# Mật mã khối

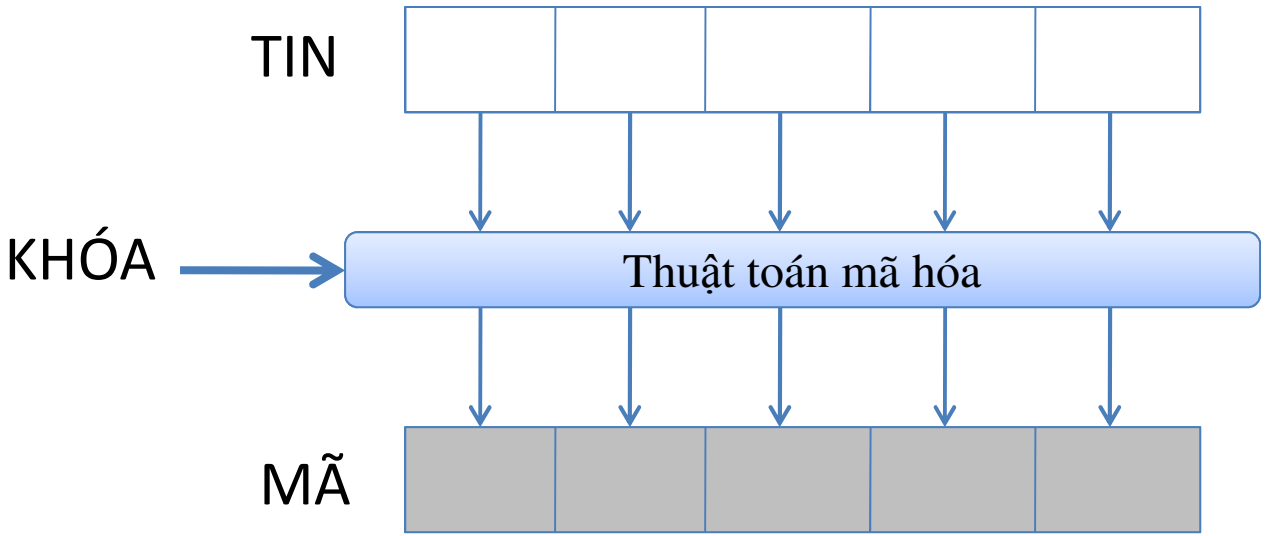
Giảng viên: Nguyễn Phi Lê

# Mục lục

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

# Khái niệm (1/2)

- Định nghĩa
  - 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

| key \ input | 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 |

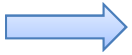
KHÓA : 1

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 |

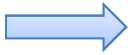
KHÓA : 1

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)

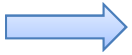
MÃ: (111) (011) (000) (000)

# Ví dụ

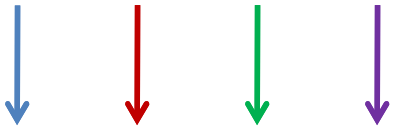
| key \ input | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001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

| key \ input | 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 phối, …

# 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

  $\Leftrightarrow$Mỗi bit của bãn mã bị ảnh hưởng bởi nhiều bit của bản tin

  $\Rightarrow$gây khó khăn trong việc phá mã dựa trên đặc tính thống kê của bản tin

# Nguyên tắc thiết kế

- **Khuếch tán (diffusion):**
  - Tin: $T = t_1 t_2 \ldots$
  - Mã: $M_n = \left( \sum_{i=1}^{k} t_{n+i} \right) \bmod 26$

  $\Rightarrow$tần suất xuất hiện của các ký tự trong bản mã "gần giống nhau" hơn trong bản tin.

