Chương 7

Chuẩn Mã hóa Cấp cao (AES)

Chương 7

Mục tiêu

Để xem lại lịch sử ngắn gọn của AES

Để xác định cấu trúc cơ bản của AES

Để xác định các phép biến đổi được AES sử dụng

Để xác định quá trình mở rộng khóa

Để thảo luận về các cách triển khai khác nhau

7-1 GIỚI THIỆU

Tiêu chuẩn mã hóa nâng cao (AES) là mật mã khối khóa đối xứng được xuất bản bởi Viện Tiêu chuẩn và Công nghệ Quốc gia (NIST) vào tháng 12 năm 2001.

Các chủ đề được thảo luận trong phần này:

7.1.1 Lịch sử 7.1.2 Tiêu chí

7.1.3 Vòng đấu

7.1.4 Đơn vị dữ liệu

7.1.5 Cấu trúc của mỗi vòng

7.1.1 Lịch sử.

Vào tháng 2 năm 2001, NIST thông báo rằng bản dự thảo của Tiêu chuẩn Xử lý Thông tin Liên bang (FIPS) đã có sẵn để công chúng xem xét và bình luận. Cuối cùng, AES được xuất bản với tên FIPS 197 trong Cục Đăng ký Liên bang vào tháng 12 năm 2001.

7.1.2 Tiêu chí

Các tiêu chí do NIST xác định để lựa chọn AES rơi vào ba lĩnh vực: 1. Bảo mật 2. Chi phí 3.

Triển khai.

7.1.3 Vòng đấu.

AES là một mật mã không phải Feistel mã hóa và giải mã khối dữ liệu 128 bit. Nó sử dụng 10, 12 hoặc 14 vòng. Kích thước khóa có thể là 128, 192 hoặc 256 bit, tùy thuộc vào số vòng.

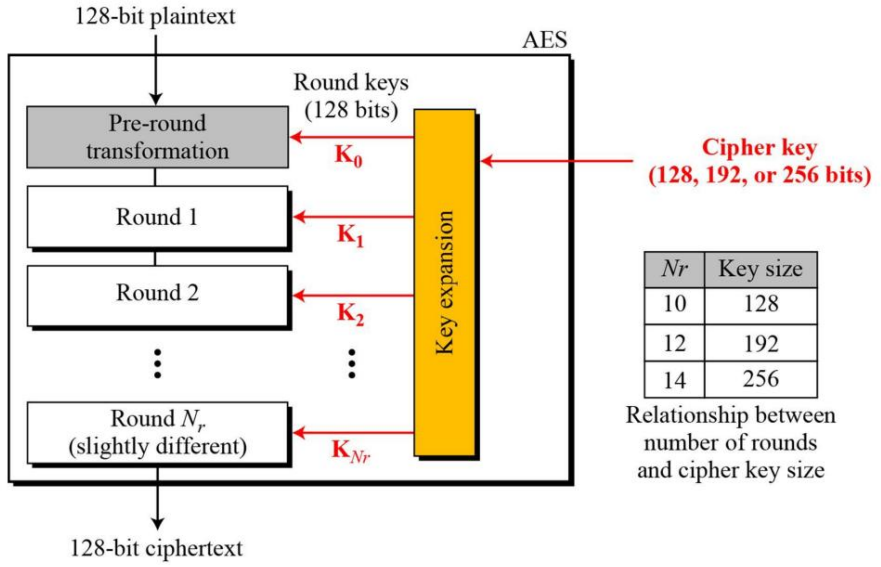
Ghi chú

AES đã xác định ba phiên bản, với 10, 12 và 14 vòng.

Mỗi phiên bản sử dụng kích thước khóa mật mã khác nhau (128, 192 hoặc 256), nhưng khóa tròn luôn là 128 bit.

