

MẬT MÃ ỨNG DỤNG TRONG AN TOÀN THÔNG TIN

(Applied Cryptography In Secure Information System)

ThS. Bùi Hữu Đông
buihuudong19@gmail.com
0903.82.36.46



Học viện Kỹ thuật Mật mã, khoa An toàn thông tin

Ngày 6 tháng 11 năm 2020

Buổi 03

MÃ HÓA ĐỐI XỨNG (Symmetric Cryptography)

MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI (Modern Symmetric Cryptography)

Tổng quan

- Nhận xét:

- Đối tượng của phương pháp mã hóa cổ điển là các bản tin ngôn ngữ, một đơn vị mã hóa là các chữ cái và có thể áp dụng các hình thức thay thế (đơn hoặc đa bảng) hoặc hoán vị
- Với sự phát triển của máy tính, bản thông tin cần mã hóa đa dạng hơn như dữ liệu hình ảnh, âm thanh... mà dữ liệu này được biểu diễn trong máy tính dưới dạng số nhị phân

Thí dụ:

Bản tin: attack

Mã ASCII: 97 116 116 97 99 107

Biểu diễn nhị phân: 01100001 01110100 01110100 01100001 01100011 01101011

Tổng quan

- Dù thông tin tồn tại dưới dạng nhị phân nhưng kẻ thám mã vẫn có thể phá mã
- Mã hóa hiện đại quan tâm tới vấn đề *chống phá mã* khi biết trước bản rõ

Mã hóa đối xứng hiện đại

- Thí dụ:

- Giả sử ta có bảng mã tương ứng 8 chữ cái với số nhị phân 3 bit:

Chữ cái	Nhị phân
A	000
B	001
C	010
D	011
E	100
F	101
G	110
H	111

- Ta có bản rõ $P = \text{"HEAD"} = \text{"111100000011"}$ và khóa $K = \text{"0101"}$.

Mã hóa đối xứng hiện đại

- Để mã hóa ta dùng phép toán XOR, cụ thể:

bản rõ: 1111 0000 0011 (head)

khóa: 0101 0101 0101

bản mã: 1010 0101 0110 (FBCG)

- ▷ Trong phép mã hóa trên ta không mã hóa theo từng chữ cái mà là một khối gồm 4 bit.
- ▷ Để giải mã ta thực hiện phép XOR ngược giữa bản mã và khóa với nhau.

Mã hóa đối xứng hiện đại

- Thực hiện mã hóa bằng XOR có nhược điểm:
 - ▷ Khóa lặp lại nhiều lần (giống mã hóa Vigenere) để khắc phục người ta dùng bộ sinh số ngẫu nhiên để tạo khóa dài (giả lập One-Time Pad) đây là cơ sở cho *mã dòng (stream cipher)*
 - ▷ Một khối được mã hóa bằng XOR không an toàn vì chỉ cần biết *một cặp khối* - bản rõ & bản mã, kẻ phá mã có thể tìm được khóa => cần phương pháp mã hóa phức tạp hơn và đây là cơ sở của *mã khối (block cipher)*

MÃ DÒNG (Stream Cipher)

Mã dòng - Stream Cipher

• Đặc điểm:

- Kích thước một đơn vị mã hóa gồm k bit. Bản rõ chia thành các đơn vị mã hóa, có nghĩa $P \rightarrow p_0, p_1 p_2 \dots p_{n-1}$ với mỗi $p_i : k$ 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:

$$SC(K) \rightarrow Z = z_0 z_1 z_2 \dots z_{n-1} \text{ với } z_i \text{ là } k \text{ bit.}$$

- Mỗi số ngẫu nhiên được XOR với đơn vị mã hóa của bản rõ để được bản mã:

$$c_0 = p_0 \oplus z_0, c_1 = p_1 \oplus z_1, \dots, C = c_0 c_1 c_2 \dots c_{n-1}$$

- Quá trình giải mã được làm ngược lại. Tức thực hiện phép XOR giữa khóa và bản mã

Mã dòng - Stream Cipher

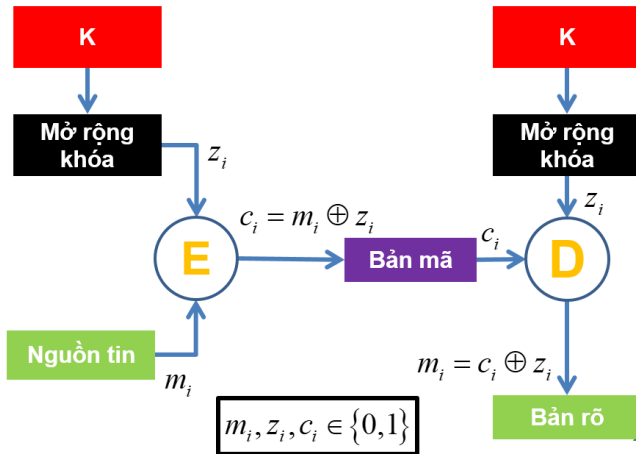
- Thí dụ mã hóa dưới đây có phải mã dòng?

bản rõ: 1111 0000 0011 (head)

khóa: 0101 0101 0101

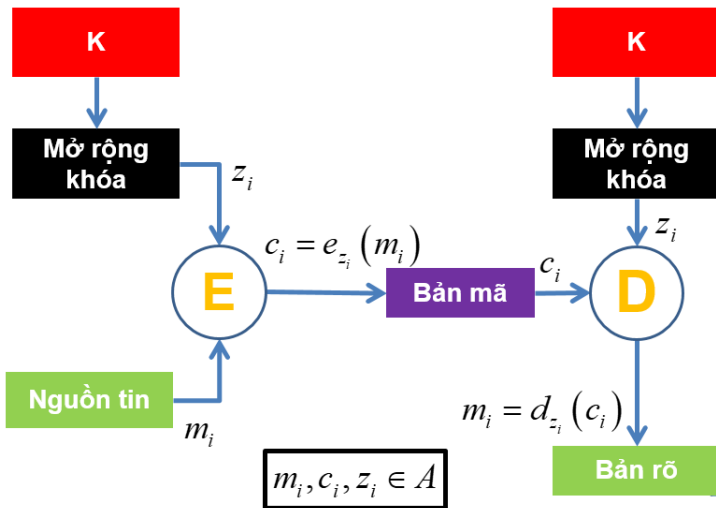
bản mã: 1010 0101 0110 (FBCG)

Mã dòng - Stream Cipher



Theo bạn mã hóa dạng mã dòng quan trọng nhất là gì?