  $\Rightarrow$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

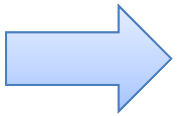
- **Khuếch tán (diffusion):**
  - Phương pháp thực hiện: Thực hiện nhiều lần thao tác: hoán vị + tác động thuật toán

  ⇒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

xtgfncrjkthpucwx

# 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 tuyến”

# Nguyên tắc thiết kế

- **Hỗn loạn (confusion):**
  - Tin: $T = t_1 t_2$ $\quad T' = t_1' t_2'$
  - Mã: $M = m_1 m_2$ $\quad M' = m_1' m_2'$

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

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

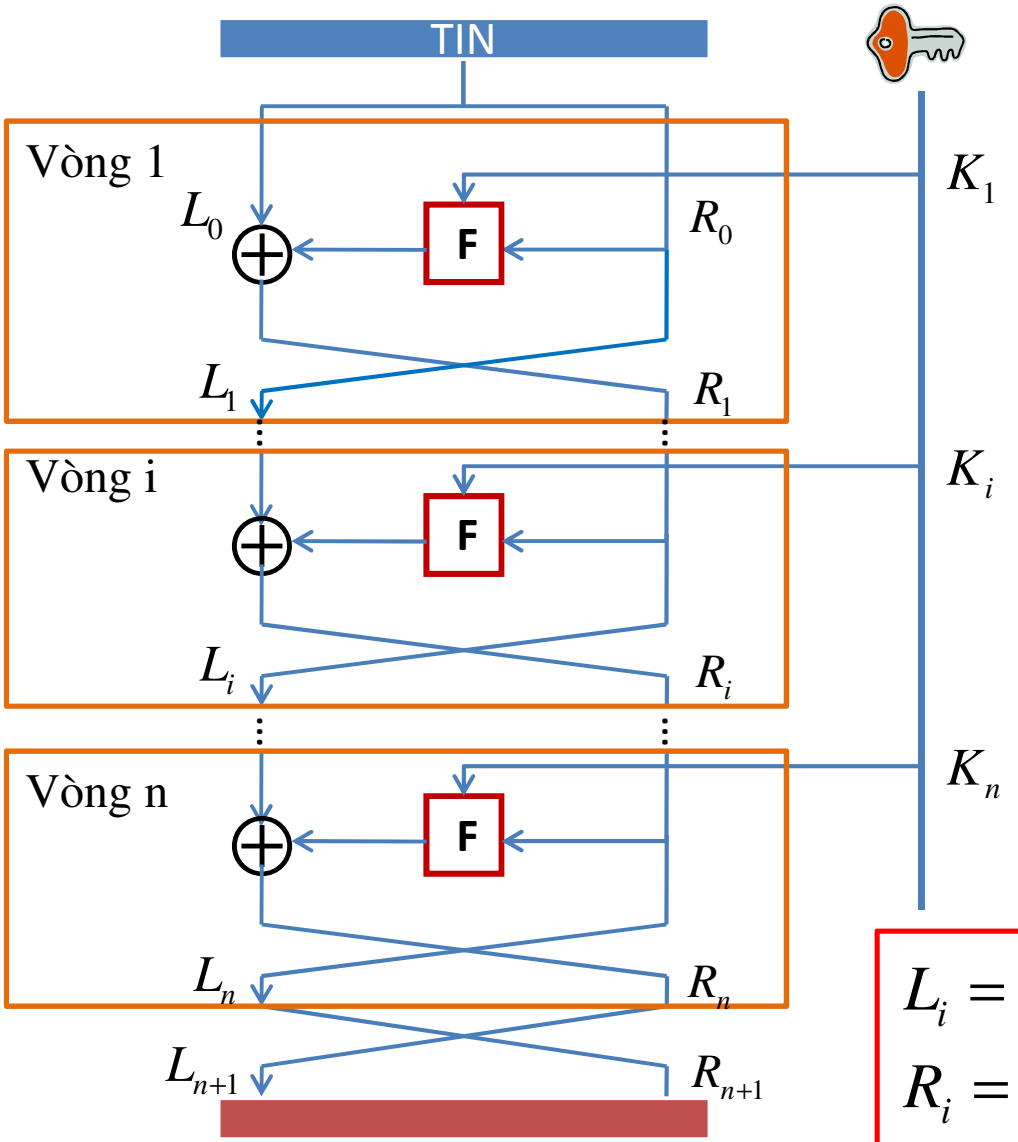
# Nguyên tắc thiết kế

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

# Cấu trúc Feistel

- Đề 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)

