

Block cipher

Mật mã khối

Giảng viên: Nguyễn Phi Lê

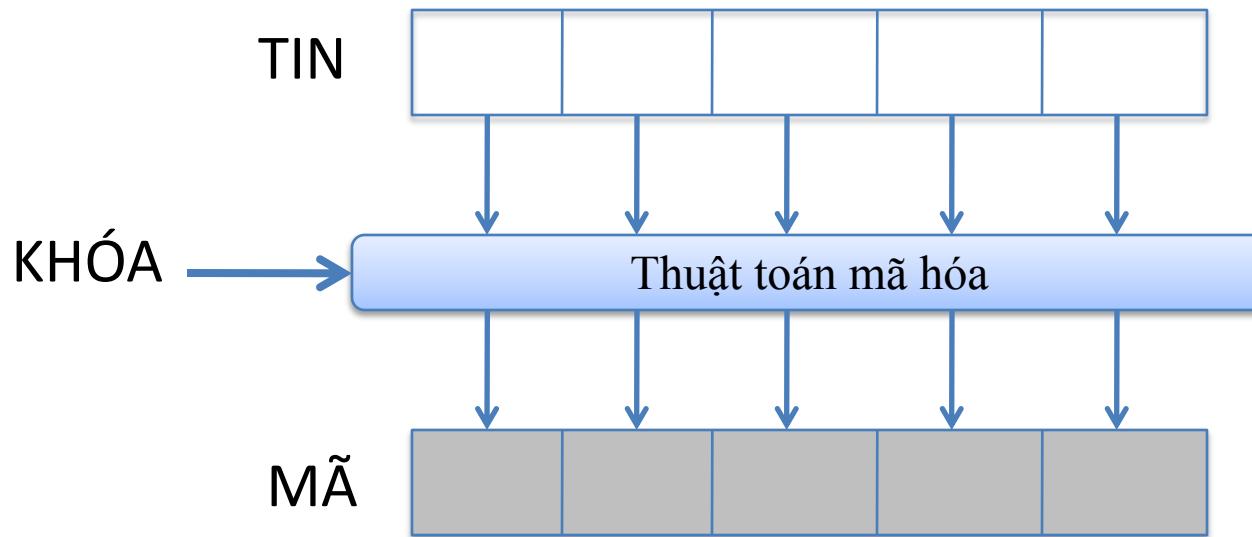
Mục lục

- Khái niệm
- Nguyên tắc thiết kế
 - Cấu trúc Feistel
- Mật mã DES

Overview (1/2)

- definition

- Mật mã khối là mật mã mà trong đó bản tin (plain text) được chia thành các khối có độ dài bằng nhau. Các khối này được mã hóa riêng biệt thành các khối mã có độ dài bằng khối đầu vào



Ví dụ

| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)

Ví dụ

| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)

Ví dụ

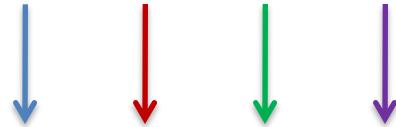
| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)



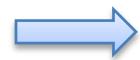
MÃ: (111) (011) (000) (000)

Ví dụ

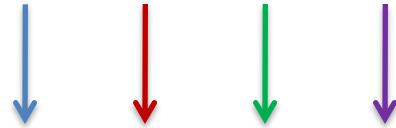
| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

KHÓA : 1

TIN: 010100110111



TIN: (010) (100) (110) (111)



MÃ: (111) (011) (000) (000)

KHÓA : 3

TIN: 010100110111

Ví dụ

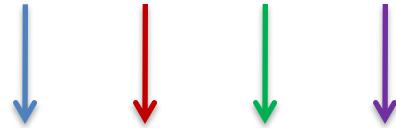
| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

KHÓA : 1

TIN: 010100110111



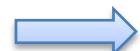
TIN: (010) (100) (110) (111)



MÃ: (111) (011) (000) (000)

KHÓA : 3

TIN: 010100110111



MÃ: (110) (000) (100) (011)

Ví dụ

| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

MÃ : 001

Ví dụ

| input key \ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

MÃ : 001  TIN: (000) Hoặc (101)

Khái niêm (2/2)

- Điều kiện an toàn
 - Kích thước khối phải đủ lớn
 - Để chống lại tấn công theo kiểu thống kê
 - Tuy nhiên lại gây ra độ trễ lớn
 - Không gian khóa phải đủ lớn
 - Để chống lại tìm kiếm vét cạn
 - Tuy nhiên lại gây ra khó khăn trong lưu trữ, phân phôi,
 - ...

