

TRUYỀN VÀ BẢO MẬT THÔNG TIN

Bài 5:

Mã hóa đối xứng hiện đại (phần II)

VŨ THỊ TRÀ

©2020 ĐH Sư Phạm – ĐH Đà Nẵng

Nội dung

- Bản tin, mã ASCII, biểu diễn nhị phân

- Mã dòng (Stream Cipher): A5/1 + RC4

- Mã khối (Block Cipher)

- Mã DES (Data Encryption Standard)

- Mã AES (Advanced Encryption Standard);
Các mô hình ứng dụng của mã khối

- Một số thuộc tính hệ MHĐXHĐ; Trao đổi
khóa bí mật bằng TT phân phối khóa KDC

Mã khối

Mã khối (Block Cipher)

- ❖ **Mã khối an toàn lý tưởng:** để phá mã (tìm được khóa) phải biết được **tất cả** các cặp **bản mã và bản rõ**; nếu kích cỡ của khối là 4 thì số dòng trong bảng khóa là $2^4=16$.
- ❖ **Mạng SP:** Shannon (1946) chỉ ra cách sử dụng một khóa có kích thước ngắn dựa trên sự kết hợp phép thay thế (**substitute, S-box**) và phép hoán vị (**permutation, P-box**)
- ❖ **Mô hình mã Feistel:** là sự kết hợp của phép thay thế và hoán vị, một cách tiếp cận khác của mạng SP; bản rõ sẽ được biến đổi qua một số vòng để cho ra bản mã cuối cùng.

Mã khối an toàn lý tưởng

- **Xét lại VD: Bảng chữ cái & Hệ nhị phân.** Giả sử dùng một **khóa K** 4 bit **0101** để mã hóa bản rõ là "**head**" bằng phép XOR:

Bản rõ:	1111 0000 0011	(head)
Khóa:	0101 0101 0101	
Bản mã:	1010 0101 0110	(FBCG)

Ví dụ về bảng tra cứu ngẫu nhiên

Bản rõ	Bản mã
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

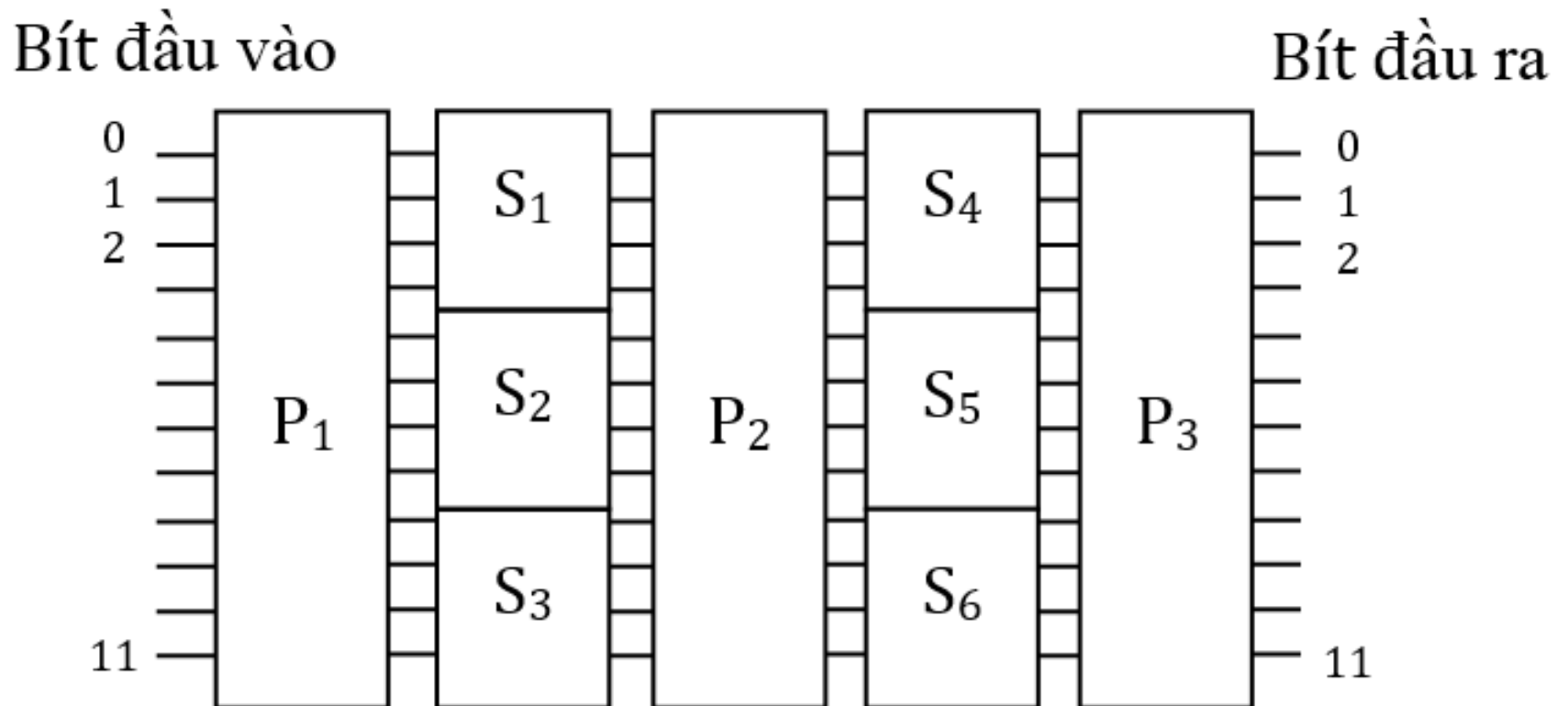
Đặc tính mã khối an toàn lý tưởng

- Nếu chọn kích thước của khối là **64 bit** thì số dòng của bảng khóa là 2^{64} , một con số rất lớn (và có khoảng 2^{64} ! bảng khóa như vậy). Lúc này việc nắm **tất cả các cặp bản rõ-bản mã** của bảng khóa là điều **không thể** đối với người phá mã. Trường hợp này ta gọi là **mã khối an toàn lý tưởng**.
- Khi kích thước **khối lớn** thì số dòng của bảng khóa cũng lớn và **gây trở ngại** cho việc **lưu trữ** cũng như **trao đổi khóa** giữa người gửi và người nhận. Bảng khóa có 2^{64} dòng mỗi dòng **2x64 bit** do đó kích thước khóa sẽ là **$2 \times 64 \times 2^{64} = 2^{71} \approx 2,36 \times 10^{21}$ bit**. Do đó mã khối an toàn lý tưởng là **không khả thi** trong thực tế.

