

CHƯƠNG 3: MẬT MÃ KHỐI VÀ KHÓA ĐỐI XỨNG

3.1. Phép toán trên bit nhị phân 0-1 và bảng mã ASCII

A ⇄	B ⇄	A&B ⇄
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Kí hiệu: A **XOR** B hoặc $A \oplus B$

CHƯƠNG 3: MẬT MÃ KHỐI VÀ KHÓA ĐỐI XỨNG

3.1. Phép toán trên bit nhị phân 0-1 và bảng mã ASCII

LETTER	ASCII VALUES	BINARY VALUES			
A	97	01100001	N	110	01101110
C	99	01100011	O	111	01101111
D	100	01100100	P	112	01110000
E	101	01100101	Q	113	01110001
F	102	01100110	R	114	01110010
G	103	01100111	S	115	01110011
H	104	01101000	T	116	01110100
I	105	01101001	U	117	01110101
J	106	01101010	V	118	01110110
K	107	01101011	W	119	01110111
L	108	01101100	X	120	01111000
M	109	01101101	Y	121	01111001
			Z	122	01111010

CHƯƠNG 3: MẬT MÃ KHỐI VÀ KHÓA ĐỐI XỨNG

3.1. Phép toán trên bit nhị phân 0-1 và bảng mã ASCII

Bản tin: attack

Mã ASCII: 97 116 116 97 99 107

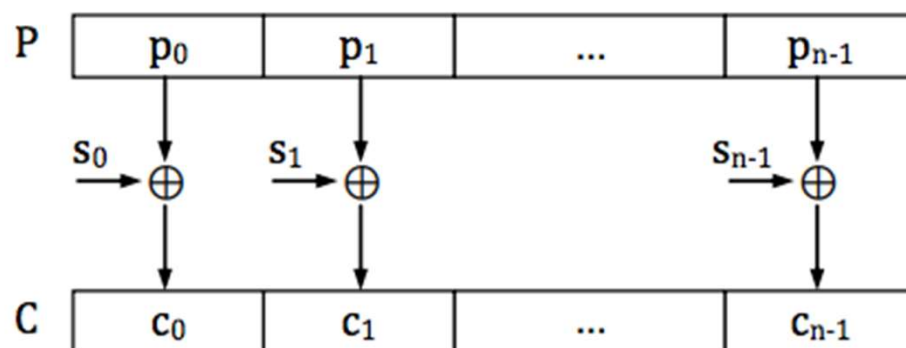
Biểu diễn nhị phân: 01100001 01110100 01110100 01100001 01100011 01101011

3.2. Mã dòng (Stream Cipher)

Mã dòng có các đặc tính sau:

- Kích thước một đơn vị mã hóa: gồm k bit. Bản rõ được chia thành các đơn vị mã hóa: $P \rightarrow p_0 p_1 p_2 \dots p_{n-1}$ ($p_i : k$ bit)
- Một bộ sinh dãy số ngẫu nhiên: dùng một khóa K ban đầu để sinh ra các số ngẫu nhiên có kích thước bằng kích thước đơn vị mã hóa:
 $StreamCipher(K) \rightarrow S = s_0 s_1 s_2 \dots s_{n-1}$ ($s_i : k$ bit)
- Mỗi số ngẫu nhiên được XOR với đơn vị mã hóa của bản rõ để có được bản mã.

$$c_0 = p_0 \oplus s_0, c_1 = p_1 \oplus s_1 \dots ; C = c_0 c_1 c_2 \dots c_{n-1}$$



3.2. Mã dòng (Stream Cipher)

Một số loại mã dòng tiêu biểu:

- **A5/1**: dùng trong mạng điện thoại **GSM**, để bảo mật dữ liệu trong quá trình liên lạc giữa máy điện thoại và trạm thu phát sóng vô tuyến. Đơn vị mã hóa của A5/1 là một bit. Bộ sinh số mỗi lần sẽ sinh ra hoặc bit 0 hoặc bit 1 để sử dụng trong phép XOR.
- **RC4**: dùng trong giao thức **SSL (Secure Sockets Layer)** để bảo mật dữ liệu trong quá trình truyền dữ liệu giữa Web Server và trình duyệt Web; hoặc trong mã hóa WEP của mạng Wireless LAN.