7.1.3 Tiếp tục

Hình 7.1 Thiết kế chung của mật mã mã hóa AES

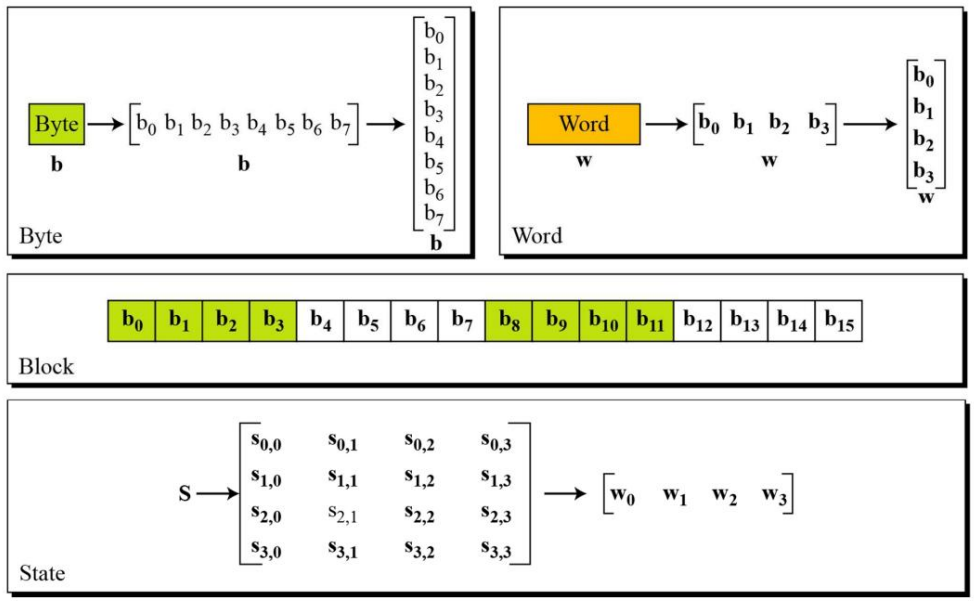


Nr	Key size
10	128
12	192
14	256

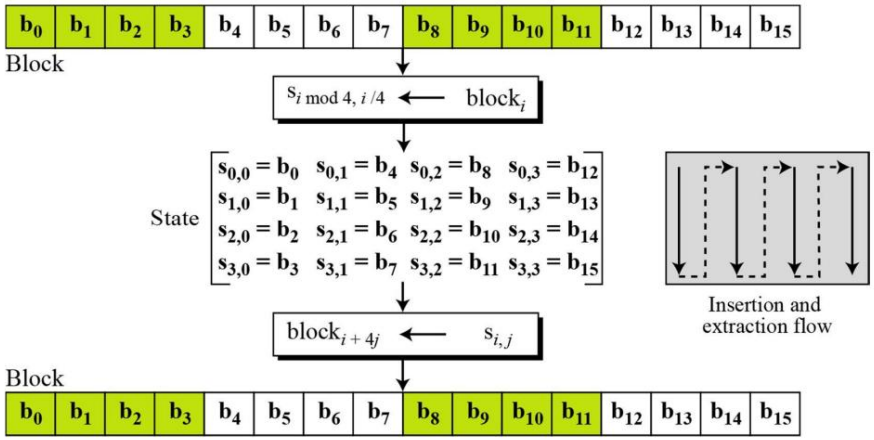
Relationship between number of rounds and cipher key size

7.1.4 Đơn vị dữ liệu.

Hình 7.2 Đơn vị dữ liệu được sử dụng trong AES

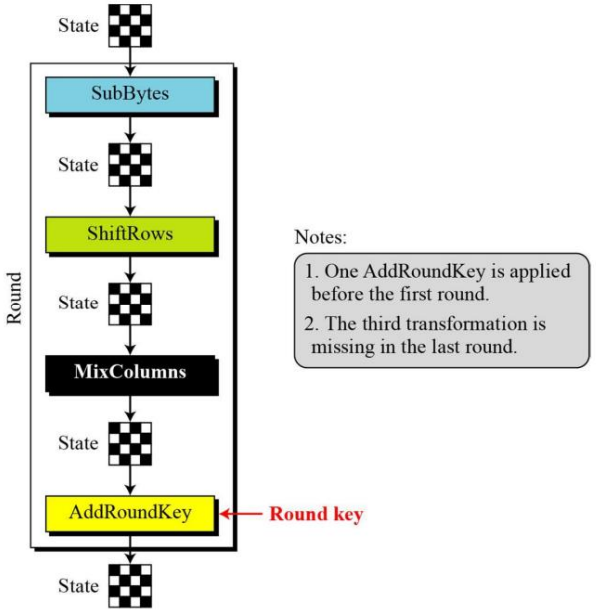


Hình 7.3 Chuyển đổi khối sang trạng thái và trạng thái thành khối



7.1.5 Cấu trúc của mỗi vòng

Hình 7.5 Cấu trúc của mỗi vòng tại vị trí mã hóa



Ví dụ 7.1 Tiếp tục

Giả sử khối văn bản là “AES sử dụng ma trận”

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$ State

Hình 7.4 Thay đổi bản rõ sang trạng thái

7-2 BIẾN ĐỔI

Để cung cấp bảo mật, AES sử dụng bốn loại chuyển đổi: thay thế, hoán vị, trộn và thêm khóa.