SPN (Substitution-Permutation Network)

- **Nhu cầu:** cần dùng một khóa có **kích thước ngắn** để giả lập một bảng tra cứu có độ an toàn xấp xỉ **độ an toàn của mã khối lý tưởng**. Cách thực hiện là **kết hợp** hai hay nhiều **mã hóa đơn giản** lại với nhau để tạo thành một **mã hóa tổng (Product cipher)**, trong đó mã hóa tổng an toàn hơn rất nhiều so với các mã hóa thành phần. Các mã hóa đơn giản thường dùng là **phép thay thế (Substitution, S-box)** và **hoán vị (Permutation, P-box)**. Do đó người ta hay gọi mã hóa tổng là **Substitution-Permutation Network (SPN hay mạng SP)**.

Mô hình SPN



Mạng SP

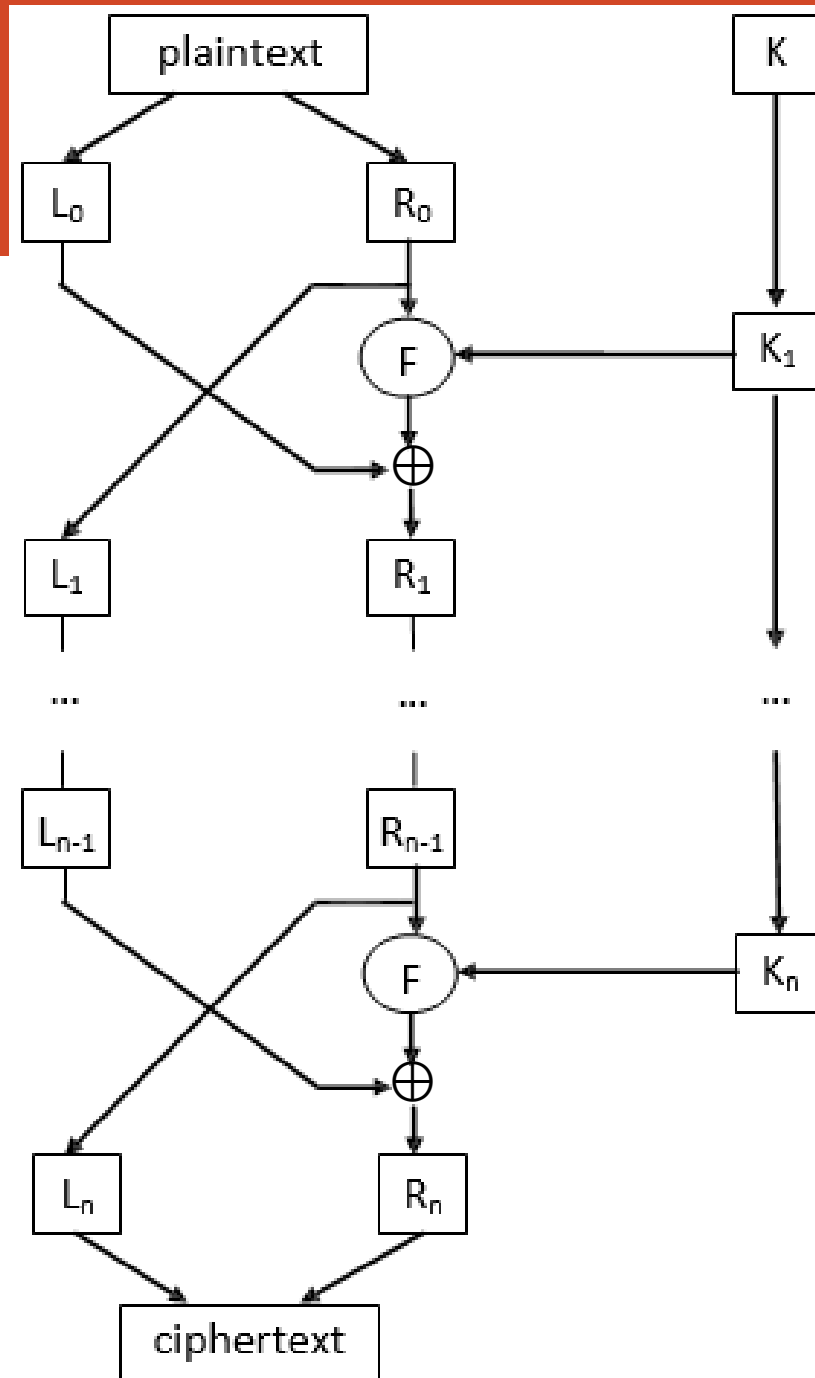
Việc kết hợp các S-box và P-box tạo ra hai tính chất quan trọng của mã hóa là **tính khuếch tán** (**diffusion**) và **tính gây lẫn** (**confusion**). Hai tính chất này do **Claude Shannon** giới thiệu vào năm **1946** là **cơ sở** của tất cả các mã khối hiện nay.

