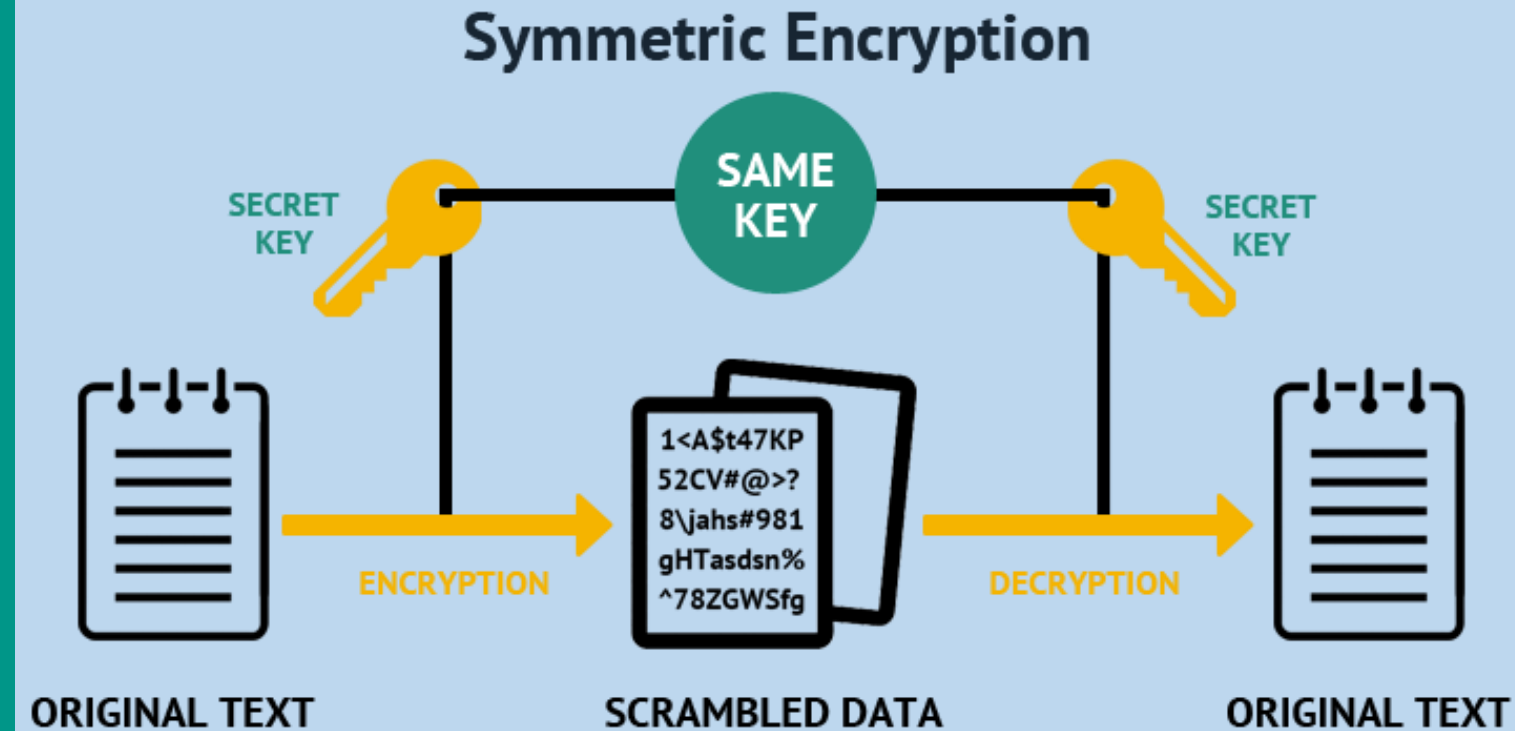


## Симетричні алгоритми шифрування



# План

1. Поняття криптографічної стійкості

2. Шифр Вернама та криптостійкість

3. Мережа Фейстеля

4. Алгоритм DES

5. Модифікації DES

# 1. Поняття криптографічної стійкості

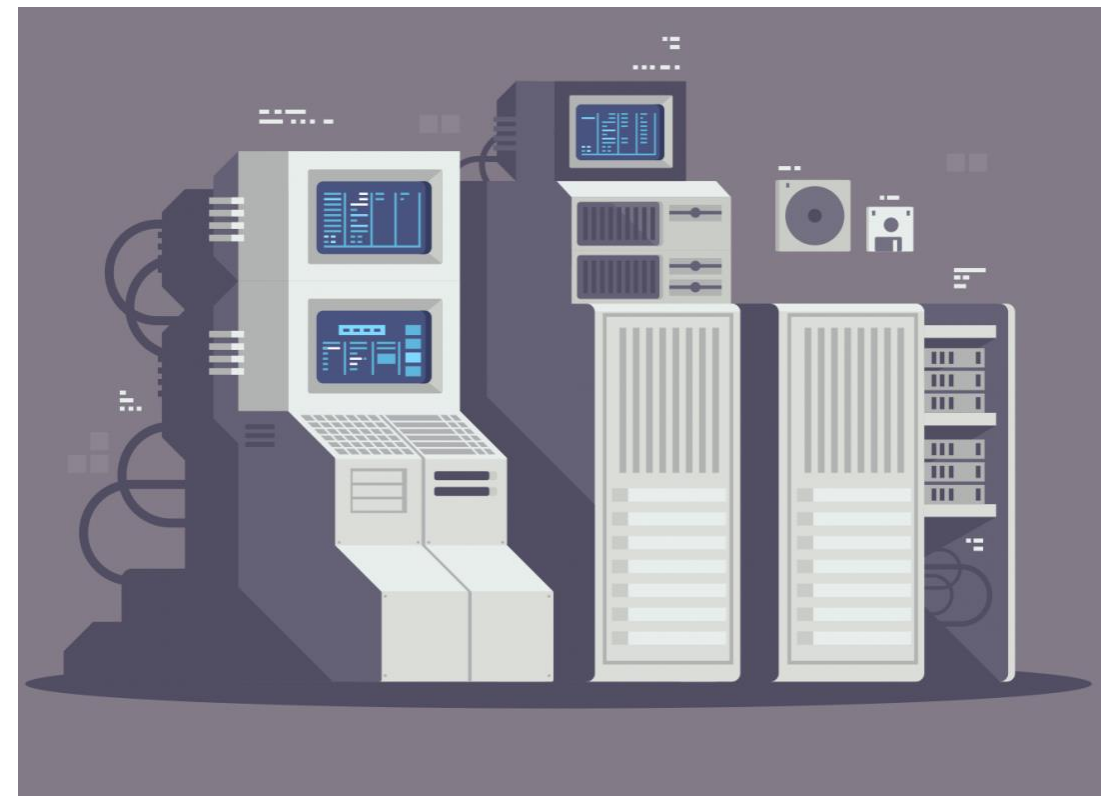
**Криптографічна стійкість** –  
здатність криптографічного  
алгоритму **протидіяти**  
криптографічному аналізу  
(потенційним атакам  
супротивника)



# 1. Поняття криптографічної стійкості

## Види стійкості шифрів (по Шеннону)

**Теоретична (абсолютна) стійкість** – стійкість криптосистеми за наявності у криптоаналітика необмеженого часу, необмежених обчислювальних ресурсів, найкращих методів криптоаналізу



**Оцінка базується на теорії інформації та теорії ймовірностей**

# 1. Поняття криптографічної стійкості

## Види стійкості шифрів (по Шеннону)

**Практична (обчислювальна) стійкість** – стійкість криптосистеми на поточний момент часу з урахуванням того, що криптоаналітик володіє сучасними методами криптоаналізу, проте час та обчислювальні ресурси обмежені