Các chủ đề được thảo luận trong phần này:

7.2.1 Thay thế
7.2.2 Hoán vị
7.2.3 Trộn
7.2.4 Thêm khóa

7.2.1 Thay thế

AES, giống như DES, sử dụng sự thay thế. AES sử dụng hai phép biến đổi khả nghịch.

SubBytes

Phép biến đổi đầu tiên, SubBytes, được sử dụng tại trang mã hóa. Để thay thế một byte, chúng tôi hiểu byte đó là hai chữ số thập lục phân.

Ghi chú

Hoạt động SubBytes bao gồm 16 phép biến đổi từng byte độc lập.

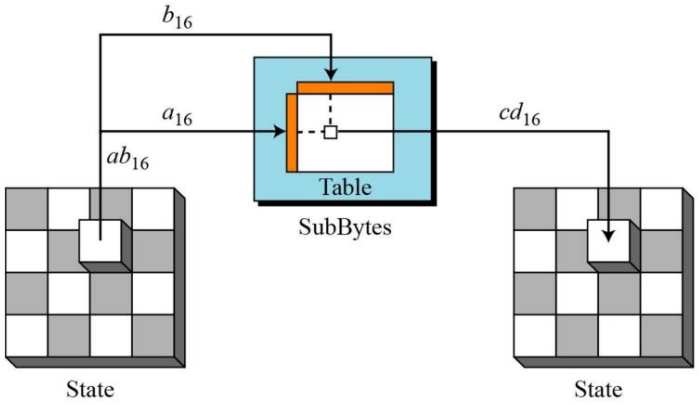
7.2.1 Tiếp tục

Table 7.1 *SubBytes transformation table*

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

7.2.1 Tiếp tục

Hình 7.6 Chuyển đổi SubByte



Byte dưới dạng hai chữ số thập lục phân. Chữ số bên trái xác định hàng và chữ số bên phải xác định cột của bảng thay thế

7.2.1 Continue

InvSubBytes là nghịch đảo của SubBytes

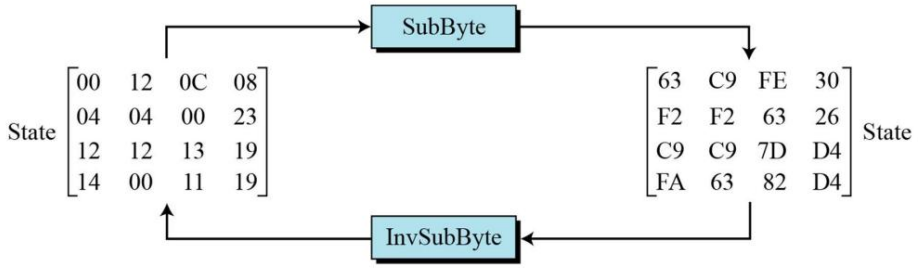
Table 7.2 *InvSubBytes transformation table*

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

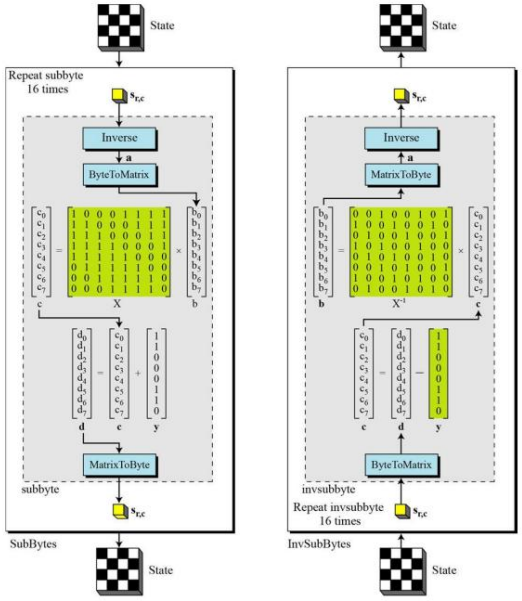
Ví dụ 7.2

Hình 7.7 cho thấy cách một trạng thái được chuyển đổi bằng cách sử dụng phép biến đổi SubBytes. Hình này cũng cho thấy phép biến đổi InvSubBytes tạo ra phép biến đổi ban đầu. Lưu ý rằng nếu hai byte có cùng giá trị thì phép biến đổi của chúng cũng giống nhau.