Nguyên tắc thiết kế

- Khuếch tán (diffusion)
- Hỗn loạn (Confusion)
 - Đề xuất bởi Shanon, năm 1945

Nguyên tắc thiết kế

- Khuếch tán (diffusion)
- Hỗn loạn (Confusion)

Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Khuếch tán đặc tính thống kê của bản tin vào bản mã
 - Mỗi bit của bản tin và khóa phải ảnh hưởng lên nhiều bit của bản mã
 - ↔ Mỗi bit của bản mã bị ảnh hưởng bởi nhiều bit của bản tin
 - ⇒ gây khó khăn trong việc phá mã dựa trên đặc tính thống kê của bản tin

Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Tin: $T = t_1t_2\dots$
 - Mã: $M_n = \left(\sum_{i=1}^k t_{n+i} \right) \bmod 26$

⇒ tần suất xuất hiện của các ký tự trong bản mã “gần giống nhau” hơn trong bản tin.

⇒ đặc tính về thống kê của các ký tự trong bản tin đã bị làm khuếch tán đi trong bản mã.

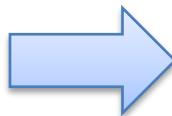
Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Phương pháp thực hiện: Thực hiện nhiều lần thao tác: hoán vị + tác động thuật toán
⇒ các bit tại các vị trí khác nhau của bản tin cùng tác động lên một bit của bản mã

Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
 - Bản tin = “computer security”
 - Tạo hoán vị bằng cách viết tin thành các hàng độ dài 5, mã lấy theo cột từ phải sang trái

| | | | | |
|---|---|---|---|---|
| c | o | m | p | u |
| t | e | r | s | e |
| c | u | r | i | t |
| y | | | | |



uetpsimrroeuctcy
computersecurity



xtgfnrcjkthpucwx

Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**
 - Sự phụ thuộc của bản mã đối với bản tin phải càng phức tạp càng tốt
 - ⇒ gây rắc rối, hỗn loạn đối với kẻ thù có ý định tìm quy luật phá mã
 - ⇒ mối quan hệ này tốt nhất là “phi tuyế”

Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**

- Tin: $T = t_1 t_2$

- Mã: $M = m_1 m_2$

$$\begin{cases} m_1 = k_{11}t_1 + k_{12}t_2 \\ m_2 = k_{21}t_1 + k_{22}t_2 \end{cases}$$

$$T' = t'_1 t'_2$$

$$M' = m'_1 m'_2$$

$$\begin{cases} m'_1 = k_{11}t'_1 + k_{12}t'_2 \\ m'_2 = k_{21}t'_1 + k_{22}t'_2 \end{cases}$$



$$\begin{cases} m_1 = k_{11}t_1 + k_{12}t_2 \\ m'_1 = k_{11}t'_1 + k_{12}t'_2 \end{cases}$$

$$\begin{cases} m_2 = k_{21}t_1 + k_{22}t_2 \\ m'_2 = k_{21}t'_1 + k_{22}t'_2 \end{cases}$$

Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**
 - Phương pháp thực hiện: Thực hiện bằng các thuật toán thay thế phức tạp

Mạng SPN

- Substitution and permutation network
 - Thuật toán sinh khoá (key schedule algorithm)
 - Input: khoá K
 - Output: N_r khoá con: $\{K^1, K^2, \dots, K^{N_r}\}$
 - Thuật toán mã hoá trong mỗi vòng lặp (round function)
 - $g(w^{r-1}, K^r)$

$$\begin{aligned} w^0 &\leftarrow x \\ w^1 &\leftarrow g(w^0, K^1) \\ w^2 &\leftarrow g(w^1, K^2) \end{aligned}$$

.....

$$\begin{aligned} w^{N_{r-1}} &\leftarrow g(w^{N_{r-2}}, K^{N_{r-1}}) \\ w^{N_r} &\leftarrow g(w^{N_{r-1}}, K^{N_r}) \\ y &\leftarrow w^{N_r} \end{aligned}$$