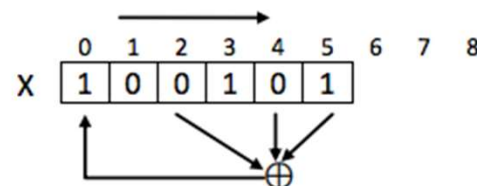
Tiny-A5/1

Bộ sinh số gồm 3 thanh ghi X, Y, Z. Thanh ghi X gồm 6 bit, ký hiệu là (x_0, x_1, \dots, x_5) . Thanh ghi Y gồm 8 bit (y_0, y_1, \dots, y_7) . Thanh ghi Z lưu 9 bit (z_0, z_1, \dots, z_8) . Khóa K ban đầu có chiều dài 23 bit và lần lượt được phân bổ vào các thanh ghi: **K \rightarrow XYZ**. Các thanh ghi X, Y, Z được biến đổi theo 3 quy tắc:

1) **Quay X** gồm các thao tác:

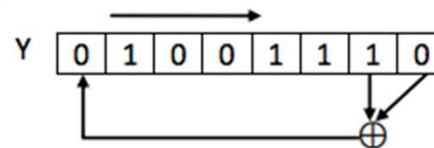
- $t = x_2 \oplus x_4 \oplus x_5$
- $x_j = x_{j-1}$ với $j = 5, 4, 3, 2, 1$
- $x_0 = t$

Ví dụ: giả sử X là 100101, dẫn đến $t = 0 \oplus 0 \oplus 1 = 1$, vậy sau khi quay giá trị của X là 110010.



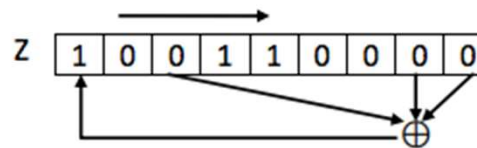
2) **Quay Y**: tương tự như quay X, quay Y là như sau:

- $t = y_6 \oplus y_7$
- $y_j = y_{j-1}$ với $j = 7, 6, 5, \dots, 1$
- $y_0 = t$



3) **Quay Z**:

- $t = z_2 \oplus z_7 \oplus z_8$
- $z_j = z_{j-1}$ với $j = 8, 7, 6, \dots, 1$
- $z_0 = t$



Hàm $maj(x, y, z)$ nếu trong 3 bit x, y, z có từ hai bit 0 trở lên thì hàm trả về giá trị 0, nếu không hàm trả về giá trị 1

Sinh khoá s_i :

$$m = maj(x_1, y_3, z_3)$$

If $x_1 = m$ then thực hiện quay X

If $y_3 = m$ then thực hiện quay Y

If $z_3 = m$ then thực hiện quay Z

Và bit được sinh ra là:

$$s_i = x_5 \oplus y_7 \oplus z_8$$

Tiny-A5/1 – Ví dụ

Ví dụ: mã hóa bản rõ P=111 (chữ h) với khóa K là 100101. 01001110.100110000.

Ban đầu giá trị của các thanh ghi X, Y, Z là:

$$X = 1\underline{0}0101$$

$$Y = 010\underline{0}1110$$

$$Z = 100\underline{1}10000$$

Bước 0: $x_1=0, y_3=0, z_3=1 \rightarrow m=0 \rightarrow$ quay X, quay Y

$$X = 1\underline{1}0010$$

$$Y = 101\underline{0}0111 \rightarrow s_0 = 0 \oplus 1 \oplus 0 = 1$$

$$Z = 100\underline{1}10000$$

Bước 1: $x_1=1, y_3=0, z_3=1 \rightarrow m=1 \rightarrow$ quay X, quay Z

$$X = 1\underline{1}1001$$

$$Y = 101\underline{0}0111 \rightarrow s_1 = 1 \oplus 1 \oplus 0 = 0$$

$$Z = 010\underline{0}11000$$

Bước 2: $x_1=1, y_3=0, z_3=0 \rightarrow m=0 \rightarrow$ quay Y, quay Z

$$X = 111001$$

$$Y = 01010011 \rightarrow s_2 = 1 \oplus 1 \oplus 0 = 0$$

$$Z = 001001100$$

Vậy bản mã là $C = 111 \oplus 100 = 011$ (chữ D)

A5/1

Về nguyên tắc bộ sinh số A5/1 hoạt động giống như TinyA5/1. Kích thước thanh ghi X, Y, Z lần lượt là 19, 22 và 23 bit. Các bước quay X, Y, Z cụ thể như sau:

1) Quay X:

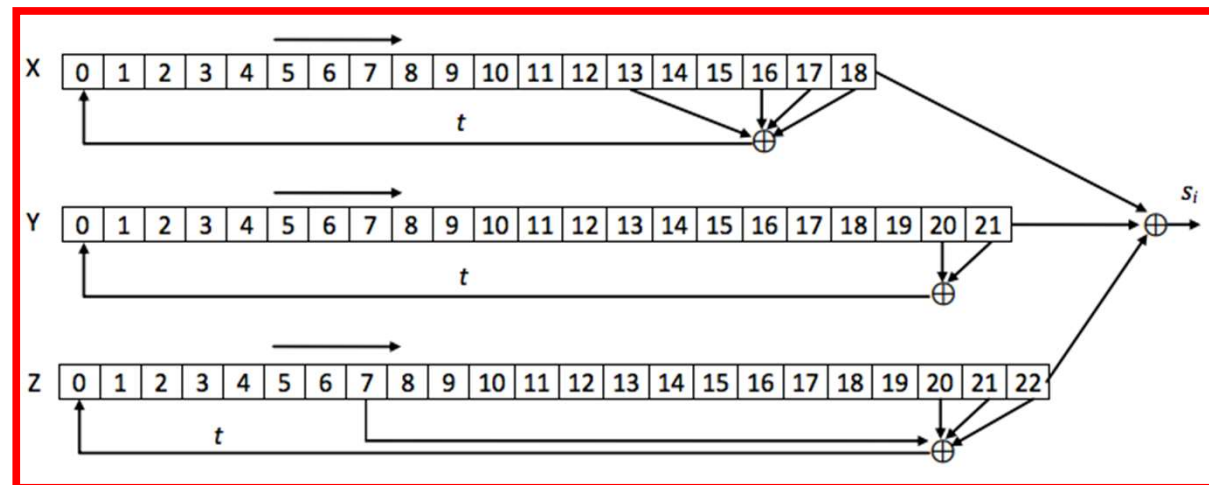
- $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
- $x_j = x_{j-1}$ với $j = 18, 17, 16, \dots, 1$
- $x_0 = t$