Hình 7.7 Chuyển đổi SubByte cho Ví dụ 7.2



Hình 7.8 Các tiến trình SubBytes và InvSubBytes



Phép biến đổi sử dụng trường GF(28) AES cũng xác định phép biến đổi đại số bằng cách sử dụng trường GF(28) với các đa thức tối giản (x8 + x4 + x3+ x + 1), như trong Hình 7.8.

$$\text{subbyte:} \quad \rightarrow \quad \mathbf{d} = \mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y}$$
$$\text{invsubbyte:} \quad \rightarrow \quad [\mathbf{X}^{-1}(\mathbf{d} \oplus \mathbf{y})]^{-1} = [\mathbf{X}^{-1}(\mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y} \oplus \mathbf{y})]^{-1} = [(s_{r,c})^{-1}]^{-1} = s_{r,c}$$

Ghi chú

Các phép biến đổi SubBytes và InvSubBytes là nghịch đảo của nhau.

Hãy để chúng tôi chỉ ra cách byte 0C được chuyển đổi thành FE bằng quy trình byte con và được chuyển đổi trở lại 0C bằng quy trình invsubbyte.

7.2.1 Tiếp tục

Algorithm 7.1 *Pseudocode for SubBytes transformation*

```
SubBytes (S)
{
   for (r = 0 to 3)
     for (c = 0 to 3)
           S_r,c = subbyte (S_r,c)
}

subbyte (byte)
{
   a ← byte^-1              // Multiplicative inverse in GF(2^8) with inverse of 00 to be 00
    ByteToMatrix (a, b)
    for (i = 0 to 7)
    {
        c_i ← b_i ⊕ b_(i+4)mod 8 ⊕ b_(i+5)mod 8 ⊕ b_(i+6)mod 8 ⊕ b_(i+7)mod 8
        d_i ← c_i ⊕ ByteToMatrix (0x63)
    }
    MatrixToByte (d, d)
    byte ← d
}
```

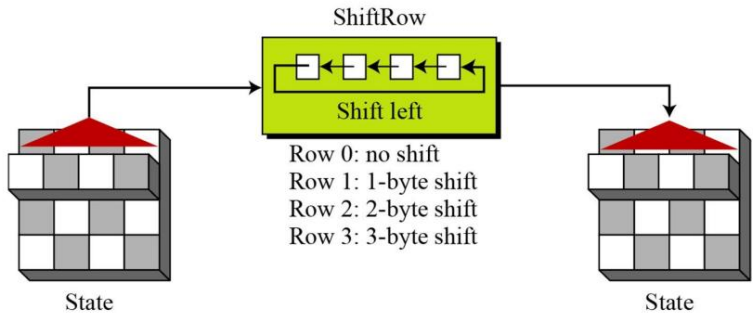

7.2.1 Tiếp tục

Một phép biến đổi khác được tìm thấy trong một vòng là phép dịch chuyển, phép này hoán vị các byte.

ShiftRows

Trong quá trình mã hóa, phép biến đổi được gọi là ShiftRows.

Hình 7.9 Phép biến đổi ShiftRows



7.2.2 Tiếp tục

InvShiftRows

Trong quá trình giải mã, phép biến đổi được gọi là InvShiftRows và phép dịch chuyển sang phải.

Algorithm 7.2 *Pseudocode for ShiftRows transformation*

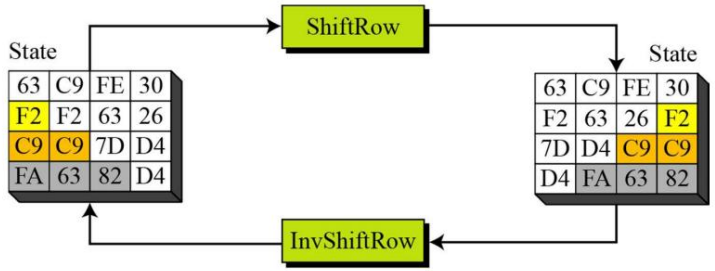
```
ShiftRows (S)
{
    for (r = 1 to 3)
        shiftrow (s_r, r)                 // s_r is the rth row
}

shiftrow (row, n)                         // n is the number of bytes to be shifted
{
   CopyRow (row, t)                       // t is a temporary row
   for (c = 0 to 3)
        row_(c – n) mod 4 ← t_c
}
```


7.2.2 Tiếp tục

Ví dụ 7.4

Hình 7.10 cho thấy cách chuyển đổi một trạng thái bằng cách sử dụng phép biến đổi ShiftRows. Hình cũng cho thấy phép biến đổi InvShiftRows tạo ra trạng thái ban đầu.