Оцінка базується на теорії складності

# 1. Поняття криптографічної стійкості

## Показники стійкості криптосистеми

1. **Розмір ключа**, який забезпечує необхідний рівень стійкості. Чим довший ключ, тим складніше зламати криптосистему
2. **Час**, необхідний для реалізації атаки. Чим більше часу потрібно для успішного зламу системи, тим вищий рівень її стійкості
3. **Обчислювальні витрати**, необхідні для проведення криптоаналізу. Деякі атаки можуть вимагати значних обчислювальних ресурсів для виконання математичних операцій, таких як факторизація великих чисел або дискретного логарифмування

# 1. Поняття криптографічної стійкості

## Властивості притаманні стійким шифрам

### Розсіювання (Diffusion)

забезпечує поширення впливу окремих символів **відкритого тексту** на увесь шифротекст (реалізується через **перестановку**)

### Перемішування (Confusion)

забезпечує поширення впливу окремих символів **ключа** на увесь шифротекст (реалізується через **підстановку (заміну)**)

## 2. Шифр Вернама та криптостійкість

Творці – **Гільберт Вернам** зі співробітниками телеграфної компанії AT&T, а також офіцер армії США **Джозеф Моборн** (1917 рік)

Шифр Вернама є єдиною системою шифрування, для якої доведена **абсолютна криптографічна стійкість** (Клод Шеннон, 1949 рік)



Гільберт  
Вернам



Джозеф  
Моборн



# 2. Шифр Вернама та криптостійкість

## Ідея автоматичного шифрування телеграфних повідомлень

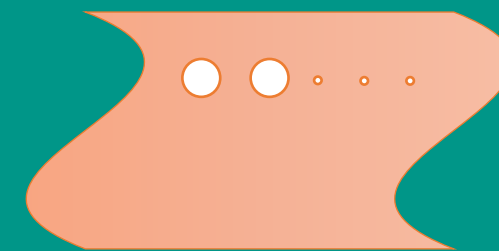
Відкритий текст представлявся у вигляді **п'ятизначних імпульсних комбінацій** на перфострічці

**Ключ:** перфострічка з випадковими знаками — «гама»

Наприклад, літера «А» мала

вигляд:

+ + - - -



«+» — отвір

«-» — його відсутність

# 2. Шифр Вернама та криптостійкість

## Шифрування:

імпульси «гами»  
електромеханічно склалися  
з імпульсами знаків  
відкритого тексту. Отримана  
сума представляла собою  
шифротекст

## Дешифрування:

імпульси, отримані по каналу  
зв'язку, електромеханічно  
склалися з імпульсами тієї  
самої «гами», в результаті  
чого відновлювалися вихідні  
імпульси повідомлення

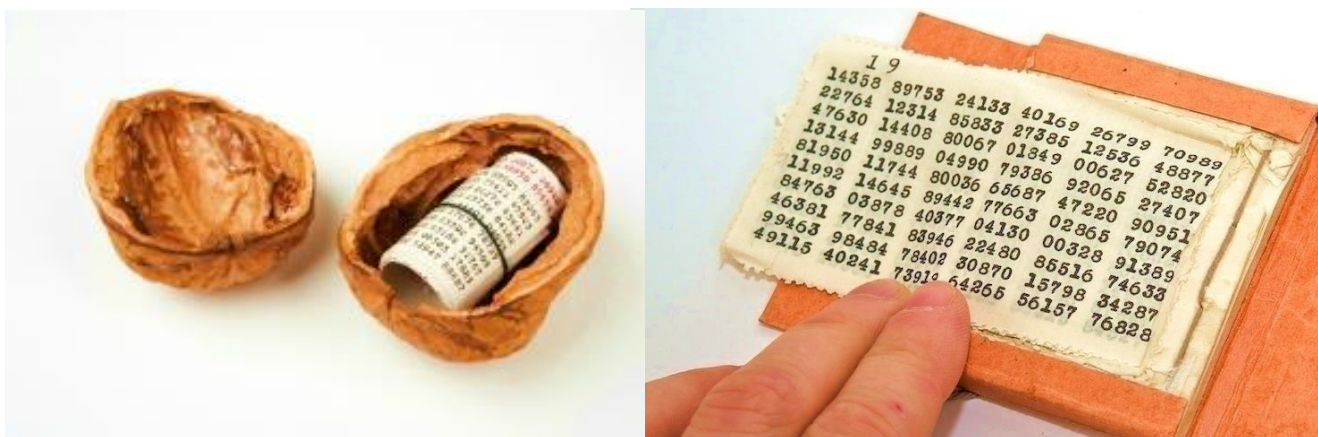
# 2. Шифр Вернама та криптостійкість

## Класичний одноразовий блокнот

Ключ: одноразовий блокнот – послідовність випадкових символів, написаних на аркушах паперу

Ключ повинен володіти трьома критично важливими властивостями:

- 1) бути дійсно випадковим;
- 2) за розміром збігатися з заданим відкритим текстом (ключ ні в якому разі не зациклюється)
- 3) застосовуватися тільки один раз!



## 2. Шифр Вернама та криптостійкість

### Шифрування:

кожен символ ключа  
використовується для  
шифрування одного символу  
повідомлення

### Дешифрування:

одержувач, використовуючи  
точно такий самий блокнот,  
дешифрує кожний символ  
шифротексту

# 2. Шифр Вернама та криптостійкість

## Приклад 1.1:

Ключ: SECRET LISTENERS

Повідомлення: MONITORING TIMES

Шифрування:

Відкритий текст	13 15 14 09 20 15 18 09 14 07 20 09 13 05 19
Ключова гама	19 05 03 18 05 20 12 09 19 20 05 14 05 18 19
Результат додавання	32 20 17 27 25 35 30 18 33 27 25 23 18 23 38
За модулем 26	06 20 17 01 25 09 04 18 07 01 25 23 18 23 12
Шифротекст	FTQAYIDRGAYWRWL

# 2. Шифр Вернама та криптостійкість

## Приклад 1.2:

Ключ: SECRET LISTENERS

Шифротекст: FTQAYIDRGAYWRWL

Дешифрування:

Шифротекст	06 20 17 01 25 09 04 18 07 01 25 23 18 23 12
Ключова гама	19 05 03 18 05 20 12 09 19 20 05 14 05 18 19
Результат віднімання	-13 15 14 -17 20 -11 -08 09 -12 -19 20 09 13 05 -07
За модулем 26	13 15 14 09 20 15 18 09 14 07 20 09 13 05 19
Відкритий текст	MONITORING TIMES

## 2. Шифр Вернама та криптостійкість

Для шифрування бінарних даних (потоків бітів)

