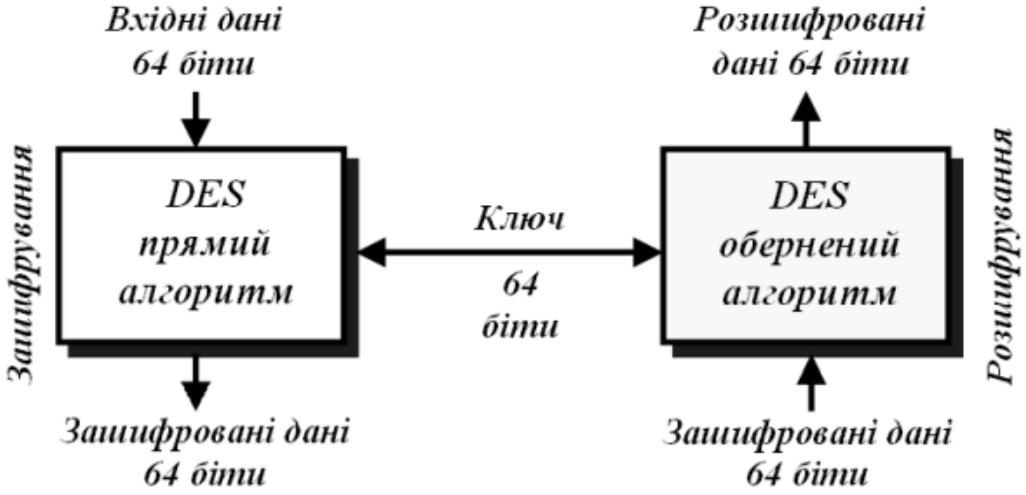
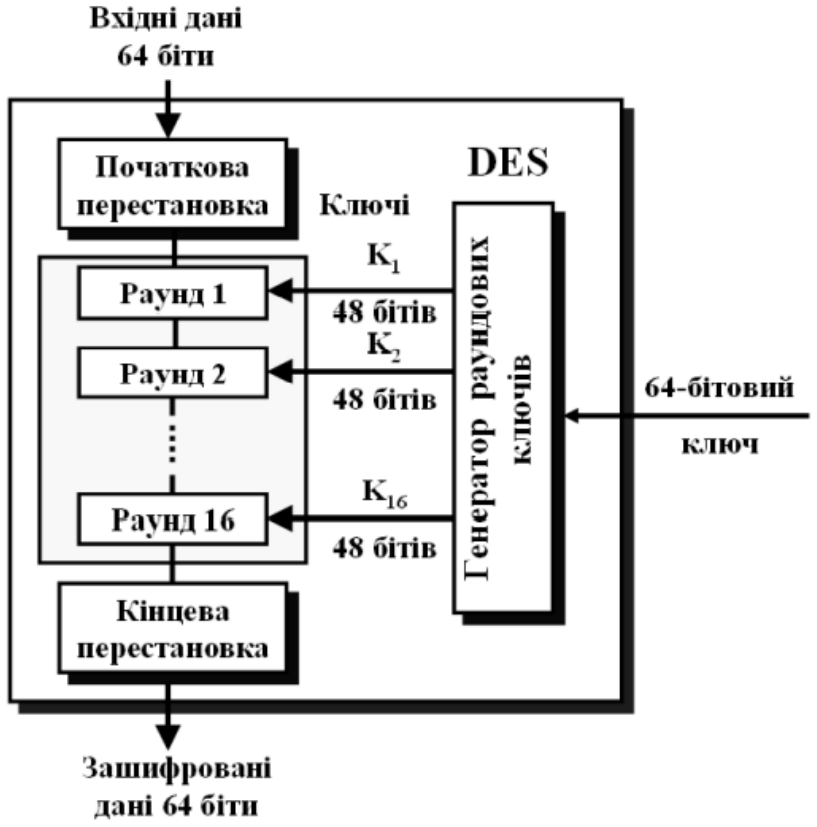
Математична криптологія

Стандарт шифрування DES
Багаторазове шифрування

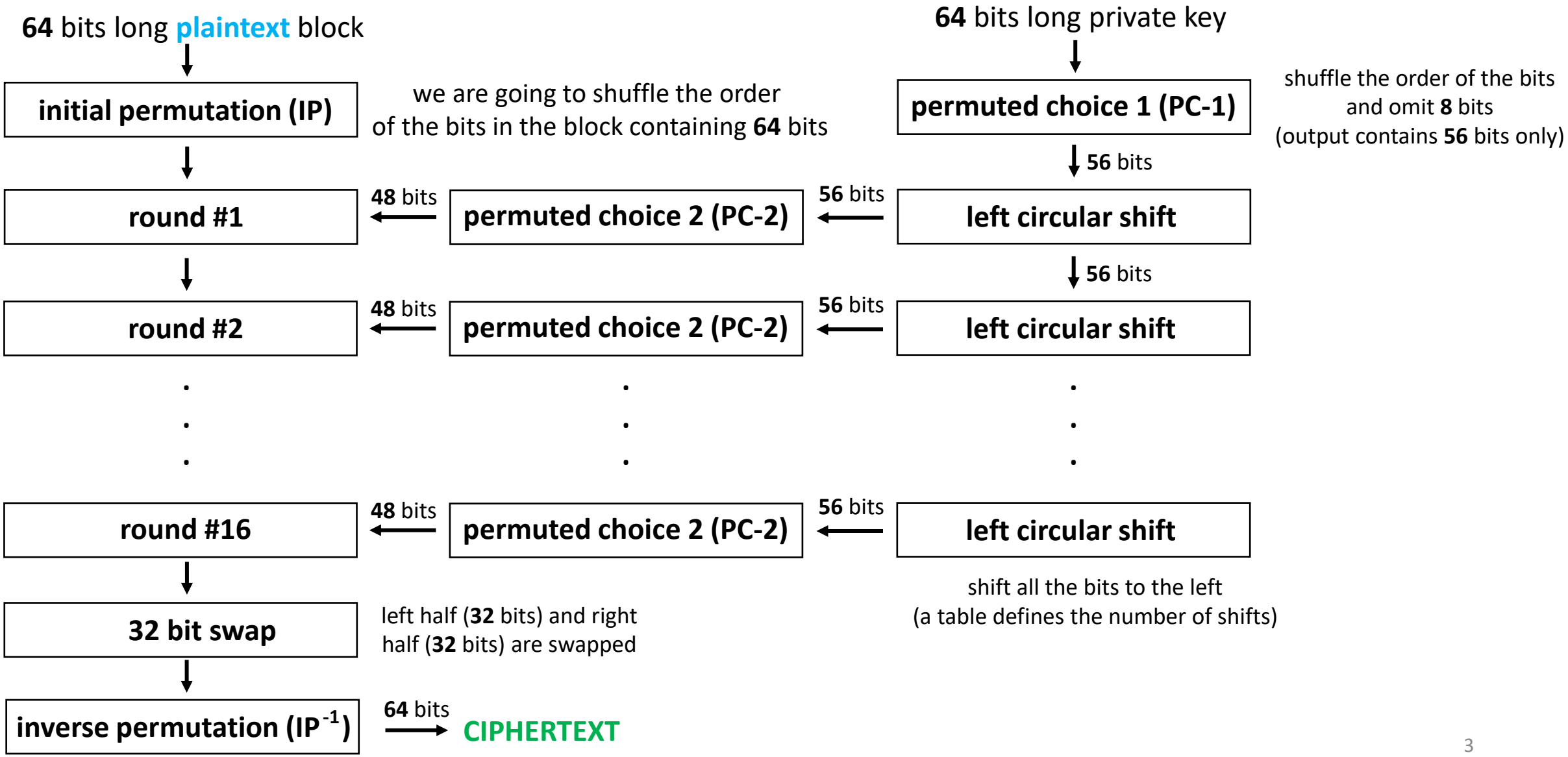
Стандарт шифрування даних DES

Запропонована IBM модифікація проекту, названа “Люцифер” (Lucifer, Horst Feistel, 1970), була прийнята як DES. DES опубліковано в ескізному вигляді у Федеральному Регістрі в березні 1976 р. як Федеральний Стандарт Обробки Інформації (Federal Information Processing Standard — FIPS). Цей алгоритм був світовим стандартом упродовж більш ніж двадцяти років і затвердився як перший офіційний алгоритм, доступний усім бажаючим. Тому його варто відмітити як найважливішу віху на шляху криптографії від суто військового використання до широко масштабного застосуванн



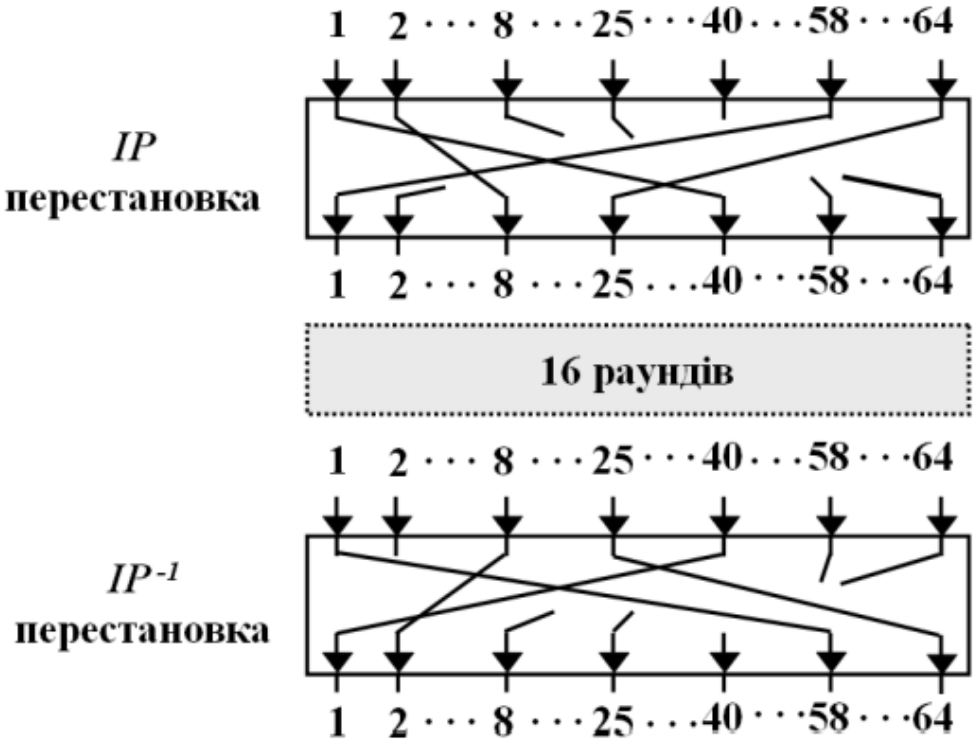


Стандарт шифрування даних DES



DES перестановки

Кожна з перестановок приймає на вхід 64-бітові блоки даних і переставляє його елементи за заданим правилом. Ця ніяк не впливає на стійкість шифру, і користувачі часто ставили питання, навіщо її взагалі робити. Один із членів творчого колективу розробників DES стверджував, що вона полегшує апаратурну реалізацію процедури шифрування.