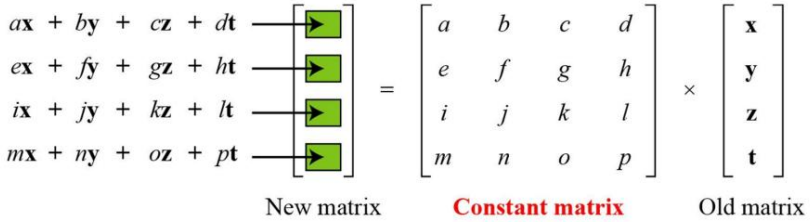
Hình 7.10 Phép biến đổi ShiftRows trong Ví dụ 7.4



7.2.3 Trộn

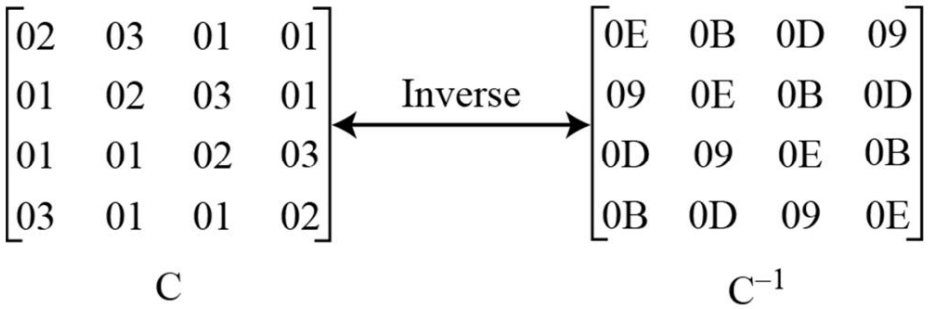
Chúng ta cần một phép biến đổi xen kẽ để thay đổi các bit bên trong một byte, dựa trên các bit bên trong các byte lân cận. Chúng ta cần trộn các byte để cung cấp khả năng khuếch tán ở cấp độ bit.

Hình 7.11 Trộn byte bằng phép nhân ma trận



7.2.3 Tiếp tục

Hình 7.12 Các ma trận không đổi được MixColumns và InvMixColumns sử dụng

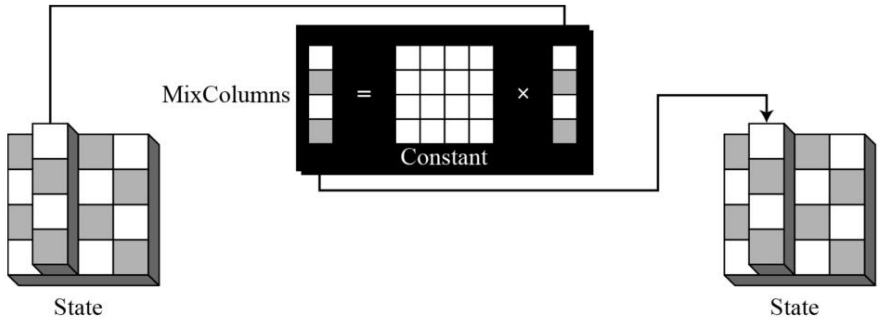


7.2.3 Tiếp tục

MixColumns Phép

biến đổi MixColumns hoạt động ở cấp độ cột; nó chuyển đổi từng cột của trạng thái thành một cột mới.

Hình 7.13 Phép biến đổi MixColumns



7.2.3 Tiếp tục

InvMixColumns Phép

biến đổi InvMixColumns về cơ bản giống như phép biến đổi MixColumns.

Ghi chú

Các phép biến đổi MixColumns và InvMixColumns là nghịch đảo của nhau.

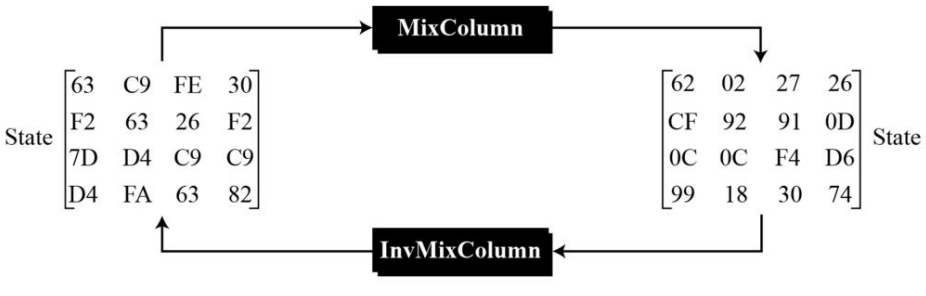
Algorithm 7.3 *Pseudocode for MixColumns transformation*

```
MixColumns (S)
{
    for (c = 0 to 3)
        mixcolumn (s_c)
}
mixcolumn (col)
{
    CopyColumn (col, t)                    // t is a temporary column
    col_0 ← (0x02) • t_0 ⊕ (0x03 • t_1) ⊕ t_2 ⊕ t_3
    col_1 ← t_0 ⊕ (0x02) • t_1 ⊕ (0x03) • t_2 ⊕ t_3
    col_2 ← t_0 ⊕ t_1 ⊕ (0x02) • t_2 ⊕ (0x03) • t_3
    col_3 ← (0x03 • t_0) ⊕ t_1 ⊕ t_2 ⊕ (0x02) • t_3
}
```