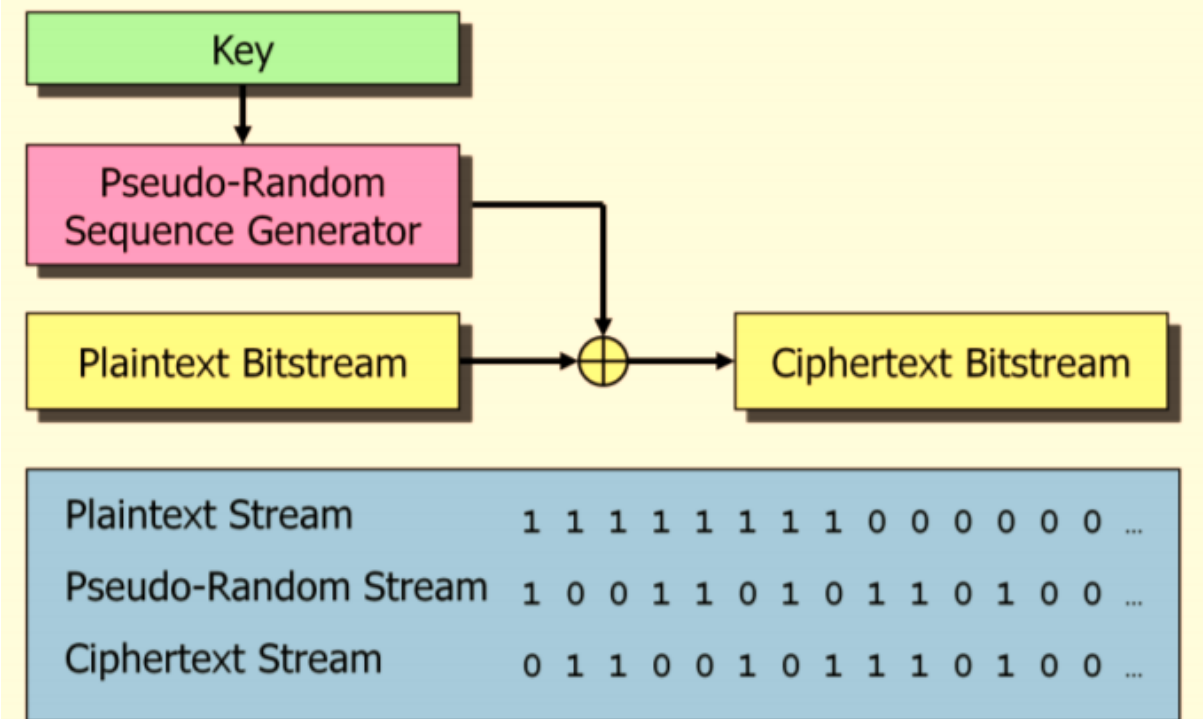
Ключ: послідовність  
випадкових бітів

$\oplus$	0	1
0	0	1
1	1	0

Виконується додавання бітів за модулем 2 (операція **XOR**, exclusive OR – виключне або)

# 2. Шифр Вернама та криптостійкість

**Потоковий шифр** – шифр, що перетворює кожен **СИМВОЛ** (літеру, біт або байт) відкритого тексту у символ шифротексту, залежно від **КЛЮЧА** та **розташування символів** у тексті



**Ключовий потік** (гама) – це бітова послідовність, що визначається за допомогою ключа шифру



# 2. Шифр Вернама та криптостійкість

## Приклад 1.3:

Ключ: 00001011 00010010 00001111

Повідомлення: SUN

Шифрування:

Відкритий текст	01010011 01010101 01001110
Ключова гама	00001011 00010010 00001111
Результат додавання за модулем 2	01011000 01000111 01000001
Шифротекст	XGA

## 2. Шифр Вернама та криптостійкість

Чому ж не використовують абсолютно стійкий шифр?  
Навіщо придумали інші шифри, якщо вони не ідеальні?

### Недоліки:

- ✓ проблема генерації та зберігання ключа;
- ✓ проблема передавання ключа для дешифування.

# 3. Мережа Фейстеля

У 1971 році **Хорст Фейстель** (дослідницький центр IBM) створив шифр **Lucifer**, на базі якого згодом було розроблено шифр DES

Мережа Фейстеля – один з **методів** побудови **блокових** шифрів



Хорст  
Фейстель

# 3. Мережа Фейстеля

Дані, представлені в **комп'ютерній пам'яті**, розбиваються на **блоки фіксованої довжини** (наприклад, 64 біта, 128 біт).

Якщо останній фрагмент **коротше довжини** блоку – його **доповнюють** будь-яким способом (наприклад, незначущими нулями)

**Ключ:** послідовність бітів **фіксованої довжини**, з якої генеруються  $N$  **раундових ключів** за яким-небудь математичним правилом

# 3. Мережа Фейстеля

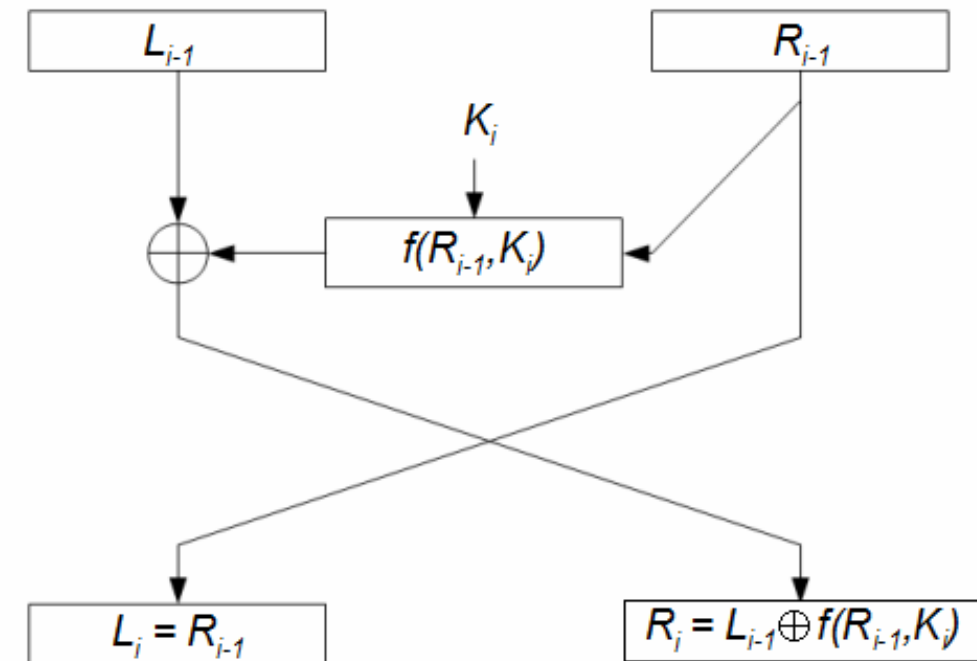
1. Кожен блок даних розбивається на дві рівні частини – **ліву** ( $L$ ) і **праву** ( $R$ )

2. Права частина видозмінюється деякою **функцією**  $f(R,K)$  залежно від **раундового ключа**  $K$

3. Здійснюється **додавання за модулем 2** лівої частини  $L$  та  $f(R,K)$

4. **Результат додавання** присвоюється **новому правому** підблоку, а **правий** підблок присвоюється **без змін** новому **лівому** підблоку (вхідні дані для наступного раунду)