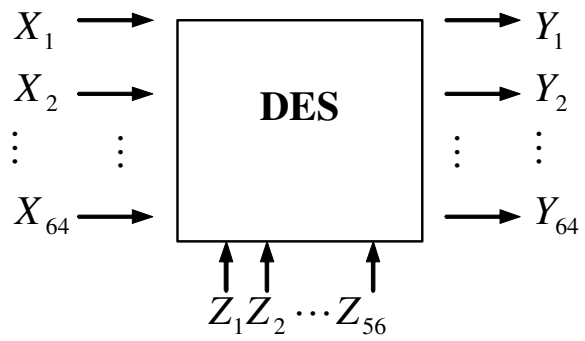
- Lịch sử
  - Vào những năm đầu thập kỷ 70, nhu cầu có một chuẩn chung về thuật toán mã hóa đã trở nên rõ ràng
    - Sự phát triển của công nghệ thông tin và của nhu cầu an toàn & bảo mật thông tin
    - Các thuật toán ‘cây nhà lá vườn’ (ad hoc) không thể đảm bảo được tính tin cậy đòi hỏi
    - Các thiết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

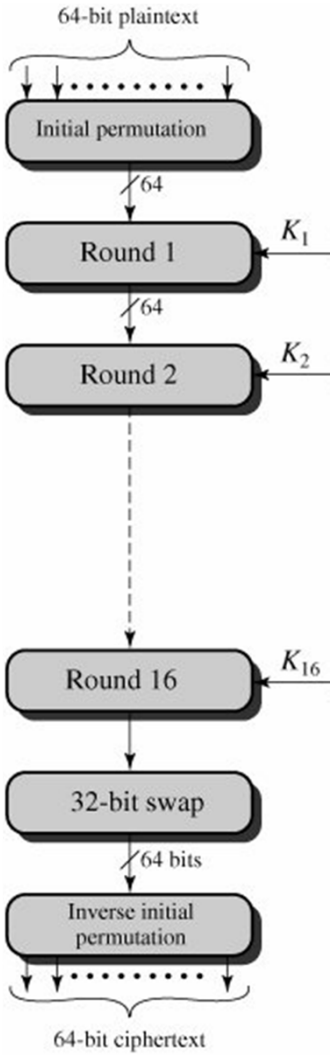
# The Data Encryption Standard (DES)

- 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
  - DES, IBM, Lucipher
  - dùng rộng rãi nhất, tranh cãi nhiều nhất
- Sơ đồ chung