Таблиця початкової перестановки *IP*

	Номери бітів															
Вхід	*58*	*50*	*42*	*34*	*26*	*18*	*10*	*02*	*60*	*52*	*44*	*36*	*28*	*20*	*12*	*04*
Вихід	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Вхід	*62*	*54*	*46*	*38*	*30*	*22*	*14*	*06*	*64*	*56*	*48*	*40*	*32*	*24*	*16*	*08*
Вихід	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
Вхід	*57*	*49*	*41*	*33*	*25*	*17*	*09*	*01*	*59*	*51*	*43*	*35*	*27*	*19*	*11*	*03*
Вихід	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*
Вхід	*61*	*53*	*45*	*37*	*29*	*21*	*13*	*05*	*63*	*55*	*47*	*39*	*31*	*23*	*15*	*07*
Вихід	*49*	*50*	*51*	*52*	*53*	*54*	*55*	*56*	*57*	*58*	*59*	*60*	*61*	*62*	*63*	*64*

DES перестановки

What is the **permuted choice 1 (PC-1)**?

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

The **64** bits long private key is splitted into two **32** bits long left- and right keys

→ note that only **56** bits of the original **64** bits are selected

→ we omit some bits (**8**, **16**, **24**, **32**, **40**, **48**, **56** and **64**)

Генерація раундових ключів

Генератор ключів створює шістнадцять ключів по 48 бітів із ключа шифру на 56 бітів. Проте ключ шифру зазвичай дається як ключ із 64 бітів, в якому 8 додаткових бітів є бітами перевірки. Вони відкидаються перед фактичним процесом генерації ключів.

$$K = abab19283746cdcd_{16}.$$

Розв'язання

Замінюючи значення ключа шифру на двійкове

$$1010\ 1011\ 1010\ 1011\ 0001\ 1001\ 0010\ 1000_2$$
$$0011\ 0111\ 0100\ 0110\ 1100\ 1101\ 1100\ 1101_2$$

$$C_0 = 1100\ 0011\ 1100\ 0000\ 0011\ 0011\ 1010_2 = c3c033a_{16};$$

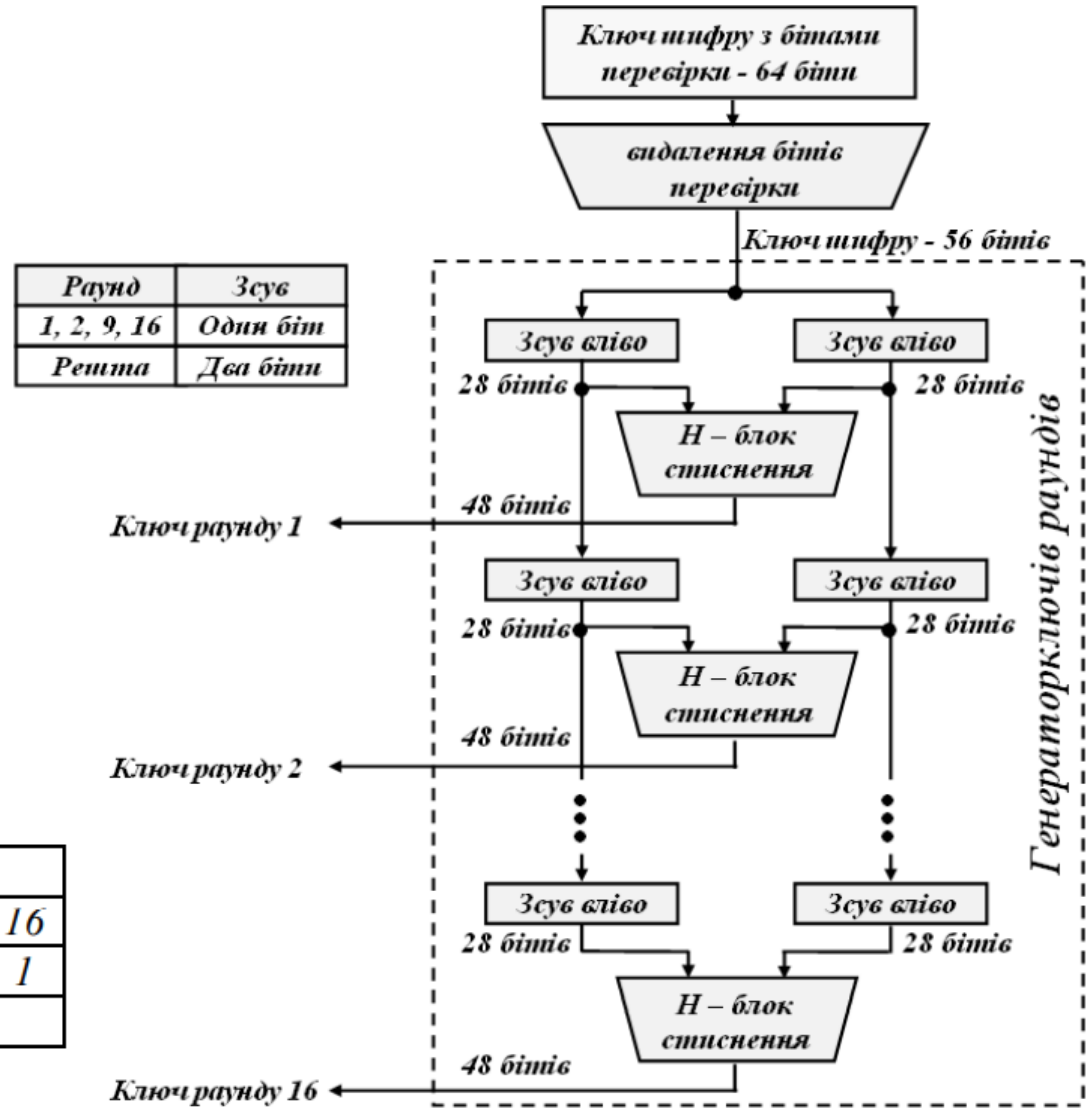
$$D_0 = 0011\ 0011\ 1111\ 0000\ 1100\ 1111\ 1010_2 = 33f0cfa_{16}.$$

$$C_2 = 0000\ 1111\ 0000\ 0000\ 1100\ 1110\ 1011_2 = 0f00ceb_{16};$$

$$D_2 = 1100\ 1111\ 1100\ 0011\ 0011\ 1110\ 1000_2 = cfc33e8_{16}.$$

Кількість бітів циклічного зсуву послідовностей C_i і D_i

Номери раундів															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Кількість бітів, що зсуваються															



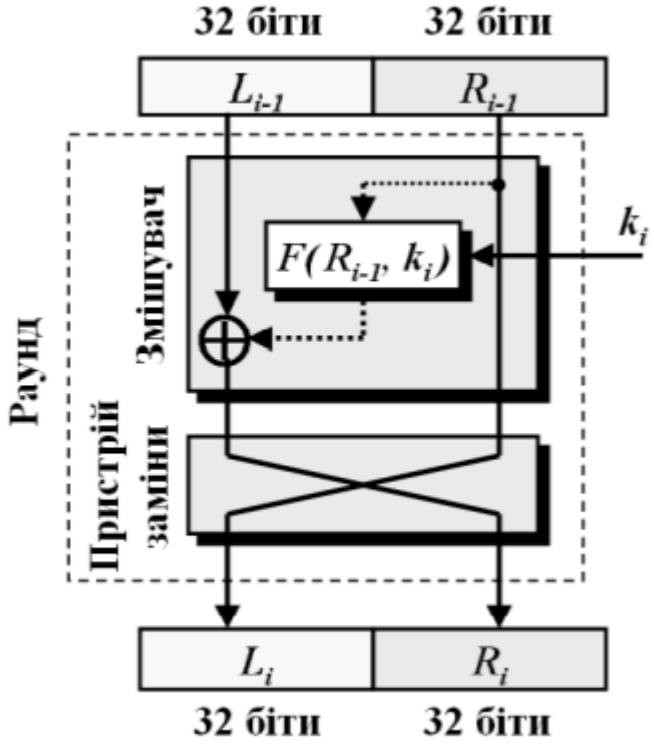
Генерація раундових ключів

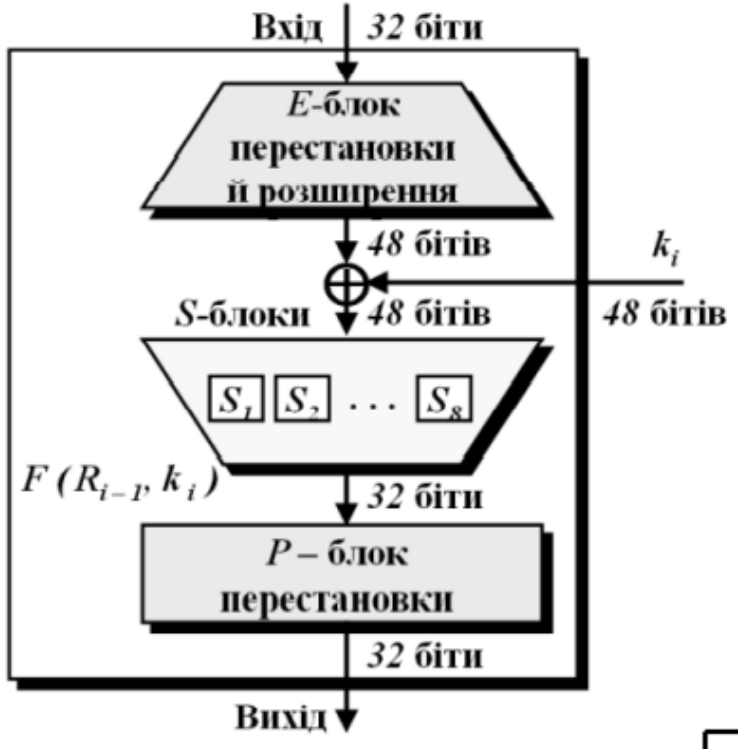
Після циклічного зсуву послідовностей Ci і Di, вони об'єднуються, щоб створити блок у 56 бітів, тобто виходить послідовність Ci||Di, де || — знак математичної операції конкатенації (об'єднання). Перестановка та стиснення (H-функція) змінює 56 бітів Ci||Di на 48 бітів ki, які використовуються як раундові ключі.

