Chủ đề 4:
Data Encryption Standard và Advanced Encryption Standard

PGS.TS. Trần Minh Triết

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung

- **Data Encryption Standard**
- **Advanced Encryption Standard**

Mã hóa tích (Product Cipher)

- Mã hóa chỉ sử dụng phép thay thế (substitution) hay phép đổi chỗ (transposition) không an toàn (do đặc tính của ngôn ngữ)
- Sử dụng liên tiếp các thao tác mã hóa đơn giản sẽ tạo ra cách mã hóa thông tin an toàn hơn
  - Substitution kết hợp với Substitution an toàn hơn 1 phép Substitution
  - Transposition kết hợp với Transposition an toàn hơn 1 phép Transposition
  - Substitution kết hợp Transposition cho kết quả an toàn hơn nhiều so với việc chỉ dùng một loại thao tác (thay thế hay đổi chỗ)
- Đây là ý tưởng mở đầu cho các phương pháp mã hóa hiện đại

Quy trình mã hóa theo khối

Secret Key

Key Schedule

Round Keys (Sub Keys)

PlainText

Data Path

Quy trình mã hóa theo khối

- Data Path:
  - Thông thường, quy trình mã hóa bao gồm nhiều chu kỳ mã hóa (round) liên tiếp nhau; mỗi chu kỳ gồm nhiều thao tác mã hóa
- Key Schedule:
  - Từ khóa gốc (secret key), phát sinh (có quy luật) các giá trị khóa sẽ được sử dụng trong mỗi chu kỳ mã hóa (round key)

Kiến trúc chu kỳ mã hóa

- Kiến trúc phổ biến của chu kỳ mã hóa:
  - Kiến trúc Fiestel
    - Ví dụ: Blowfish, Camellia, CAST-128, DES, FEAL, KASUMI, LOKI97, Lucifer, MARS, MAGENTA, MISTY1, RC5, TEA, Triple DES, Twofish, XTEA
  - Kiến trúc SPN
    - Ví dụ: Rijndael – AES, Anubis…

Data Encryption Standard

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Data Encryption Standard

- Mã hóa theo khối (block cipher)
- Ý tưởng: mã hóa tích (product cipher)
  - Key: 56 bit
  - Block: 64 bit
- Được IBM phát triển từ phương pháp **Lucifer**
- Chính thức công bố năm **1975**
- Được chọn là Chuẩn xử lý thông tin liên bang (Federal Information Processing Standard - FIPS) năm **1976**
- Giải thuật mã hóa và giải mã được công bố
- Cơ sở Toán học và mật mã của việc thiết kế DES: thông tin bí mật

Quy trình Mã hóa theo kiến trúc Feistel

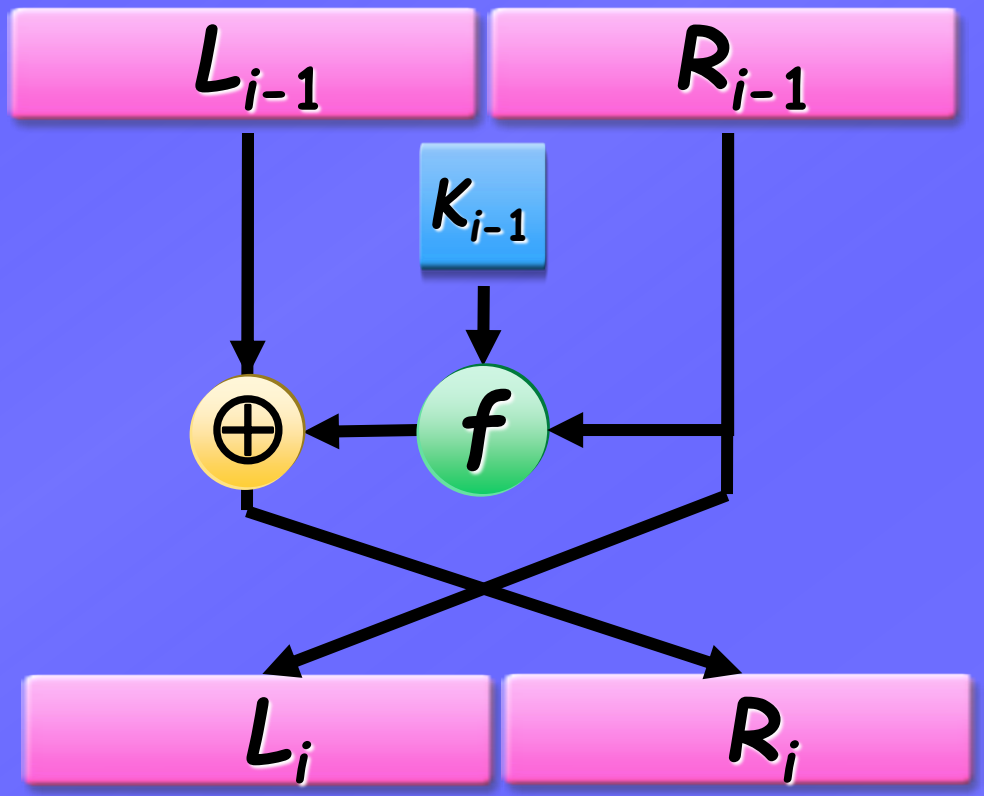
Chu kỳ mã hóa 1

...

Chu kỳ mã hóa *i*

...