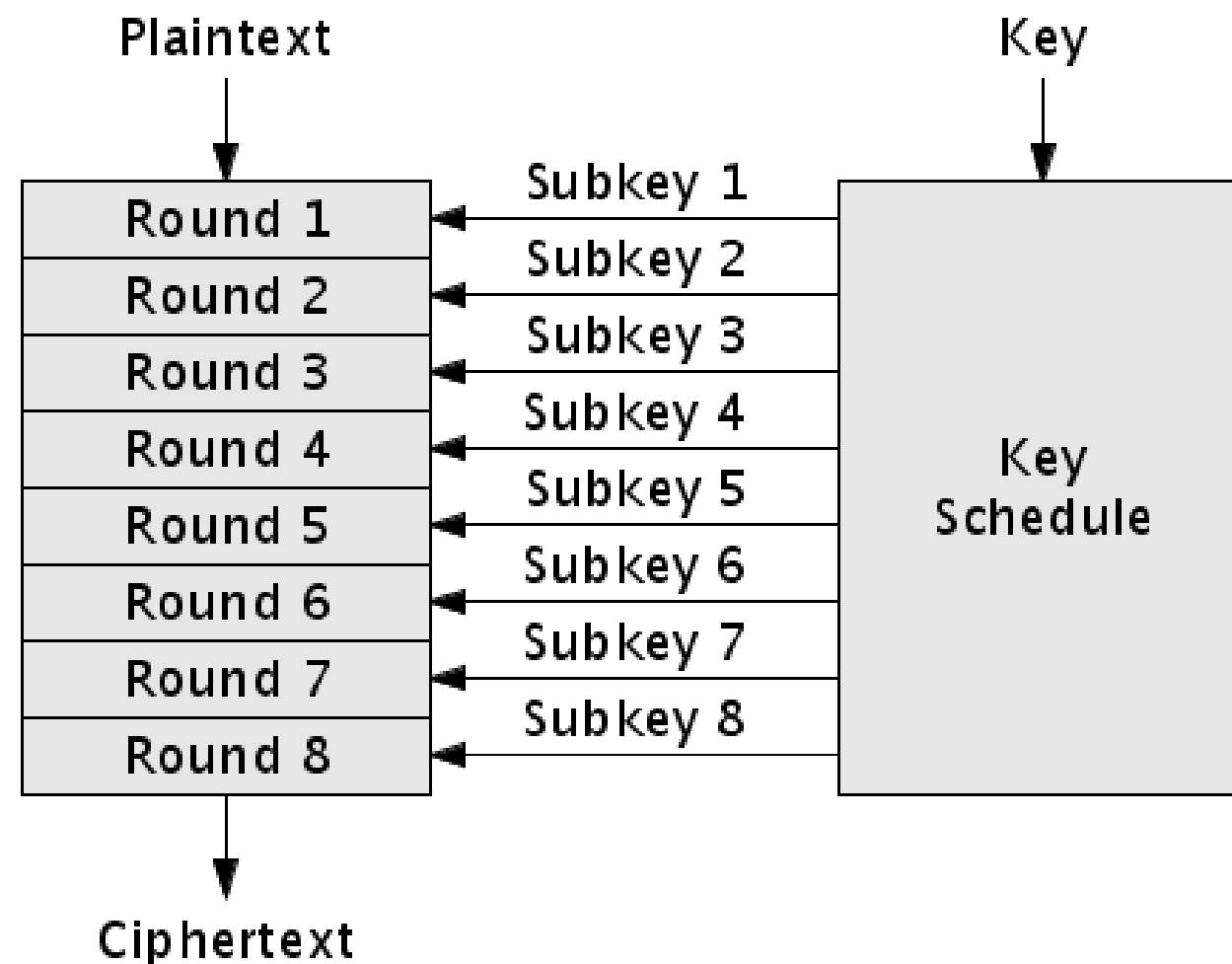
**Шифрування:**



# 3. Мережа Фейстеля

Описані операції повторюються  $N-1$  разів, де  $N$  – кількість раундів.

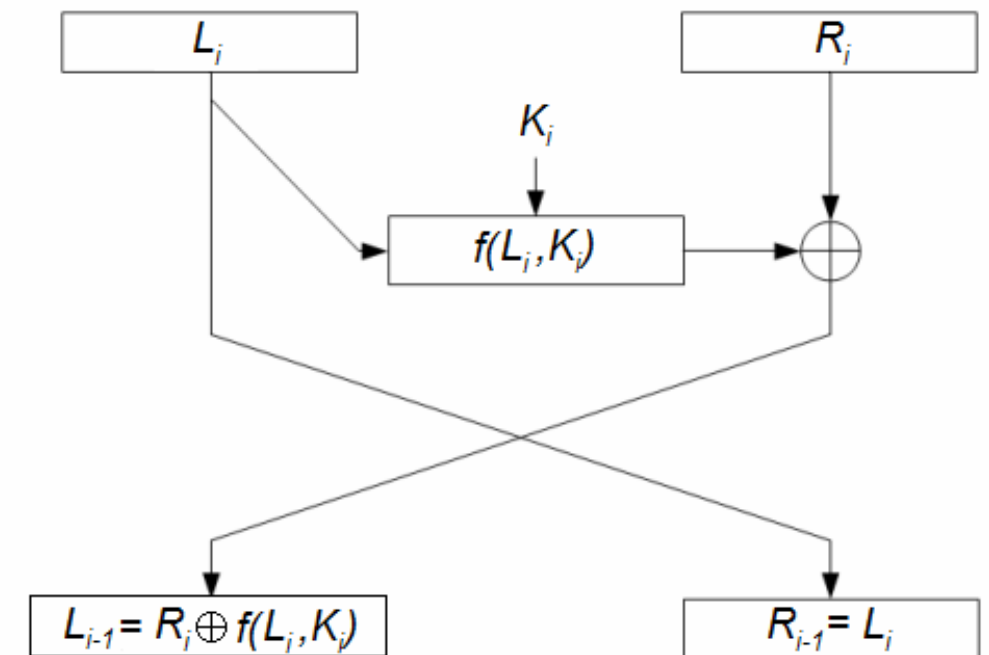
При переході від одного раунду до іншого **мінються раундові ключі** ( $K_0$  на  $K_1$  і т.д.)



# 3. Мережа Фейстеля

Дешифрування відбувається так само, як і шифрування, за винятком лише того, що ключі йдуть у **зворотному порядку**, тобто не від першого до  $N$ -го, а від  $N$ -го до першого

Дешифрування:



# 4. Алгоритм DES

**DES (Data Encryption Standard)** розроблений фірмою IBM і затверджений урядом США у 1976 році як офіційний **стандарт шифрування**

**Ключ: 56 бітний** блок + **8 біт** контролю парності, що розміщені в позиціях 8, 16, 24, 32, 40, 48, 56, 64 (використовується при знаходженні помилок при обміні та зберіганні ключів)