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

# The Data Encryption Standard (DES)

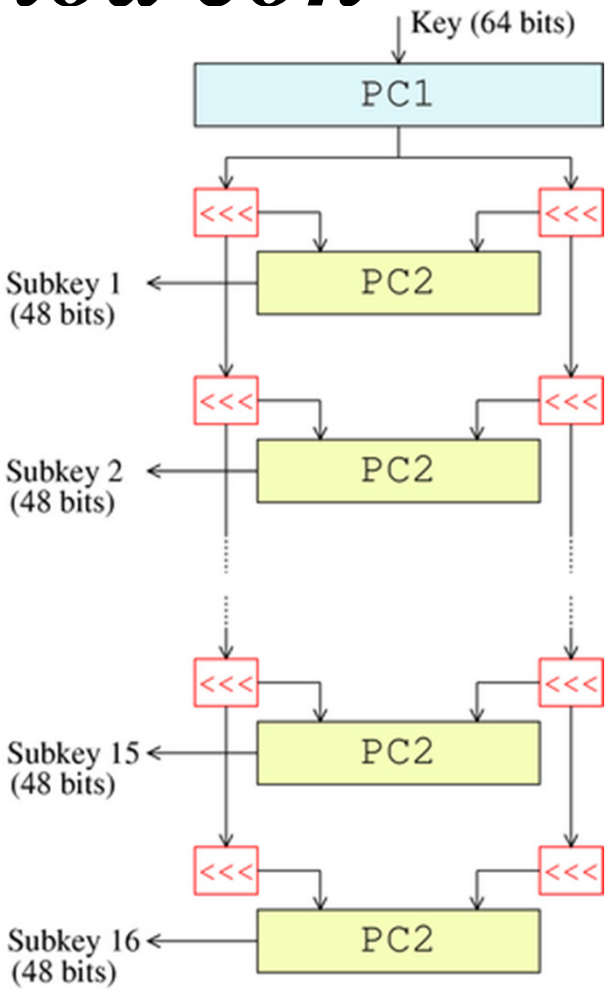


# The Data Encryption Standard (DES)

- 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:

$$DES = (IP)^{-1} \bullet f_{16} \bullet T \bullet f_{15} \bullet T \bullet \ ... \ \bullet f_2 \bullet T \bullet f_1 \bullet (IP)$$

# Thuật toán giải mã DES

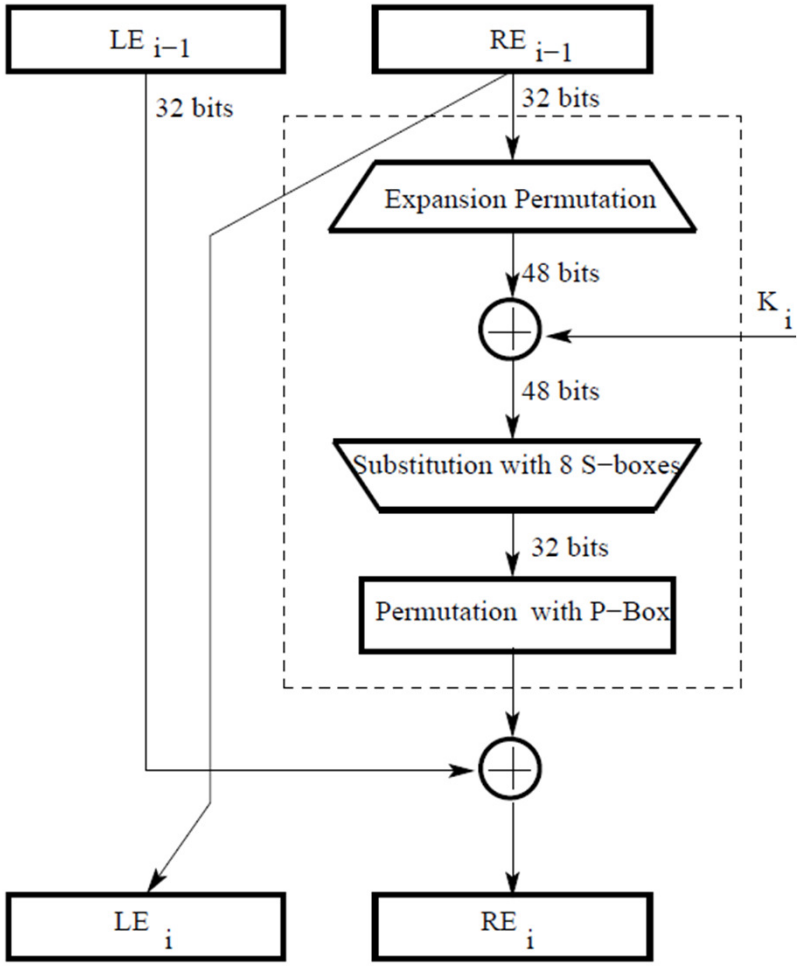
- Giống hệt như thuật toán sinh mã nhưng có các khóa con được sử dụng theo thứ tự ngược lại
  - Vì vậy, thuật toán giải mã có thể được viết lại dưới dạng công thức sau:

$$DES^{-1} = (IP)^{-1} \bullet f_1 \bullet T \bullet f_2 \bullet T \bullet \ldots \bullet f_{15} \bullet T \bullet f_{16} \bullet (IP)$$