Chu kỳ mã hóa *Nr*



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

Quy trình Giải mã theo kiến trúc Feistel

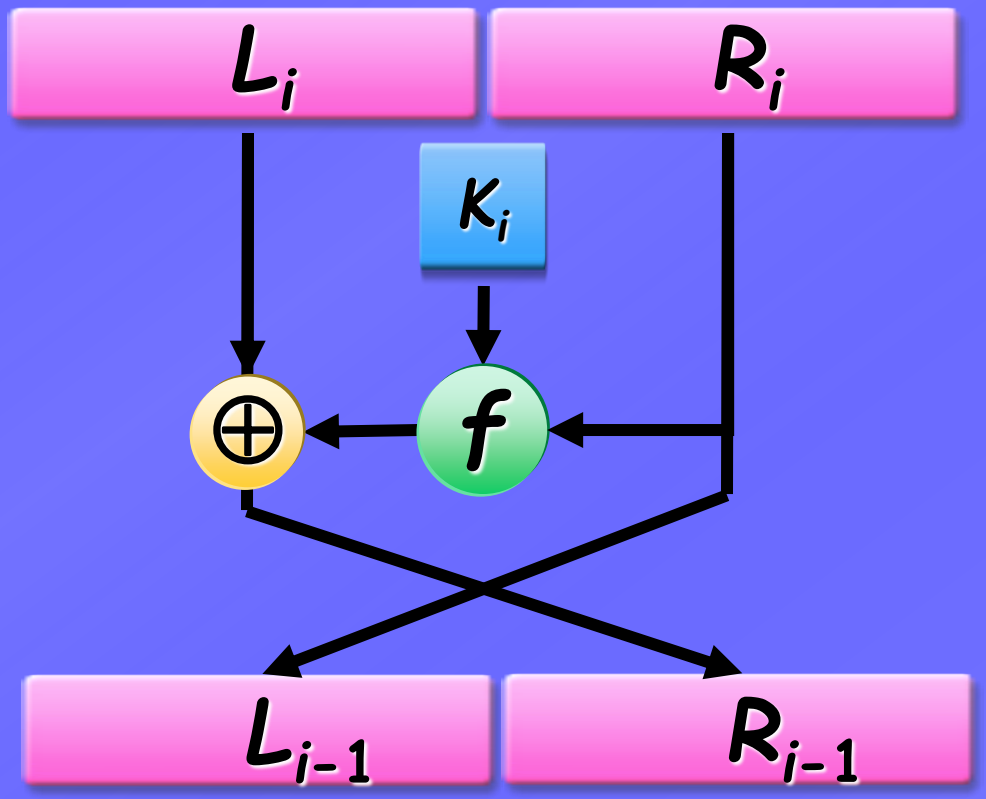
Chu kỳ giải mã *Nr*

...

Chu kỳ giải mã *i*

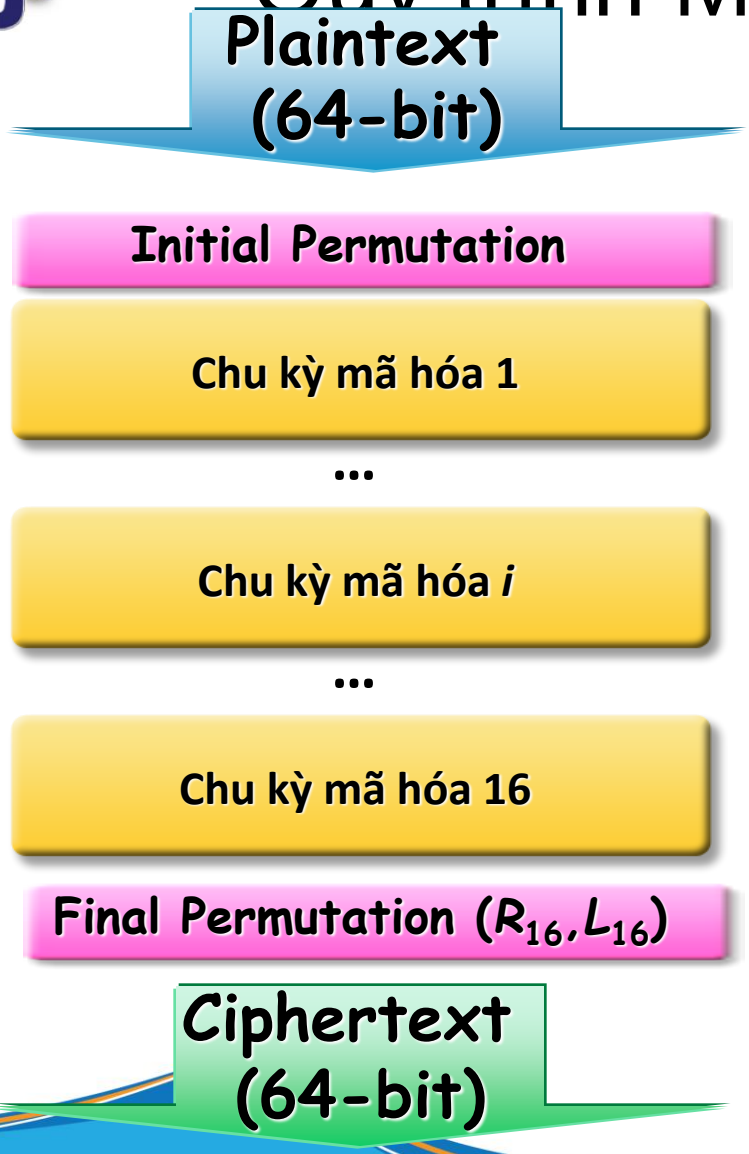
...

Chu kỳ giải mã *1*



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

Quy trình Mã hóa của giải thuật DES



IP: Initial Permutation
FP: Final Permutation

$FP = IP^{-1}$

Ghi chú:
FP và IP không có ý nghĩa về mặt mã hóa, chỉ có tác dụng để nạp dữ liệu vào và ra các khối dữ liệu (theo cơ chế phần cứng giữa thập niên 1970!!!)

Initial Permutation

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
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

Ví dụ: Bit thứ 58 của *x* trở thành bit đầu tiên của IP(x)

Bit thứ 50 của x trở thành bit thứ hai của IP(x)

Final Permutation

IP							
40	8	48	16	56	24	64	32
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

Ví dụ: Bit thứ 58 của *x* trở thành bit đầu tiên của $IP(x)$

Bit thứ 50 của x trở thành bit thứ hai của $IP(x)$

Expansion

Bảng E: quy tắc mở rộng từ 32 bit thành 48 bit

Bảng chọn lựa bit E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Hàm f trong DES

Nửa khối
(32 bit)

Expansion

48 bit

⊕

Sub Key
(48 bit)

S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8

Permutation

S-box

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Ví dụ: $B_j = b_1 b_2 b_3 b_4 b_5 b_6$ thì $S_j(B_i) = S_j[b_1 b_6][b_2 b_3 b_4 b_5]$