# 4. Алгоритм DES

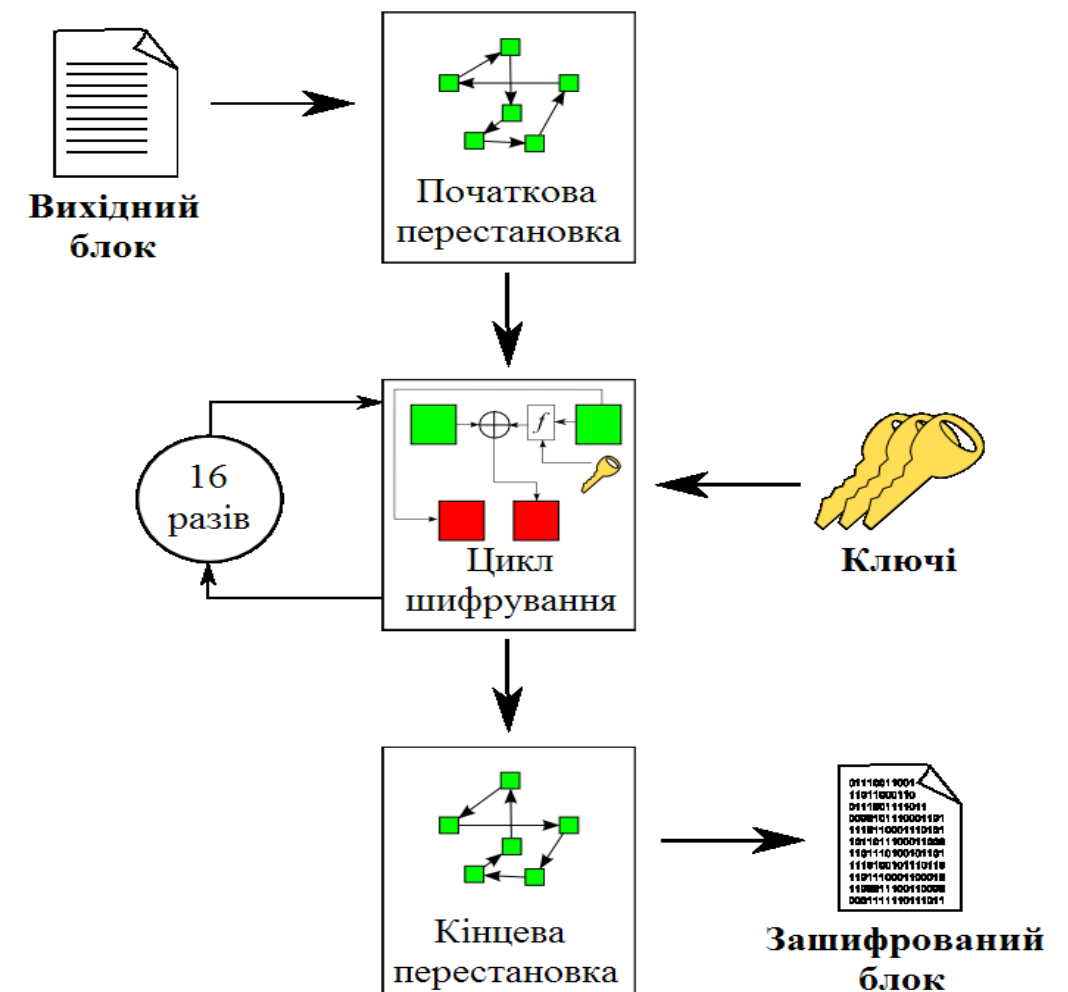
## Загальна схема алгоритму DES

Вихідний текст – блок 64 біт

Процес **шифрування** складається з

- початкової перестановки;
- 16-ти раундів шифрування;
- кінцевої перестановки

Зашифрований текст – блок 64 біт



# 4. Алгоритм DES

## Початкова перестановка

Початковий текст  $T$  (блок 64 біт)  
перетворюється за допомогою  
**початкової перестановки**  $IP$  (Initial  
Permutation) за таблицею

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

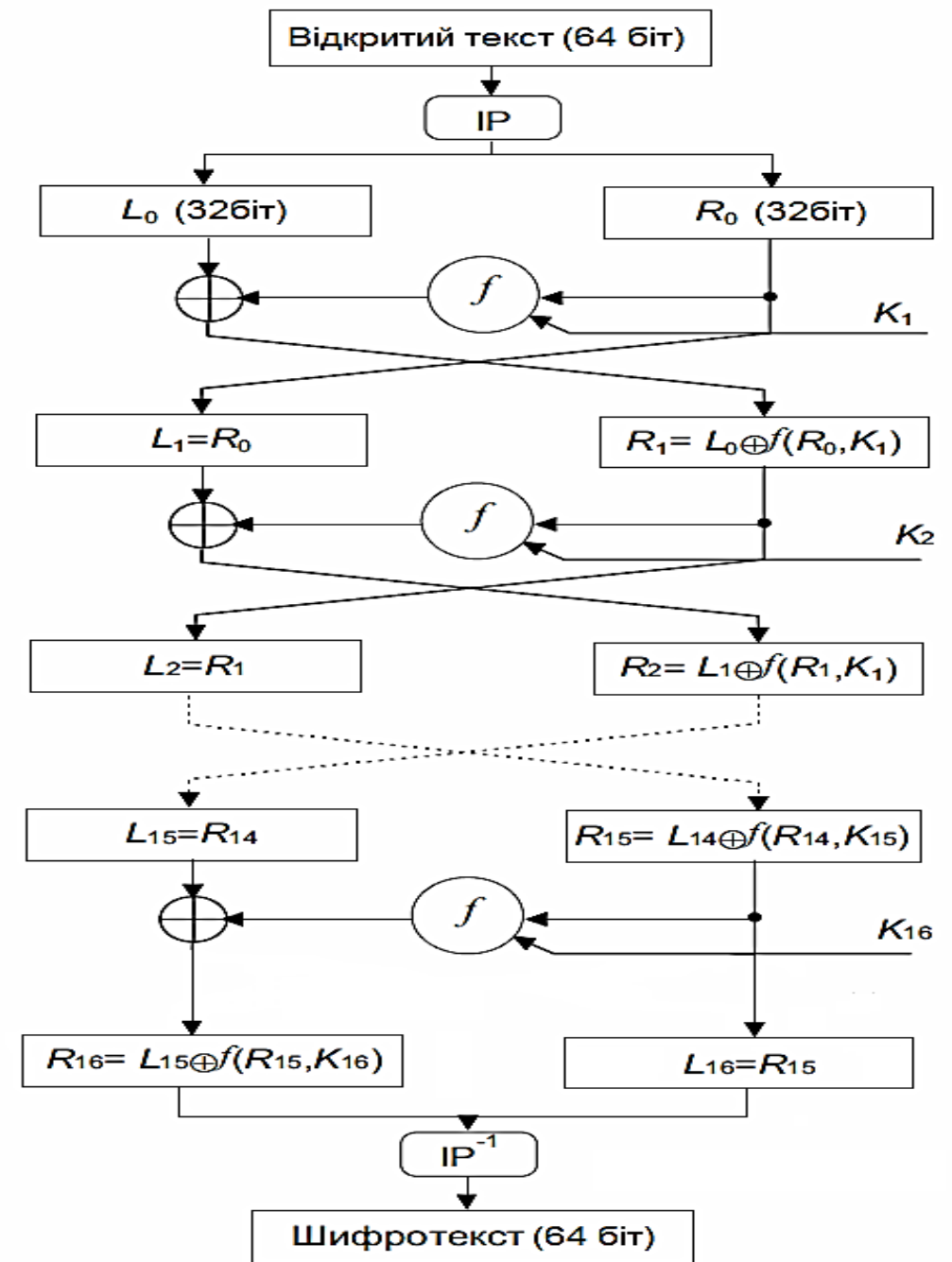
# 4. Алгоритм DES

## Раунди шифрування за мережею Фейстеля

1. Розбити  $IP(T)$  на **дві половини**  $L_0$   $R_0$  де  $L_0$  – 32 старших біта, а  $R_0$  – 32 молодших біта даного блоку

2. Права половина  $R_i$  – це бітове **додавання**  $L_{i-1}$  та  $f(R_{i-1}, K_i)$  за модулем 2:  
 $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

3. Ліва половина  $L_i$  **дорівнює** правій половині попереднього блоку  $R_{i-1}$  без змін:  $L_i = R_{i-1}$