Mạng SPN

- Độ dài khối: lm
- Round function
 - Thay thế (Substitution): π_S
 - S-box: thay thế các khối l bit bằng l khác
 - Hoán vị (Permutation): π_P
 - Thay đổi vị trí của lm bit
- Thuật toán mã hóa trong 1 vòng

$$u^r \leftarrow w^{r-1} \oplus K^r$$

$$v^r \leftarrow \pi_S(u^r)$$

$$w^r \leftarrow \pi_P(v^r)$$

Mạng SPN

Algorithm 3.1: SPN($x, \pi_S, \pi_P, (K^1, \dots, K^{Nr+1})$)

Số khoá con: $N_r + 1$

$w^0 \leftarrow x$

for $r \leftarrow 1$ **to** $Nr - 1$

do $\begin{cases} u^r \leftarrow w^{r-1} \oplus K^r \\ \textbf{for } i \leftarrow 1 \textbf{ to } m \\ \quad \textbf{do } v_{*}^r \leftarrow \pi_S(u_{*}^r) \\ \quad w^r \leftarrow (v_{\pi_P(1)}^r, \dots, v_{\pi_P(\ell m)}^r) \end{cases}**$

$u^{Nr} \leftarrow w^{Nr-1} \oplus K^{Nr}$

for $i \leftarrow 1$ **to** m

do $v_{*}^{Nr} \leftarrow \pi_S(u_{*}^{Nr})**$

$y \leftarrow v^{Nr} \oplus K^{Nr+1}$

output (y)

Mang SPN

- Ví dụ: $l = m = N^r = 4$

| z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| $\pi_S(z)$ | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

| z | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| $\pi_P(z)$ | 1 | 5 | 9 | 13 | 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 | 12 | 16 |