S-box

S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-box

S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box

S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

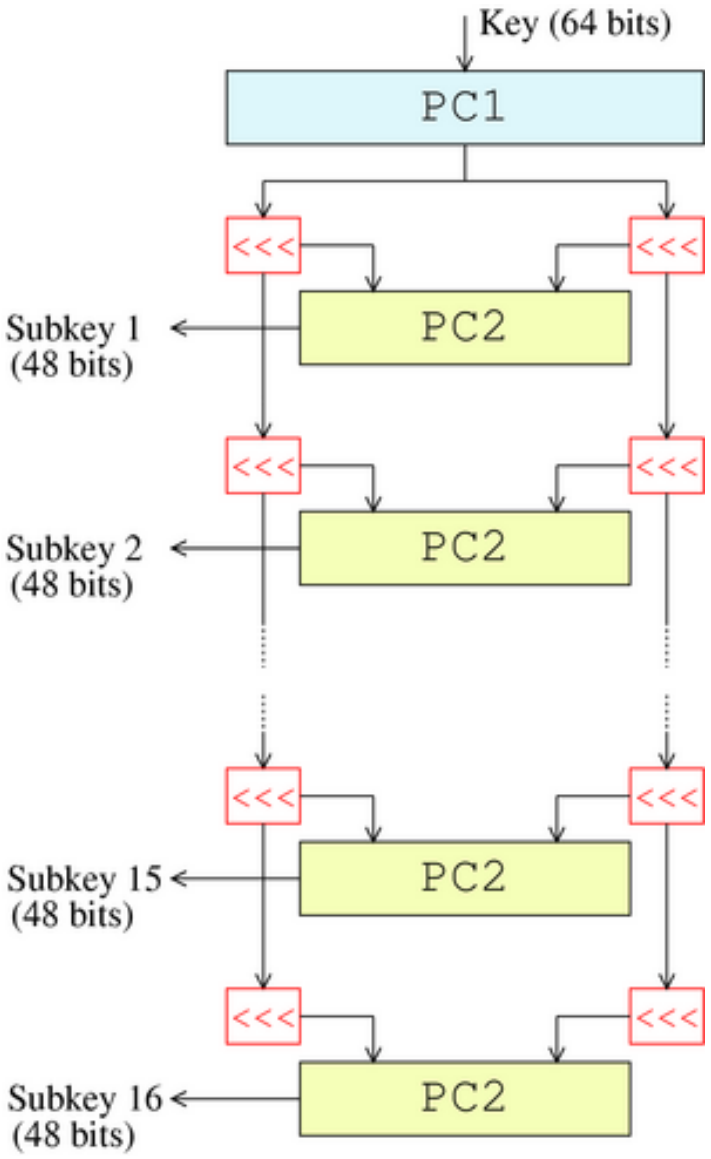
Bảng hoán vị P

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Key Schedule

- Thao tác xoay vòng bit
  - <<<: Xoay vòng sang trái
  - >>>: Xoay vòng sang phải

- Với subkey thứ 1, 2, 9, 16: xoay vòng 1 vị trí
- Với subkey còn lại: xoay vòng 2 vị trí



Các hoán vị trong Key Schedule

PC-1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC-2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Chọn 56 bit
(bỏ bit 8, 16, 24, 32, 40, 48, 56, 64

Chọn 48 bit
(bỏ bit 9, 18, 22, 25, 35, 38, 43, 54)

Một số nhận xét

- 4 khóa yếu (weak key):
  - Gồm toàn bit 0
  - Gồm toàn bit 1
  - Gồm ½ là bit 0 (liên tiếp), ½ là bit 1 (liên tiếp)
- 12 khóa "tương đối yếu" (semi-weak key)
  - Tính chất: $\text{Encrypt}_k(P) = P$
  - Khóa có dạng: 7 bit 0 (liên tiếp), 7 bit 1 (liên tiếp)
- Khóa bù (complement key)
  - $\text{Encrypt}_k(P) = C \rightarrow \text{Encrypt}_{k^*}(P^*) = C^*$
  - Với x^* được tạo bằng cách đảo ngược các bit của x

Advanced Encryption Standard

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Phương pháp mã hóa Rijndael

- Phương pháp Rijndael do Vincent Rijmen và Joan Daeman đề nghị
- Viện Tiêu chuẩn và Công nghệ Hoa Kỳ (National Institute of Standards and Technology – NIST) chọn làm chuẩn mã hóa nâng cao (Advanced Encryption Standard) từ 02 tháng 10 năm 2000

Phương pháp mã hóa Rijndael

- Phương pháp mã hóa theo khối (block cipher) có kích thước khối và mã khóa thay đổi linh hoạt với các giá trị 128, 192 hay 256 bit.
- Phương pháp này thích hợp ứng dụng trên nhiều hệ thống khác nhau từ các thẻ thông minh cho đến các máy tính cá nhân

Một số khái niệm Toán học

- Đơn vị thông tin được xử lý trong thuật toán Rijndael là byte
- Mỗi byte xem như một phần tử của trường Galois GF(2^8) được trang bị phép cộng (ký hiệu $\oplus$) và phép nhân (ký hiệu $\bullet$)
- Mỗi byte được biểu diễn theo nhiều cách khác nhau:
  - Dạng nhị phân: $\{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\}$
  - Dạng thập lục phân: $\{h_1 h_0\}$
  - Dạng đa thức có các hệ số nhị phân

Phép toán trên GF (2^8)

- Phép cộng trên GF(2^8)

  $\{a_7a_6a_5a_4a_3a_2a_1a_0\} \oplus \{b_7b_6b_5b_4b_3b_2b_1b_0\}$

  $= \{c_7c_6c_5c_4c_3c_2c_1c_0\}$

  với $c_i = a_i \oplus b_i$, $0 \leq i \leq 7$

- Phép nhân trên GF(2^8)