Mã dòng - Stream Cipher



Mã dòng - Stream Cipher

- Phần mở rộng khóa:

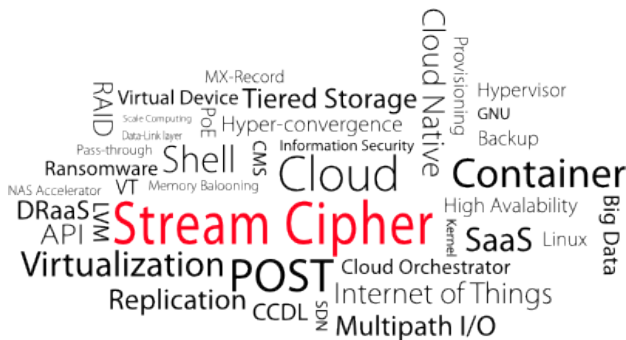
- Là bộ sinh số giả ngẫu nhiên (PRNG)
- Là quan trọng nhất
- Quyết định độ an toàn của mã dòng

- Phân loại:

- z_i chỉ phụ thuộc vào khóa K thì gọi là mã dòng đồng bộ
- z_i phụ thuộc $c_{i-n}, c_{i-n+1}, \dots, c_{i-1}$: gọi là mã dòng tự đồng bộ

Ứng dụng của Mã dòng

- Mã dòng có tốc độ cao do cơ chế sinh dòng khóa và mã hóa khá đơn giản so với mã khối
- Mã dòng có thể mã hóa lượng dữ liệu bất kì, không cần phải chờ đợi kích thước đầu vào đạt đến giá trị nhất định như mã khối



Phương pháp Mã dòng A5/1

- **Mô tả:** A5/1 được 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.

Để hiểu được A5/1 ta tìm hiểu dạng thu nhỏ là **TinyA5/1**. Với cơ chế như sau:

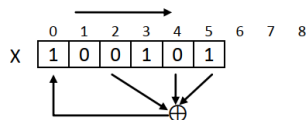
- Bộ sinh số gồm 3 thanh ghi X, Y, Z .
 - X gồm 6 bit (x_0, x_1, \dots, x_5)
 - Y gồm 8 bit (y_0, y_1, \dots, y_7)
 - Z gồm 9 bit (z_0, z_1, \dots, z_8)
- Khóa K có chiều dài 23 bit và được phân bổ vào các thanh ghi $K \rightarrow XYZ$

Phương pháp Mã dòng A5/1

- Các thanh ghi X,Y,Z được biến đổi theo 3 quy tắc:

1) **Quay X** gồm các thao tác:

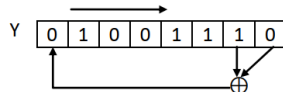
- $t = x_2 \oplus x_4 \oplus x_5$
- $x_j = x_{j-1}$ với $j = 5, 4, 3, 2, 1$
- $x_0 = t$



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.

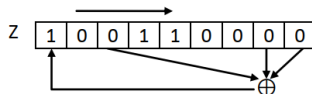
2) **Quay Y**: tương tự như quay X, quay Y là như sau:

- $t = y_6 \oplus y_7$
- $y_j = y_{j-1}$ 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 $j = 8, 7, 6, \dots, 1$
- $z_0 = t$



Phương pháp Mã dòng A5/1

- Cho 3 bit x, y, z ta định nghĩa hàm $maj(x, y, z)$, nếu trong 3 bit này có từ 2 bit 0 trở lên thì hàm trả về 0, ngược lại trả về 1.
- Tại bước sinh số thứ i , các phép tính sau được thực hiện:
 - $m = maj(x_1, y_3, z_3)$
 - Nếu $x_1 = m$ thì thực hiện quay X
 - Nếu $y_3 = m$ thì thực hiện quay Y
 - Nếu $z_3 = m$ thì thực hiện quay Z

Và bit được sinh ra là: $s_i = x_5 \oplus y_7 \oplus z_8$

Bit s_i được XOR với bit thứ i trong bản rõ để có được bit thứ i trong bản mã theo quy tắc của mã dòng.

Phương pháp Mã dòng A5/1

- Thí 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

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

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

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

Phương pháp Mã dòng A5/1

○ Thí dụ <tiếp>:

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)

Phương pháp Mã dòng A5/1

• Phương pháp mã dòng A5/1:

Về nguyên tắc, bộ sinh số **A5/1** hoạt động như **TinyA5/1** nhưng các thanh ghi có size lần tương ứng X , Y , Z là 19, 22 và 23 bit. Các bước quay 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:

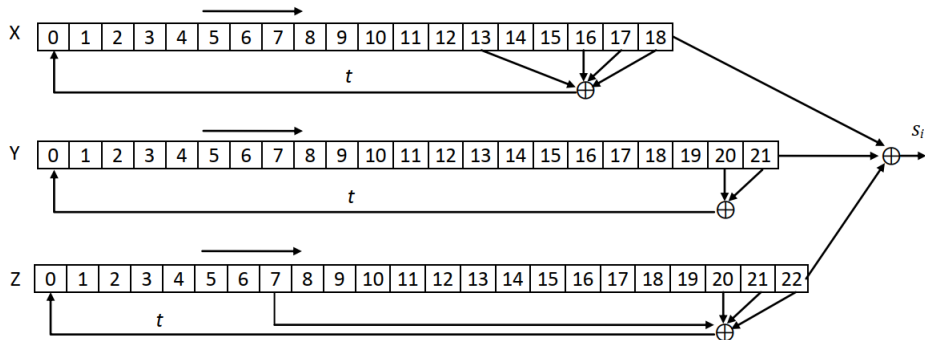
- $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
- $z_j = z_{j-1}$ với $j = 22, 21, 20, \dots, 1$
- $z_0 = t$

