування немає підтвердження достеменності повідомлень.

Алгоритм Рабіна

Алгоритм Рабіна було розроблено в 1979 році. Його застосовують тільки для зашифровування і розшифровування даних. Безпека алгоритму полягає в складності пошуку коренів за модулем складного числа.

Для генерації ключів алгоритму вибирають пару простих чисел, таких, що

p=3mod4 і q=3mod4.

Ці прості числа і є таємним ключем. Відкритим ключем є число $N = p \cdot q$

Для шифрування повідомлення M, де (M<N) обчислюють $C = M^2 \bmod N$

Для розшифровування повідомлення за допомогою китайської теореми про залишки обчислюють:

$$m_1 = C^{\frac{p+1}{4}} \bmod p \quad m_2 = \left(p - C^{\frac{p+1}{4}} \right) \bmod p \quad m_3 = C^{\frac{q+1}{4}} \bmod q$$

$$m_4 = \left(p - C^{\frac{q+1}{4}} \right) \bmod q$$

Потім вибирають два цілих числа: $a = q\left(q^{-1} \bmod p\right) \quad b = p\left(p^{-1} \bmod q\right)$

Чотири можливих рішення: $M_1 = \left(am_1 + bm_3\right) \bmod N \quad M_2 = \left(am_1 + bm_4\right) \bmod N$

$M_3 = \left(am_2 + bm_3\right) \bmod N \quad M_4 = \left(am_2 + bm_4\right) \bmod N$

Один з чотирьох результатів $M_1 \quad M_2 \quad M_3 \quad M_4$ є повідомлення M.

Кроки алгоритму Рабіна

1. Для формування таємних ключів алгоритму вибрати пару простих чисел p і q, таких що p=3mod4 і q=3mod4;

2.Сформувати відкритий ключ алгоритму $N = p \cdot q$

3.Для шифрування повідомлення M обчислити $C = M^2 \bmod N$, де (M<N);

4.Для розшифровування повідомлення за допомогою китайської теореми про залишки обчислити:

$$m_1 = C^{\frac{p+1}{4}} \bmod p \qquad m_2 = \left(p - C^{\frac{p+1}{4}} \right) \bmod p$$

$$m_3 = C^{\frac{q+1}{4}} \bmod q \qquad m_4 = \left(p - C^{\frac{q+1}{4}} \right) \bmod q$$

5.вибирати два цілих числа: $a = q\left(q^{-1} \bmod p\right) \quad b = p\left(p^{-1} \bmod q\right)$

6.вибрати одне з чотирьох можливих рішень: $M_1 = \left(am_1 + bm_3\right) \bmod N$

$M_2 = \left(am_1 + bm_4\right) \bmod N \quad M_3 = \left(am_2 + bm_3\right) \bmod N \quad M_4 = \left(am_2 + bm_4\right) \bmod N$

по знаходженню повідомлення M.

Приклад 10.3. Виконайте шифрування повідомлення $M = \{15\}$ використовуючи алгоритм шифрування Рабіна.

Розв'язання. Використовуючи крок 1 алгоритму, виберемо пари простих чисел p і q:p=11,q=7. Тоді згідно з кроком 2 алгоритму, відкритий ключ буде дорівнювати N=77, а шифротекст згідно з кроком 3 алгоритму, - $C = 15^2 \bmod 77 = 71$. Використовуючи крок 4 алгоритму, обчислюємо $m_1 = 4 \quad m_2 = 7 \quad m_3 = 1 \quad m_4 = 6$ Згідно з кроком 5 алгоритму,

Згідно з кроком 5 алгоритму, обчислюємо значення чисел: $a = 7\left(7^{-1} \bmod 11\right) = 7 \cdot 8 = 56$

$b = 11\left(11^{-1} \bmod 7\right) = 11 \cdot 2 = 22$. Використовуючи крок 6 алгоритму, обчислюємо одне з чотирьох рішень: $M_1 = 15$ $M_2 = 48$ $M_3 = 36$ $M_4 = 69$

Дійсно, одне з чотирьох значень, а саме $M_1 = 15$ і є повідомлення *M*.

Алгоритм Діффі-Хеллмана

Алгоритм Діффі-Хеллмана дає змогу обмінюватися секретним ключем для симетричних криптосистем з використанням каналу, захищеного від модифікації, тобто дані, які передаються цим каналом, можуть бути прослухані, але не змінені. У цьому випадку дві сторони можуть створити однаковий секретний ключ, жодного разу не передавши його до мережі, згідно з таким алгоритмом.