# 4. Алгоритм DES

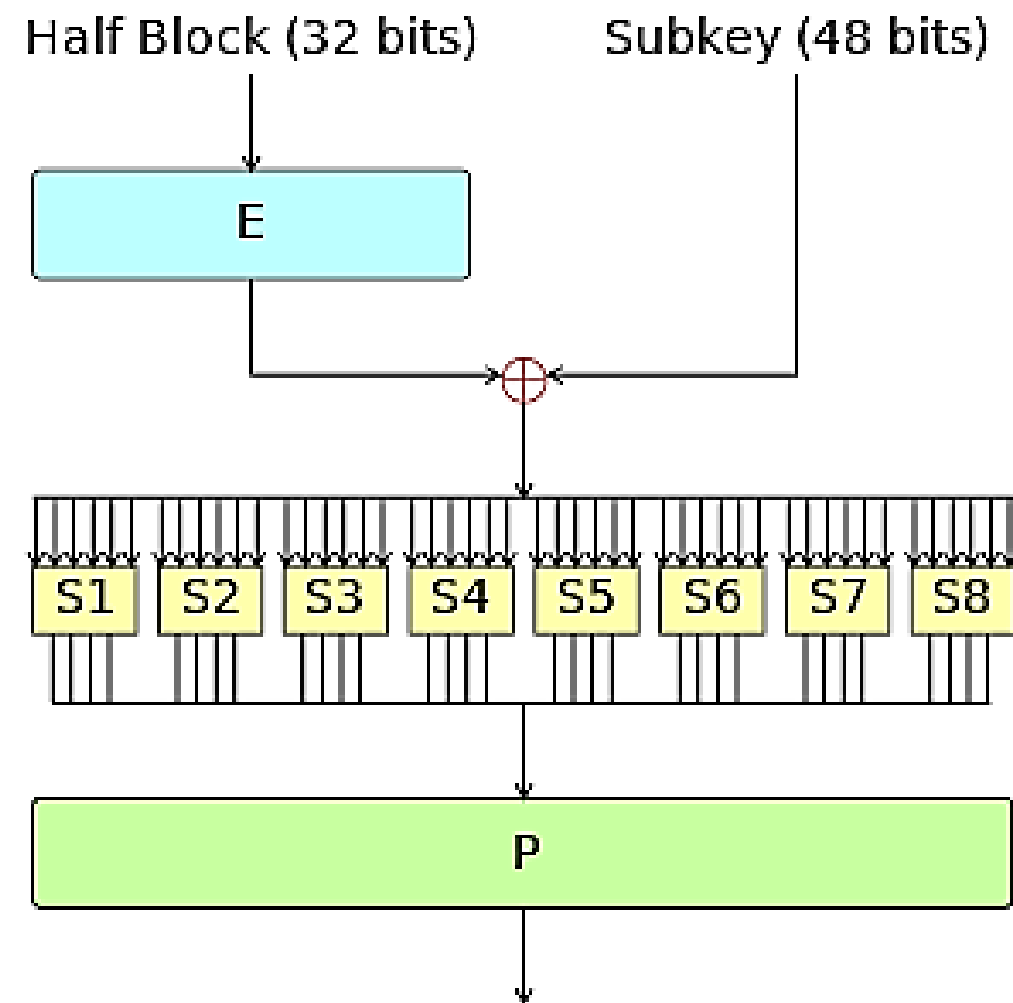
## Основна функція шифрування (функція Фейстеля)

Аргументи функції  $f$ :

32-бітовий вектор  $R_{i-1}$  та  
48-бітовий раундовий ключ  $K_i$

Для обчислення функції  $f$   
використовуються:

- функція розширення  $E$ ;
- перетворення  $S$ , яке складається з 8 перетворень  $S$ -блоків;
- перестановка  $P$



# 4. Алгоритм DES

Функція  $E$  розширює 32-бітовий вектор  $R_{i-1}$  до 48-бітового вектора  $E(R_{i-1})$  шляхом дублювання деяких бітів  $R_{i-1}$

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

Отриманий після розширення блок  $E(R_{i-1})$  додається за модулем 2 із раундовими ключами  $K_i$ . Потім представляється у вигляді восьми послідовних блоків  $E(R_{i-1}) = V_1 V_2 \dots V_8$

# 4. Алгоритм DES

Кожний  $V_j$  являється **6-бітовим** блоком, що перетворюється у **4-бітовий** блок  $V'_j$  за допомогою S-перетворень, що визначаються таблицями

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

# 4. Алгоритм DES

## Приклад 3.1:

$B_3 = 101111$ . Знайти  $B'_3$  - ?

Перший і останній розряди  $B_3$  – двійковий запис числа  $a$ ,  $0 \leq a \leq 3$ .

$$a = 11_2 = 3_{10}$$

Середні чотири розряди  $B_3$  – двійковий запис числа  $b$ ,  $0 \leq b \leq 15$ .

$$b = 0111_2 = 7_{10}$$

Пара чисел  $(a, b)$  визначає число, що знаходиться на **перетині** рядка  $a$  та стовпця  $b$  в таблиці S-перетворень.

У таблиці S3, число обумовлене парою  $(3, 7) = 7$ . Звідси  $B'_3 = 0111$ .

# 4. Алгоритм DES

Отриманий 32-бітовий блок  $V'_1V'_2\dots V'_8$  перетворюється за допомогою **перестановки  $P$** , що задана таблицею

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



# 4. Алгоритм DES

## Генерація ключів

З ключа розміром **56 біт** генерується 16 штук **48-бітних** раундових ключів

**Видаляються 8 контрольних** бітів та виконується **перестановка** початкового 56-бітного ключа

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

# 4. Алгоритм DES

56-бітовий ключ ділиться на **дві половини** по 28 біт  $C_0$  та  $D_0$ , які потім **циклічно зсуваються** на один чи два біти ліворуч в залежності від раунду

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


# 4. Алгоритм DES

## Приклад 2.2:

Побудуємо  $C_1, D_1$  з  $C_0, D_0$ .

Для цього виконаємо

**циклічний зсув ліворуч** на  
один біт кожної половини

57	49	41	33	25	17	9	$C_0$
1	58	50	42	34	26	18	
10	2	59	51	43	35	27	
19	11	3	60	52	44	36	
<hr/>							
63	55	47	39	31	23	15	$D_0$
7	62	54	46	38	30	22	
14	6	61	53	45	37	29	
21	13	5	28	20	12	4	
<hr/>							
							
49	41	33	25	17	9	1	$C_1$
58	50	42	34	26	18	10	
2	59	51	43	35	27	19	
11	3	60	52	44	36	57	
<hr/>							
55	47	39	31	23	15	7	$D_1$
62	54	46	38	30	22	14	
6	61	53	45	37	29	21	
13	5	28	20	12	4	63	

# 4. Алгоритм DES

Після зсуву  $C_i$ ,  $D_i$  вибирається 48 бітів з 56 бітів та міняється їх **порядок** за 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
44	49	39	56	34	53
46	42	50	36	29	32

# 4. Алгоритм DES

## Кінцева перестановка

Кінцева перестановка  $IP^{-1}$  є оберненою до перестановки  $IP$  та використовується для **відновлення позицій**. Кінцева перестановка визначається таблицею