Phương pháp Mã dòng A5/1

Hàm *maj* được tính trên 3 bit x_8, y_{10}, z_{10} , với bit sinh ra là:

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

Mô hình sinh mã dòng của A5/1:



Ứng dụng của phương pháp Mã dòng A5/1

- A5/1 đã từng được sử dụng để mã hóa các dữ liệu real-time như các dãy بیت audio
- A5/1 được sử dụng để mã hóa dữ liệu cuộc gọi trong mạng điện thoại GSM.

Giới thiệu phương pháp Mã dòng RC4

- RC4 được thiết kế để đạt hiệu năng cao khi cài đặt bằng phần mềm
- Xây dựng bởi Ron Rivest năm 1987 nhưng đến năm 1994 mới được tiết lộ
- Được ứng dụng rộng rãi
- Kích thước khóa: 40-2048 bit

Để đơn giản, ta sẽ tìm hiểu mô hình thu nhỏ của **RC4** đó là **TinyRC4**

Phương pháp Mã dòng TinyRC4

- Giới thiệu:

- ◇ Khác với A5/1, đơn vị mã hóa của TinyRC4 là 3 bit và nó dùng hai mảng S và T , mỗi mảng gồm 8 số nguyên 3 bit (từ 0 đến 7)
- ◇ Khóa là một dãy gồm N số nguyên 3 bit với N có thể lấy giá trị từ 1 đến 8
- ◇ Bộ sinh số mỗi lần sinh ra 3 bit để sử dụng trong phép XOR. Quá trình sinh số của TinyRC4 gồm hai giai đoạn:

Phương pháp Mã dòng TinyRC4

◇ Quá trình sinh số của TinyRC4 gồm hai giai đoạn:

a) Giai đoạn khởi tạo:

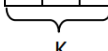
```
/* Khởi tạo dãy số S và T */  
for i = 0 to 7 do  
    S[i] = i;  
    T[i] = K[i mod N];  
next i  
/* Hoán vị dãy S */  
j = 0;  
for i = 0 to 7 do  
    j = (j + S[i] + T[i]) mod 8;  
    Swap(S[i], S[j]);  
next i
```

Phương pháp Mã dòng TinyRC4

- Mô tả giai đoạn khởi tạo:
 - Dãy S gồm các số nguyên 3 bit từ 0 đến 7 được sắp thứ tự tăng dần
 - Dựa trên các phần tử của khóa K , các phần tử của S được hoán vị lẫn nhau đến một mức độ ngẫu nhiên nào đó.
- Thí dụ: mã hóa bản rõ $P = 001000110$ (từ "bag") với khóa K gồm 3 số 2, 1, 3 ($N = 3$)

- Khởi tạo S và T

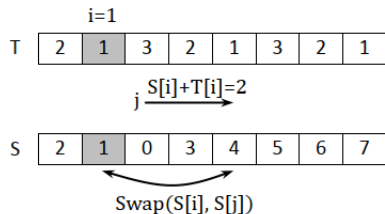
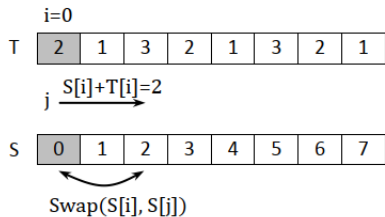
S	0	1	2	3	4	5	6	7
T	2	1	3	2	1	3	2	1



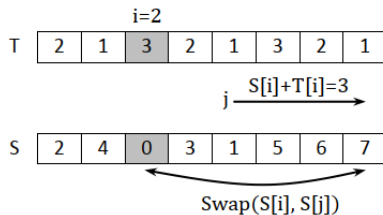
 K

Phương pháp Mã dòng TinyRC4

- Hoán vị S



Phương pháp Mã dòng TinyRC4



Quá trình thực hiện đến khi $i = 7$ và lúc đó dãy S là 6 0 7 1 2 3 5 4

Phương pháp Mã dòng TinyRC4

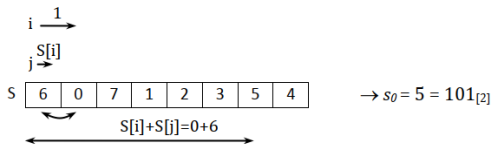
b) Giai đoạn sinh số:

```
i, j = 0;
while (true)
    i = (i + 1) mod 8;
    j = (j + S[i]) mod 8;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 8;
    k = S[t];
end while;
```

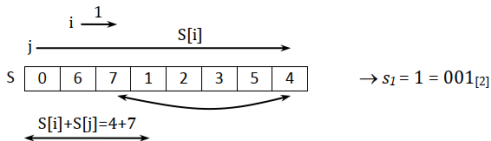
Phương pháp Mã dòng TinyRC4

- Mô tả giai đoạn sinh số:
 - Các phần tử của S tiếp tục được hoán vị
 - Tại mỗi bước sinh số, hai phần tử của dãy S được chọn để tính ra số k 3 bit là số được dùng để XOR với đơn vị mã hóa của bản rõ
- Thí dụ: quá trình sinh số mã hóa bản rõ "bag" thực hiện như sau:

Bước 0:

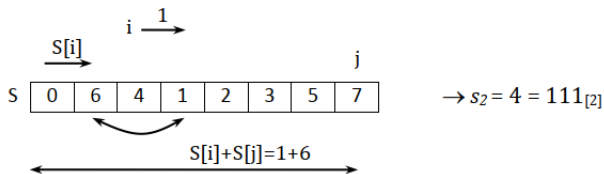


Bước 1:



Phương pháp Mã dòng TinyRC4

Bước 2:



Vậy bản mã là $C = 001.000.110 \oplus 101.001.111 = 100.001.001$ (từ EBB)

Phương pháp Mã dòng RC4

- Giới thiệu:

Cơ chế hoạt động của **RC4** cũng giống như **TinyRC4** với các đặc tính sau:

- Đơn vị mã hóa của RC4 là một byte 8 bit.
- Mảng S và T gồm 256 số nguyên 8 bit
- Khóa K là một dãy gồm N số nguyên 8 bit với N có thể lấy giá trị từ 1 đến 256
- Bộ sinh số mỗi lần sinh ra một byte để sử dụng trong phép XOR.

Phương pháp Mã dòng RC4

Hai giai đoạn của RC4 là:

a) Giai đoạn khởi tạo:

```
/* Khởi tạo day S và T*/  
for i = 0 to 255 do  
    S[i] = i;  
    T[i] = K[i mod N];  
next i  
/* Hoan vị day S */  
j = 0;  
for i = 0 to 255 do  
    j = (j + S[i] + T[i]) mod 256;  
    Swap(S[i], S[j]);  
next i
```

Phương pháp Mã dòng RC4

b) Giai đoạn sinh số:

```
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
end while;
```

Đánh giá phương pháp Mã dòng RC4

- ◇ Thuật toán đơn giản, rõ ràng
- ◇ Kích thước từ có thể thay đổi (ví dụ, dùng 4 bit thay vì 8 bit)
- ◇ Quá trình sinh số của RC4 cũng sinh ra dãy số ngẫu nhiên, khó đoán trước
- ◇ RC4 đạt được mức độ an toàn cao theo tinh thần của mã hóa One-Time Pad
- ◇ Mã hóa RC4 hoàn toàn được thực hiện trên các số nguyên một byte do đó tối ưu cho việc thiết lập bằng phần mềm và tốc độ thực hiện nhanh hơn so với mã khối

Ứng dụng của mã dòng RC4

- ◇ RC4 được dùng trong giao thức SSL
- ◇ RC4 được sử dụng trong mã hóa WEP (Wired Equivalent Privacy) của mạng Wireless LAN.
- ◇ BitTorrent protocol encryption
- ◇ Opera Mini
- ◇ Remote Desktop Protocol
- ◇ PDF
- ◇ Skype
- ◇ Secure shell*

MÃ KHỐI (Block Cipher)

Mã khối an toàn lý tưởng

- Phép toán XOR đảm bảo cho tốc độ mã hóa, nhưng chỉ cần biết cặp khối (bản rõ và bản mã) thì có thể xác định được khóa và có thể dùng khóa này để giải các khối mã hóa khác.
- Một cách tổng quát, nếu giữa bản rõ P và bản mã C có mối liên hệ toán học thì kẻ phá mã có thể tìm được khóa
- Vậy để an toàn ta phải làm cho P và C không có mối quan hệ toán học nào bằng cách lập bảng tra cứu ngẫu nhiên giữa bản rõ và bản mã

Mã khối an toàn lý tưởng

- Thí dụ: bảng ngẫu nhiên giữa bản rõ và bản mã:

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

Mã khối an toàn lý tưởng

- Khóa lúc này chính là bảng trên
- Kẻ phá mã chỉ có thể biết một phần bảng map trên, không thể biết hết từng mã như trên
- Nếu ta tăng kích thước của khối lên 64 bit \Rightarrow càng khó để phá mã vì số lượng bảng khóa rất lớn - cỡ $2^{64} \Rightarrow$ mã khối an toàn lý tưởng
- Tuy nhiên, nếu size của khóa tăng thì sẽ khó khăn cho việc lưu trữ và trao đổi. Bảng khóa có 2^{64} dòng mỗi dòng 64 bit do đó kích thước khóa sẽ là $64 \times 2^{64} = 2^{70} \approx 10^{21}$ bit \Rightarrow mã khối an toàn lý tưởng là không khả thi trong thực tế.

Nguyên lý thuyết kế mã khối

- Nguyên tắc Kerckhoffs: được Auguste Kerckhoffs đưa ra trong thế kỷ 19:
 - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge
 - The enemy knows the system (Claude Shannon)
- Nguyên lý Khuếch tán và xáo trộn:
 - Xáo trộn (Confusion): mỗi bit của bản mã phải phụ thuộc vào một phần của khóa và sao cho che dấu được mối liên quan giữa khóa và bản mã
 - Khuếch đại (Diffusion): nếu ta thay đổi một bit của bản rõ thì một nửa số bit trong bản mã sẽ thay đổi và ngược lại.

Nguyên lý thuyết kế mã khối

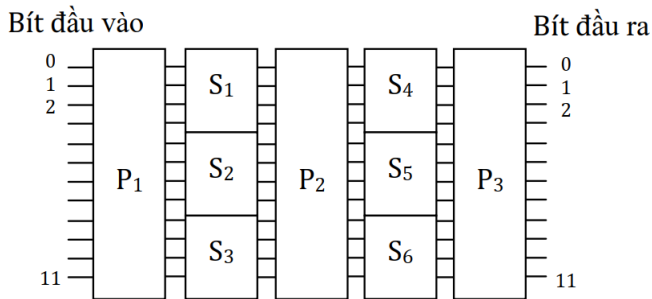
MẠNG SPN (Substitution-permutation network)