- **Tính khuếch tán:** dựa vào sử dụng P-box kết hợp S-box, **một bit** của bản rõ tác động đến **tất cả các bit** của **bản mã** → làm **giảm tối đa** mối **liên quan** giữa bản rõ và bản mã → ngăn chặn việc suy ra lại khóa.
- **Tính gây lẫn:** dựa vào sử dụng S-box làm **phức tạp hóa** mối liên quan giữa bản mã và khóa → ngăn chặn việc suy ra lại khóa.

Mô hình mã Feistel

Mô hình mã Feistel do **Horst Feistel** đề xuất là một dạng tiếp cận khác so với mạng SP

- Hàm **F** đóng vai trò **thay thế**
- Các **L_i** , **R_i** đóng vai trò **hoán vị**
- Khóa con **K_i** được sinh ra từ khóa **K** ban đầu theo luật sinh khóa **$K \rightarrow K_1 \rightarrow \dots \rightarrow K_n$**



Lập mã theo mô hình mã Feistel

Trong hệ mã Feistel, **bản rõ** sẽ được biến đổi **qua một số vòng** để cho ra **bản mã** cuối cùng:

$$P \xrightarrow{K_1} C_1 \xrightarrow{K_2} C_2 \xrightarrow{K_3} \dots \xrightarrow{K_n} C_n$$

- 1) Bản rõ **P** và bản mã **C_i** được chia thành 2 nửa trái & phải:

$$P = (L_0, R_0)$$

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

- 2) Biến đổi nửa trái, nửa phải qua các vòng được thực hiện như sau:

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

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

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

Giải mã theo mô hình mã Feistel

Quá trình được thực hiện qua các vòng theo thứ tự ngược lại:

$$\mathbf{C} \rightarrow (\mathbf{L}_n, \mathbf{R}_n) \quad i = n, n-1, \dots, 1$$

$$\mathbf{R}_{i-1} = \mathbf{L}_i \quad \text{thuận } \mathbf{L}_i = \mathbf{R}_{i-1}$$

$$\mathbf{L}_{i-1} = \mathbf{R}_i \oplus \mathbf{F}(\mathbf{R}_{i-1}, \mathbf{K}_i) \quad \text{thuận } \mathbf{R}_i = \mathbf{L}_{i-1} \oplus \mathbf{F}(\mathbf{R}_{i-1}, \mathbf{K}_i)$$

$$\mathbf{P} = (\mathbf{L}_0, \mathbf{R}_0)$$

Đặc tính của hệ mã Feistel

- Hệ mã Feistel chia các bản mã thành **hai nửa trái phải** giúp cho hàm F không cần **khả nghịch** (không cần có F^{-1}). Mã hóa và giải mã đều dùng một hàm F . Hàm F và thuật toán **sinh khóa** càng phức tạp thì càng khó phá mã.
- Ứng với các **hàm F** và thuật toán **sinh khóa** khác nhau thì ta sẽ có các phương pháp **mã hóa** khác nhau, phần tiếp theo sẽ trình bày **mã hóa DES**, phương pháp mã hóa dựa trên nguyên tắc của hệ mã Feistel.

Mã DES

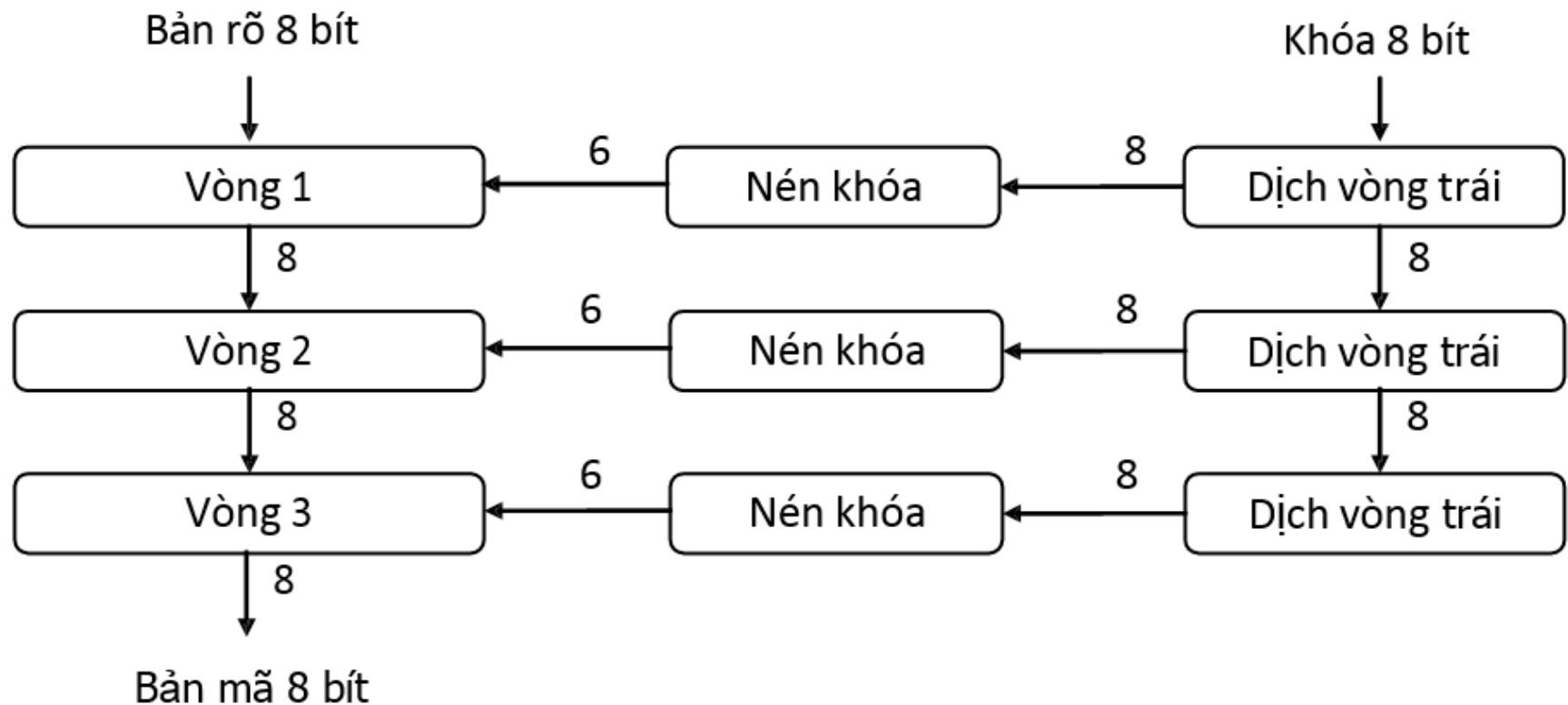
DES (Data Encryption Standard)