  $a(x) \bullet b(x) = a(x) \times b(x) \mod (x^8 + x^4 + x^3 + x + 1)$

Đa thức với hệ số trên $GF(2^8)$

$$a(x) = \sum_{i=0}^{3} a_i x^i \qquad a_i \in \mathrm{GF}(2^8)$$

$$b(x) = \sum_{i=0}^{3} b_i x^i \qquad b_i \in \mathrm{GF}(2^8)$$

$$a(x) + b(x) = \sum_{i=0}^{3} (a_i \oplus b_i) x^i$$

Đa thức với hệ số trên $GF(2^8)$

$$c(x) = a(x) \otimes b(x) = a(x) \bullet b(x) \bmod x^4 + 1$$

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

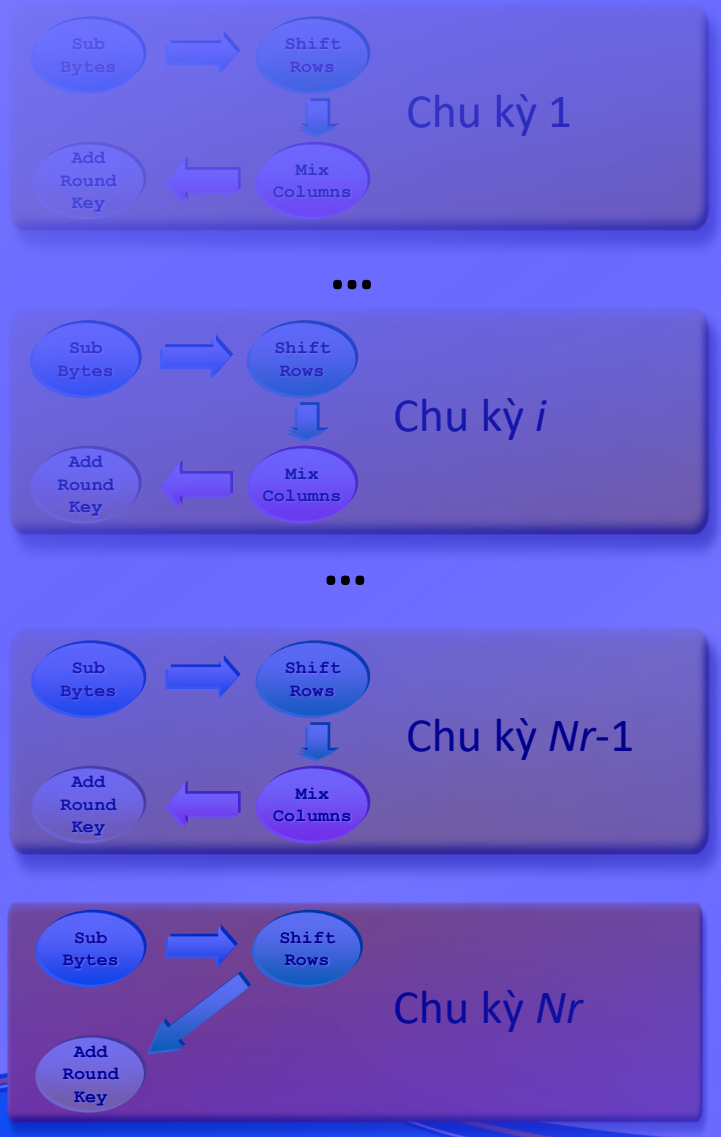
Phương pháp Rijndael

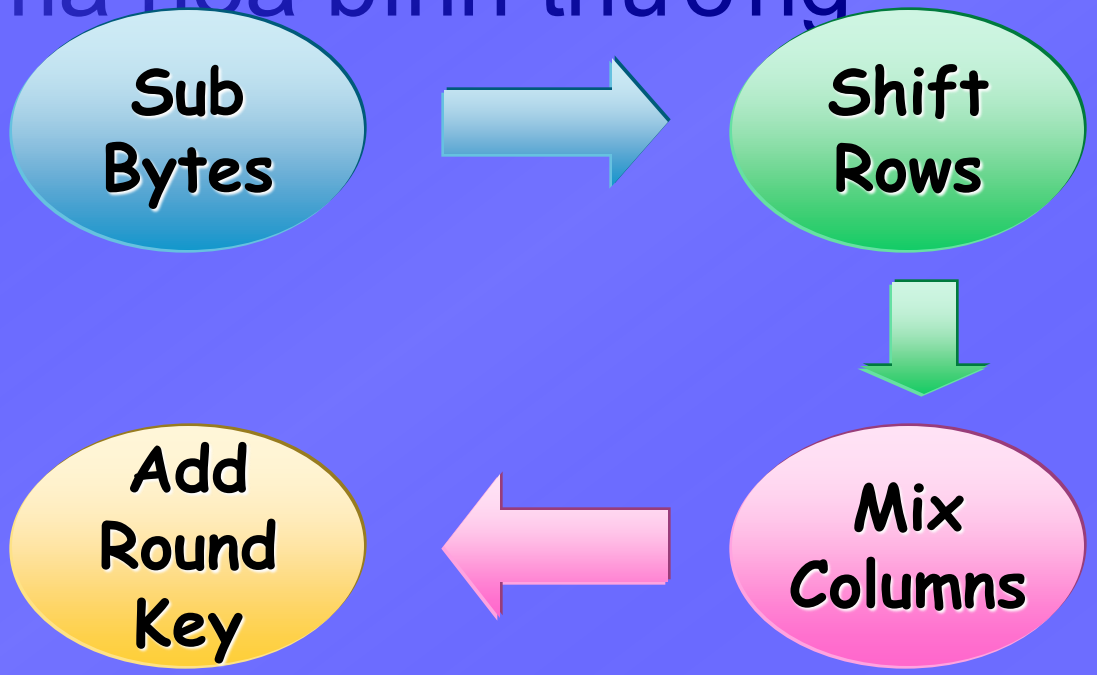
- Kết quả trung gian giữa các bước biến đổi được gọi là trạng thái (state)
- Một trạng thái được biểu diễn dưới dạng ma trận gồm 4 dòng và *Nb* cột với *Nb* bằng độ dài khối chia cho 32
- Mã khóa chính (Cipher Key) được biểu diễn dưới dạng ma trận gồm 4 dòng và *Nk* cột với *Nk* bằng độ dài khóa chia cho 32
- Số lượng chu kỳ $Nr = \max\{Nb, Nk\}+6$