Ví dụ 7.5

Hình 7.14 cho thấy cách chuyển đổi một trạng thái bằng cách sử dụng phép biến đổi MixColumns. Hình này cũng cho thấy phép biến đổi InvMixColumns tạo ra phép biến đổi ban đầu.

Hình 7.14 Phép biến đổi MixColumns trong Ví dụ 7.5



7.2.4 Thêm khóa

AddRoundKey

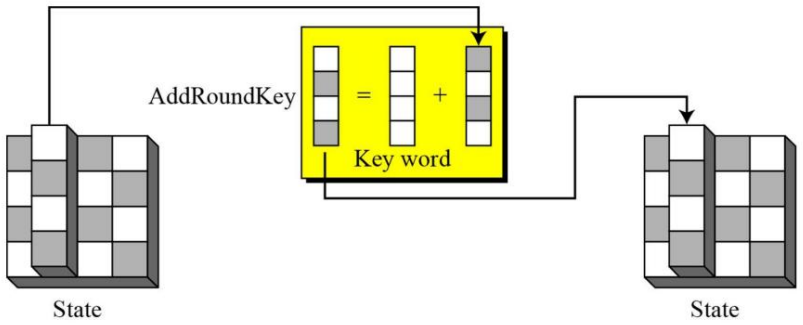
AddRoundKey tiến hành mỗi lần một cột.

AddRoundKey thêm một từ khóa tròn vào mỗi ma trận cột trạng thái; phép toán trong AddRoundKey là phép cộng ma trận.

Ghi chú

Phép biến đổi AddRoundKey là nghịch đảo của chính nó.