2) Quay Y:

- $t = y_{20} \oplus y_{21}$
- $y_j = y_{j-1}$ với $j = 21, 20, 19, \dots, 1$
- $y_0 = t$

3) Quay Z:

- $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
- $z_j = z_{j-1}$ với $j = 22, 21, 20, \dots, 1$
- $z_0 = t$



Hàm maj được tính trên 3 bit x_8, y_{10}, z_{10} . Sau khi quay xong bit sinh ra là:

$$s_i = x_{18} \oplus y_{21} \oplus z_{22}$$


Bài tập:

1. Viết chương trình bằng Python mô phỏng hệ mã hoá A5/1
2. Trình bày hệ mã hoá Tiny-RC4 và RC4
3. Viết chương trình bằng Python mô phỏng hệ mã hoá RC4

3.3. Mã khối (block cipher)

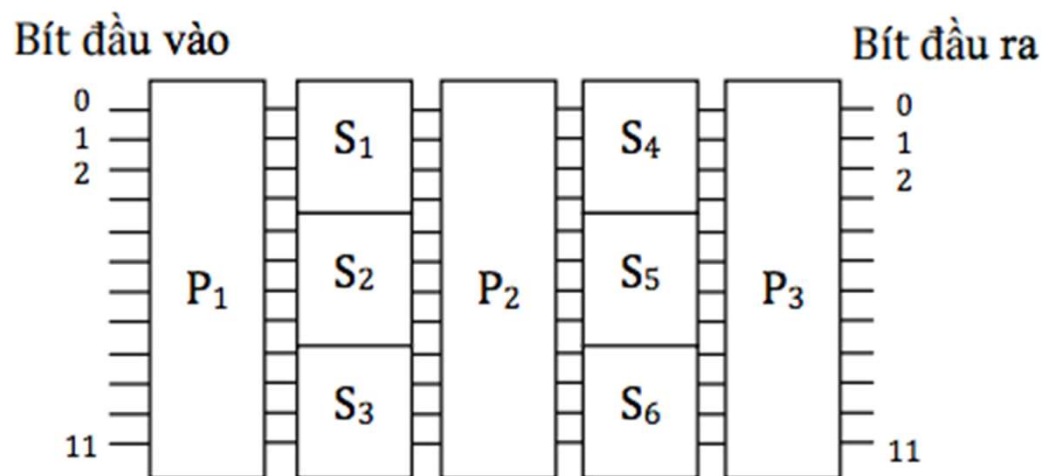
bản rõ: 1111 0000 0011 (head)
khóa: 0101 0101 0101
bản mã: 1010 0101 0110 (FBCG)

Nếu biết bản mã $c_0 = 1010$ có bản rõ tương ứng là $p_0 = 1111$, thì có thể dễ dàng suy ra khóa là 0101. Nói một cách tổng quát, nếu giữa bản rõ P và bản mã C có mối liên hệ toán học thì việc biết một số cặp bản rõ-bản mã giúp ta có thể tính được khóa K . (như trong trường hợp mã Hill)

Nếu chọn kích thước của khối là 64 bit thì số dòng
của bảng khóa là 2^{64}  ***mã khối an toàn lý tưởng***

Mạng Substitution-Permutation Network (SPN)

- Kết hợp hai hay nhiều mã hóa đơn giản lại với nhau để tạo thành một mã hóa tích hợp (**product cipher**)
- Các mã hóa đơn giản thường là phép thay thế (**substitution, S-box**) và hoán vị (**Permutation, P-box**)
- Mã tích hợp thường được gọi là **mạng SPN**.