Biểu diễn khối dữ liệu và mã khóa

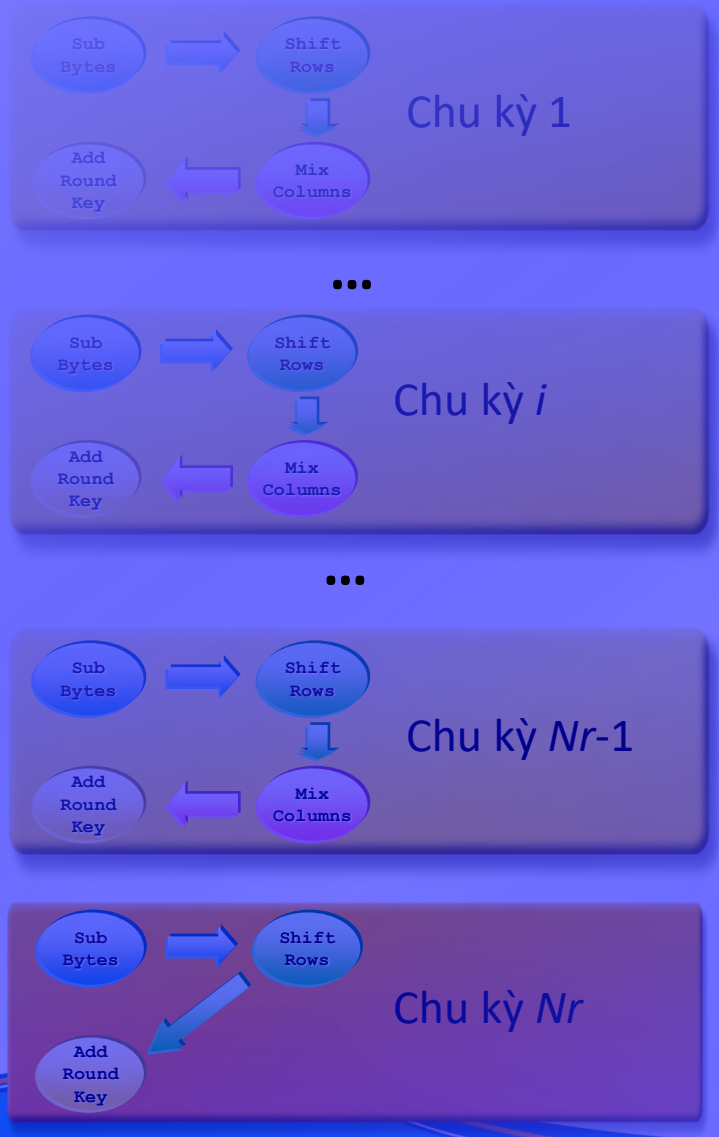
Nb=8

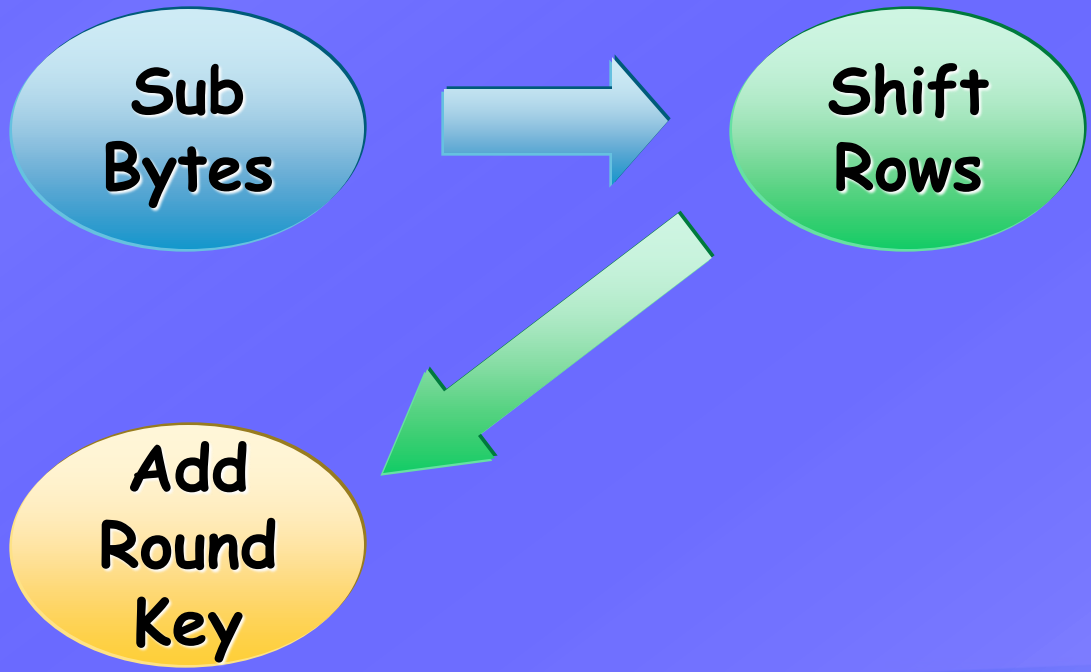
Chu kỳ mã hóa bình thường



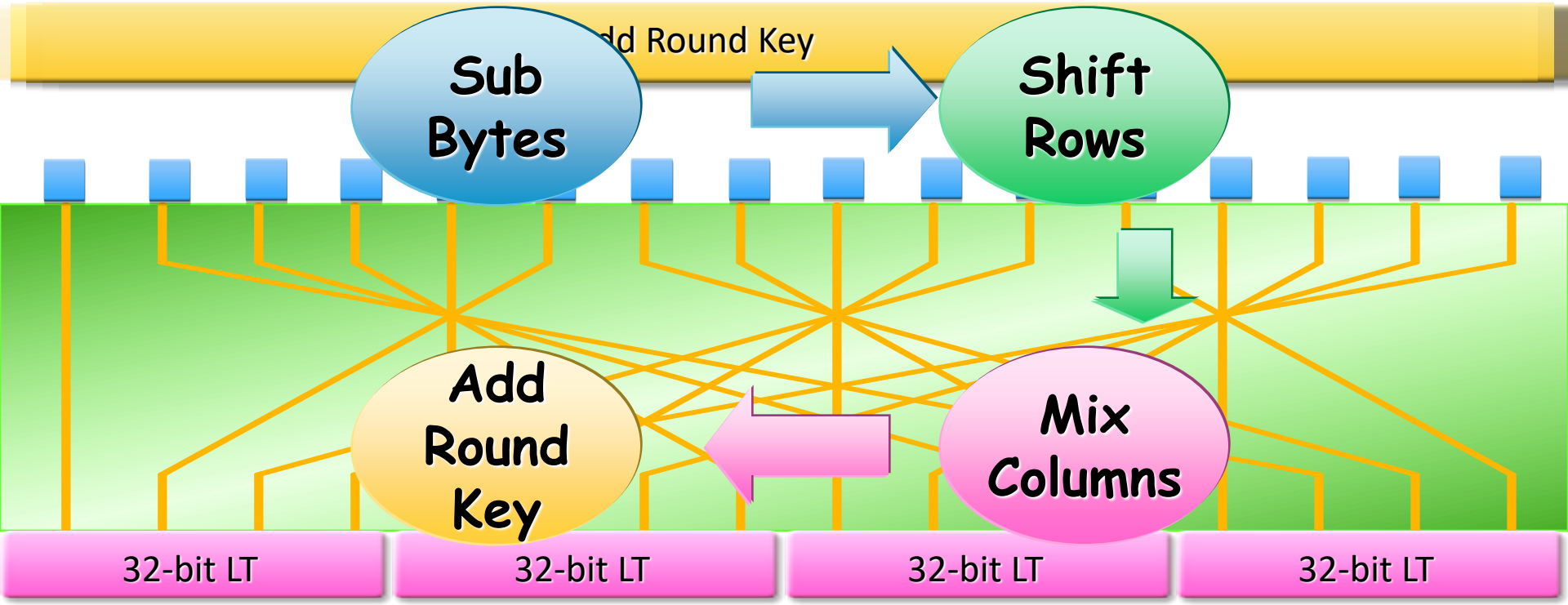


Chu kỳ mã hóa cuối





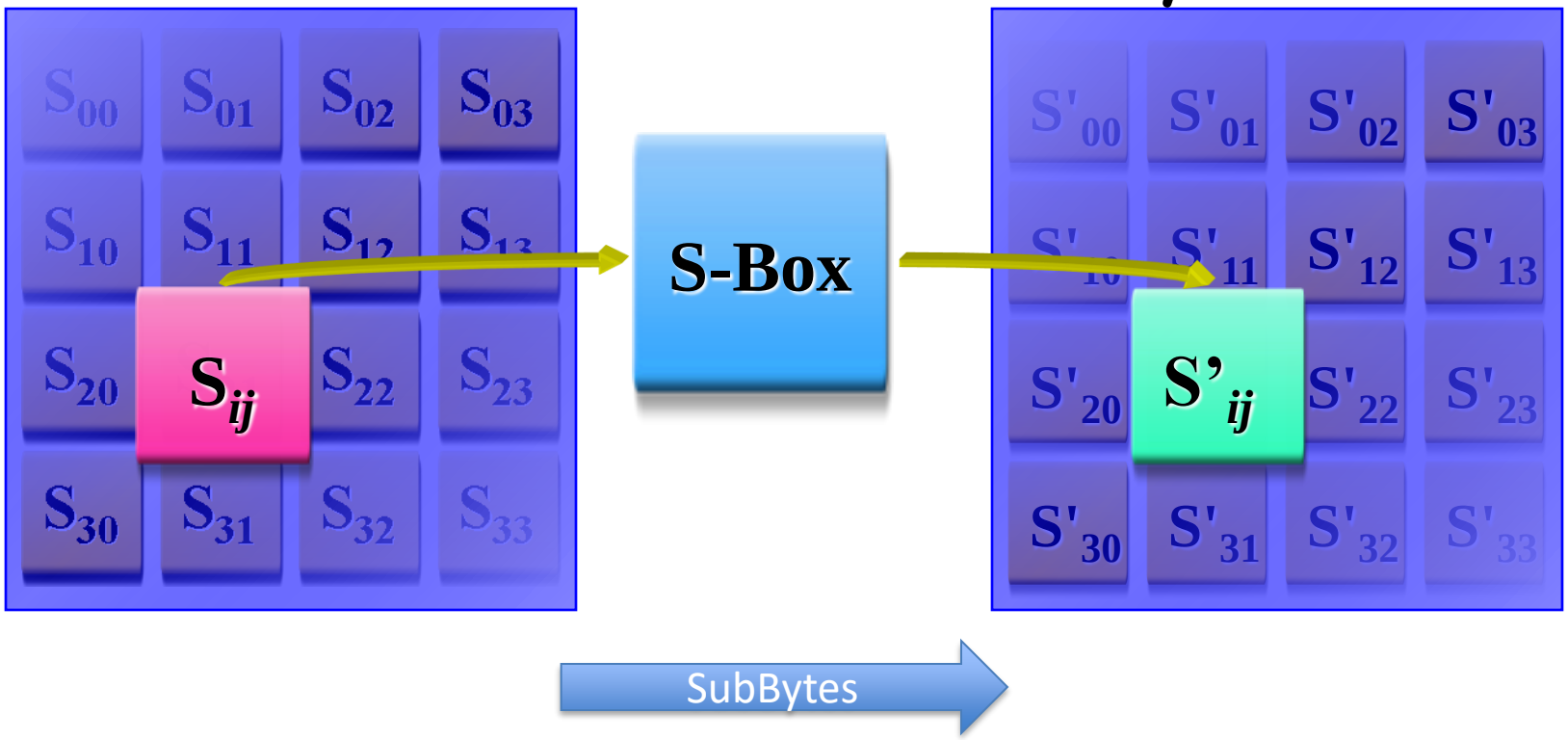
Kiến trúc Substitution-Permutation Network



Quy trình mã hóa của thuật toán Rijndael

```
Cipher( byte in[4 * Nb], byte out[4 * Nb], word w[Nb*(Nr + 1)])
begin
    byte   state[4,Nb]
    state =    in
    AddRoundKey(state, w)
    for round = 1 to Nr – 1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w + round * Nb)
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w + Nr * Nb)
    out = state
end
```

Phép biến đổi SubBytes



Phép biến đổi SubBytes

- Phép thay thế byte phi tuyến thông qua bảng thay thế (S-box)