$K = 0011\ 1010\ 1001\ 0100\ 1101\ 0110\ 0011\ 1111$

$K^1 = 0011\ 1010\ 1001\ 0100$

$K^2 = 1010\ 1001\ 0100\ 1101$

$K^3 = 1001\ 0100\ 1101\ 0110$

$K^4 = 0100\ 1101\ 0110\ 0011$

$K^5 = 1101\ 0110\ 0011\ 1111$

Plain text $x = 0010\ 0110\ 1011\ 0111$

Mang SPN

$w^0 = 0010\ 0110\ 1011\ 0111$

$K^1 = 0011\ 1010\ 1001\ 0100$

$u^1 = 0001\ 1100\ 0010\ 0011$

$v^1 = 0100\ 0101\ 1101\ 0001$

$w^1 = 0010\ 1110\ 0000\ 0111$

$K^2 = 1010\ 1001\ 0100\ 1101$

$u^2 = 1000\ 0111\ 0100\ 1010$

$v^2 = 0011\ 1000\ 0010\ 0110$

$w^2 = 0100\ 0001\ 1011\ 1000$

$K^3 = 1001\ 0100\ 1101\ 0110$

$u^3 = 1101\ 0101\ 0110\ 1110$

$v^3 = 1001\ 1111\ 1011\ 0000$

$w^3 = 1110\ 0100\ 0110\ 1110$

$K^4 = 0100\ 1101\ 0110\ 0011$

$u^4 = 1010\ 1001\ 0000\ 1101$

$v^4 = 0110\ 1010\ 1110\ 1001$

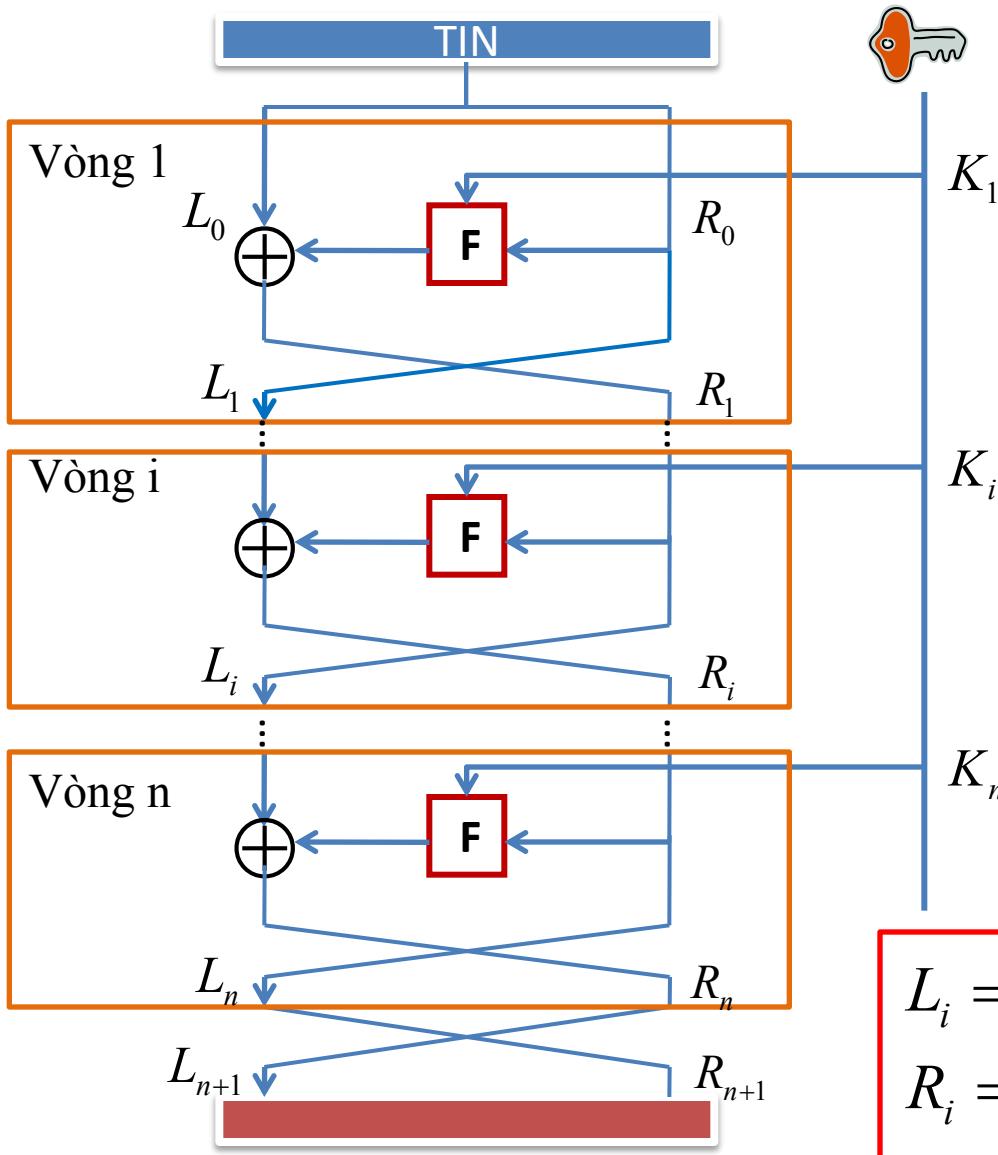
$K^5 = 1101\ 0110\ 0011\ 1111,$

$y = 1011\ 1100\ 1101\ 0110$

Cấu trúc Feistel

- Đầu tiên xuất bởi Feistel, năm 1973
- Được sử dụng bởi phần lớn các mật mã khối hiện nay
- Bao gồm nhiều vòng lặp,
 - Tại mỗi vòng lặp sẽ thực hiện các thao tác hoán vị và thay thế
 - Đầu vào của mỗi vòng lặp là đầu ra của vòng lặp trước đó và một khóa con được sinh ra từ khóa đầy đủ bởi thuật toán sinh quá
- Giải mã là một quá trình ngược với các khóa con cho mỗi vòng sẽ được phát sinh theo thứ tự ngược

Cấu trúc Feistel



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

The Data Encryption Standard (DES)

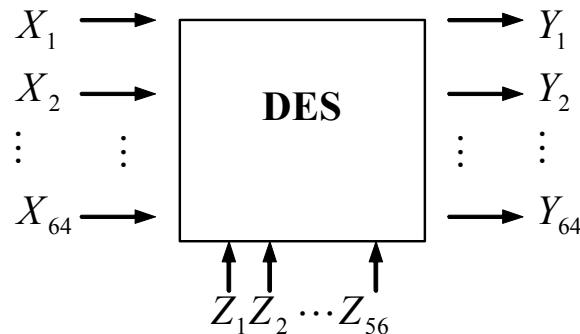
- Lịch sử
 - Vào những năm đầu thập kỷ 70, nhu cầu có một chuẩn chung về thuật toán mã hóa đã trở nên rõ ràng
 - Sự phát triển của công nghệ thông tin và của nhu cầu an toàn & bảo mật thông tin
 - Các thuật toán ‘cây nhà lá vườn’ (ad hoc) không thể đảm bảo được tính tin cậy đòi hỏi
 - Các thiết bị khác nhau đòi hỏi sự trao đổi thông tin mã hóa