Таблиця кінцевої обробки раундових ключів *H*

Вхід	*14*	*17*	*11*	*24*	*01*	*05*	*03*	*28*	*15*	*06*	*21*	*10*	*23*	*19*	*12*	*04*
Вихід k_i	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Вхід	*26*	*08*	*16*	*07*	*27*	*20*	*13*	*02*	*41*	*52*	*31*	*37*	*47*	*55*	*30*	*40*
Вихід k_i	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
Вхід	*51*	*45*	*33*	*48*	*44*	*49*	*39*	*56*	*34*	*53*	*46*	*42*	*50*	*36*	*29*	*32*
Вихід k_i	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*

Процес шифрування даних

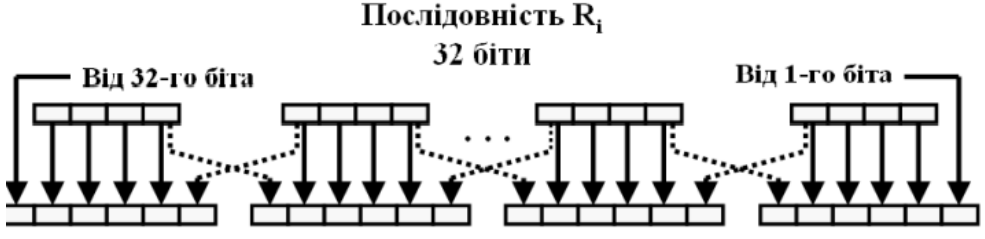




Оскільки на вході R_{i-1} має довжину 32 біти, а ключ k_i - довжину 48 бітів, спочатку потрібно розширити R_{i-1} до 48 бітів. R_{i-1} розділяється на 8 секцій по 4 біти. Кожна секція на 4 біти розширюється до 6 бітів. Ця перестановка з розширенням відбувається за заздалегідь визначеними правилами.

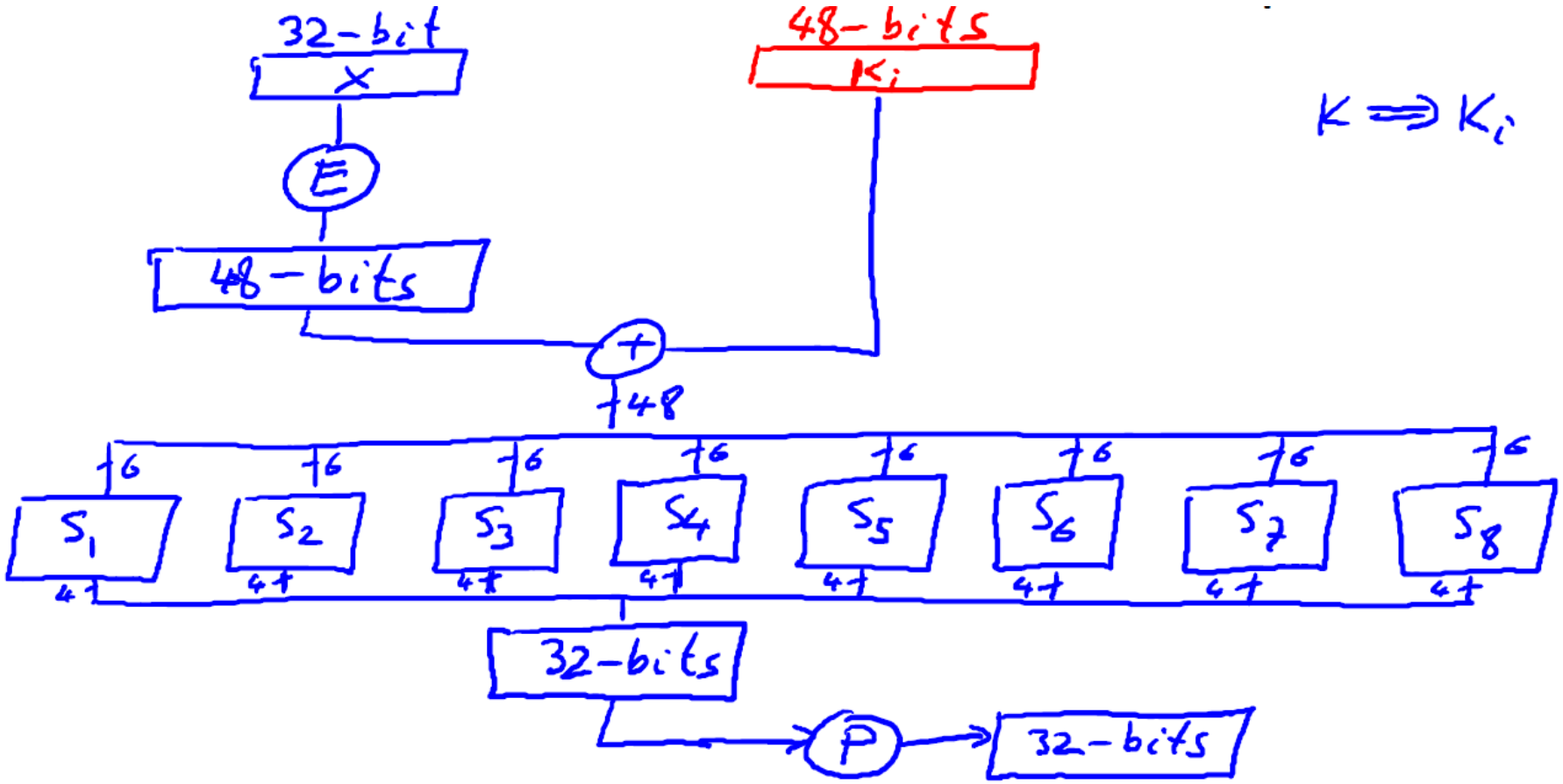
$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus F(R_{i-1}, k_i), i = 1,2, \ldots 16.$

$E_K(M) = R_{16}L_{16}$



$R_{i-1}^E = E(R_{i-1})$

Функція F(x, k_i)



S-box: function $\{0,1\}^6 \longrightarrow \{0,1\}^4$, implemented as look-up table.

Розширення

What is the **expansion function**?

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

In this phase the **DES** algorithm transforms a **32** bits input into a **48** bits output

→ of course it means some bits are duplicated

→ **16** extra bits will be added: the bits in the left column and the bits in the right column

Перестановка PC-2

What is the **permuted choice 2 (PC-2)**?

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

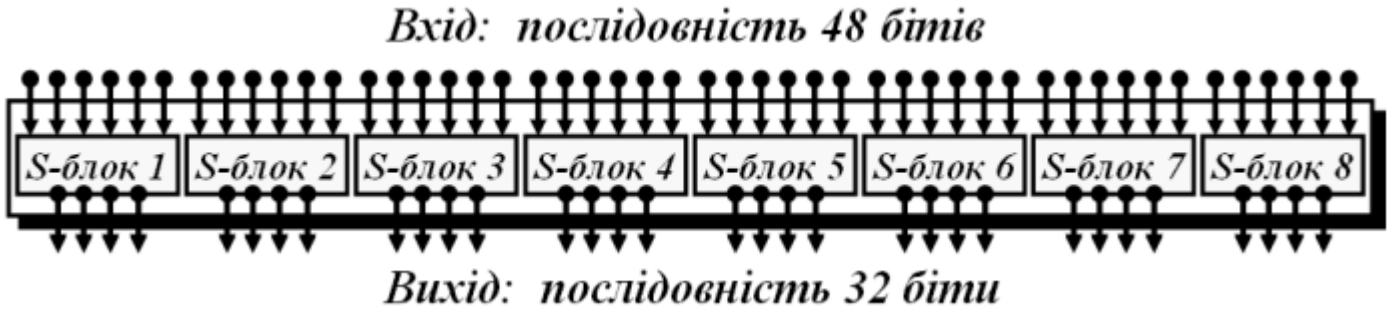
This table again defines the location of the given bits

→ some bits are not used

→ this is why this **PC-2** selects **48** bits from the original **56** bits long key

S - блоки заміни функції DES

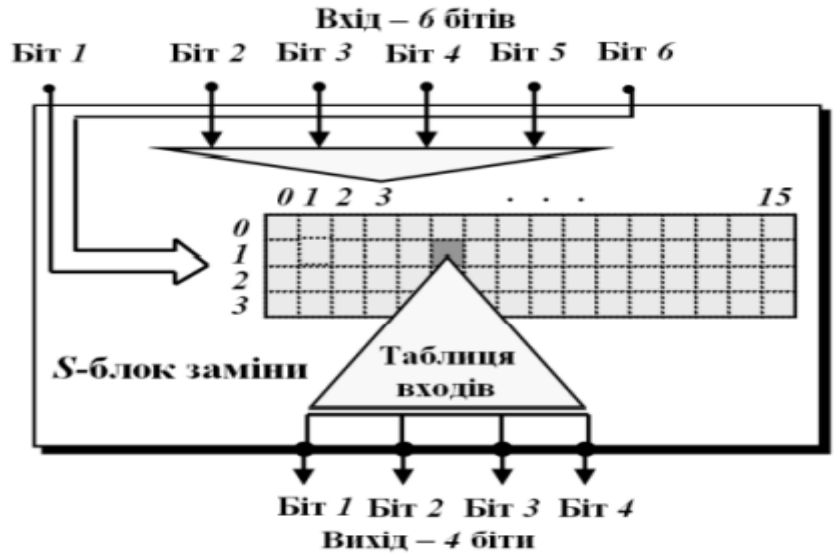
S-блоки заміни змішують інформацію (операція перемішування). DES використовує S-блоки, кожний із 6 вхідними бітами і 4 вихідними. Послідовність із 48 бітів розділяється на вісім частин по 6 бітів, і кожна частина надходить у відповідний S-блок. Результат кожного S-блока — дані завдовжки 4 біти:



$$R_{i-1}^{S} = S(R_{i-1}^{\oplus})$$

$$R_{i-1}^{\oplus} = B_1 \,||\, B_2 \,||\, B_3 \,||\, B_4 \,||\, B_5 \,||\, B_6 \,||\, B_7 \,||\, B_8$$

$$B_j = \underbrace{b_1 \,||\, b_6}_{\text{Номер рядка}} \,||\, \underbrace{b_2 \,||\, b_3 \,||\, b_4 \,||\, b_5}_{\text{Номер стовпця}}$$



https://en.wikipedia.org/wiki/DES_supplementary_material

Лінійні S-boxes

Suppose:

$$S_i(x_1, x_2, \ldots, x_6) = \left(x_2 \oplus x_3,\ \ x_1 \oplus x_4 \oplus x_5,\ \ x_1 \oplus x_6,\ \ x_2 \oplus x_3 \oplus x_6 \right)$$

or written equivalently: $S_i(\mathbf{x}) = A_i \cdot \mathbf{x} \pmod 2$

$$\begin{bmatrix} 0&1&1&0&0&0 \\ 1&0&0&1&1&0 \\ 1&0&0&0&0&1 \\ 0&1&1&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_2 \oplus x_3 \\ x_1 \oplus x_4 \oplus x_5 \\ x_1 \oplus x_6 \\ x_2 \oplus x_3 \oplus x_6 \end{bmatrix}$$

We say that S_i is a linear function.