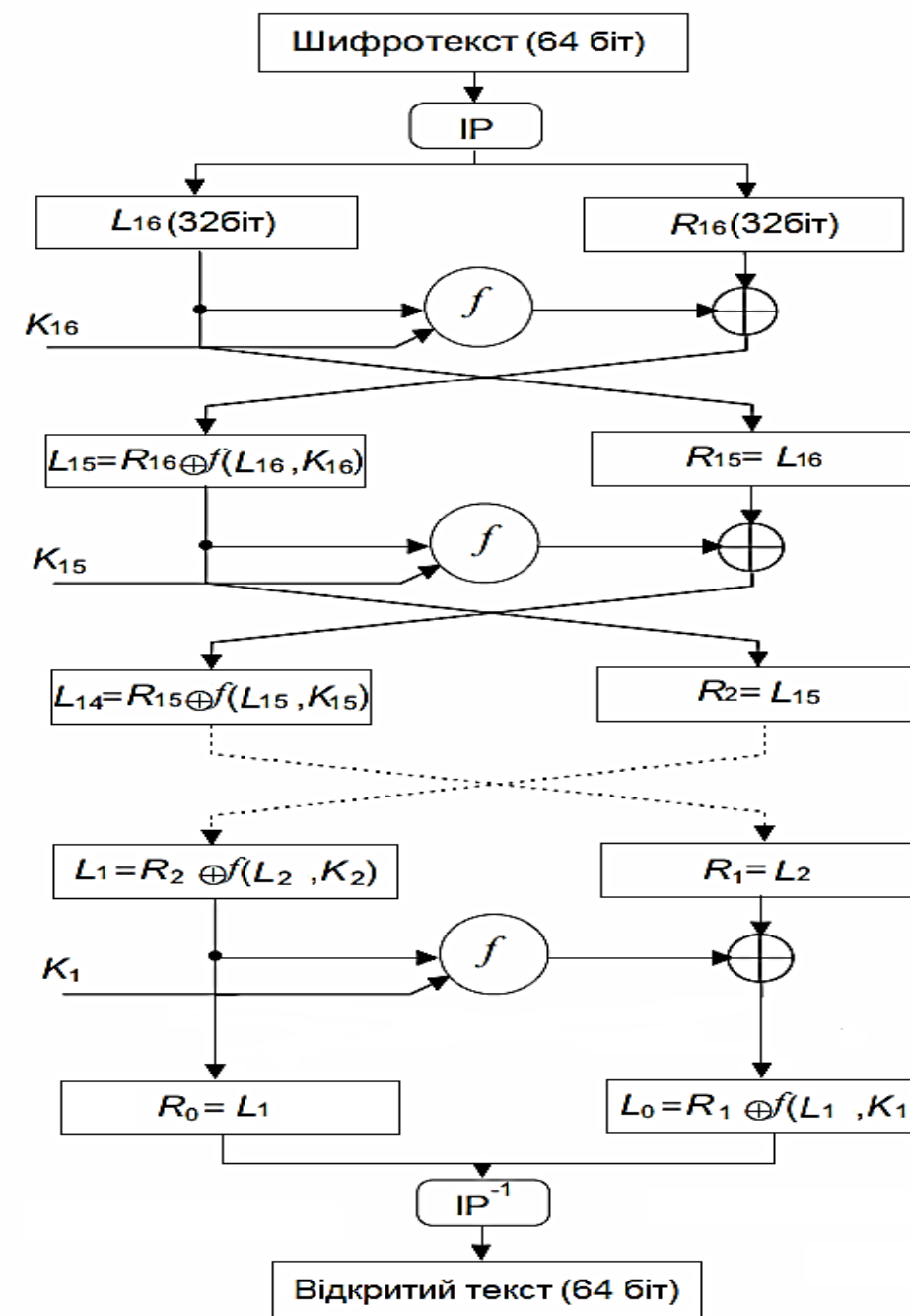
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

# 4. Алгоритм DES

## Дешифрування

При дешифруванні даних всі дії відбуваються в **зворотному порядку**. Ключі також застосовуються в зворотному порядку

Функція  $f$ , перестановки  $IP$  і  $IP^{-1}$  такі самі як і в процесі зашифрування



# 4. Алгоритм DES

## DES Visualization

# 4. Алгоритм DES

## Лавинний ефект

**Лавинний ефект** означає, що невеликі зміни в початкових даних (чи в ключі) можуть викликати значні зміни в зашифрованих даних (в DES проявляється вже на 4-5 раунді).

### Приклад 3.1:

	<i>Перший випадок</i>	<i>Другий випадок</i>
<b>Відкритий текст</b>	0000000000000000 <sub>16</sub>	000000000000000001 <sub>16</sub>
<b>Ключ</b>	23234513987aba23 <sub>16</sub>	23234513987aba23 <sub>16</sub>
<b>Шифротекст</b>	4789fd476e82a5f1 <sub>16</sub>	0a4ed5c15a63fea3 <sub>16</sub>

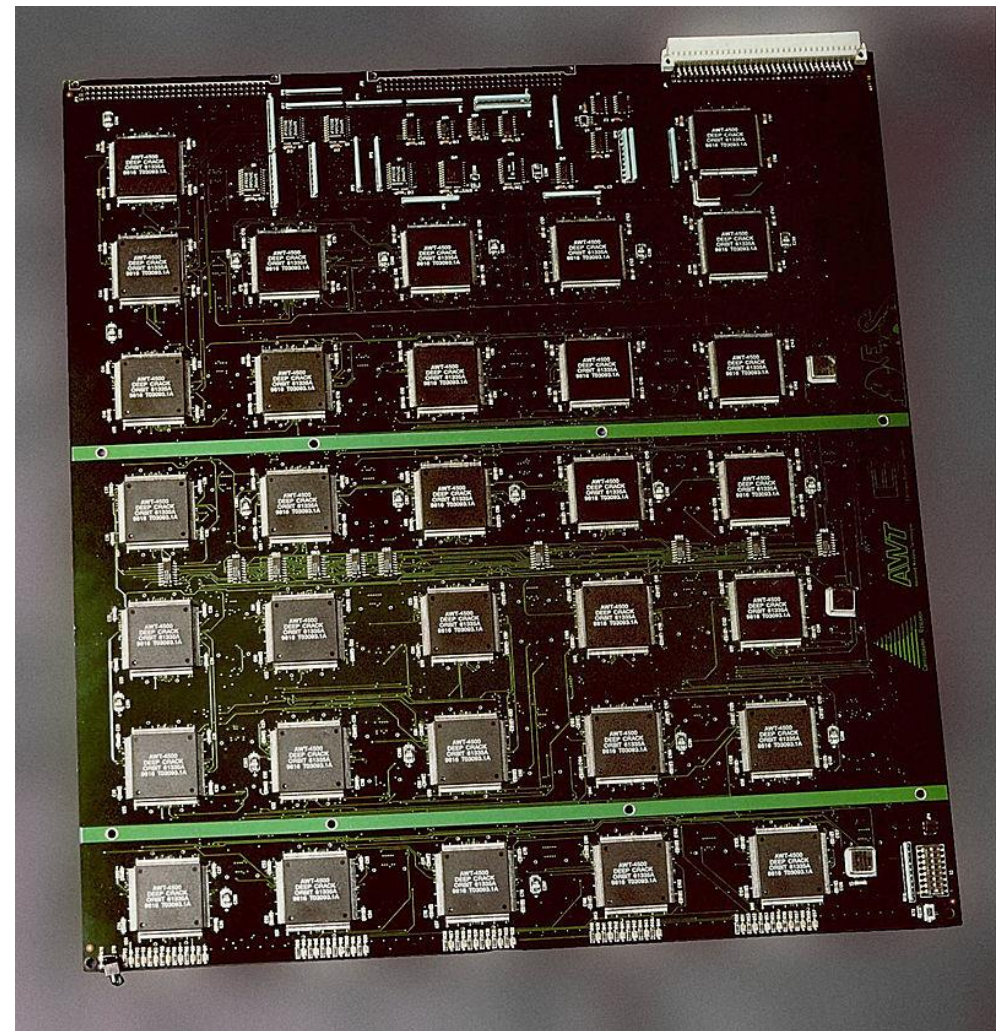


# 4. Алгоритм DES

## Довжина ключа

Для здійснення атаки «грубою силою» потрібно перевірити  $2^{56}$  ключів.