- Năm 1973, khi nhu cầu ứng dụng bảo mật vào các mục đích dân sự được đặt ra, Cục tiêu chuẩn quốc gia Hoa Kỳ kêu gọi các công ty Mỹ thiết lập một chuẩn mã hóa quốc gia. Mã hóa Lucifer của công ty IBM được chọn và sau một vài sửa đổi của cơ quan an ninh Hoa Kỳ, mã hóa Lucifer đã trở thành mã tiêu chuẩn DES.
- Qua quá trình sử dụng mã DES đã chứng tỏ độ an toàn cao và được sử dụng rộng rãi.

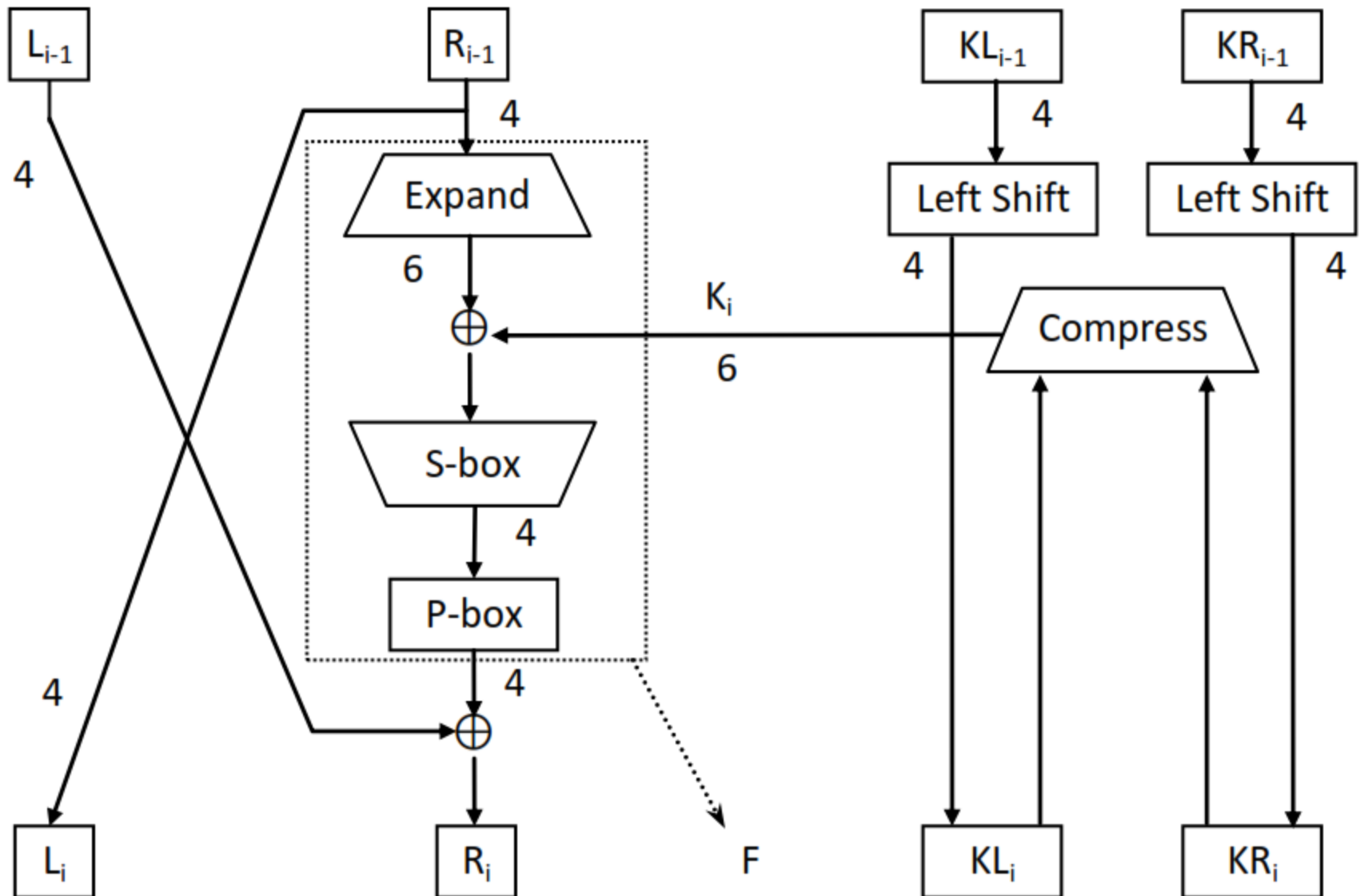
TinyDES

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

Các vòng Feistel của mã TinyDES



Cấu trúc một vòng của mã TinyDES



TinyDES

Hàm F của Feistel là:

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

Trong đó, hàm

- **Expand:** thực hiện hoán vị và mở rộng 4 bit vào thành 6 bit ra. Nếu $R_{i-1} = b_0b_1b_2b_3$ thì $\mathit{Expand}(R_{i-1}) = b_2b_3b_1b_2b_1b_0$
- **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 theo *Bảng S-box*
- **P-box:** thực hiện hoán vị 4 bit $b_0b_1b_2b_3$ cho ra kết quả $b_2b_0b_3b_1$