Лінійні S-boxes

Then entire DES cipher would be linear: ∃fixed binary matrix B s.t.

$$\text{DES}(k,m) = \underset{64}{\overset{832}{B}} \cdot \begin{pmatrix} m \\ k_1 \\ k_2 \\ \vdots \\ k_{16} \end{pmatrix} = c \pmod 2$$

But then: $\text{DES}(k,m_1) \oplus \text{DES}(k,m_2) \oplus \text{DES}(k,m_3) = \text{DES}(k, m_1 \oplus m_2 \oplus m_3)$

$$B \begin{pmatrix} m_1 \\ k \end{pmatrix} \oplus B \begin{pmatrix} m_2 \\ k \end{pmatrix} \oplus B \begin{pmatrix} m_3 \\ k \end{pmatrix} = B \begin{pmatrix} m_1 \oplus m_2 \oplus m_3 \\ k \oplus k \oplus k \end{pmatrix}$$

$K = \begin{pmatrix} k_1 \\ \vdots \\ k_{16} \end{pmatrix}$

Лінійні S-boxes

Choosing the S-boxes and P-box at random would result in an insecure block cipher (key recovery after $\approx 2^{24}$ outputs) [BS'89]

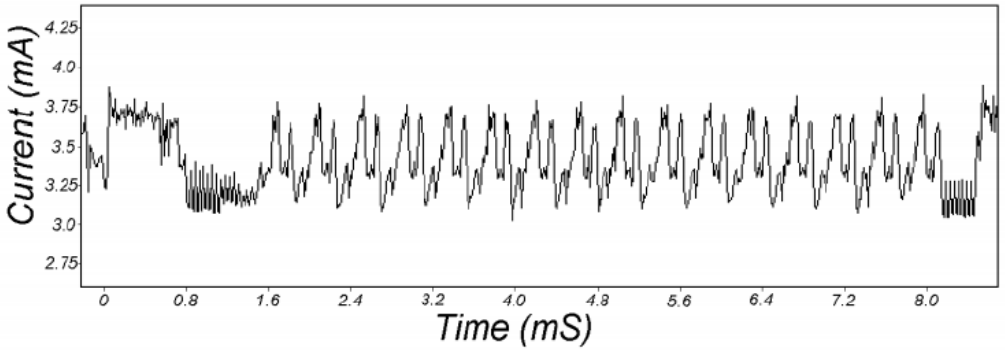
Several rules used in choice of S and P boxes:

- No output bit should be close to a linear func. of the input bits

Атаки на реалізацію

1. Side channel attacks:
   - Measure **time** to do enc/dec, measure **power** for enc/dec

smartcard



[Kocher, Jaffe, Jun, 1998]

2. Fault attacks:
   - Computing errors in the last round expose the secret key k

⇒ do not even implement crypto primitives yourself …

Модель ідеального шифру

Def: In the ideal cipher model, we assume the block cipher *is* a random permutation for *every* key. Furthermore, we treat these permutations as independent.

.

Example: Suppose DES is an ***ideal cipher***

It is a collection of 2^{56} independent random permutations, one for each key

Атака грубої сили (or exhaustive search)

Goal: given a few input output pairs $(m_i, c_i = E(k, m_i))$ i=1,..,3
find key k.

Lemma: Suppose DES is an ***ideal cipher***

(2^{56} random invertible functions $\pi_1, \ldots, \pi_{2^{56}} : \{0,1\}^{64} \to \{0,1\}^{64}$)

Then ∀ m, c there is at most **one** key k s.t. c = DES(k, m)

Proof: with prob. ≥ 1 – 1/256 ≈ 99.5%

$$\Pr[\exists k' \neq k : c = DES(k,m) = DES(k',m)] \leq$$

$$\leq \sum_{k' \in \{0,1\}^{56}} \Pr[DES(k,m) = DES(k',m)] \leq 2^{56} \cdot \frac{1}{2^{64}} = \frac{1}{2^8}$$

Атака грубої сили (or exhaustive search)

For two DES pairs $(m_1, c_1=DES(k, m_1))$, $(m_2, c_2=DES(k, m_2))$

unicity prob. $\approx 1 - 1/2^{71}$

For AES-128: given two inp/out pairs, unicity prob. $\approx 1 - 1/2^{128}$

$\Rightarrow$ two input/output pairs are enough for exhaustive key search for DES but not AES.

DES challenge

msg = "The unknown messages is: XXXX … "

CT = c_1 c_2 c_3 c_4

Goal: find $k \in \{0,1\}^{56}$ s.t. $DES(k, m_i) = c_i$ for i=1,2,3

1997: Internet search -- **3 months**

1998: EFF machine (deep crack) -- **3 days** (250K $)

1999: combined search -- **22 hours**

2006: COPACOBANA (120 FPGAs) -- **7 days** (10K $)

⇒ 56-bit ciphers should not be used !! (128-bit key ⇒ 2^{72} days)

Атака грубої сили (Brute Force Crack)

→ of course again we can use brute-force approach to check all the possible values for the keys

DES keyspace's size = $\mathbf{2^{56}}$

→ the small size of the keyspace is the reason why **DES** cryptosystem is no longer secure

„**Deep Crack**" has managed to crack **DES** with brute-force attack within **22 hours**
~ it does not use any internal structure of the cryptosystem
just considers all the possible keys (linear search)

This is why **DES** was replaced by triple DES (**TDES**) and later with **AES**

Лінійний криптоаналіз

Linear cryptoanalysis needs **N** plaintext / ciphertext pairs

→ for cracking **DES** we need $\mathbf{2^{47}}$ known plaintexts so this approach is not practical when cracking **DES**

→ usually **linear cryptoanalysis** is faster than brute-force approach

This approach assumes a linear relationship between the elements (individual bits) of the plaintext, the ciphertext and the key

So this approach tries to find an **f** linear approximation such that **ciphertext = f(plaintext,key)**

WE ARE LOOKING FOR INFORMATION LEAKING (NON-RANDOM BEHAVIOR)

~ the aim of **permutations** and mainly the **S-box** is to end up with a random sequence

Лінійний криптоаналіз

Given *many* inp/out pairs, can recover key in time less than 2^{56} .

Linear cryptanalysis (overview) : let c = DES(k, m)

Suppose for random k,m :

$$\Pr\left[\underbrace{m[i_1] \oplus \cdots \oplus m[i_r]}_{\text{subset of msg bits}} \oplus \underbrace{c[j_1] \oplus \cdots \oplus c[j_v]}_{\text{subset of ciphertext bits}} = \underbrace{k[l_1] \oplus \cdots \oplus k[l_u]}_{\text{subset of key bits}} \right] = \tfrac{1}{2} + \varepsilon$$

For some ε. For DES, this exists with $\varepsilon = 1/2^{21} \approx 0.0000000477$

A tiny bit of linearity in S_5 lead to a 2^{42} time attack.

Лінійний криптоаналіз

$$\Pr\left[\, m[i_1]\oplus\cdots\oplus m[i_r] \oplus c[j_j]\oplus\cdots\oplus c[j_v] = k[l_1]\oplus\cdots\oplus k[l_u] \,\right] = \tfrac{1}{2} + \varepsilon$$

Thm: given $1/\varepsilon^2$ random $(m, c=DES(k, m))$ pairs then

$$k[l_1,\ldots,l_u] = \mathrm{MAJ}\left[\, m[i_1,\ldots,i_r] \oplus c[j_j,\ldots,j_v] \,\right]$$

with prob. ≥ 97.7%

$\Rightarrow$ with $1/\varepsilon^2$ inp/out pairs can find $k[l_1,\ldots,l_u]$ in time $\approx 1/\varepsilon^2$.