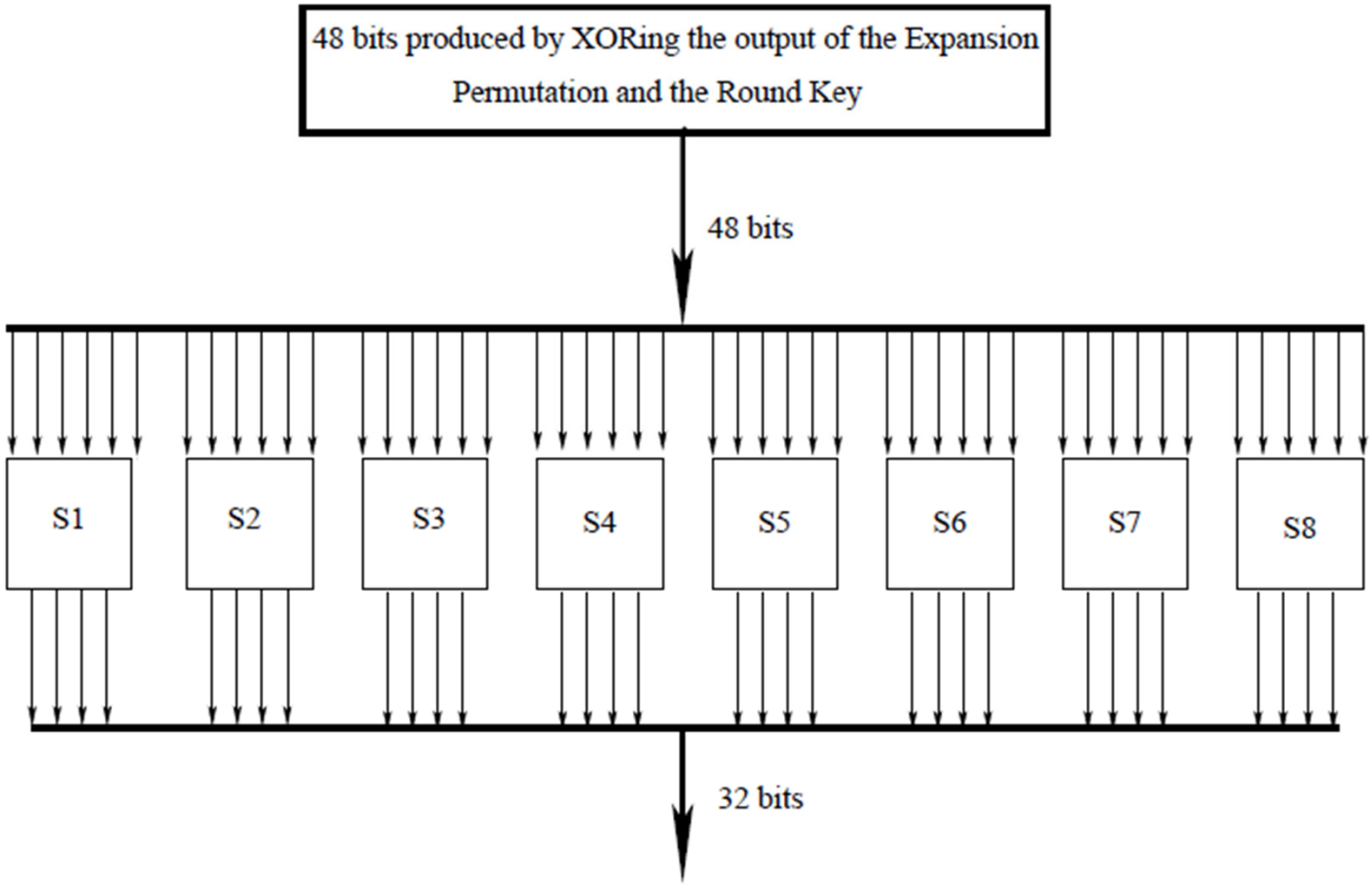
- Chú ý rằng mỗi hàm T hoặc f đều là các hàm có tính chất đối hợp ($f=f^{-1}$, hay $f(f(x)=x)$ ➔ thực hiện $DES \bullet DES^{-1}$ sẽ thu được phép đồng nhất.
  - Điều đó giải thích tại sao thuật toán giải mã lại giống hệt như sinh mã chỉ có khác thứ tự dùng khóa con.