Tổng quan mạng SPN

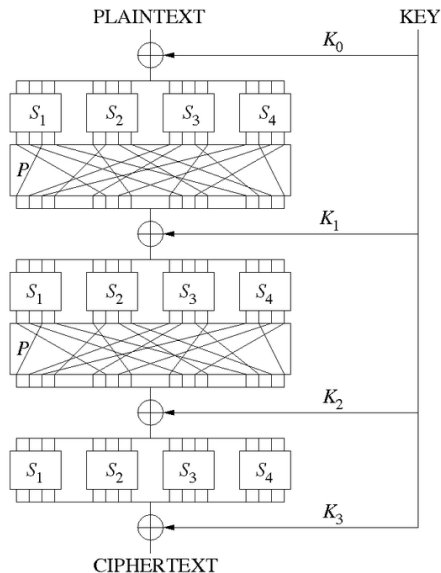
- Để tạo mã khối an toàn lý tưởng là rất khó khăn. Tuy nhiên, trong thực tế người ta tìm cách để chỉ dùng một khóa có size ngắn để giả lập một bảng tra cứu có độ an toàn xấp xỉ độ an toàn của mã khối lý tưởng.
- Để làm được, ta thực hiện kết hợp hai hay nhiều mã hóa đơn giản \Rightarrow mã hóa tổng.
- Mã hóa tổng an toàn hơn rất nhiều so với mã hóa thành phần.
- Các mã hóa đơn giản thường là các 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 (mạng SPN).

Tổng quan mạng SPN - Mô hình đơn giản



Tổng quan mạng SPN - Mô hình thực



Tổng quan mạng SPN

- Sự kết hợp giữa S-Box và P-Box tạo ra hai tính chất quan trọng của mã hóa là khuếch tán (diffusion) và xáo trộn/gây lộn (confusion) => cơ sở của tất cả mã khối hiện nay.
 - *Tính khuếch tán*: một bit của bản rõ tác động tới tất cả các bit của bản mã. Như thế, nhằm giảm tối đa mối liên hệ giữa bản rõ và bản mã => khó suy ra khóa (tính chất này nhờ kết hợp giữa S-Box và P-Box).
 - *Tính xáo trộn*: làm phức tạp mối liên quan giữa khóa và bản mã => ngăn chặn suy ra khóa (tính chất này nhờ S-Box)

Ứng dụng mạng SPN

SPN được dùng trong các giải thuật mã khối như:

- AES (Rijndael)
- 3-Way
- Kalyna
- Square
- Present
- Safer
- Shark
- Kuznyechik

Nguyên lý thuyết kế mã khối

CẤU TRÚC LƯỚI FEISTEL (Feistel network)

Tổng quan Cấu trúc lưới Feistel - Feistel Network

- Mô hình mã hóa lưới Feistel là một dạng tiếp cận khác so với mạng SPN. Mô hình này cũng là sự kết hợp giữa phép thay thế và hoán vị.



Horst Feistel

- Born in Germany
- January 30, 1915
- Died November 14, 1990
- Moved to US in 1934

Tổng quan Cấu trúc lưới Feistel - Feistel Network

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

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

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

$$P = (L_0, R_0)$$

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

Tổng quan Cấu trúc lưới Feistel - Feistel Network

- Quy tắc biến đổi các nửa trái phải qua các vòng được thực hiện:

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

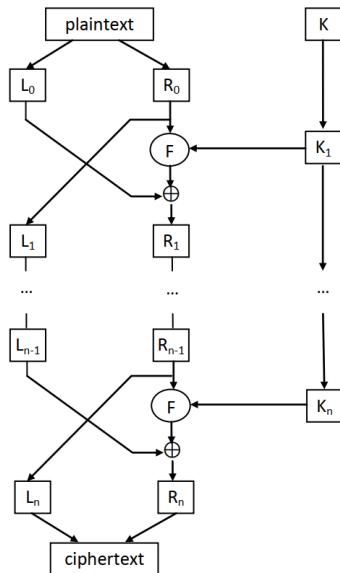
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

Trong đó:

- K_i là một khóa con cho vòng thứ i . Khóa K_i được sinh ra từ K ban đầu theo một thuật toán sinh khóa con: $K \rightarrow K_1 \rightarrow K_2 \rightarrow \dots \rightarrow K_n$
- F là một hàm mã hóa chung cho tất cả các vòng. Hàm F đóng vai trò như phép thay thế còn việc hoán đổi các nửa trái phải đóng vai trò hoán vị
- Bản mã C được tính từ kết xuất của vòng cuối cùng:

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

Mô hình cấu trúc lưới Feistel - Feistel Network



Tổng quan Cấu trúc lưới Feistel - Feistel Network

- Quá trình giải mã được thực hiện qua các vòng ngược lại:

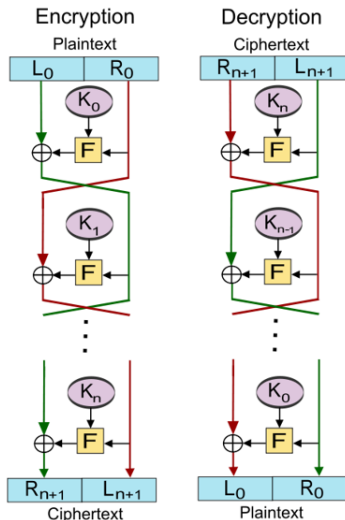
$$C \rightarrow L_n, R_n$$

$$R_{i-1} = L_i \text{ (theo mã hóa } L_i = R_{i+1})$$

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

Và cuối cùng ta được bản rõ là: $P = (L_0, R_0)$

Mô hình cấu trúc lưới Feistel - Feistel Network



F: round function
 K_i : round key
 L_i : left half
 R_i : right half

Mã hóa:

$$\begin{cases} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus F(R_i, K_i) \end{cases}$$

Giải mã

$$\begin{cases} R_i = L_{i+1} \\ L_i = R_{i+1} \oplus F(R_{i+1}, K_i) \end{cases}$$

Nhận xét mã hóa khối với Feistel Network

- ◇ Tại mỗi vòng i , chỉ một nửa khối được mã hóa \rightarrow cần nhiều vòng \rightarrow giảm hiệu năng
- ◇ Việc mã hóa và giải mã là giống hệt nhau, chỉ khác ở trật tự sử dụng khóa vòng \rightarrow chỉ cần 1 hàm/1 mạch điện tử để thực hiện cả mã hóa và giải mã
- ◇ Nửa khối được mã hóa là bên trái \Rightarrow kết quả mã hóa không nằm bên trái mà được chuyển sang phải, còn bên phải không mã hóa mà chuyển nguyên sang bên trái
- ◇ Nếu viết 1 hàm để thực hiện cả mã hóa và giải mã thì tham số sẽ là gì?