Лінійний криптоаналіз

For DES, $\varepsilon = 1/2^{21} \Rightarrow$

with 2^{42} inp/out pairs can find $k[l_1,\ldots,l_u]$ in time 2^{42}

Roughly speaking: can find 14 key "bits" this way in time 2^{42}

Brute force remaining $56-14=42$ bits in time 2^{42}

Total attack time $\approx 2^{43}$ ($<< 2^{56}$) with 2^{42} random inp/out pairs

Diferenciální kryptoanalýza

Differential cryptoanalysis needs **N** plaintext / ciphertext pairs

AGAIN WE ARE LOOKING FOR INFORMATION LEAKING (NON-RANDOM BEHAVIOR)

~ the aim of **permutations** and mainly the **S-box** is to end up with a random sequence

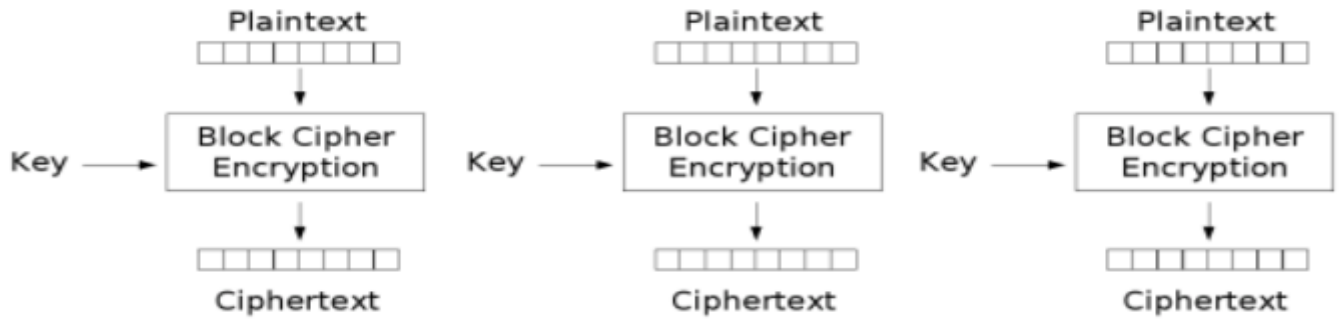
This approach aim to map bitwise **ΔX** differences in the input (plaintext) to **ΔY** differences in the output (ciphertext)

→ of course the aim is to reverse-engineer the cryptosystem

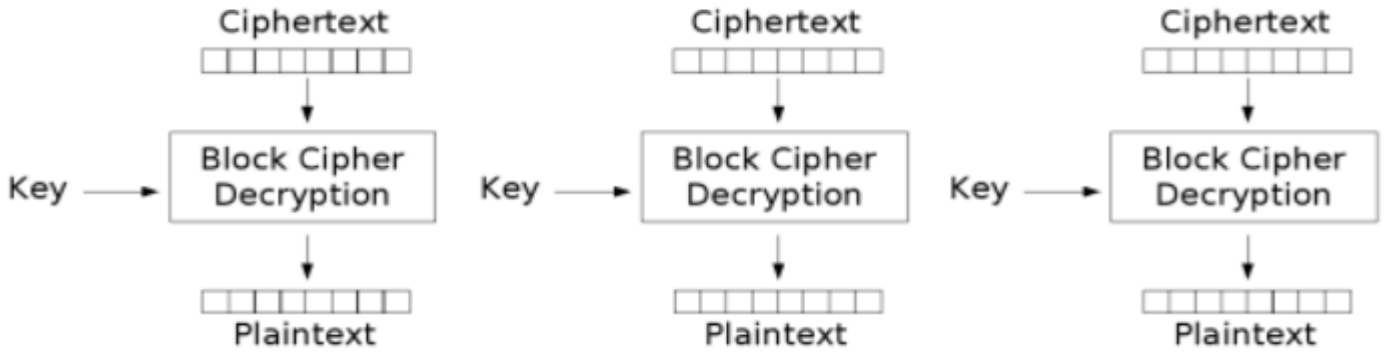
→ we analyse what will happen to the output if we make a little change in the input

Режим електронної кодової книги (ECB)

Режим електронної кодової книги (Electronic Code Book — ECB). У цьому режимі виконання кожний блок незашифрованих даних шифрується незалежно від інших блоків, із застосуванням того самого ключа шифрування. Типові застосування — безпечна передача поодиноких значень (наприклад криптографічного ключа)



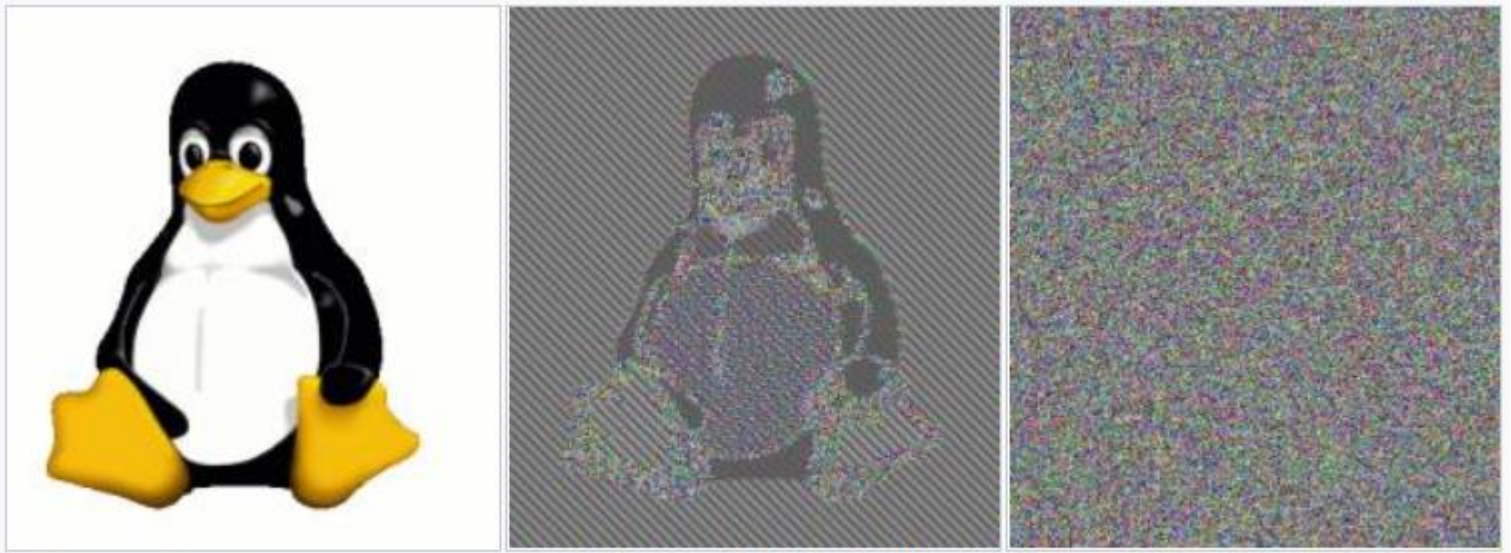
Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Режим електронної кодової книги (ECB)

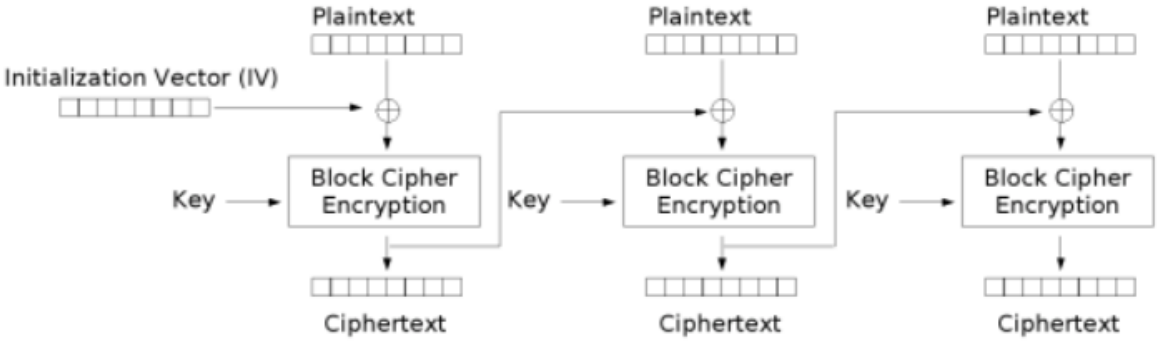
Типовий приклад ступеня збереження шаблонів ECB відкритого тексту в шифротексті можна побачити, коли ECB використовують для шифрування bitmap зображень, які використовують великі площі однорідних кольорів. Хоча колір кожного окремого пікселя зашифровано, загальну картинку можна розрізнити як маску однокольорових пікселів картинки на вході.



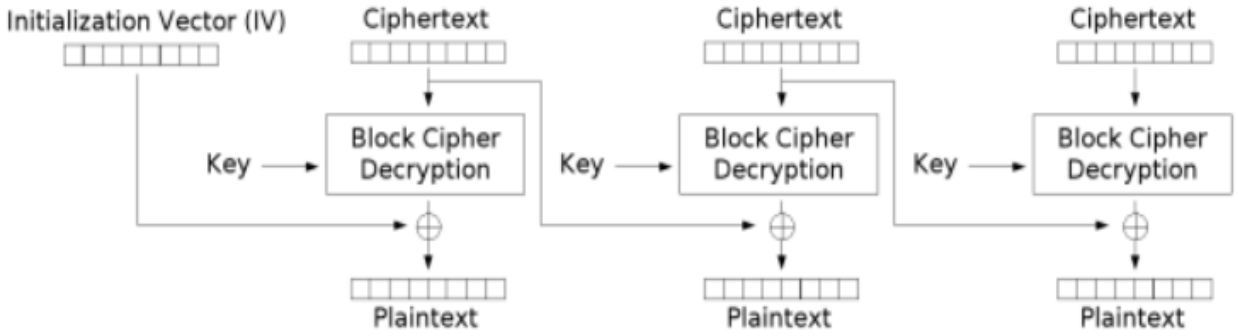
Первісне зображення | зашифроване в режимі ECB | Псевдовипадковий вислід режимів відмінних від ECB

Режим зчеплення шифроблоків (CBC)

IBM винайшла режим зчеплення шифроблоків (англ. cipher-block chaining, CBC) у 1976.[5] В CBC режимі, кожен блок відкритого тексту XOR-ять з попереднім шифроблоком перед шифруванням. Так, кожен шифроблок, залежить від усіх блоків оброблених до нього. Для отримання унікальних повідомлень потрібно використовувати ініціалізаційний вектор у першому блоці.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