The Data Encryption Standard (DES)

- Một chuẩn chung cần thiết phải có với các thuộc tính như
 - Bảo mật ở mức cao
 - Thuật toán được đặc tả và công khai hoàn toàn, tức là tính bảo mật không được phép dựa trên những phần che giấu đặc biệt của thuật toán
 - Việc cài đặt phải dễ dàng để đem lại tính kinh tế
 - Phải mềm dẻo để áp dụng được cho muôn vàn nhu cầu ứng dụng

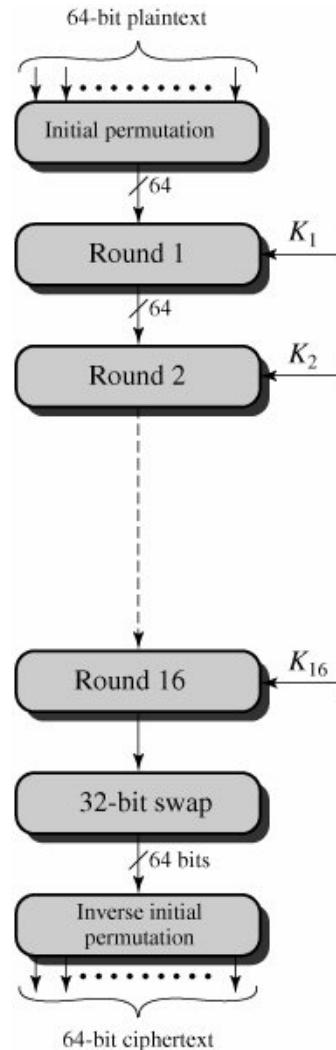
The Data Encryption Standard (DES)

- Năm 1973, Cục quản lý các chuẩn quốc gia của Mỹ đã có văn bản cổ động cho các hệ thống mã hóa ở cơ quan đăng ký liên bang của Mỹ. Điều đó cuối cùng đã dẫn đến sự phát triển của Data Encryption Standard, viết tắt là DES
- Sơ đồ chung



- Độ dài khối : 64 bit
- Độ dài khóa: 56 bit

The Data Encryption Standard (DES)

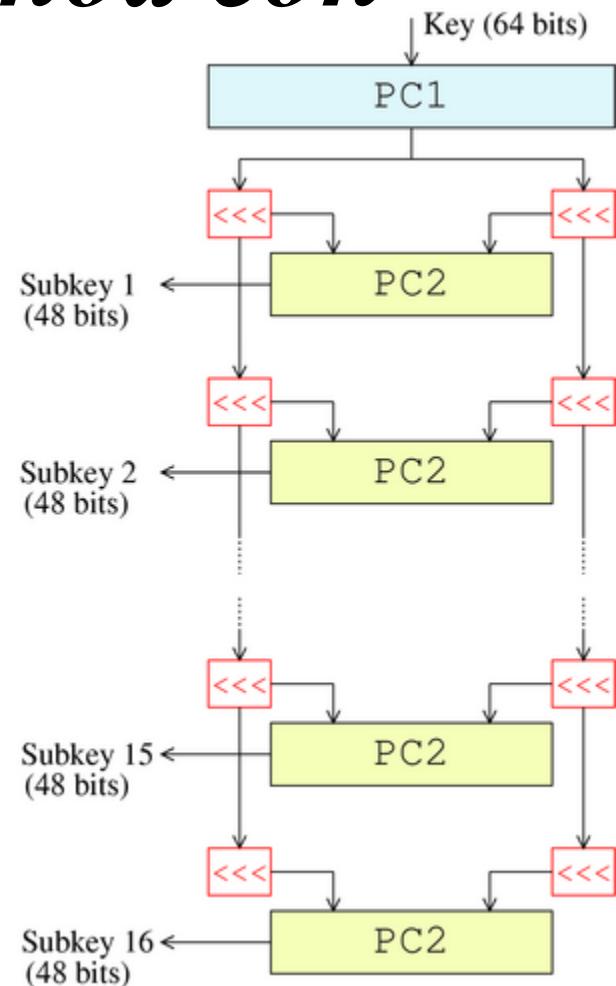


The Data Encryption Standard (DES)