TinyDES:: Bảng S-box

Hệ nhị phân

		$b_1b_2b_3b_4$															
		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

Hệ thập lục phân

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
	1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
	2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
	3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

TinyDES:: Thuật toán sinh khóa con

- 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 **1**, KL_0 và KR_0 được dịch vòng trái **1 bit** để có được KL_1 và KR_1 .
- Tại vòng thứ **2**, KL_1 và KR_1 được dịch vòng trái **2 bit** để có được KL_2 và KR_2 .
- Tại vòng thứ **3**, KL_2 và KR_2 được dịch vòng trái **1 bit** để có KL_3 và KR_3 .
- Khóa K_i tại 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** : $k_5k_1k_3k_2k_7k_0$

TinyDES:: Ví dụ

Mã hóa bản rõ $P = 0101.1100$ (5C) với khóa $K = 1001.1010$ (9A)

- **Bước 0:**

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

- **Bước 1:** Vòng 1 Feistel

1) $L_1 = R_0 = 1100 \rightarrow \text{Expand}(R_0) = 001011$

2) $KL_1 = KL_0 \ll 1 = 0011, KR_1 = KR_0 \ll 1 = 0101$

$\rightarrow K_1 = \text{Compress}(KL_1, KR_1) = 101110$

3) $\text{Expand}(R_0) \oplus K_1 = 100101$

4) $S\text{-box}(100101) = 1000$

5) $F_1 = P\text{-box}(1000) = 0100$

6) $R_1 = L_0 \oplus F_1 = 0001$

TinyDES:: Ví dụ

- **Bước 2:** Vòng 2 Feistel

1) $L_2 = R_1 = 0001 \rightarrow \text{Expand}(R_1) = 010000$

2) $KL_2 = KL_1 \ll 2 = 1100, KR_2 = KR_1 \ll 2 = 0101$

$\rightarrow K_2 = \text{Compress}(KL_2, KR_2) = 110011$

3) $\text{Expand}(R_1) \oplus K_2 = 100011$

4) $S\text{-box}(100011) = 1100$

5) $F_2 = P\text{-box}(1100) = 0101$

6) $R_2 = L_1 \oplus F_2 = 1001$

TinyDES:: Ví dụ

- **Bước 3:** Vòng 3 Feistel

1) $L_3 = R_2 = 1001 \rightarrow \text{Expand}(R_2) = 010001$

2) $KL_3 = KL_2 \ll 1 = 1001, KR_3 = KR_2 \ll 1 = 1010$

$\rightarrow K_3 = \text{Compress}(KL_3, KR_3) = 001001$

3) $\text{Expand}(R_2) \oplus K_3 = 011000$

4) $S\text{-box}(011000) = 0101$

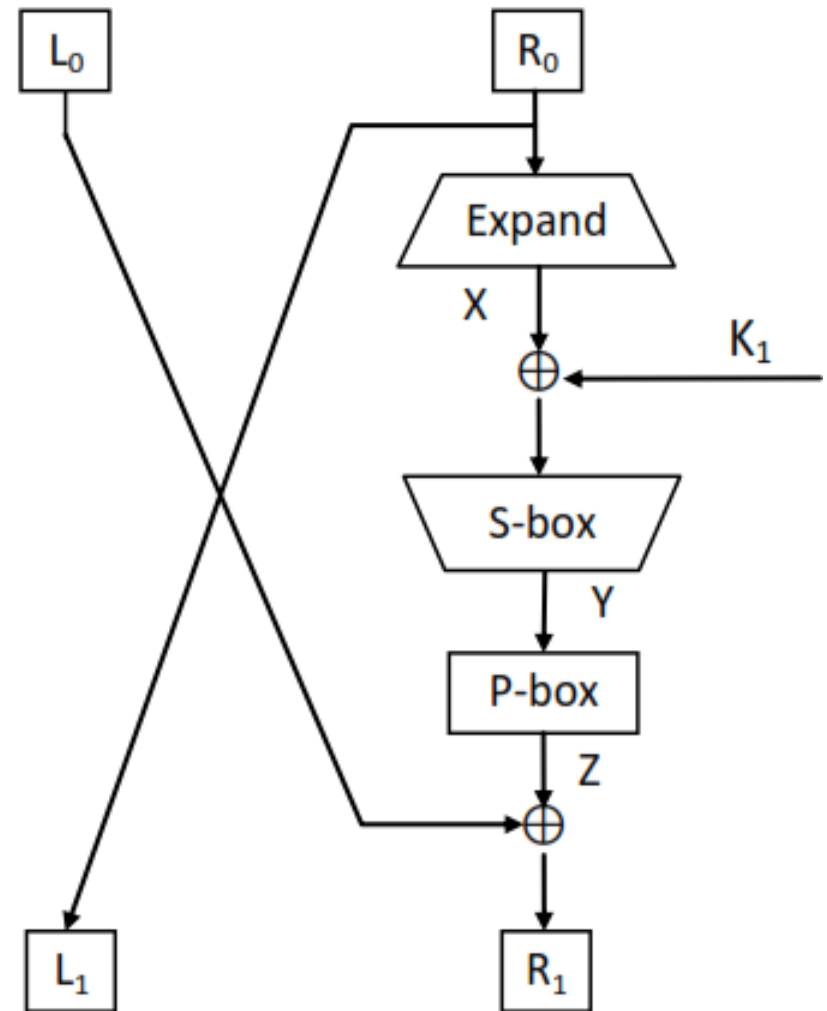
5) $F_3 = P\text{-box}(0101) = 0011$

6) $R_3 = L_2 \oplus F_3 = 0010$

- **Kết quả:** $C = L_3 R_3 = 1001.0010$ (92)

TinyDES:: Mã thám

- Xét trường hợp TinyDES chỉ có **1 vòng Feistel**. Khi đó, $C = (L_1, R_1)$ và $P = (L_0, R_0)$



TinyDES:: Mã thám

Trong trường hợp này người phá mã biết P và C , tuy nhiên không biết K . Giả sử $P = 0101.1100$ và $C = 1100.0001$.