Mã Feistel

- là một dạng tiếp cận khác so với mạng SPN, do **Horst Feistel** đề xuất năm 1973, cũng là sự kết hợp các phép thay thế và hoán vị: bản rõ sẽ được biến đổi qua một số vòng để cho ra bản mã cuối cùng

$$P \xrightarrow{K_1} C_1 \xrightarrow{K_2} C_2 \xrightarrow{K_3} \dots \xrightarrow{K_{n-1}} C_n$$

Trong đó bản rõ P và các bản mã C_i được chia thành nửa trái và nửa phải:

$$P = (L_0, R_0)$$

$$C_i = (L_i, R_i) \quad i = 1, 2, \dots, n$$

Quy tắc biến đổi các nửa trái phải này qua các vòng được thực hiện như sau:

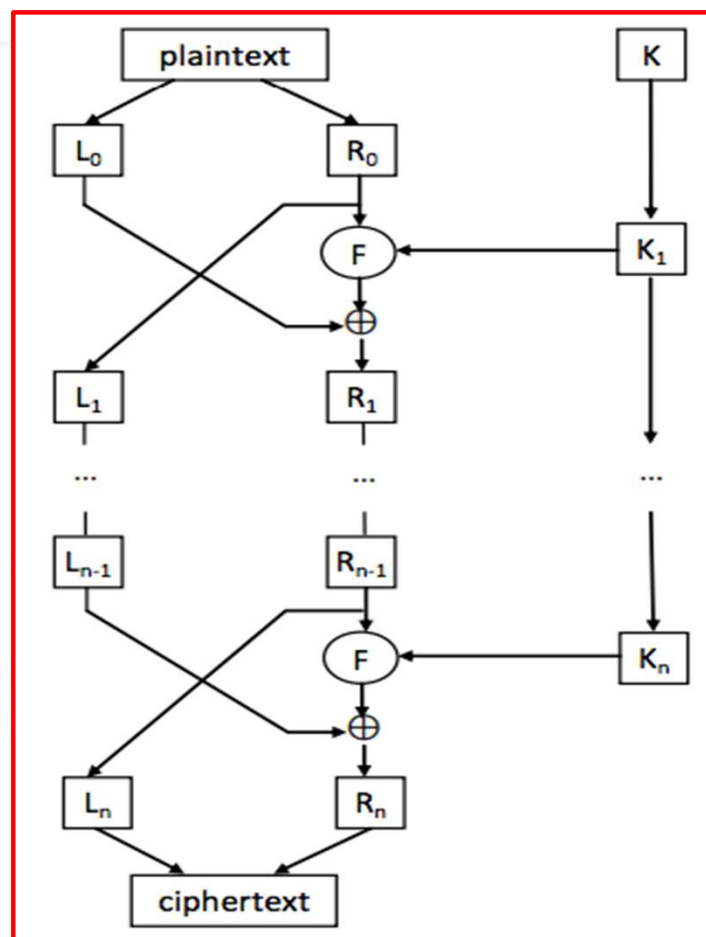
$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

K_i là một khóa con cho vòng thứ i . Khóa con này được sinh ra từ khóa K ban đầu theo một thuật toán sinh khóa con (key schedule): $K \rightarrow K_1 \rightarrow K_2 \rightarrow \dots \rightarrow K_n$

F là một hàm mã hóa dùng chung cho tất cả các vòng. Hàm F đóng vai trò như là phép thay thế còn việc hoán đổi các nửa trái phải có vai trò hoán vị. Bản mã C được tính từ kết xuất của vòng cuối cùng:

$$C = C_n = (L_n, R_n)$$



Mã Feistel

Đặc điểm của hệ mã Feistel:

- Hàm F và thuật toán sinh khóa con càng phức tạp thì càng khó phá mã.
- Ứng với các hàm F và thuật toán sinh khóa con khác nhau thì ta sẽ có các phương pháp mã hóa khác nhau: **DES** (Data Encryption Standard)

Mã DES (Data Encryption Standard)

Mã hoá Lucifer (IBM)