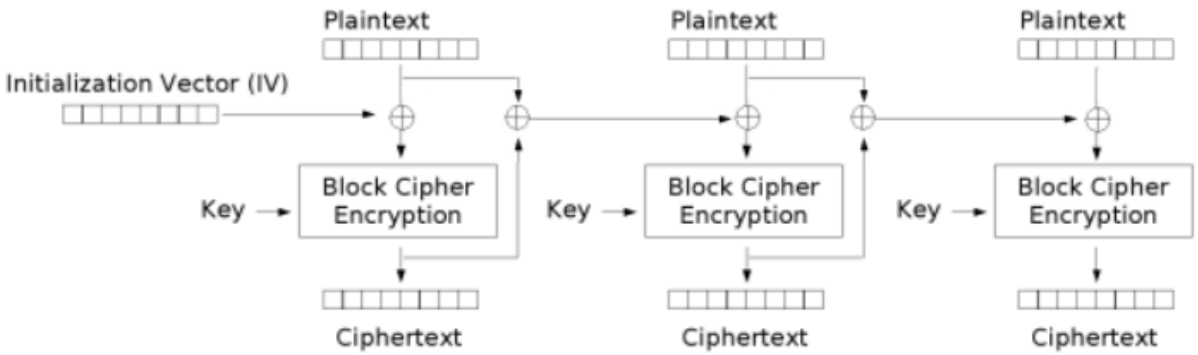
$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV,$$

$$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV.$$

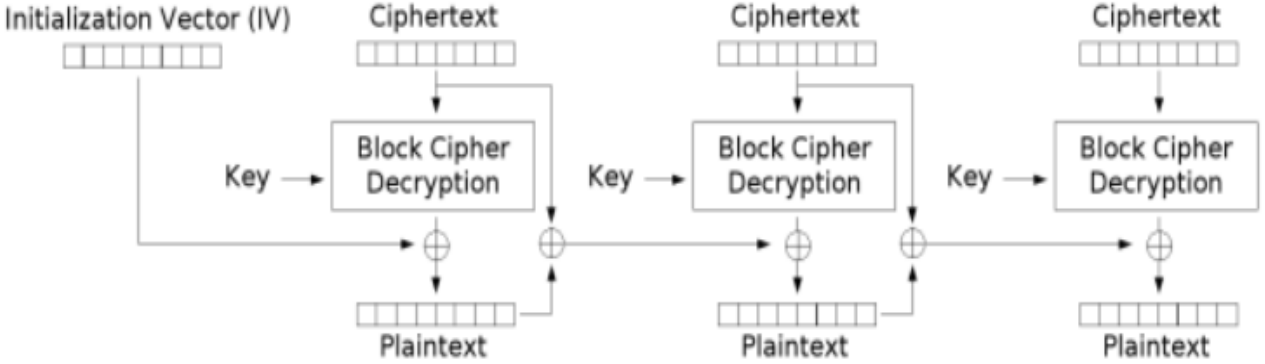
CBC найчастіше використовний режим. Основна його вада це властива послідовність (тобто не можливість розпаралелення), і необхідність доповнення повідомлення до розміру кратного розміру блоку. Зауважте, що зміна одного біта в відкритому тексті або IV впливає на всі наступні шифроблоки. Розшифрування з неправильним IV спричиняє пошкодження першого блоку відкритого тексту, але наступні блоки будуть правильними. Це відбувається через можливість відновити відкритий текст з двох суміжних блоків шифротексту. Як наслідок, розшифрування можна розпаралелити.

Режим зчеплення шифроблоків з поширенням (PCBC)

Режим зчеплення шифроблоків із поширенням (англ. propagating cipher-block chaining, PCBC) спроектували з метою щоб незначні зміни в шифротексті нескінченно поширювались при розшифруванні.



Propagating Cipher Block Chaining (PCBC) mode encryption



Propagating Cipher Block Chaining (PCBC) mode decryption

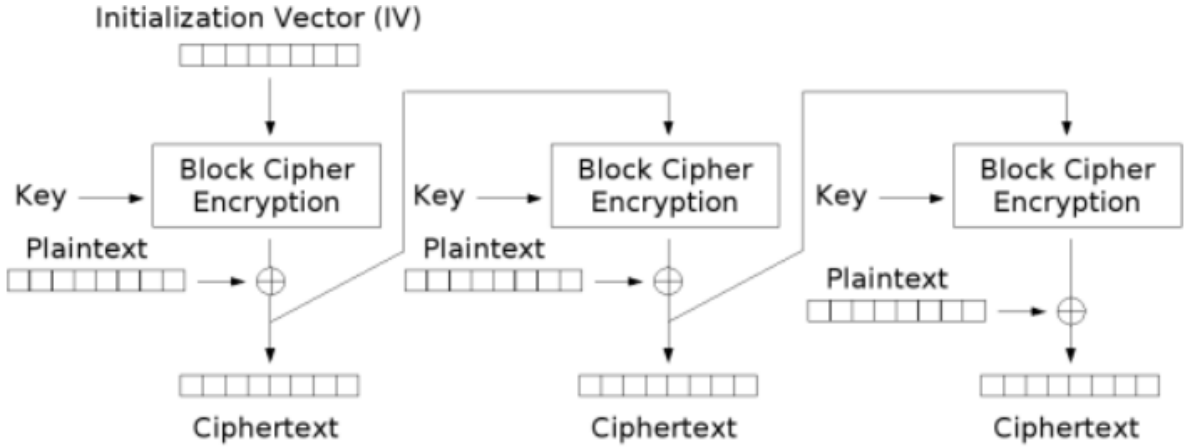
$$C_i = E_K(P_i \oplus P_{i-1} \oplus C_{i-1}), P_0 \oplus C_0 = IV$$

$$P_i = D_K(C_i) \oplus P_{i-1} \oplus C_{i-1}, P_0 \oplus C_0 = IV$$

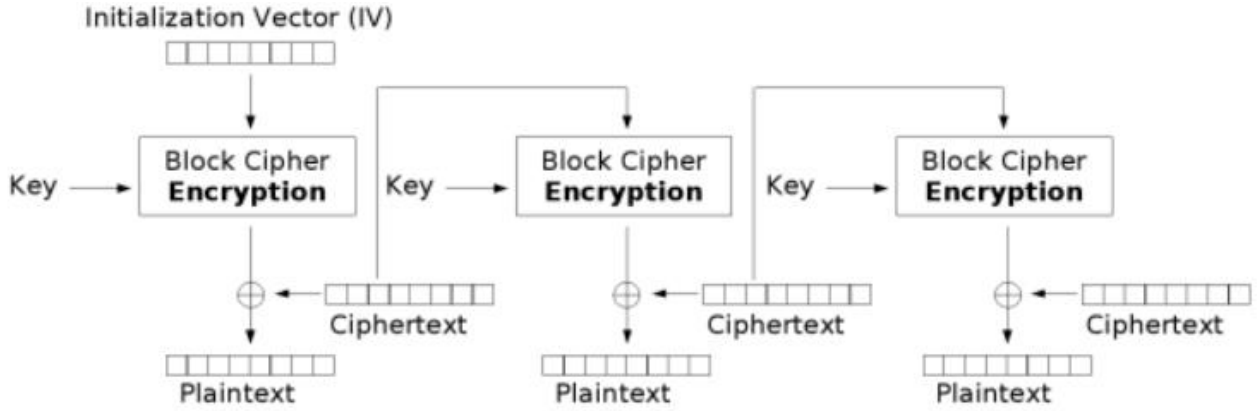
Kerberos v4 використовують PCBC. Якщо повідомлення зашифроване у PCBC режимі, тоді переставка двох сусідніх блоків не впливає на розшифрування наступних блоків. Через це, Kerberos v5 не використовує PCBC.

Зворотний зв'язок за криптотекстом (CFB)

Режим зворотного зв'язку за криптотекстом (англ. cipher feedback, CFB), близький родич для CBC, перетворює блоковий шифр на потоковий, що самосинхронізується. Дія дуже схожа; зокрема, розшифрування CFB майже тотожне до шифрування CBC виконаного навпаки:



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

$$C_i = E_K(C_{i-1}) \oplus P_i$$

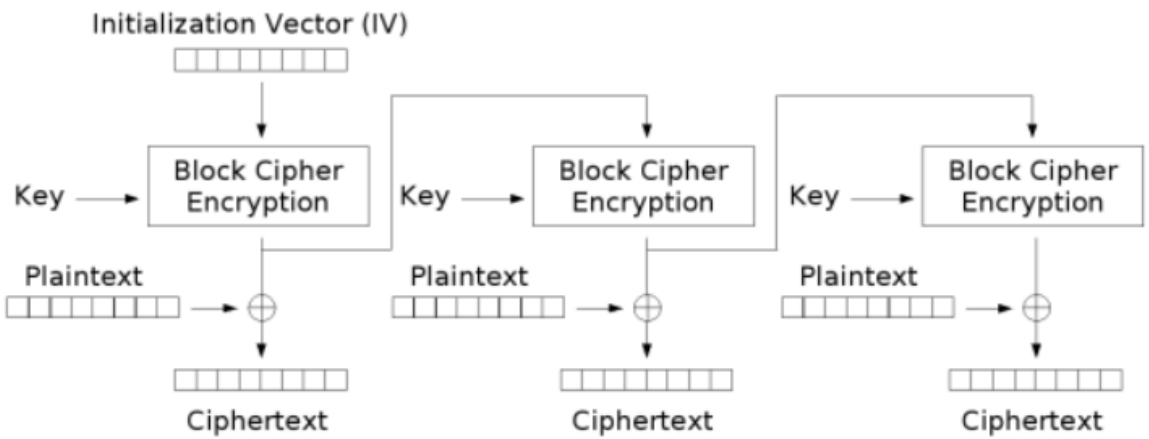
$$P_i = E_K(C_{i-1}) \oplus C_i$$

$$C_0 = \mathrm{IV}$$

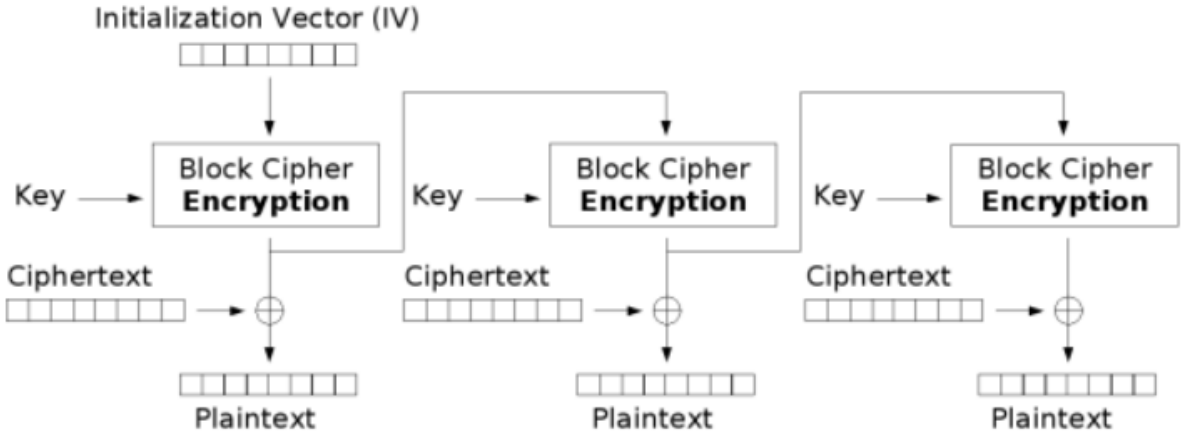
Подібно до режиму CBC, зміни в відкритому тексту поширюються на весь шифротекст, і шифрування не можна розпаралелити. При розшифруванні, зміна одного біту в шифротексті впливає на два блоки відкритого тексту: однобітова зміна в відповідному блоці відкритого тексту і повне пошкодження наступного блоку. Подальші блоки розшифровуються нормально.