- Bao gồm 16 bước lặp được kẹp bởi hai toán tử hoán vị IP, IP^{-1}
 - Các vòng lặp tuân theo cấu trúc Feistel
 - Toán tử IP, IP^{-1} mang ý nghĩa về mặt thực tiễn hơn là thuật toán (đơn giản hóa việc “chip hóa”)
- Hàm F là nguồn gốc của sức mạnh trong DES
 - Sự lặp lại nhiều lần các bước lặp với tác dụng của F là nhằm tăng cường thêm mãnh lực của F về mặt lượng

Thuật toán sinh khóa con

- 16 vòng lặp của DES chạy cùng thuật toán như nhau nhưng với các khóa khác nhau, được gọi là các khóa con
 - sinh ra từ khóa chính của DES bằng một thuật toán sinh khóa con.
- Khóa chính K, 64 bit, qua 16 bước biến đổi, mỗi bước sinh 1 khóa con 48 bit.
- Thực sự chỉ có 56 bit của khóa chính được sử dụng
 - 8 parity bits, lọc ra qua PC1.
 - Các bộ biến đổi PC1 và PC2 là các bộ vừa chọn lọc vừa hoán vị.
 - R1 và R2 là các phép đẩy bit trái 1 và hai vị trí.



Cấu trúc vòng lặp DES

- Mỗi vòng lặp của DES thực hiện trên cơ sở công thức sau:
$$(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f(R_{i-1}, K_i))$$
- Ta cũng có thể viết lại
$$(L_i, R_i) = T \bullet F (L_{i-1}, R_{i-1})$$
 - Trong đó F là phép thay thế L_{i-1} bằng $L_{i-1} \oplus f(R_{i-1}, K_i)$
 - T là phép đổi chỗ hai thành phần L và R.
 - Tức là mỗi biến đổi vòng lặp của DES có thể coi là một tích hàm số của F và T (trừ vòng cuối cùng không có T).
- Viết lại toàn bộ **thuật toán sinh mã DES** dưới dạng công thức:

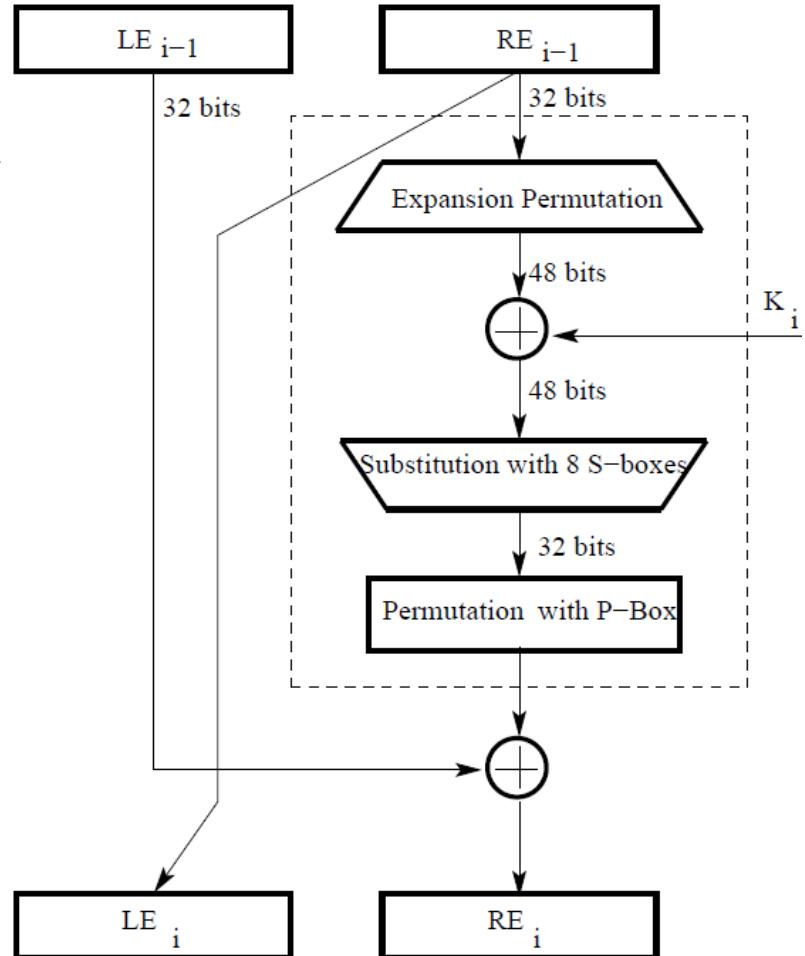
$$DES = (IP)^{-1} \bullet F_{16} \bullet T \bullet F_{15} \bullet T \bullet \dots \bullet F_2 \bullet T \bullet F_1 \bullet (IP)$$