Đặc điểm cấu trúc lưới Feistel - Feistel Network

- ▶ Feistel chia các bản mã thành hai nửa trái và phải giúp cho hàm F không cần khả nghịch (tức không cần có F^{-1})
- ▶ Quá trình mã hóa và giải mã đều dùng chiều thuận của hàm F
- ▶ Hàm F và thuật toán sinh khóa con càng phức tạp thì càng khó phá mã
- ▶ Với mỗi hàm F và thuật toán sinh khóa con khác nhau \Rightarrow có phương pháp mã hóa khác nhau.

Các hệ mật sử dụng mạng Feistel

- | | | |
|-----------------|-----------|--------------|
| ◇ Blowfish | ◇ KASUMI | ◇ Simon |
| ◇ Camellia | ◇ LOKI97 | ◇ TEA |
| ◇ CAST-128 | ◇ Lucifer | ◇ Triple DES |
| ◇ DES | ◇ MARS | ◇ Twofish |
| ◇ FEAL | ◇ MAGENTA | ◇ XTEA |
| ◇ GOST 28147-89 | ◇ MISTY1 | |
| ◇ ICE | ◇ RC5 | |

Ứng dụng khác của mạng Feistel

- ♦ Một số hệ mật sử dụng biến thể của mạng Feistel (CAST-256, CLEFIA, MacGuffin, RC2, RC6, Skipjack, SMS4)
- ♦ Mạng Feistel còn được sử dụng cho mục đích khác với xây dựng mã khối, ví dụ, sử dụng trong lược đồ OAEP (Optimal Asymmetric Encryption Padding)
- ♦ Mạng Feistel còn được sử dụng cho mục đích khác với xây dựng mã khối, ví dụ, sử dụng trong lược đồ OAEP (Optimal Asymmetric Encryption Padding)

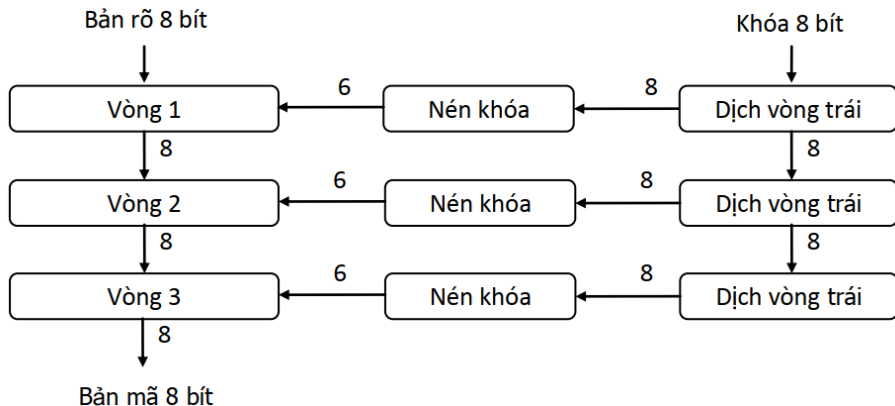
Phương pháp mã hóa dựa trên mạng Feistel

PHƯƠNG PHÁP MÃ HÓA DES (Data Encryption Standard)

Phương pháp mã hóa TinyDES

- Đặc điểm:
 - ◇ 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.

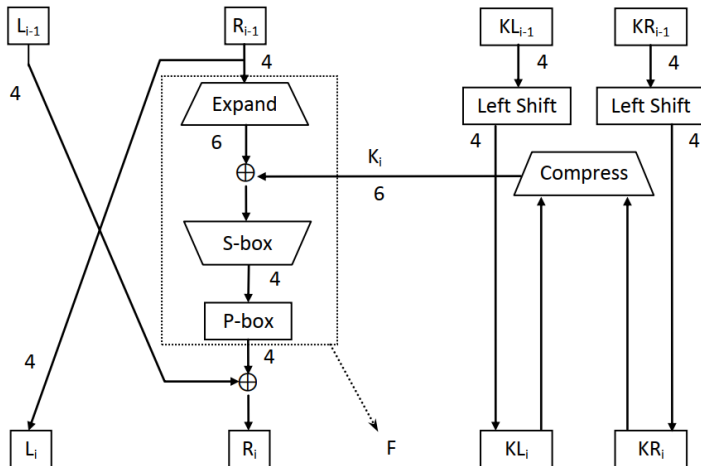
Mô hình các vòng Feistel của mã TinyDES



• Nhận xét:

Theo sơ đồ, mã TinyDES trên gồm 2 phần, thứ nhất là các vòng Feistel, thứ 2 là thuật toán sinh khóa con từ khóa K ban đầu.

Các vòng của TinyDES



Hình: Cấu trúc một vòng của mã TinyDES

Các vòng của TinyDES

- Nhận xét:

- Hàm F của Feistel là:

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

- Hàm *Expand* vừa mở rộng vừa hoán vị R_{i-1} từ 4 bit lên 6 bit
- Hàm *S-boxes* biến đổi một số 6 bit đầu vào thành một số 4 bit đầu ra
- Hàm *P-box* là một hoán vị 4 bit

Các vòng của TinyDES

- Chi tiết các hàm trong mã TinyDES:
 - Hàm *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: $b_2b_3b_1b_2b_1b_0$
 Thí dụ: $R_0 = 0110 \Rightarrow \text{Expand}(R_0) = 101110$
 - Hàm *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.