- Tác động độc lập lên từng byte trong trạng thái hiện hành

Phép biến đổi SubBytes

- Quá trình thay thế byte x trong **SubBytes**:
  - Xác định phần tử nghịch đảo x^{-1} (có biểu diễn nhị phân $\{x_7x_6x_5x_4x_3x_2x_1x_0\}$). Quy ước $\{00\}^{-1} = \{00\}$
  - Phép biến đổi affine:

$$y_i = x_i \oplus x_{(i+4)\bmod 8} \oplus x_{(i+5)\bmod 8} \oplus x_{(i+6)\bmod 8} \oplus x_{(i+7)\bmod 8} \oplus c_i$$

với $\{c_7c_6c_5c_4c_3c_2c_1c_0\} = \{63\}$

Phép biến đổi SubBytes

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Phép biến đổi ngược InvSubBytes

Quá trình thay thế byte *y* trong **InvSubBytes**:

- Phép biến đổi affine:

$$x_i = y_{(i+2)\bmod 8} \oplus y_{(i+5)\bmod 8} \oplus y_{(i+7)\bmod 8} \oplus d_i$$

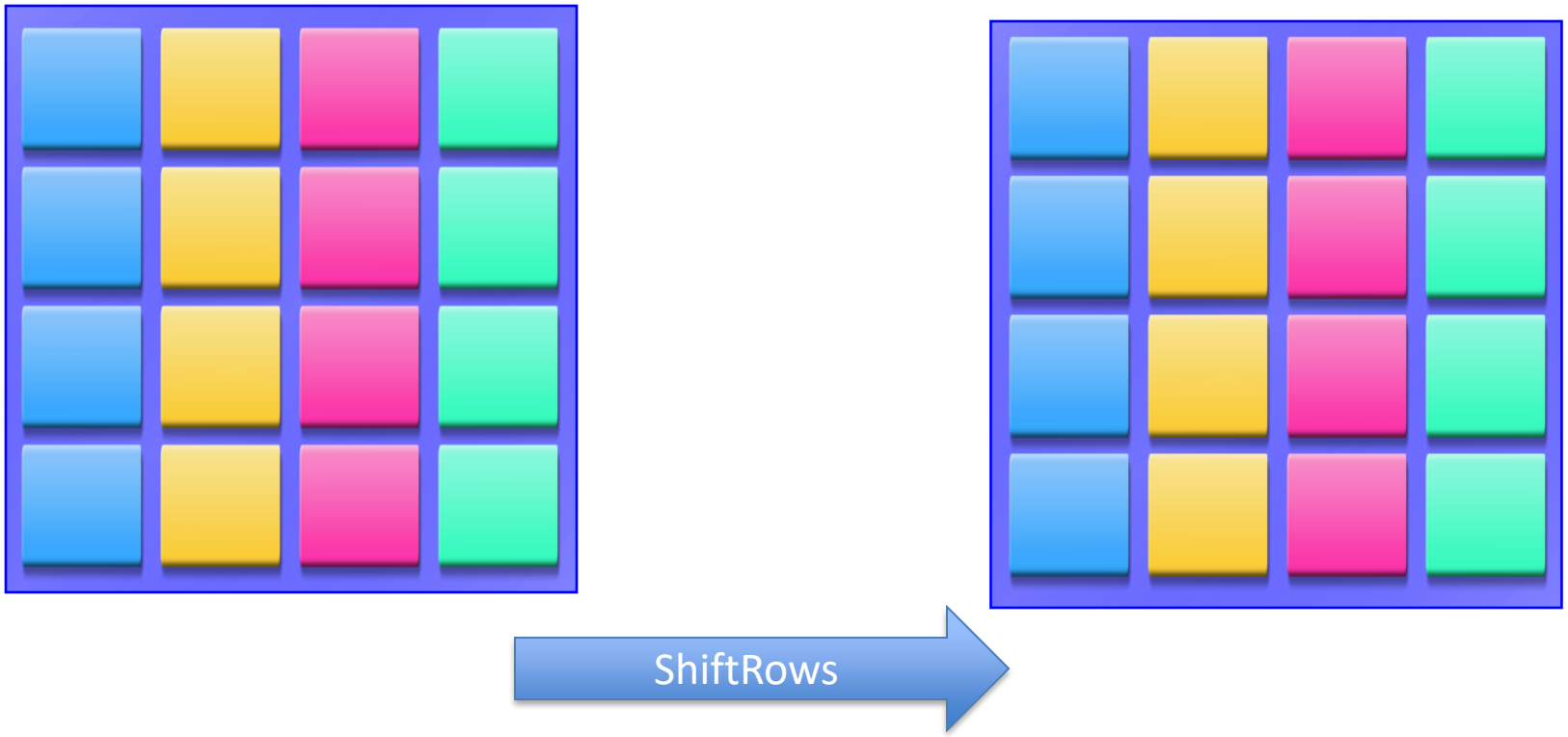
với $\{d_7\, d_6\, d_5\, d_4\, d_3\, d_2\, d_1\, d_0\} = \{05\}$