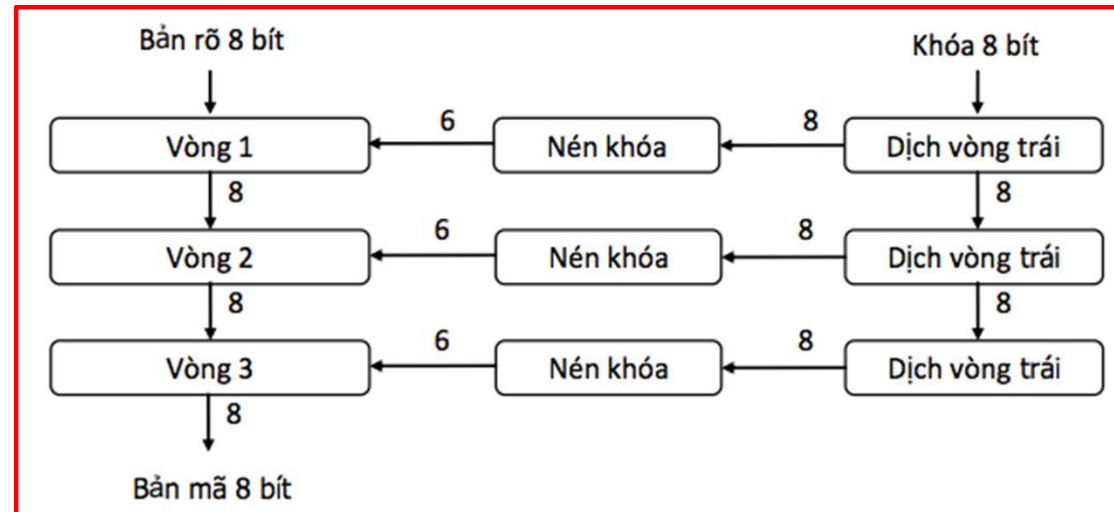
Sử dụng nguyên tắc của hệ mã Feistel

Cục tiêu chuẩn
quốc gia Mỹ

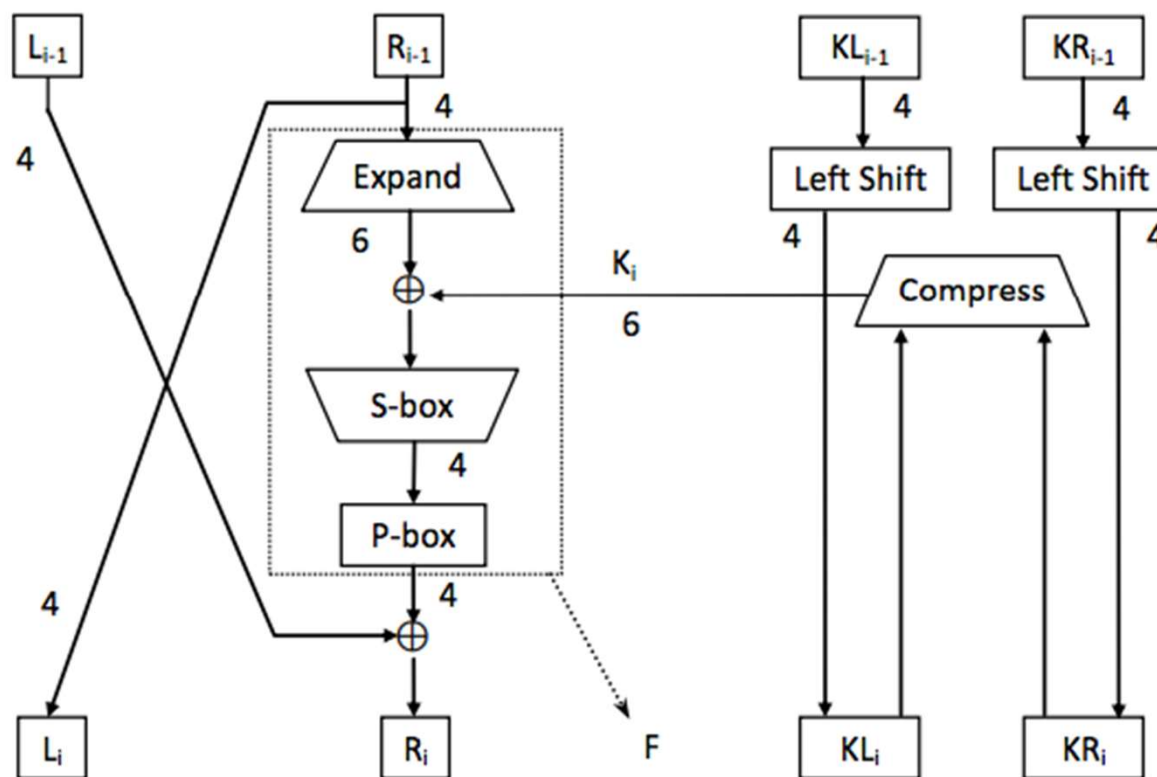
Mã hoá DES (1973-1974)

Mã **Tiny-DES** có các tính chất sau:

- Là mã thuộc hệ mã Feistel gồm **3 vòng**
- Kích thước của khối là **8 bit**.
- Kích thước khóa là **8 bit**
- Mỗi vòng của DES dùng khóa con có kích thước **6 bit** được trích ra từ khóa chính



Cấu tạo một vòng của Tiny-DES



$$F(R_{i-1}, K_i) = P\text{-box}(S\text{-box}(Expand(R_{i-1}) \oplus K_i))$$

$$F(R_{i-1}, K_i) = P\text{-box}(S\text{-box}(\text{Expand}(R_{i-1}) \oplus K_i))$$

- *Expand*: gọi 4 bit của R_{i-1} là $b_0b_1b_2b_3$. Hàm *Expand* hoán vị và mở rộng 4 bit thành 6 bit cho ra kết quả: $b_2b_3b_1b_2b_1b_0$.
Ví dụ: $R_0 = 0110 \Rightarrow \text{Expand}(R_0) = 101110$
- *S-box*: Gọi $b_0b_1b_2b_3b_4b_5$ là 6 bit đầu vào của *S-box*, ứng với mỗi trường hợp của 6 bit đầu vào sẽ có 4 bit đầu ra. Việc tính các bit đầu ra dựa trên bảng sau:

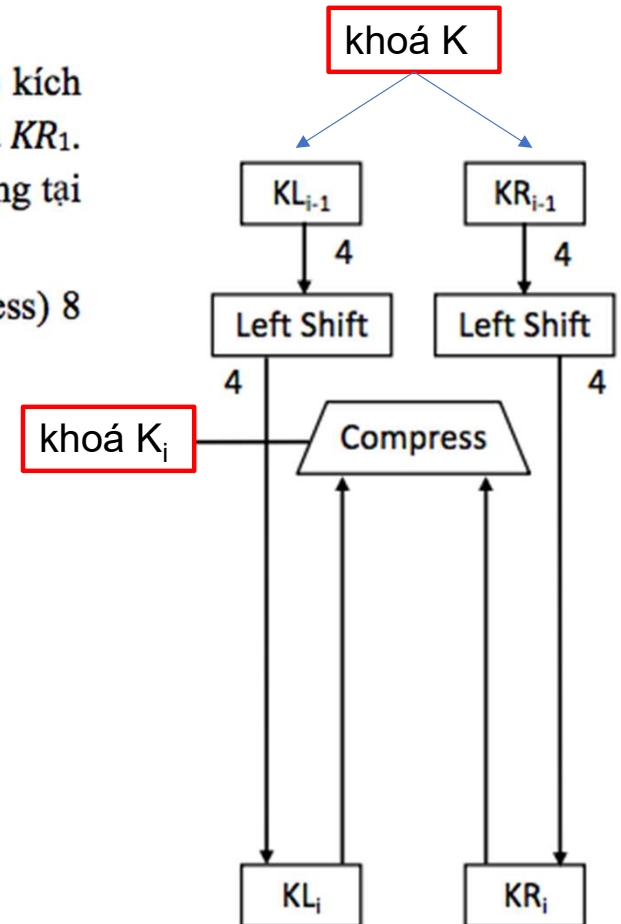
		$b_1b_2 \ b_3b_4$															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
b_0b_5	00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
	01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
	10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
	11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Ví dụ: $X = 101010$. Tra bảng ta có $S\text{-box}(X) = 0110$.

Thuật toán sinh khoá con của Tiny-DES

Khóa K 8 bit ban đầu được chia thành 2 nửa trái phải KL_0 và KR_0 , mỗi nửa có kích thước 4 bit. Tại vòng thứ nhất KL_0 và KR_0 được dịch vòng trái 1 bit để có được KL_1 và KR_1 . Tại vòng thứ hai KL_1 và KR_1 được dịch vòng trái 2 bit để có được KL_2 và KR_2 . Tại vòng tại vòng thứ 3 KL_2 và KR_2 được dịch vòng trái 1 bit để có KL_3 và KR_3 .

Cuối cùng khóa K_i của mỗi vòng được tạo ra bằng cách hoán vị và nén (compress) 8 bit của KL_i và KR_i ($k_0k_1k_2k_3k_4k_5k_6k_7$) thành kết quả gồm 6 bit : $k_5k_1k_3k_2k_7k_0$.



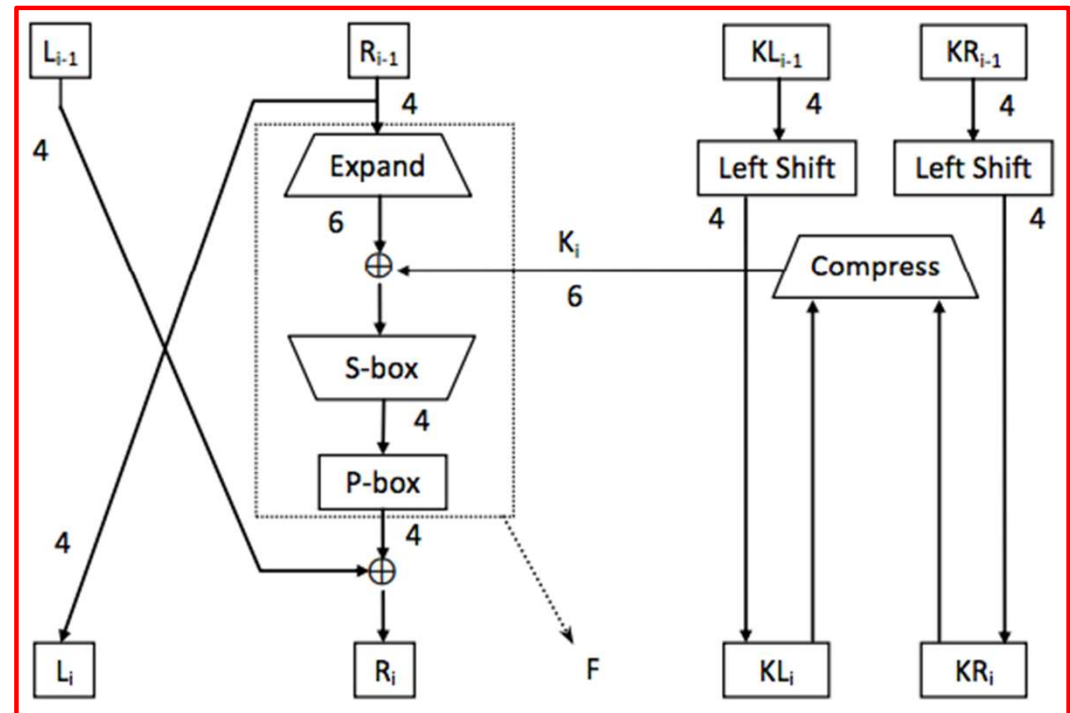
Tiny-DES

Ví dụ: mã hóa bản rõ $P = 0101.1100$ với khóa $K = 1001.1010$

- $L_0 = 0101, R_0 = 1100, KL_0 = 1001, KR_0 = 1010$

• Vòng 1:

- $L_1 = R_0 = 1100, \text{Expand}(R_0) = 001011$
- $KL_1 = KL_0 \ll 1 = 0011, KR_1 = KR_0 \ll 1 = 0101$
- $K_1 = \text{Compress}(KL_1KR_1) = 101110$
- $\text{Expand}(R_0) \oplus K_1 = 100101$
- $S\text{-box}(100101) = 1000$
- $F_1 = P\text{-box}(1000) = 0100$
- $R_1 = L_0 \oplus F_1 = 0001$



Tiny-DES Ví dụ: mã hóa bản rõ $P = 0101.1100$ với khóa $K = 1001.1010$

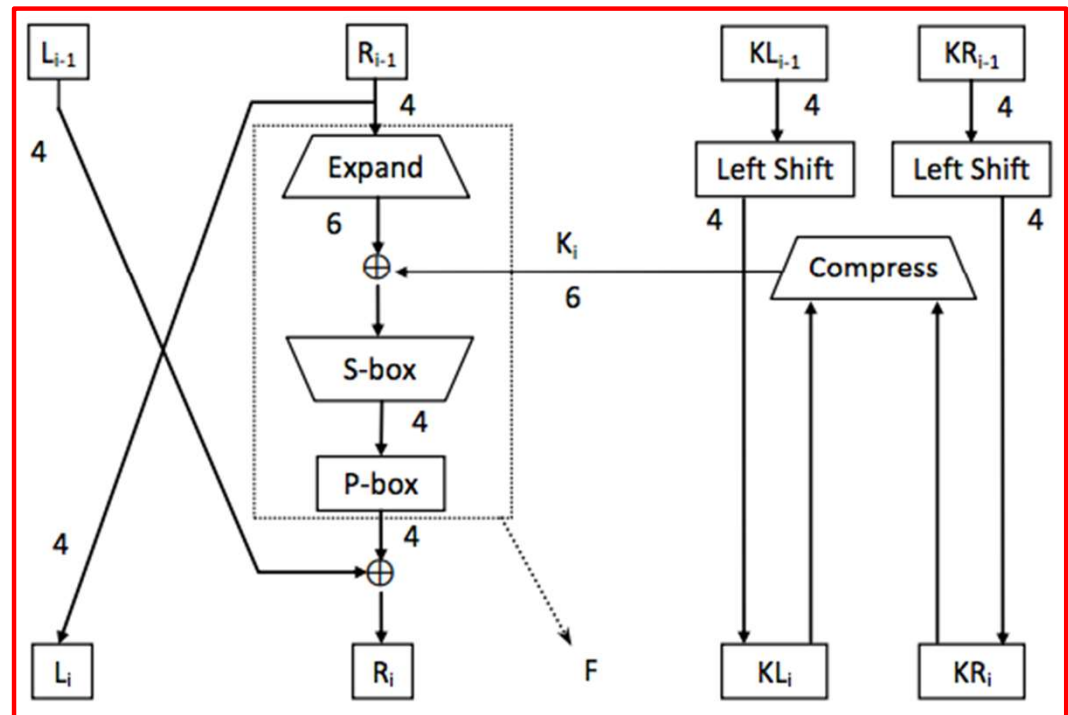
- $L_0 = 0101, R_0 = 1100, KL_0 = 1001, KR_0 = 1010$

• Vòng 1:

- $L_1 = R_0 = 1100, \text{Expand}(R_0) = 001011$
- $KL_1 = KL_0 \ll 1 = 0011, KR_1 = KR_0 \ll 1 = 0101$
- $K_1 = \text{Compress}(KL_1KR_1) = 101110$
- $\text{Expand}(R_0) \oplus K_1 = 100101$
- $S\text{-box}(100101) = 1000$
- $F_1 = P\text{-box}(1000) = 0100$
- $R_1 = L_0 \oplus F_1 = 0001$

• Vòng 2:

- $L_2 = R_1 = 0001, \text{Expand}(R_1) = 010000$
- $KL_2 = KL_1 \ll 2 = 1100, KR_2 = KR_1 \ll 2 = 0101$
- $K_2 = \text{Compress}(KL_2KR_2) = 110011$
- $\text{Expand}(R_1) \oplus K_2 = 100011$
- $S\text{-box}(100011) = 1100$
- $F_2 = P\text{-box}(1100) = 0101$
- $R_2 = L_1 \oplus F_2 = 1001$