Người phá mã tiến hành tính K như sau:

- Từ R_0 tính $X = 001011$.
- Từ L_0 và R_1 tính $Z = 0100$, và từ Z tính $Y = 1000$.
- Tra cứu bảng S-box với đầu ra là 1000, ta xác định được các đầu ra X và K_1 có thể xảy ra là: $\{100101, 100111, 001110, 011111\}$
- Thử tiếp với 1 vài cặp bản rõ-bản mã khác ta sẽ tìm được $K_1 = 101110$.
- Từ K_1 để tìm được $K = 1001.1010$?

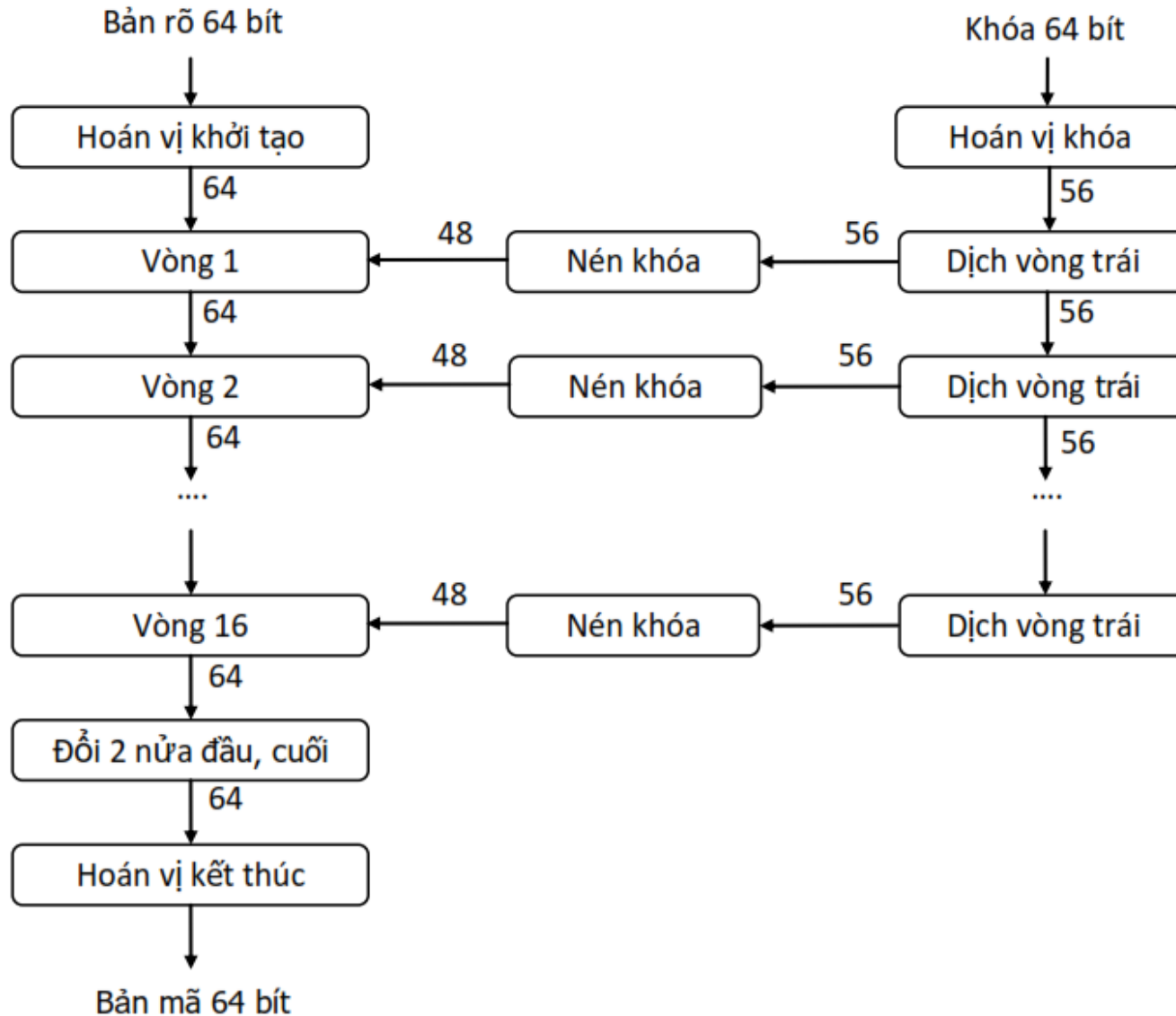
TinyDES:: Mã thám

- Tuy nhiên với mã **TinyDES ba vòng**, việc phá mã không còn đơn giản, người phá mã chỉ biết được input của vòng đầu là P và output của vòng cuối là C , giá trị trung gian L_1R_1, L_2R_2 bị ẩn giấu nên không thể giới hạn miền tìm kiếm của các khóa con K_1, K_2, K_3
- Dưới tác động của S -box, việc thay đổi 1 bit trong bản rõ hoặc khóa K sẽ ảnh hưởng đến nhiều bit khác nhau trong các giá trị trung gian L_1R_1, L_2R_2 , nên khó phân tích mối liên quan giữa bản rõ, bản mã và khóa

Mã DES

- 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 kết thúc 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 gốc 64 bit được trích rút khởi tạo thành khóa chính 56 bit.
- Mỗi vòng của DES dùng khóa con có kích thước 48 bit được trích ra từ khóa chính.