Thuật toán giải mã DES

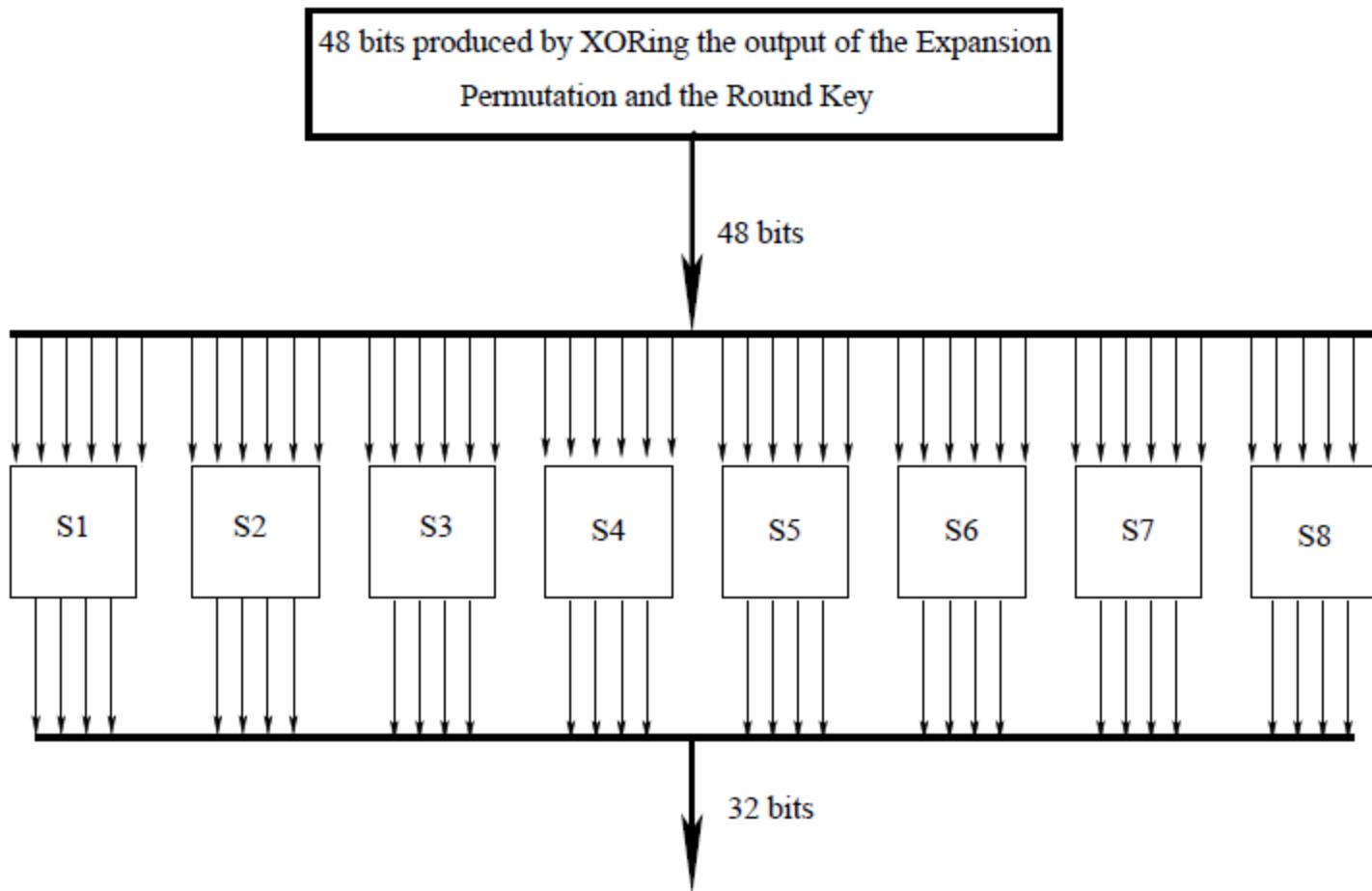
- Giống hệt như thuật toán sinh mã nhưng có các khóa con được sử dụng theo thứ tự ngược lại
 - Vì vậy, thuật toán giải mã có thể được viết lại dưới dạng công thức sau:
$$\text{DES}^{-1} = (\text{IP})^{-1} \bullet F_1 \bullet T \bullet F_2 \bullet T \bullet \dots \bullet F_{15} \bullet T \bullet F_{16} \bullet (\text{IP})$$
- Chú ý rằng mỗi hàm T hoặc F đều là các hàm có tính chất đối hợp ($f=f^{-1}$, hay $f(f(x))=x$) \rightarrow thực hiện DES • DES⁻¹ sẽ thu được phép đồng nhất.
 - Điều đó giải thích tại sao thuật toán giải mã lại giống hệt như sinh mã chỉ có khác thự từ dùng khóa con.