# *Cấu trúc cụ thể hàm F*

- 32 bit của $R_{i-1}$ được mở rộng thành 48 bit thông qua E rồi đem 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 (Fi).



# S-boxes


48 bits produced by XORing the output of the Expansion Permutation and the Round Key
48 bits
S1
S2
S3
S4
S5
S6
S7
S8
32 bits

# *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 bên trong những cửa bẫy (trapdoor) mà qua đó họ có thể dễ dàng phá giải mã hơn người bình thường.

# Các điểm yếu của DES

- Tính bù

Ký hiệu $\bar{u}$ là phần bù của u (e.g. 0100101 và 1011010 là bù của nhau) thì DES có tính chất sau:

$$y = DES_z(x) \Rightarrow \bar{y} = DES_{\bar{z}}(\bar{x})$$

Cho nên nếu biết MÃ y được mã hóa từ TIN x với khóa z thì ta suy ra $\bar{y}$ được mã hóa từ TIN $\bar{x}$ với khóa $\bar{z}$.

  - Tính chất này chính là một điểm yếu của DES bởi vì nhờ đó kẻ địch có thể loại trừ một nửa số khóa cần phải thử khi tiến hành phép thử-giải mã theo kiểu vét cạn (tiếp)

# Khóa yếu

- Các khóa yếu là các khóa mà theo thuật toán KS sinh khóa con thì tất cả 16 khóa con đều như nhau

$$Z_1 = Z_2 = Z_3 = ... = Z_{15} = Z_{16}$$

điều đó khiến cho phép sinh mã và giải mã đối với các khóa yếu này là giống hệt nhau

$$DES_z = DES^{-1}_z$$

- Có tất cả 4 khóa yếu như sau:

1) [00000001 00000001 ... ... 00000001]
2) [11111110 11111110 ... ... 11111110]
3) [11100000 11100000 11100000 11100000 11110001 11110001 11110001 11110001]
4) [00011111 00011111 00011111 00011111 00001110 00001110 00001110 00001110]

# *Tấn công bằng phương pháp vét cạn (brute-force attack)*

- DES có $2^{56}=10^{17}$ khóa. Nếu như biết một cặp TIN/Mã thì chúng ta có thể thử tất cả $10^{17}$ khả năng này để tìm ra khóa cho kết quả khớp.
  - Giả sử như một phép thử mất quãng $10^{-6}$s, thì chúng ta sẽ thử mất $10^{11}$s tức là 7300 năm!
  - Xử lý song song: một thiết bị với $10^{7}$ con chip mật mã DES chạy song song,mỗi con chip chỉ thực hiện $10^{10}$ phép thử.
  - Chip mã DES ngày nay có thể xử lý tới tốc độ là 4.5 x $10^{7}$bits/s tức là có thể làm được hơn 105 phép mã DES trong một giây.
- Diffie và Hellman (1977) ước lượng: máy tính chuyên dụng vét cạn không gian khóa DES trong1/2 ngày với giá 20 triệu đô la. Giảm xuống $200,000 vào năm 1987.
  - Thực tế DES có thể bị phá trong vòng mấy chục giờ, với giá chỉ $10,000