Các vòng của TinyDES

- Hàm *S-box* <tiếp>: Việc tính các bit đầu ra dựa trên bảng sau:

		$b_1b_2b_3b_4$															
		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

Hai bit b_0b_5 xác định thứ tự hàng, bốn bit $b_1b_2b_3b_4$ xác định thứ tự cột của bảng, Từ đó dựa vào bảng tính được 4 bit đầu ra.

Các vòng của TinyDES

- Hàm *S-box* <tiếp>: Để đơn giản ta có thể mô tả dưới dạng 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

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

- Hàm *P-Box*: thực hiện hoán vị 4 bit đầu $b_0b_1b_2b_3$ cho ra kết quả $b_2b_0b_3b_1$.

Thuật toán sinh khóa con của TinyDES

- Thuật toán:

- ▶ 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 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 : $k_5k_1k_3k_2k_7k_0$.

Thí dụ về TinyDES

- Thí dụ: mã hóa bản rõ $P = 0101.1100$ (5C) với khóa $K = 1001.1010$

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

▷ Vòng 1:

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

Thí dụ về TinyDES

▷ Vòng 2:

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

Thí dụ về TinyDES

▷ Vòng 3:

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

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

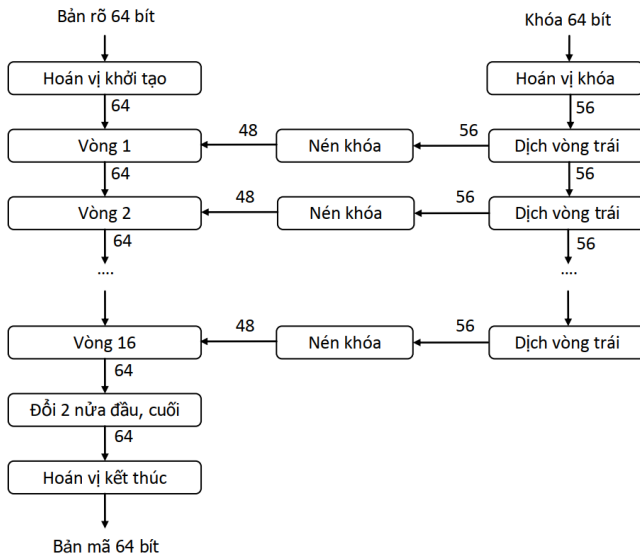
Nhận xét về TinyDES

- Với mã TinyDES 3 vòng, người phá mã chỉ biết 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 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 (hiệu ứng lan truyền) \Rightarrow khó phân tích mối liên quan giữa bản rõ, bản mã và khóa.

Giới thiệu phương pháp mã hóa DES (Data Encryption Standard)

- Mã DES thuộc hệ mã Feistel gồm 16 vòng cùng một hoán vị khởi tạo trước vào vòng 1 và một hoán vị khởi tạo sau vòng 16.
- Kích thước của khối là 64 bit: thí 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.

Mô hình các vòng Feistel của mã DES



Mô hình các vòng Feistel của mã DES

- **Mô tả:** Sơ đồ mã DES trên gồm 3 phần:
 - Phần thứ nhất là các hoán vị khởi tạo và hoán vị kết thúc
 - Phần thứ hai là các vòng Feistel
 - Phần thứ ba là thuật toán sinh khóa con

Quá trình hoán vị khởi tạo và hoán vị kết thúc

Ta đánh số các bit của khối 64 bit theo thứ tự từ trái sang phải là $b_0 b_1 b_2, \dots, b_{62} b_{63}$.

- Hoán vị khởi tạo sẽ hoán đổi các bit theo quy tắc:

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

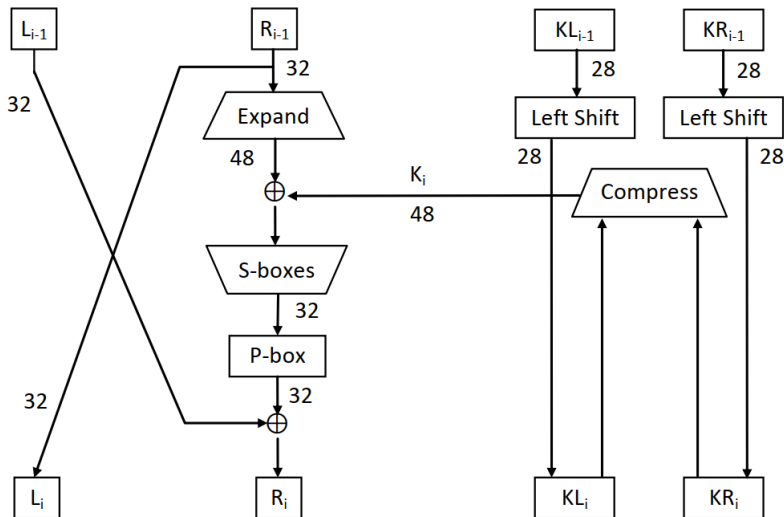
$$(b_0 b_1 b_2 \dots b_{62} b_{63} \rightarrow b_{57} b_{49} b_{41} \dots b_{14} b_6)$$

Quá trình hoán vị khởi tạo và hoán vị kết thúc

- Hoán vị kết thúc hoán đổi các bit theo quy tắc sau:

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

Các vòng của DES - Sơ đồ một vòng Feistel của DES



Mô tả sơ đồ một vòng Feistel của DES

- Với DES, hàm F của Feistel:

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

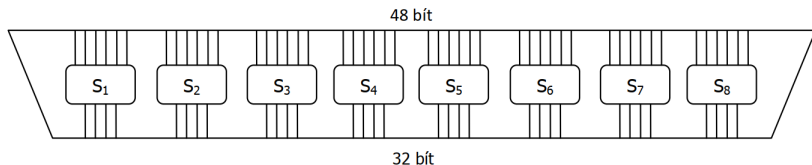
- Hàm *Expand*: đánh số các bit của R_{i-1} theo thứ tự từ trái sang phải là 0, 1, 2, ..., 31. Hàm Expand thực hiện vừa hoán vị vừa mở rộng 32 bit thành 48 bit theo quy tắc:

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

Mô tả sơ đồ một vòng Feistel của DES

- Hàm *S-boxes*: thực hiện biến đổi một số 48 bit thành một số 32 bit. Để giảm kích thước của bảng tra cứu (2^{16} dòng và 2^{32} cột), người ta chia hàm S-boxes thành 8 hàm S-box con, mỗi hàm biến đổi số 6 bit thành số 4 bit.



Mô tả sơ đồ một vòng Feistel của DES

- Hàm *S-boxes* <tiếp>: Hàm S-box đầu tiên, hộp S1, giống hoàn toàn như S-box của TinyDES:

		b ₁ b ₂ b ₃ b ₄															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
b ₀ b 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

Mô tả các hàm S-box của DES

		$b_1b_2b_3b_4$															
		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

Mô tả các hàm S-box của DES

		$b_1b_2b_3b_4$															
		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

Mô tả các hàm S-box của DES

		$b_1b_2b_3b_4$															
		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

Mô tả các hàm S-box của DES

		$b_1b_2b_3b_4$															
		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

Nhận xét hàm S-box của DES

- Nhận xét S-box: Có thể thấy, mỗi hàm S-box con là một phép thay thế Substitution. Các hàm S-box con không khả nghịch, do đó hàm S-boxes cũng không khả nghịch. Sự phức tạp này của S-boxes là yếu tố chính làm cho DES có độ an toàn cao.
- Hàm P-box: hàm P-box cũng 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

Thuật toán sinh khóa con của DES

- Khóa K 64 bit ban đầu được rút trích và hoán vị thành một khóa 56 bit (tức chỉ sử dụng 56 bit) theo quy 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

Thuật toán sinh khóa con của DES

- 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:

$$r_i = \begin{cases} 1 & \text{nếu } i \in \{1, 2, 9, 16\} \\ 2 & \text{với những } i \text{ khác} \end{cases}$$

Thuật toán sinh khóa con của DES

- 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 (Avalanche Effect)

- Hiệu ứng lan truyền trong các thuật toán mã hóa là chỉ cần thay đổi một bit trong bản rõ hay khóa thì dẫn đến sự thay đổi của nhiều bit bản mã
- Nhờ có tính chất này mà người phá mã không thể giới hạn miền tìm kiếm của bản rõ hay của khóa \Rightarrow vét cạn khóa
- DES có hiệu ứng lan truyền. Thí dụ: xét hai bản rõ (64 bit):

P1: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

P2: 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Hai bản rõ trên được mã hóa bằng DES với khóa:

K : 00000011001011010010011000100011100001100000111000110010

Hiệu ứng lan truyền (Avalanche Effect)

- Bảng sau cho biết số bit khác nhau của bản mã tương ứng với P_1 và P_2 qua các vòng của DES:

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

Hiệu ứng lan truyền (Avalanche Effect)

- Chỉ cần đến vòng thứ 2, số bit khác nhau giữa hai bản mã đã là 21 bit, sau 16 vòng số bit khác nhau là 34 bit (khoảng 1/2 tổng số bit của bản rõ)

Xét bản rõ sau (64 bit):

P: 01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

Dùng hai khóa sau đây để mã hóa bản rõ trên (hai khóa này chỉ khác nhau 1 bit):

K1: 1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

K2: 0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Hiệu ứng lan truyền (Avalanche Effect)

- Bảng sau cho biết số bit khác nhau của bản mã tương ứng với K_1 và K_2 qua các vòng của DES. Sau 16 vòng, số bit khác nhau là 35 bit, cũng khoảng $1/2$ tổng số bit của bản rõ:

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

Sinh viên thực hiện lại các vấn đề

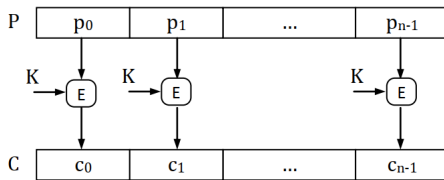
- Độ an toàn của DES
- Tấn công bạo lực (Brute force attack)
- Phá mã vi sai
- Phá mã Davies
- Một số phương pháp mã khối khác: Triple DES, AES
- Thực nghiệm bằng các thí dụ cụ thể.

Các mô hình ứng dụng mã khối

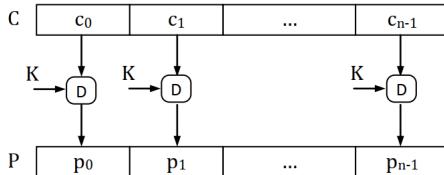
- Mã khối DES được áp dụng mã hóa một khối dữ liệu xác định, để mã hóa một bản rõ P ta chia P thành nhiều khối $P = P_0P_1, \dots, P_{n-1}$ và áp dụng mã khối cho từng khối một
- Có nhiều mô hình áp dụng mã khối là:
 - ◇ ECB: Electronic Codebook Mode
 - ◇ CBC: Cipher Block Chaining Mode
 - ◇ OFB: Output Feedback Mode
 - ◇ CFB: Cipher Feedback Mode
 - ◇ CTR: Counter Mode

Mô hình Electronic Codebook – ECB

- Trong mô hình ECB, mỗi khối được mã hóa một cách riêng rẽ, dùng chung một khóa K



a) Quá trình mã hóa



b) Quá trình giải mã