- Xác định phần tử nghịch đảo $x^{-1} \in GF(2^8)$ của x. Quy ước $\{00\}^{-1} = \{00\}$

Phép biến đổi ngược InvSubBytes

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Phép biến đổi ShiftRows

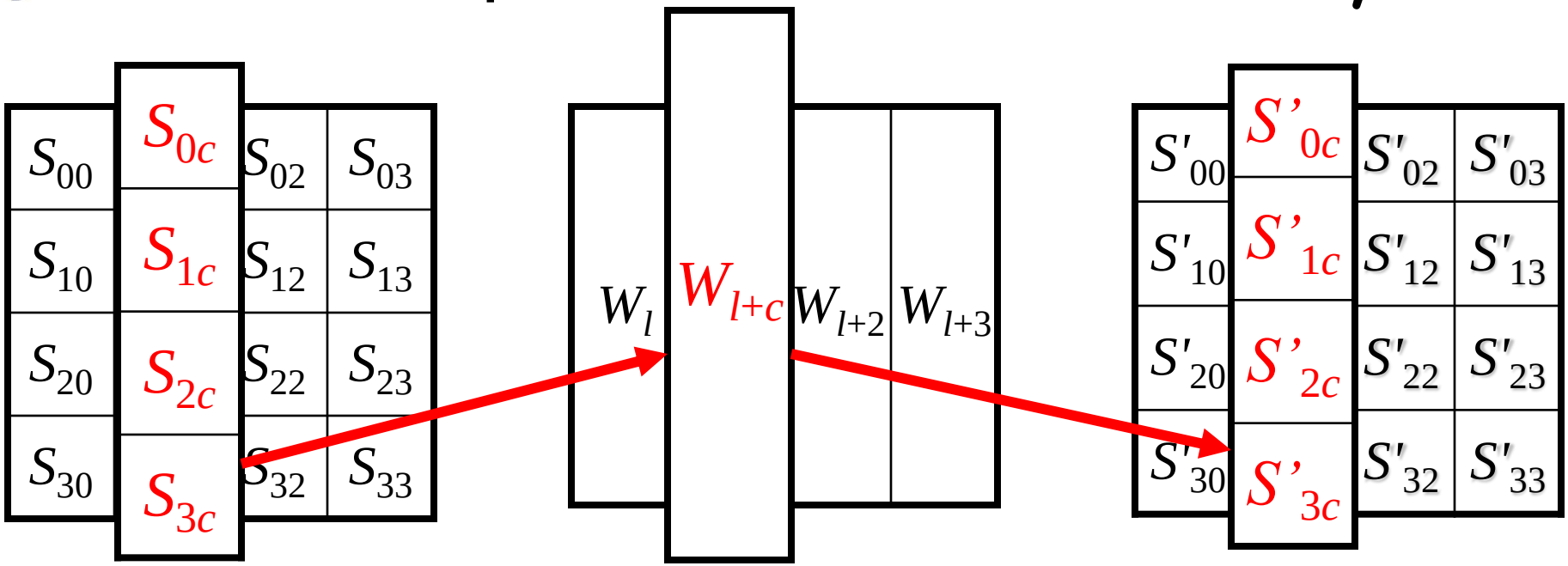


Phép biến đổi ShiftRows

- Mỗi dòng của trạng thái hiện hành được dịch chuyển xoay vòng đi một số vị trí
- Byte $s_{r,c}$ tại dòng r cột c sẽ dịch chuyển đến cột $(c + shift(r, Nb)) \bmod Nb$
- Phép biến đổi ngược **InvShiftRows**: Byte $s_{r,c}$ tại dòng r cột c sẽ dịch chuyển đến cột $(c - shift(r, Nb)) \bmod Nb$