Hình 7.15 phép biến đổi AddRoundKey



Algorithm 7.4 *Pseudocode for AddRoundKey transformation*

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_round + 4c
}
```

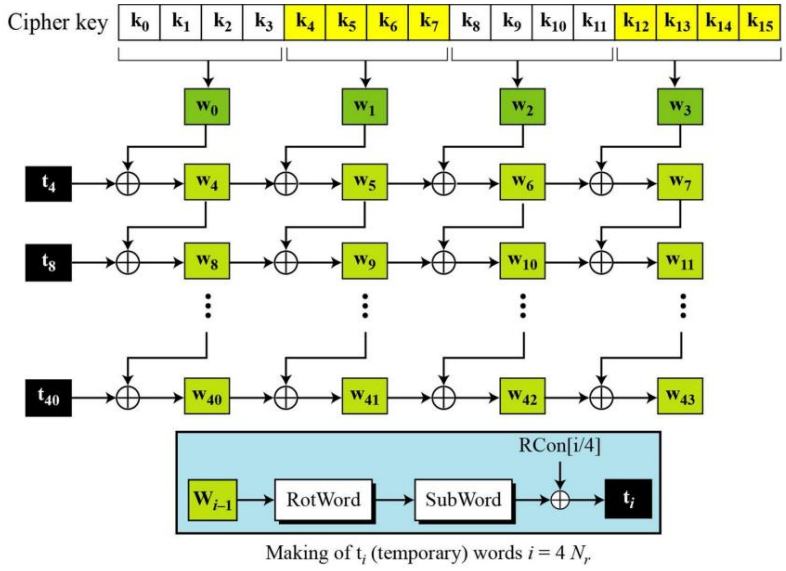
7-3 KHÓA MỞ RỘNG

Để tạo khóa tròn cho mỗi vòng, AES sử dụng quy trình mở rộng khóa. Nếu số vòng là Nr thì quy trình mở rộng khóa sẽ tạo Nr + 1 khóa tròn 128 bit từ một khóa mật mã 128 bit duy nhất.

Các chủ đề được thảo luận trong phần này: 7.3.1 Mở rộng khóa trong AES-128 7.3.2 Mở rộng khóa trong AES-192 và AES-256 7.3.3 Phân tích mở rộng khóa

7.3.1 Mở rộng khóa trong AES-128

Hình 7.16 Mở rộng khóa trong AES



Making of t_i (temporary) words $i = 4\ N_r$

7-3 Tiếp tục

Table 7.3 *Words for each round*

Round	*Words*			
Pre-round	w_0	w_1	w_2	w_3
1	w_4	w_5	w_6	w_7
2	w_8	w_9	w_{10}	w_{11}
. . .	. . .			
N_r	w_{4N_r}	w_{4N_r+1}	w_{4N_r+2}	w_{4N_r+3}

7.3.1 Tiếp tục

Table 7.4 *RCon constants*

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	(**01** 00 00 00)$_{16}$	6	(**20** 00 00 00)$_{16}$
2	(**02** 00 00 00)$_{16}$	7	(**40** 00 00 00)$_{16}$
3	(**04** 00 00 00)$_{16}$	8	(**80** 00 00 00)$_{16}$
4	(**08** 00 00 00)$_{16}$	9	(**1B** 00 00 00)$_{16}$
5	(**10** 00 00 00)$_{16}$	10	(**36** 00 00 00)$_{16}$

Thủ tục mở rộng khóa có thể sử dụng bảng trên khi tính toán các từ hoặc sử dụng trường GF(28) để tính toán động byte ngoài cùng bên trái, như được hiển thị bên dưới (số nguyên tố là đa thức tối giản):

$$
\begin{array}{llllll}
RC_1 & \rightarrow x^{1-1} & = x^0 & \text{mod } prime & = 1 & \rightarrow 00000001 \rightarrow 01_{16} \\
RC_2 & \rightarrow x^{2-1} & = x^1 & \text{mod } prime & = x & \rightarrow 00000010 \rightarrow 02_{16} \\
RC_3 & \rightarrow x^{3-1} & = x^2 & \text{mod } prime & = x^2 & \rightarrow 00000100 \rightarrow 04_{16} \\
RC_4 & \rightarrow x^{4-1} & = x^3 & \text{mod } prime & = x^3 & \rightarrow 00001000 \rightarrow 08_{16} \\
RC_5 & \rightarrow x^{5-1} & = x^4 & \text{mod } prime & = x^4 & \rightarrow 00010000 \rightarrow 10_{16} \\
RC_6 & \rightarrow x^{6-1} & = x^5 & \text{mod } prime & = x^5 & \rightarrow 00100000 \rightarrow 20_{16} \\
RC_7 & \rightarrow x^{7-1} & = x^6 & \text{mod } prime & = x^6 & \rightarrow 01000000 \rightarrow 40_{16} \\
RC_8 & \rightarrow x^{8-1} & = x^7 & \text{mod } prime & = x^7 & \rightarrow 10000000 \rightarrow 80_{16} \\
RC_9 & \rightarrow x^{9-1} & = x^8 & \text{mod } prime & = x^4 + x^3 + x + 1 & \rightarrow 00011011 \rightarrow 1B_{16} \\
RC_{10} & \rightarrow x^{10-1} & = x^9 & \text{mod } prime & = x^5 + x^4 + x^2 + x & \rightarrow 00110110 \rightarrow 36_{16}
\end{array}
$$

Algorithm 7.5 *Pseudocode for key expansion in AES-128*

```
KeyExpansion ([key0 to key15], [w0 to w43])
{
    for (i = 0 to 3)
        wi ← key4i + key4i+1 + key4i+2 + key4i+3