Các vòng Feistel của mã DES



DES:: Hoán vị khởi tạo

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
56	48	40	32	24	16	8	0
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6

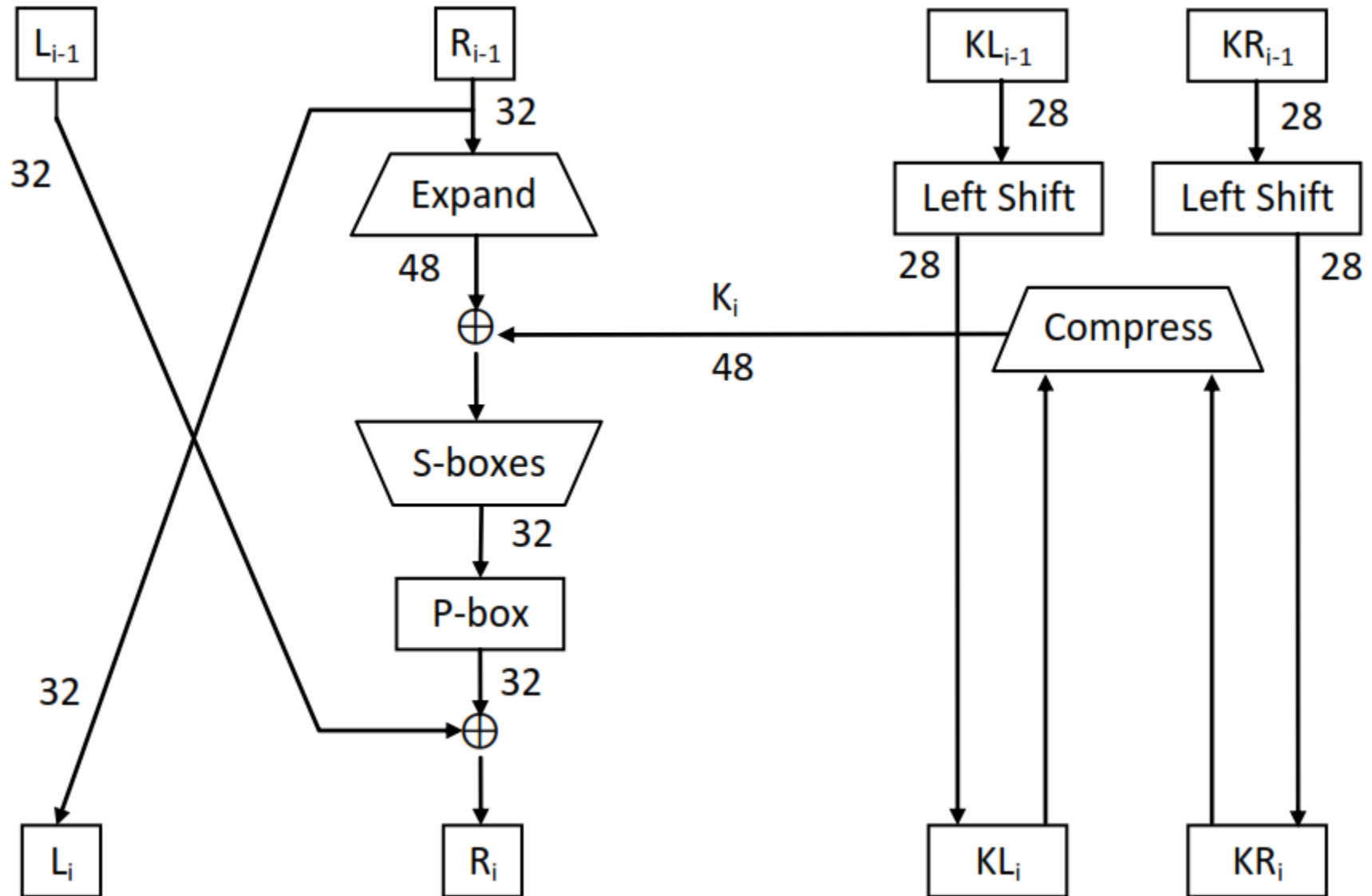
$(b_0 b_1 \dots b_{62} b_{63} \rightarrow b_{57} b_{49} \dots b_{14} b_6)$

DES:: Hoán vị kết thúc

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
32	0	40	8	48	16	56	24

$(b_0 b_1 \dots b_{62} b_{63} \rightarrow b_{39} b_7 \dots b_{56} b_{24})$


Cấu trúc 1 vòng của mã DES



DES:: Hàm *Expand*

- Hàm Expand thực hiện vừa hoán vị vừa mở rộng 32 bit thành 48 bit theo quy tắc:

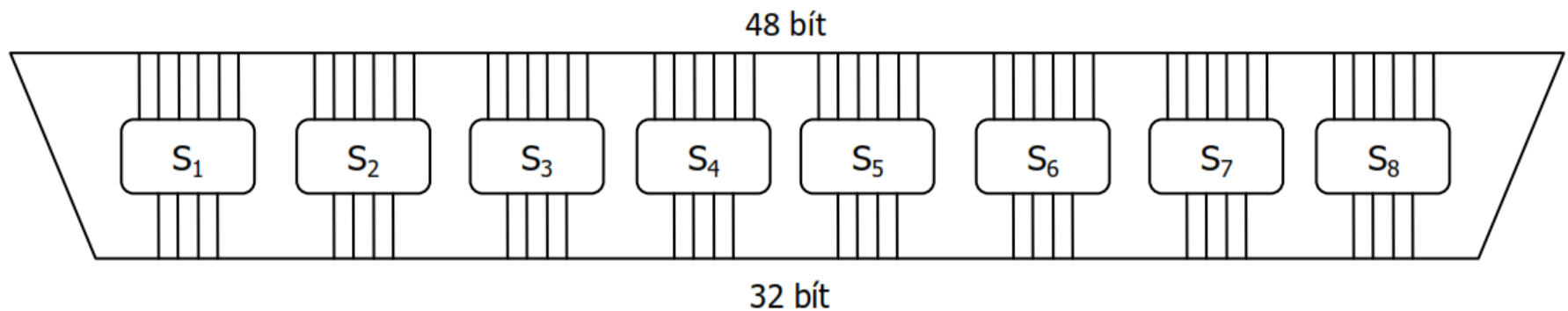
31	0	1	2	3	4
3	4	5	6	7	8
7	8	9	10	11	12
11	12	13	14	15	16
15	16	17	18	19	20
19	20	21	22	23	24
23	24	25	26	27	28
27	28	29	30	31	0