Зворотний зв'язок за виходом (OFB)

Режим зворотного зв'язку за виходу (англ. output feedback, OFB) утворює з блокового шифру синхронний потоковий шифр. Він утворює потік ключа, який потім XOR-иться з блоками відкритого тексту для утворення шифротескту. Як і передбачає ім'я, OFB використовує попередній вихід блокового шифру, а не попередній шифротекст як CFB, що робить шифротекст незалежним від відкритого тексту і шифротексту, саме через це OFB є синхронним потоковим шифром.



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

$$C_j = P_j \oplus O_j$$

$$P_j = C_j \oplus O_j$$

$$O_j = E_K(I_j)$$

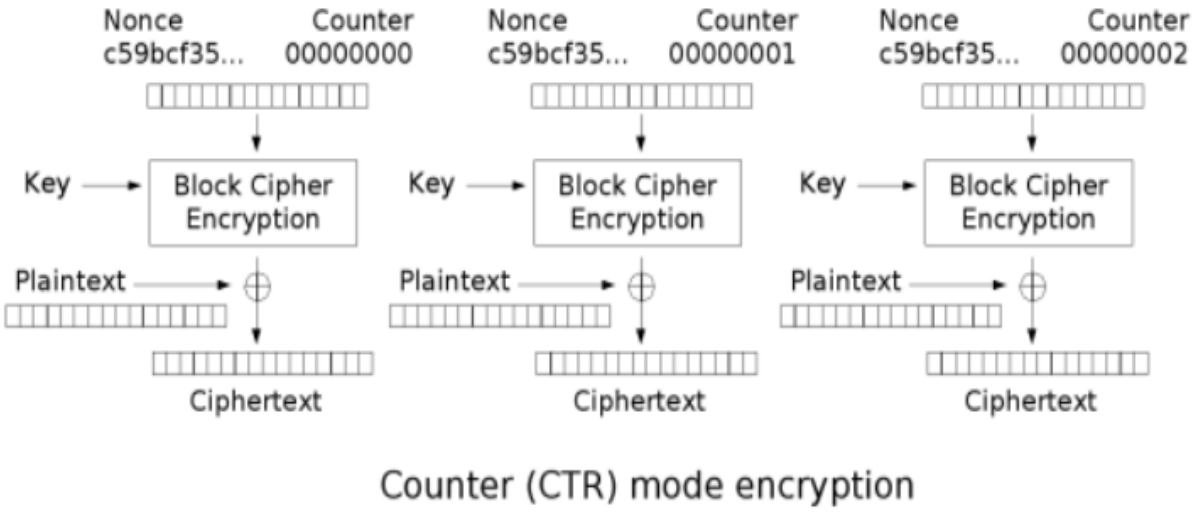
$$I_j = O_{j-1}$$

$$I_0 = \mathrm{IV}$$

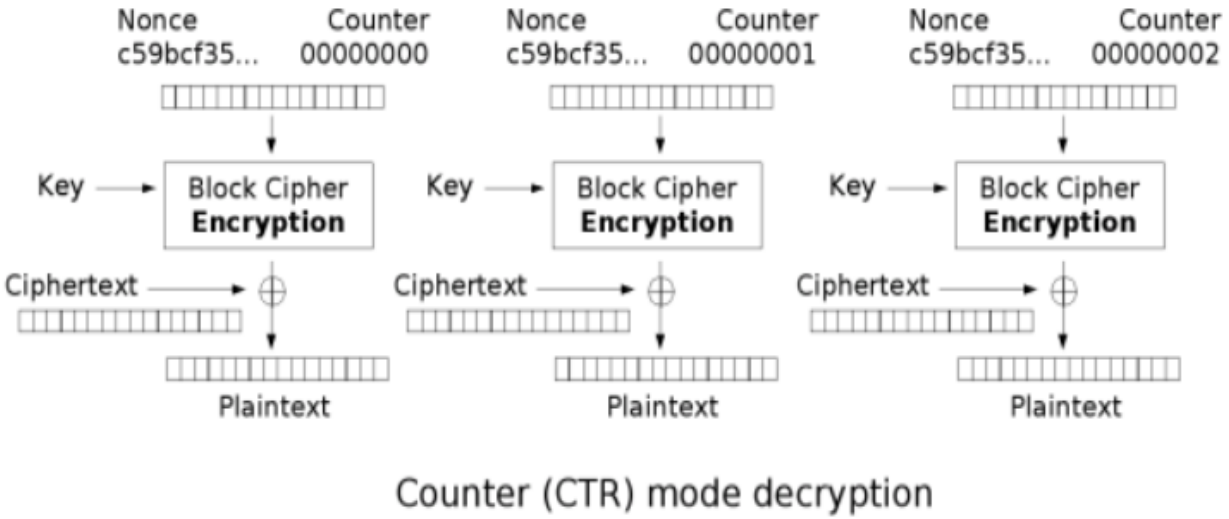
На відміну від CFB, помилка в одному біті відкритого тексту зачіпає лише відповідний біт в шифротексті, наступні блоки залишаються неушкодженими. Ця властивість разом із можливістю високої швидкодії OFB робить його підходящим для шифрування потоків даних голосу і відео.

Лічильник (CRT)

Подібно до OFB, режим лічильника перетворює блоковий шифр в потоковий шифр. Він породжує наступний блок потоку ключа шифруванням послідовних значень «лічильника». Лічильник може бути будь-якою функцією, що видає послідовність, яка гарантовано не повторюється впродовж тривалого часу, насправді найпростішими і найпоширенішими є прості лічильники, що на кожному кроці збільшуються на одиницю.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Режим CTR має подібні до OFB характеристики, але також має можливість довільного доступу під час розшифрування. Режим CTR добре підходить для використання на багатопроцесорній машині, де блоки можна шифрувати паралельно.

Galois/Counter Mode (GCM)

Шифрування:

$$H = E(K, 0^{128})$$

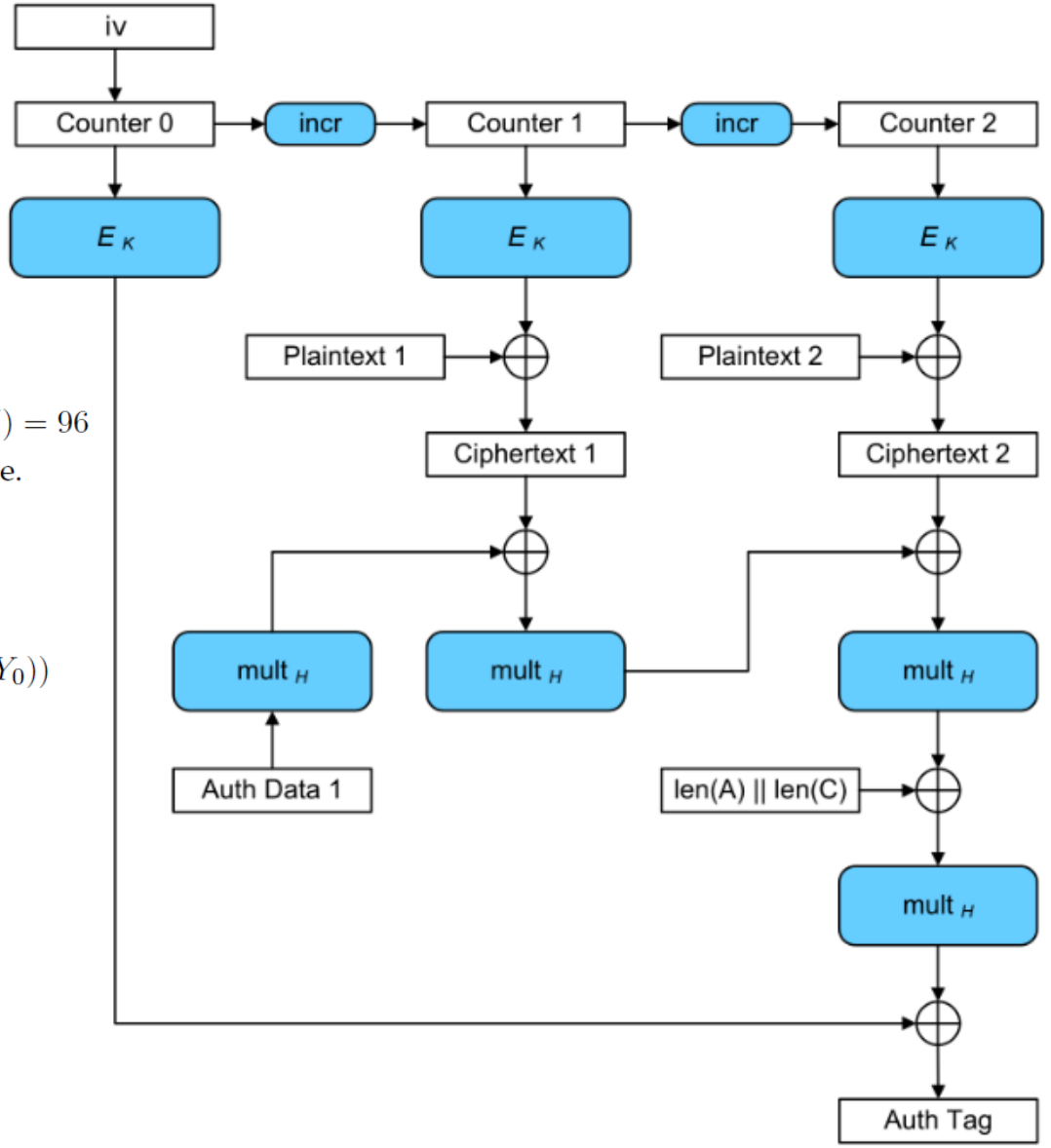
$$Y_0 = \begin{cases} IV \| 0^{31}1 & \text{if } \text{len}(IV) = 96 \\ \text{GHASH}(H, \{\}, IV) & \text{otherwise.} \end{cases}$$

$$Y_i = \text{incr}(Y_{i-1}) \text{ for } i = 1, \ldots, n$$

$$C_i = P_i \oplus E(K, Y_i) \text{ for } i = 1, \ldots, n-1$$

$$C_n^* = P_n^* \oplus \text{MSB}_u(E(K, Y_n))$$

$$T = \text{MSB}_t(\text{GHASH}(H, A, C) \oplus E(K, Y_0))$$



Дешифрування:

$$H = E(K, 0^{128})$$

$$Y_0 = \begin{cases} IV \| 0^{31}1 & \text{if } \text{len}(IV) = 96 \\ \text{GHASH}(H, \{\}, IV) & \text{otherwise.} \end{cases}$$

$$T' = \text{MSB}_t(\text{GHASH}(H, A, C) \oplus E(K, Y_0))$$

$$Y_i = \text{incr}(Y_{i-1}) \text{ for } i = 1, \ldots, n$$

$$P_i = C_i \oplus E(K, Y_i) \text{ for } i = 1, \ldots, n$$

$$P_n^* = C_n^* \oplus \text{MSB}_u(E(K, Y_n))$$

Кількоразове шифрування

Алгоритм, який полягає у шифруванні повідомлення за допомогою одного шифу, а потім до отриманого криптотексту ще одного шифру називається композицією або добутком двох шифрів.