Tiny-DES Ví dụ: mã hóa bản rõ $P = 0101.1100$ với khóa $K = 1001.1010$

- $L_0 = 0101, R_0 = 1100, KL_0 = 1001, KR_0 = 1010$

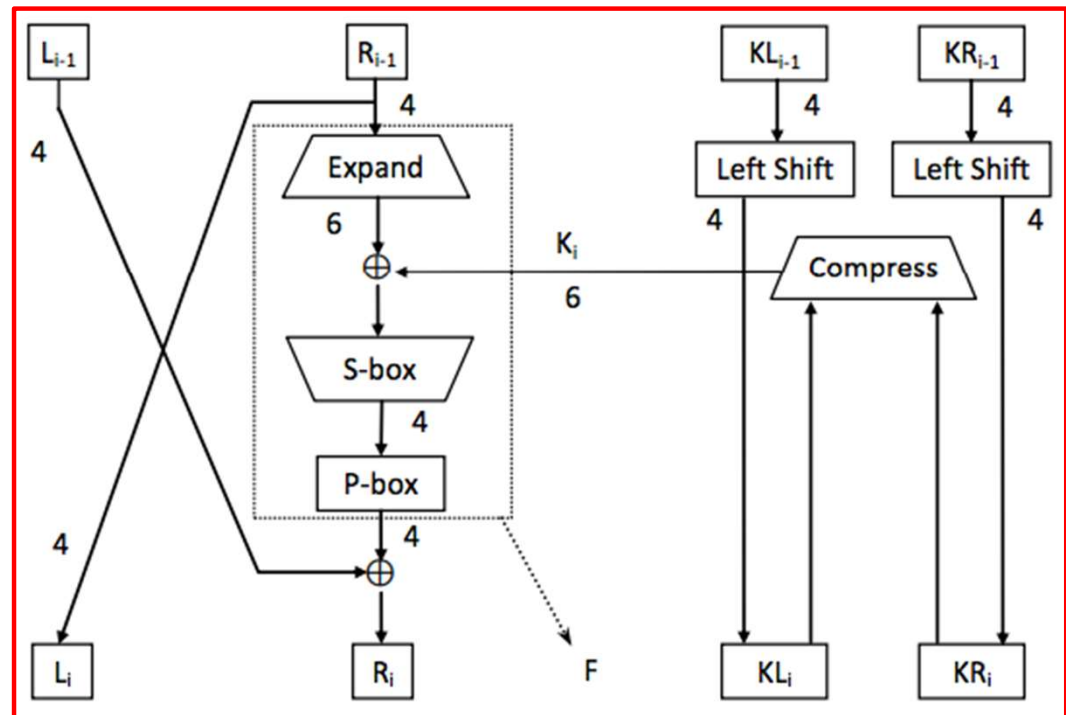
• **Vòng 2:**

- $L_2 = R_1 = 0001, \text{Expand}(R_1) = 010000$
- $KL_2 = KL_1 \ll 2 = 1100, KR_2 = KR_1 \ll 2 = 0101$
- $K_2 = \text{Compress}(KL_2KR_2) = 110011$
- $\text{Expand}(R_1) \oplus K_2 = 100011$
- $S\text{-box}(100011) = 1100$
- $F_2 = P\text{-box}(1100) = 0101$
- $R_2 = L_1 \oplus F_2 = 1001$

• **Vòng 3:**

- $L_3 = R_2 = 1001, \text{Expand}(R_2) = 010001$
- $KL_3 = KL_2 \ll 1 = 1001, KR_3 = KR_2 \ll 1 = 1010$
- $K_3 = \text{Compress}(KL_3KR_3) = 001001$
- $\text{Expand}(R_2) \oplus K_3 = 011000$
- $S\text{-box}(011000) = 0101$
- $F_3 = P\text{-box}(0101) = 0011$
- $R_3 = L_2 \oplus F_3 = 0010$

- Kết quả $C = L_3R_3 = 1001.0010$ (hệ thập lục phân: 92)



DES (Data Encryption Standard)

Mã **DES** có các tính chất sau:

- Là mã thuộc hệ mã Feistel gồm **16 vòng**, ngoài ra DES có thêm một hoán vị khởi tạo trước khi vào vòng 1 và một hoán vị khởi tạo sau vòng 16
- Kích thước của khối là **64 bit**: ví dụ bản tin “***meetmeafterthetogaparty***” biểu diễn theo mã ASCII thì mã DES sẽ mã hóa làm 3 lần, mỗi lần 8 chữ cái (64 bit):

meetmeaf - tertheto - gaparty.

- Kích thước khóa là **56 bit**
- Mỗi vòng của DES dùng khóa con có kích thước **48 bit** được trích ra từ khóa chính

Bài tập:

1. Viết chương trình bằng Python mô phỏng hệ mã hoá Tiny-DES
2. Trình bày hệ mã hoá DES