48 bit

DES:: Hàm *S-box*

- Hàm *S-box* của DES biến đổi một số 48 bit thành một số 32 bit



- Hàm *S-box* đầu tiên, hộp S_1 , giống hoàn toàn như *S-box* của TinyDES

		$b_1b_2 \quad b_3b_4$															
		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

DES:: Hàm S-box

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
	1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
	2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
	3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

DES S-box 1

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
	1	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
	2	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
	3	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9

DES S-box 2

DES:: Hàm S-box

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
	1	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
	2	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
	3	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C

DES S-box 3

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
	1	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
	2	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
	3	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E

DES S-box 4

DES:: Hàm S-box

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
	1	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
	2	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
	3	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3

DES S-box 5

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
	1	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
	2	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
	3	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D

DES S-box 6

DES:: Hàm S-box

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
	1	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
	2	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
	3	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C

DES S-box 7

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b_0b_5	0	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
	1	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
	2	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
	3	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

DES S-box 8

DES:: Hàm *P-box*

- Hàm *P-box* thực hiện hoán vị 32 bit đầu vào theo quy tắc

15	6	19	20	28	11	27	16
0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8
18	12	29	5	21	10	3	24

DES:: Thuật toán sinh khóa

- Khóa K 64 bit ban đầu được rút trích và hoán vị thành một khóa 56 bit theo qui tắc:

56	48	40	32	24	16	8
0	57	49	41	33	25	17
9	1	58	50	42	34	26
18	10	2	59	51	43	35
62	54	46	38	30	22	14
6	61	53	45	37	29	21
13	5	60	52	44	36	28
20	12	4	27	19	11	3

56 bit

DES:: Thuật toán sinh khóa

- Khóa 56 bit này được chia thành 2 nửa trái phải KL_0 và KR_0 , mỗi nửa có kích thước 28 bit.
- Tại vòng thứ i ($i = 1, 2, 3, \dots, 16$), KL_{i-1} và KR_{i-1} được dịch vòng trái r_i bit để có được KL_i và KR_i , với r_i được định nghĩa

$$r_i = \begin{cases} 1 & \text{nếu } i \in \{1, 2, 6, 9\} \\ 2 & \text{với những } i \text{ khác} \end{cases}$$

DES:: Thuật toán sinh khóa

- S cuối cùng khóa K_i của mỗi vòng được tạo ra bằng cách hoán vị và nén 56 bit của KL_i và KR_i thành 48 bit theo quy tắc:

13	16	10	23	0	4	2	27
14	5	20	9	22	18	11	3
25	7	15	6	26	19	12	1
40	51	30	36	46	54	29	39
50	44	32	47	43	48	38	55
33	52	45	41	49	35	28	31

48 bit

Hiệu ứng lan truyền của DES (**Avalanche Effect**)

- **Ví dụ 1:**

P1: 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000

P2: 10000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000

K: 0000001 1001011 0100100 1100010 0011100 0011000 0011100
0110010

- **Ví dụ 2:**

P: 01101000 10000101 00101111 01111010 00010011 01110110
11101011 10100100

K1: 1110010 1111011 1101111 0011000 0011101 0000100 0110001
11011100

K2: 0110010 1111011 1101111 0011000 0011101 0000100 0110001
11011100

Hiệu ứng lan truyền trong VD1 & VD2

Vòng thứ	Số bit khác nhau
0	1
1	6
2	21
3	35
4	39
5	34
6	32
7	31
8	29
9	42
10	44
11	32
12	30
13	30
14	26
15	29
16	34

Vòng thứ	Số bit khác nhau
0	0
1	2
2	14
3	28
4	32
5	30
6	32
7	35
8	34
9	40
10	38
11	31
12	33
13	28
14	26
15	34
16	35

Độ an toàn của DES

1) Tấn công vét cạn khóa (**Brute Force Attack**):

- Vì khóa của mã DES có chiều dài là 56 bit nên để tiến hành brute-force attack, cần kiểm tra 2^{56} khóa khác nhau. Hiện nay với những thiết bị phổ dụng, thời gian để thử khóa là rất lớn nên việc phá mã là **không khả thi**.
- Năm **1998**, tổ chức **Electronic Frontier Foundation (EFF)** thông báo đã xây dựng được một thiết bị phá mã DES gồm nhiều máy tính chạy song song, trị giá khoảng **250.000\$**. Thời gian thử khóa là **3 ngày**.
- Hiện nay, mã DES vẫn còn được sử dụng trong thương mại, tuy nhiên người ta đã bắt đầu áp dụng những phương pháp mã hóa khác có **chiều dài khóa lớn hơn** (**128 bit** hay **256 bit**) như TripleDES hoặc AES.

Độ an toàn của DES

2) Phá mã DES theo phương pháp vi sai (differential cryptanalysis):

- Năm 1990, Biham và Shamir đã giới thiệu phương pháp phá mã vi sai. Phương pháp vi sai tìm khóa ít tốn thời gian hơn brute-force. Tuy nhiên phương pháp phá mã này lại đòi hỏi phải có 2^{47} cặp bản rõ - bản mã được lựa chọn (chosen-plaintext).

→ Bất khả thi

Độ an toàn của DES

3) Phá mã DES theo phương pháp thử tuyến tính (linear cryptanalysis)

- Năm 1997, Matsui đưa ra phương pháp phá mã tuyến tính. Trong phương pháp này, cần phải biết trước 2^{43} cặp bản rõ-bản mã (known-plaintext).

→ Không khả thi