	ADFGVX
A	C08XF4
D	MK3AZ9
F	NWL0JD
G	5SIYHU
V	P1VB6R
X	EQ7T2G

Таблиця білітеральної підстановки-подрібнення.

Повідомлення:

D O N'T P U T I T O F F T I L L T O M O R R O W

Проміжний криптотекст:

FXADFAXGVAGXXGGFXGADAVAVXGGFFFFFXGADDAADVXVXADFD

GARDEN
416235
FXADFA
XGVAGX
XGGFXG
ADAVAV
XGGFFF
FFXGAD
DAADVX
VXADFD

Матриця перестановки. Запис проміжного криптотексту рядками і зчитування колонками в порядку, визначеному ключовим словом GARDEN.

Криптотекст:

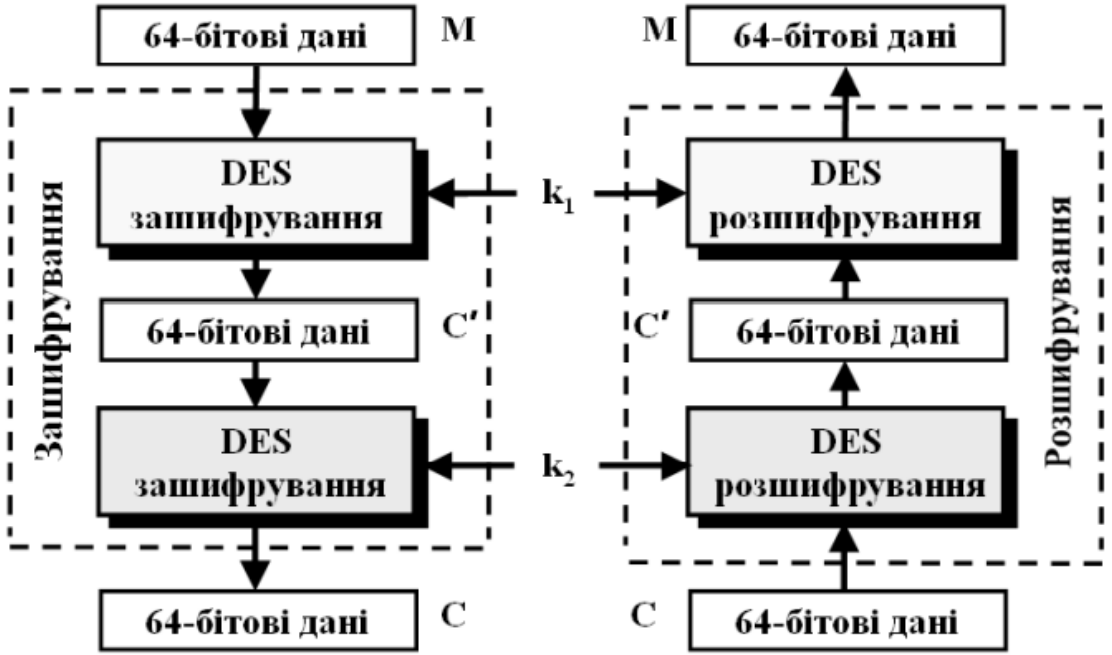
XGGDG FAXDA FVFGD DFGXA
FAVFF XXAXF DVAXG VFDXD
AVGAG XAA

Інволютивні криптосистеми – алгоритм шифрування збігається з алгоритмом дешифрування.

Шифри ***утворюють групу***, якщо подвійне шифрування з двома різними ключами дає той самий результат, що і шифрування з деяким іншим ключем. Розробники шифрів намагаються проектувати шифр так щоб він не утворював групу. Одним з таких шифрів є DES.

Подвійний DES

Перший підхід полягає в тому, щоб використати подвійний DES (Double DES). За цього підходу застосовуємо два типи прямих шифрів DES для зашифрування та два типи обернених шифрів для розшифрування. Кожний тип використовує різний ключ, що означає — розмір ключа тепер подвоївся (112 бітів). Проте подвійний DES уразливий до атаки знання відкритих даних.



Find (k_1, k_2) s.t.
$E(k_1, E(k_2, M)) = C$
Equivalently:
$E(k_2, M) = D(k_1, C)$

step 1: build table.

sort on 2nd column

$k^0 = 00\ldots00$	$E(k^0, M)$
$k^1 = 00\ldots01$	$E(k^1, M)$
$k^2 = 00\ldots10$	$E(k^2, M)$
⋮	⋮
$k^N = 11\ldots11$	$E(k^N, M)$

2^{56} entries

Step 2: for all $k \in \{0,1\}^{56}$ do:

test if $D(k, C)$ is in 2nd column.

if so then $E(k^i, M) = D(k, C) \Rightarrow (k^i, k) = (k_2, k_1)$

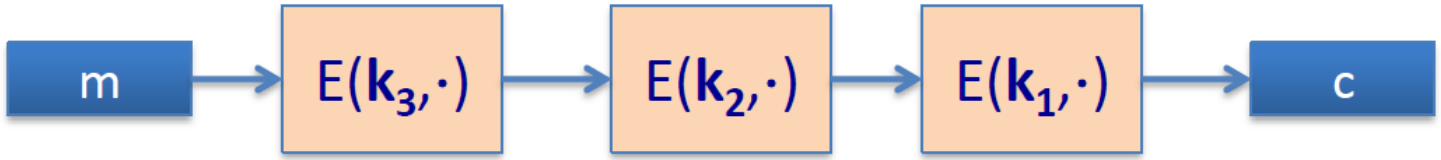
Подвійний DES

Time = $2^{56}\log(2^{56})$ + $2^{56}\log(2^{56})$ < 2^{63} << 2^{112} , space ≈ 2^{56}

build + sort table

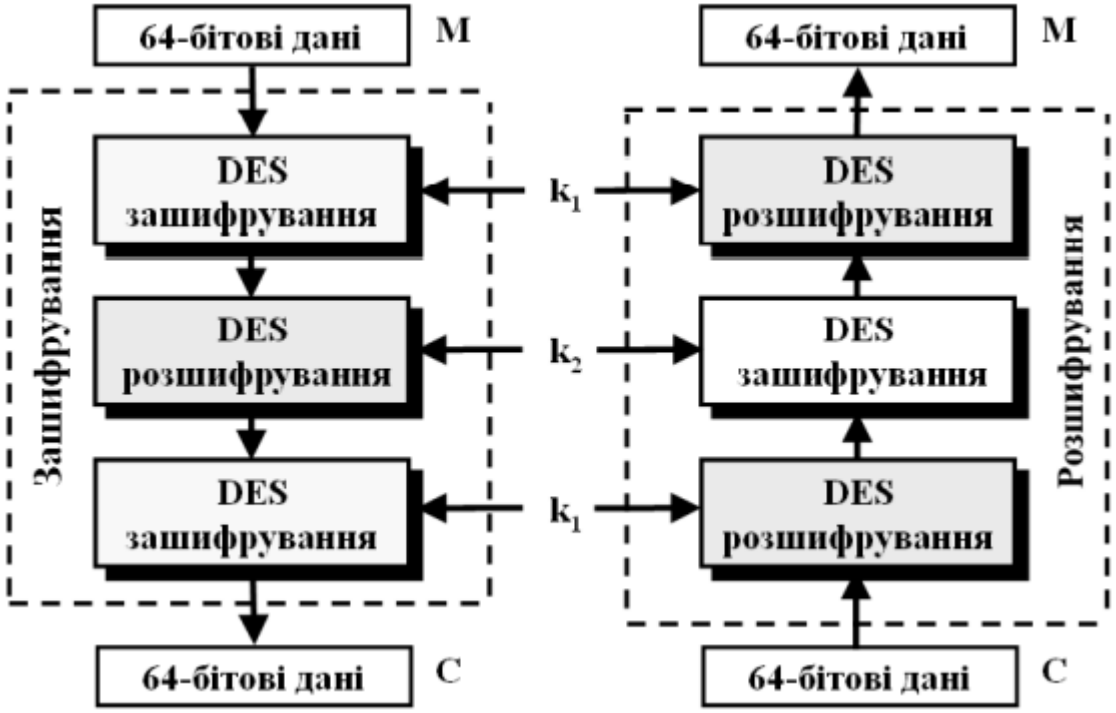
search in table

Same attack on 3DES: Time = 2^{118} , space ≈ 2^{56}



Потрійний DES

Для того щоб поліпшити безпеку DES, було запропоновано потрійний DES (Triple DES). Він використовує три каскади DES для зашифрування й розшифрування. Сьогодні використовуються дві версії потрійних DES: потрійний DES із двома ключами та потрійний DES із трьома ключами.



Потрійний DES

Method 1: **Triple-DES**

- Let $E : K \times M \longrightarrow M$ be a block cipher
- Define **3E**: $K^3 \times M \longrightarrow M$ as

$$\mathbf{3E}\big((k_1,k_2,k_3), m\big) = E(k_1, D(k_2, E(k_3, m)))$$

$k_1 = k_2 = k_3 \Rightarrow$ single DES

For 3DES: key-size = 3×56 = 168 bits. 3×slower than DES.

(simple attack in time $\approx 2^{118}$)

Потрійний DESX

$E : K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ a block cipher

Define EX as $EX\big((k_1,k_2,k_3), m\big) = k_1 \oplus E(k_2, m \oplus k_3)$

For DESX: key-len = 64+56+64 = 184 bits

… but easy attack in time $2^{64+56} = 2^{120}$ (homework)

Note: $k_1 \oplus E(k_2, m)$ and $E(k_2, m \oplus k_1)$ does nothing !!