Cấu trúc cụ thể hàm f

- 32 bit của R_{i-1} được mở rộng thành 48 bit thông qua E rồi đệm XOR với 48 bit của K_i .
- 48 bit kết quả sẽ được phân thành 8 nhóm 6 bit; mỗi nhóm này sẽ qua một biến đổi đặc biệt, S-box, và biến thành 4 bit.
 - có 8 S-box khác nhau ứng với mỗi nhóm 6 bit
- 32 bit hợp thành từ 8 nhóm 4 bit (sau khi qua các S-box) sẽ được hoán vị lại theo P rồi đưa ra kết quả cuối cùng của hàm f (F_i).



S-boxes



Cấu trúc của các S-Box

- Mỗi S-box như một bộ biến đổi gồm 4 bảng biến đổi, mỗi bảng biến đổi 1 đầu vào 4 bit thành đầu ra cũng 4 bit (bảng 16 dòng).
 - Đầu vào 4 bit chính là lấy từ các bit 2-5 của nhóm 6 bit.
 - Các bit 1 và 6 sẽ dùng để xác định 1 trong 4 bảng biến đổi của S-box. Vì thế chúng được gọi là các bit điều khiển (CL và CR: left control và right control bit).

| S ₅ | | Middle 4 bits of input | | | | | | | | | | | | | | | |
|----------------|----|------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| | | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Outer bits | 00 | 0010 | 1100 | 0100 | 0001 | 0111 | 1010 | 1011 | 0110 | 1000 | 0101 | 0011 | 1111 | 1101 | 0000 | 1110 | 1001 |
| | 01 | 1110 | 1011 | 0010 | 1100 | 0100 | 0111 | 1101 | 0001 | 0101 | 0000 | 1111 | 1010 | 0011 | 1001 | 1000 | 0110 |
| | 10 | 0100 | 0010 | 0001 | 1011 | 1010 | 1101 | 0111 | 1000 | 1111 | 1001 | 1100 | 0101 | 0110 | 0011 | 0000 | 1110 |
| | 11 | 1011 | 1000 | 1100 | 0111 | 0001 | 1110 | 0010 | 1101 | 0110 | 1111 | 0000 | 1001 | 1010 | 0100 | 0101 | 0011 |

Các thuộc tính của S-Box

- Các nguyên tắc thiết kế của 8 S-boxes được đưa vào lớp ‘Classified information’ ở Mỹ.
- NSA đã tiết lộ 3 thuộc tính của S-boxes, những thuộc tính này bảo đảm tính confusion & diffusion của thuật toán.
 1. Các bít ra (output bit) luôn phụ thuộc không tuyến tính vào các bít vào (input bit).
 2. Sửa đổi ở một bit vào làm thay đổi ít nhất là hai bit ra.
 3. Khi một bit vào được giữ cố định và 5 bit con lại cho thay đổi thì S-boxes thể hiện một tính chất được gọi là ‘phân bố đồng nhất’ (uniform distribution): so sánh số lượng bit số 0 và 1 ở các đầu ra luôn ở mức cân bằng.
 - Tính chất này khiến cho việc áp dụng phân tích theo lý thuyết thông kê để tìm cách phá S-boxes là vô ích.

- 3 tính chất này đảm bảo tốt confusion & diffusion.
 - Sau 8 vòng lặp tất cả các bit ra của DES sẽ chịu ảnh hưởng của tất cả các bit vào và tất cả các bit của khóa.
- Tuy nhiên cấu tạo của S-box đã gây tranh luận mạnh mẽ từ hàng thập kỷ qua về khả năng cơ quan NSA (National Security Agency), Mỹ, vẫn còn che dấu các một số đặc tính của S-box hay cài bén trong những cửa bẫy (trapdoor) mà qua đó họ có thể dễ dàng phá giải mã hơn người bình thường.