Припустимо, що обом учасникам обміну відомі деякі два натуральних числа *M* і *N* , де *N* – просте число. Ці числа можуть бути відомі й усім іншим зацікавленим особам. Щоб створити більш нікому невідомий ключ, обидві сторони генерують випадкові числа: перший абонент – число *X*, другий – число *Y*.

Потім перший абонент обчислює значення $L_A = M^X \bmod N$ і пересилає його другому учаснику, а другий учасник обчислює значення $L_B = M^Y \bmod N$ і передає його першому.

Зловмисник отримує обидва значення L_A і L_B, але модифікувати їх (втрутитися в процес передачі) не може.

На другому етапі перший абонент, на підставі наявного в нього числа *X* та отриманого по мережі числа L_B, обчислює значення $K_A = L^X \bmod N$, а другий абонент за наявними в нього числами *Y* та L_A обчислює значення $K_B = L^Y \bmod N$

Неважко переконатися, що в обох учасників вийшло те саме число $K_A = K_B$

Доведемо це: $K_A = L_B^X \bmod N = \left(M^Y\right)^X \bmod N = M^{YX} \bmod N = M^{XY} \bmod N = L_A^Y \bmod N = K_B$

Числа K_A і K_B можна використовувати як ключі при обміні інформацією з використанням традиційних систем шифрування.

Кроки алгоритму Діффі-Хеллмана

1. абонентами *A* і *B* взяти деякі прості відомі їм самим натуральні числа *M* і *N*;
2. абонентом *A* згенерувати випадкове число *X* й обчислити значення $L_A = M^X \bmod N$
3. абонентом *B* згенерувати випадкове число *Y* й обчислити значення $L_B = M^Y \bmod N$
4. абонент *A* передає абонентові *B* значення L_A, а абонент *B* передає абонентові *A* значення L_B
5. кожний з абонентів *A* і *B* обчислюють таємний ключ для розшифровування інформації, якою вони будуть обмінюватися між собою, згідно з прийнятими даними L_A та L_B

Приклад 10.4. Для двох простих чисел *M*=17 і *N*=23 і двох випадкових чисел *X*=3 і *Y*=5 розрахуйте таємний ключ, з яким будуть працювати абоненти *A* і *B*.

Розв'язання. Абонент *A*, знаючи випадкове число *X*=3, обчислює значення $L_A = 17^3 \bmod 23 = 14$. Абонент *B*, знаючи випадкове число *Y*=5, обчислює значення $L_B = 17^5 \bmod 23 = 21$. Абонент *A* передає абонентові *B* значення L_A, а абонент *B* передає абонентові *A* значення L_B

Кожний з абонентів *A* та *B* обчислюють таємний ключ згідно з прийнятими значеннями L_A та L_B: $K_A = 21^3 \bmod 23 = 15$ $K_B = 14^5 \bmod 23 = 15$

Якщо супротивник перехопить значення L_A і L_B то він зіштовхнеться із задачею дискретного логарифмування, що є важко розв'язною при визначенні випадкових *X* та *Y*. Необхідно ще раз сказати, що цей алгоритм працює тільки на лінію зв'язку, надійно захищену від модифікацій.

Алгоритм Поліга-Хеллмана

Алгоритм Поліга-Хеллмана подібний до *RSA*. Цей алгоритм не є симетричним, оскільки використовуються різні ключі для шифрування і розшифрування. З іншого боку алгоритм Поліга-Хеллмана не можна віднести й до класу крипто алгоритмів з відкритим ключем, тому що ключі шифрування та розшифрування виходять один з одного, а це говорить про те, що обидва ключі (відкритий і таємний) необхідно тримати в секреті.

У цьому алгоритмі так само, як і в алгоритмі *RSA*, криптограму C і відкритий текст M визначають співвідношеннями $C = M^{K_A} \bmod N$ $M = C^{K_B} \bmod N$, де $K_A \cdot K_B = 1 \bmod$ (деяке складне число). На відміну від алгоритму *RSA*, в цьому алгоритмі число N повинно залишатися частиною таємного ключа. Якщо хто-небудь довідається про значення K_A та N, то він зможе обчислити значення K_B. Але не знаючи значень K_A або K_B зловмисник буде змушений обчислювати значення $K_A = (\log_M C) \bmod N$, що зробити дуже складно.