***shift*(*r*, *Nb*)**		***r***		
		1	**2**	**3**
Nb	**4**	**1**	**2**	**3**
	6	**1**	**2**	**3**
	8	**1**	**3**	**4**

Phép biến đổi AddRoundKey



$l = round*Nb$

AddRoundKey

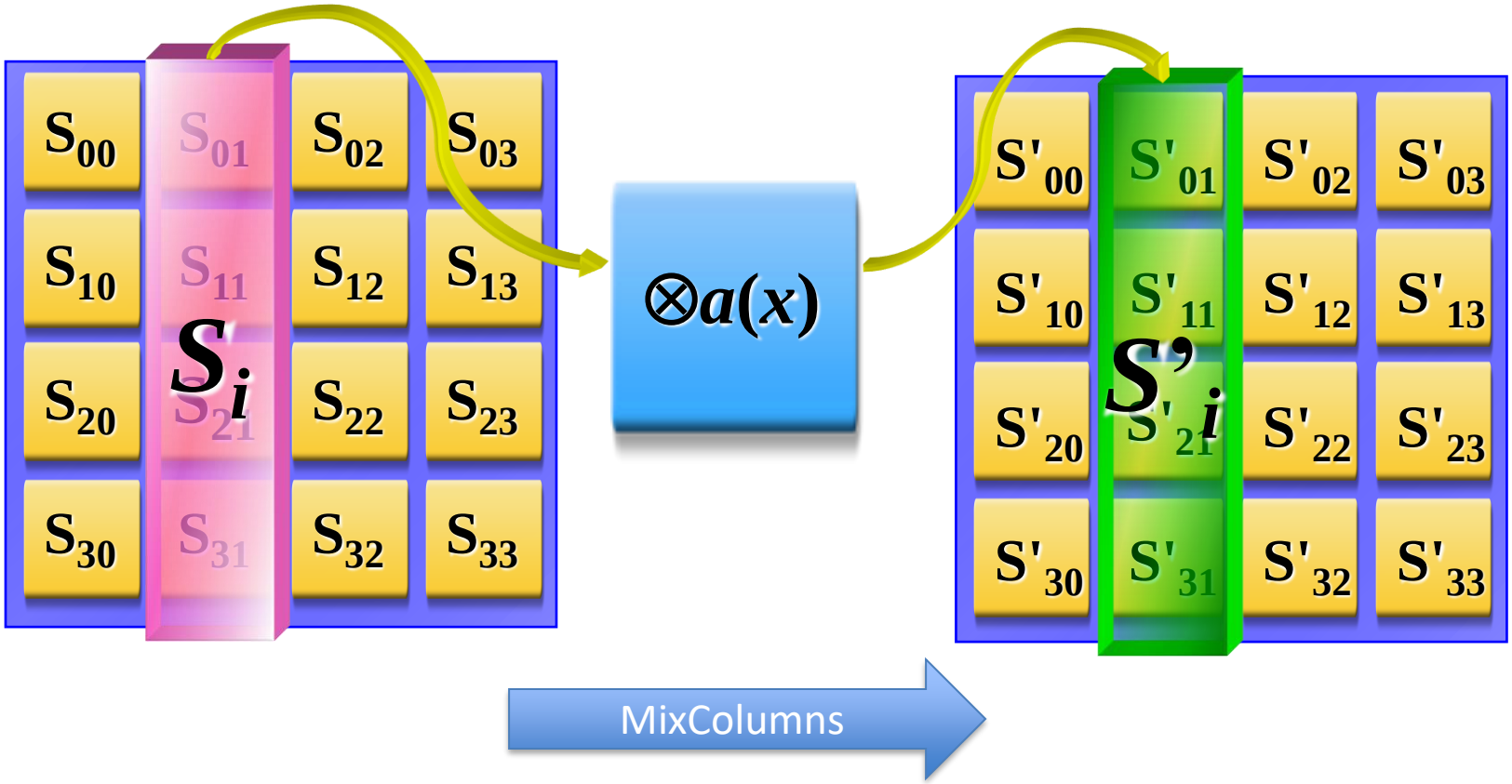
Phép biến đổi AddRoundKey

- Từng byte của trạng thái sẽ được XOR với byte tương ứng trong mã khóa của chu kỳ hiện hành.

$$s'_{r,c} = k_{r,c} \oplus s_{r,c} \text{ , } 0 < r < 8, 0 \leq c < Nb$$

- Phép biến đổi ngược của **AddRoundKey** cũng chính là **AddRoundKey**.

Phép biến đổi MixColumns



Phép biến đổi MixColumns

- Mỗi cột của trạng thái hiện hành được biểu diễn dưới dạng đa thức $s(x)$ có các hệ số trên GF(2^8).
- Thực hiện phép nhân

$$s'(x) = a(x) \otimes s(x)$$

với

$$a(x) = \{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Phép biến đổi ngược InvMixColumns

- Mỗi cột của trạng thái hiện hành được biểu diễn dưới dạng đa thức $s(x)$ có các hệ số trên GF(2^8).
- Thực hiện phép nhân

$$s'(x) = a^{-1}(x) \otimes s(x)$$

với

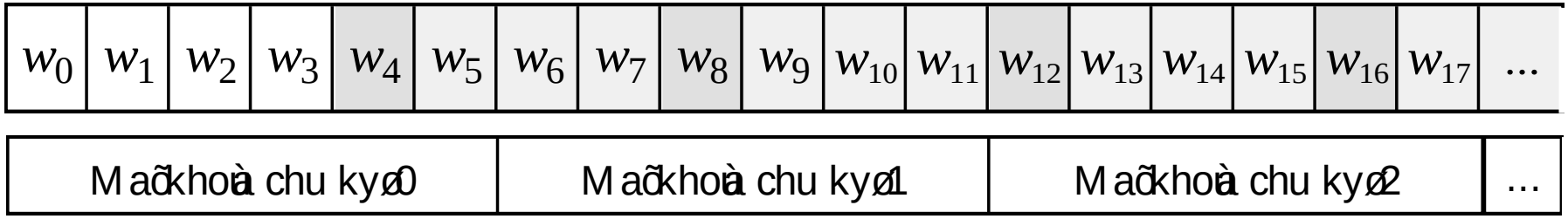
$$a^{-1}(x) = \{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Phát sinh mã khóa mỗi chu kỳ

- Bảng mã khóa mở rộng là mảng 1 chiều chứa các từ (có độ dài 4 byte)
- Hàm phát sinh bảng mã khóa mở rộng phụ thuộc vào giá trị *Nk*, tức là phụ thuộc vào độ dài của mã khóa chính

Phát sinh mã khóa mỗi chu kỳ



Bảng mã khóa mở rộng và cách xác định mã khóa của chu kỳ (*Nb*=6, *Nk*=4)

Phát sinh mã khóa mỗi chu kỳ

```
KeyExpansion(byte key[4 * Nk], word w[Nb * (Nr + 1)], Nk)
begin
    i=0
    while (i < Nk)
        w[i] = word[key[4*i], key[4*i+1],  key[4*i+2],key[4*i+3]]
        i = i + 1
    end while
    i = Nk
    while (i < Nb * (Nr + 1))
        word temp = w[i - 1]
        if  (i mod Nk = 0) then
            temp = SubWord(RotWord(temp)) xor Rcon[i / Nk]
        else
            if (Nk = 8) and (i mod Nk = 4) then
                temp = SubWord(temp)
        end if
        w[i] = w[i - Nk] xor temp
        i = i + 1
    end while
end
```

Phát sinh mã khóa mỗi chu kỳ

- Rcon[*i*] = (RC[*i*], {00}, {00}, {00}) với RC[*i*]$\in$GF(2^8)
- RC[1]=1 ({01})
- RC[*i*] =*x* ({02})•(RC[*i*-1]) = $x^{(i-1)}$

Sbox		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	B5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	Bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Sbox^{-1}		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d