Спеціальний комп'ютер **Deep Crack** був створений 1998 р., завдяки йому ключ було знайдено за **56 год.**



# 4. Алгоритм DES

## Диференційний криптоаналіз DES

Запропонований у 1990 р. ізраїльськими математиками Елі Біхамом та Аді Шаміром

**Ідея:** із множини відкритих текстів (шифротекстів) обирати пари із заданою **різницею** та на підставі аналізу кожної пари знаходити можливі варіанти ключа

# 4. Алгоритм DES

## Лінійний криптоаналіз DES

Метод винайдений у 1993 р. Японцем Міцуру Мацуї.

**Ідея:** будуються **співвідношення** між відкритим текстом, шифротекстом та ключем (лінійна апроксимація) та застосовуються із відомими парами відкритий текст – шифротекст

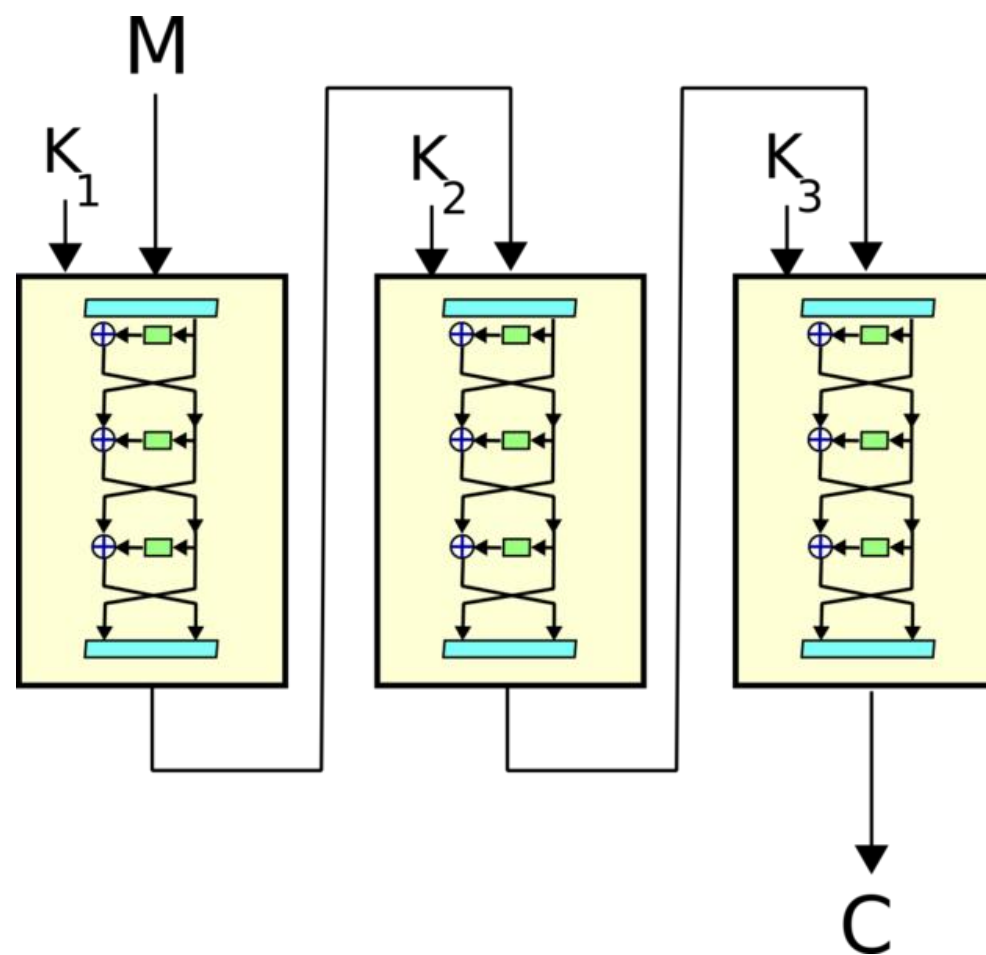
# 5. Модифікації DES

Модифікації DES	Опис
<b>Подвійний DES (2DES) Потрійний DES (3DES)</b>	збільшено довжину ключів
<b>DES з незалежними підключами</b>	використання різних підключів на кожному раунді, не створюючи їх з одного 56-бітового ключа
<b>DES зі зміненими S-блоками</b>	використовується змінний порядок S-блоків або міняється вміст самих S-блоків
<b>DESX</b>	використовується метод відбілювання ключа – операція накладання фрагмента ключа на вхід і (або) вихід алгоритму

# 5. Модифікації DES

## 3DES

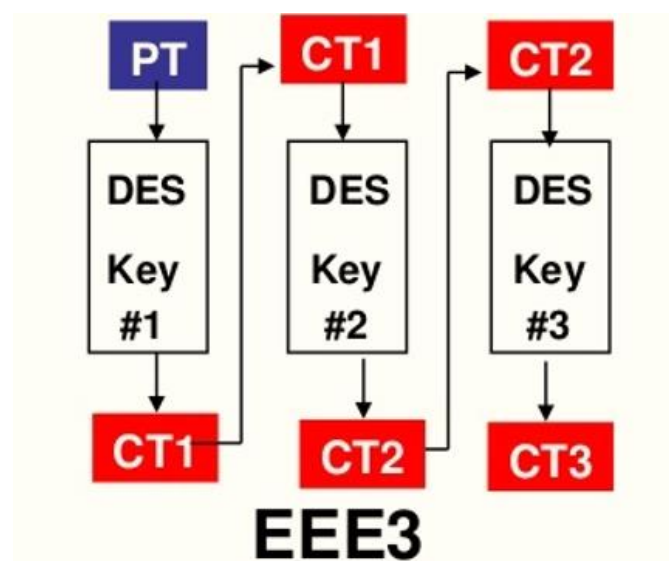
З метою ефективного збільшення довжини ключа, в 1978 році був запропонований алгоритм **3DES** або потрійний DES (TripleDES).



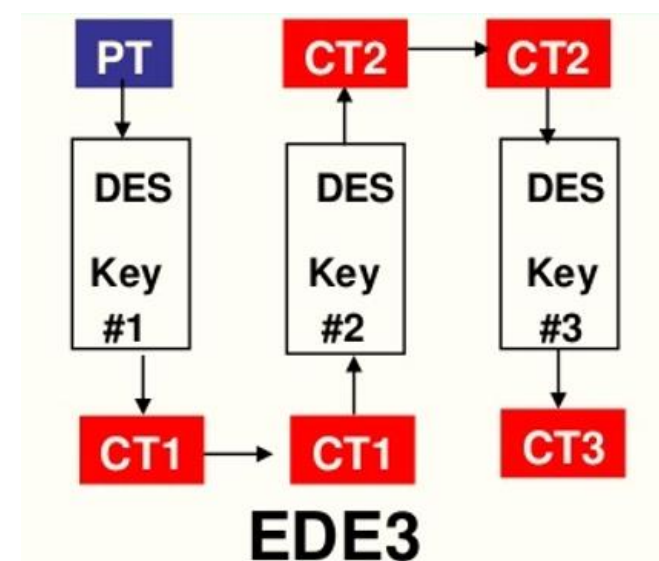
# 5. Модифікації DES

## Варіанти 3DES

**DES-EEE3** – шифрування виконується три рази з трьома різними ключами



**DES-EDE3** – виконуються операції шифрування, дешифрування та знову шифрування з трьома різними ключами



# 5. Модифікації DES

## Варіанти 3DES

**DES-EEE2** – як DES-EEE3, але на першому і третьому кроці використовується однаковий ключ

**DES-EDE2** – як DES-EDE3, але на першому і третьому кроці використовується однаковий ключ