    for (i = 4 to 43)
    {
        if (i mod 4 ≠ 0)    wi ← wi−1 + wi−4
        else
        {
            t ← SubWord (RotWord (wi−1)) ⊕ RCon i/4        // t is a temporary word
            wi ← t + wi−4
        }
    }
}
```


Ví dụ 7.6

Bảng 7.5 cho thấy cách tính toán các khóa cho mỗi vòng với giả định rằng khóa mật mã 128-bit được Alice và Bob thỏa thuận là (24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87)16.

Table 7.5 *Key expansion example*

Round	*Values of t's*	*First word in the round*	*Second word in the round*	*Third word in the round*	*Fourth word in the round*
—		$w_{00} = 2475A2B3$	$w_{01} = 34755688$	$w_{02} = 31E21200$	$w_{03} = 13AA5487$
1	AD20177D	$w_{04} = 8955B5CE$	$w_{05} = BD20E346$	$w_{06} = 8CC2F146$	$w_{07} = 9F68A5C1$
2	470678DB	$w_{08} = CE53CD15$	$w_{09} = 73732E53$	$w_{10} = FFB1DF15$	$w_{11} = 60D97AD4$
3	31DA48D0	$w_{12} = FF8985C5$	$w_{13} = 8CFAAB96$	$w_{14} = 734B7483$	$w_{15} = 2475A2B3$
4	47AB5B7D	$w_{16} = B822deb8$	$w_{17} = 34D8752E$	$w_{18} = 479301AD$	$w_{19} = 54010FFA$
5	6C762D20	$w_{20} = D454F398$	$w_{21} = E08C86B6$	$w_{22} = A71F871B$	$w_{23} = F31E88E1$
6	52C4F80D	$w_{24} = 86900B95$	$w_{25} = 661C8D23$	$w_{26} = C1030A38$	$w_{27} = 321D82D9$
7	E4133523	$w_{28} = 62833EB6$	$w_{29} = 049FB395$	$w_{30} = C59CB9AD$	$w_{31} = F7813B74$
8	8CE29268	$w_{32} = EE61ACDE$	$w_{33} = EAFE1F4B$	$w_{34} = 2F62A6E6$	$w_{35} = D8E39D92$
9	0A5E4F61	$w_{36} = E43FE3BF$	$w_{37} = 0EC1FCF4$	$w_{38} = 21A35A12$	$w_{39} = F940C780$
10	3FC6CD99	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

Ví dụ 7.7

Mỗi khóa vòng trong AES phụ thuộc vào khóa vòng trước đó.

Tuy nhiên, sự phụ thuộc là phi tuyến tính do chuyển đổi SubWord. Việc bổ sung các hằng số tròn cũng đảm bảo rằng mỗi khóa tròn sẽ khác với khóa trước đó.

Ví dụ 7.8

Hai bộ khóa tròn có thể được tạo từ hai khóa mật mã chỉ khác nhau một bit.

Cipher Key 1: 12 45 A2 A1 23 31 A4 A3 B2 CC A**A** 34 C2 BB 77 23
Cipher Key 2: 12 45 A2 A1 23 31 A4 A3 B2 CC A**B** 34 C2 BB 77 23

Tiếp tục

Table 7.6 *Comparing two sets of round keys*

R.	Round keys for set 1	Round keys for set 2	B. D.
—	1245A2A1 2331A4A3 B2CCAA34 C2BB7723	1245A2A1 2331A4A3 B2CCAB34 C2BB7723	01
1	F9B08484 DA812027 684D8A13 AAF6FD30	F9B08484 DA812027 684D8B13 AAF6FC30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9BDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83BDAB0 FDD00348 A0A90064 D2EBF651	52

Ví dụ 7.9

Khái niệm khóa yếu, như chúng ta đã thảo luận về DES trong Chương 6, không áp dụng cho AES. Giả sử rằng tất cả các bit trong khóa mật mã đều là 0. Phần sau đây hiển thị các từ cho một số vòng:

Pre-round:	00000000	00000000	00000000	00000000
Round 01:	62636363	62636363	62636363	62636363
Round 02:	9B9898C9	F9FBFBAA	9B9898C9	F9FBFBAA
Round 03:	90973450	696CCFFA	F2F45733	0B0FAC99
. . .	. . .	. . .	. . .	. . .
Round 10:	B4EF5BCB	3E92E211	23E951CF	6F8F188E

Những lời ở vòng trước và vòng đầu đều giống nhau. Ở vòng thứ hai, từ đầu tiên khớp với từ thứ ba; từ thứ hai khớp với từ thứ tư. Tuy nhiên, sau vòng thứ hai, hình mẫu này biến mất; mỗi từ đều khác nhau.

7.3.2 Mở rộng khóa trong AES-192 và AES-256

Các thuật toán mở rộng khóa trong phiên bản AES-192 và AES-256 rất giống với thuật toán mở rộng khóa trong AES-128, với những điểm khác biệt sau:

7.3.3 Phân tích mở rộng khóa

Cơ chế mở rộng khóa trong AES đã được thiết kế để cung cấp một số tính năng cản trở người giải mã.

7-4 MẬT Mã

AES sử dụng bốn loại biến đổi để mã hóa và giải mã. Trong tiêu chuẩn, thuật toán mã hóa được gọi là mật mã và thuật toán giải mã được gọi là mật mã nghịch đảo.

Các chủ đề được thảo luận trong phần này:

7.4.1 Thiết kế ban đầu

7.4.2 Thiết kế thay thế

7.4.1 Tiếp tục

Thuật toán

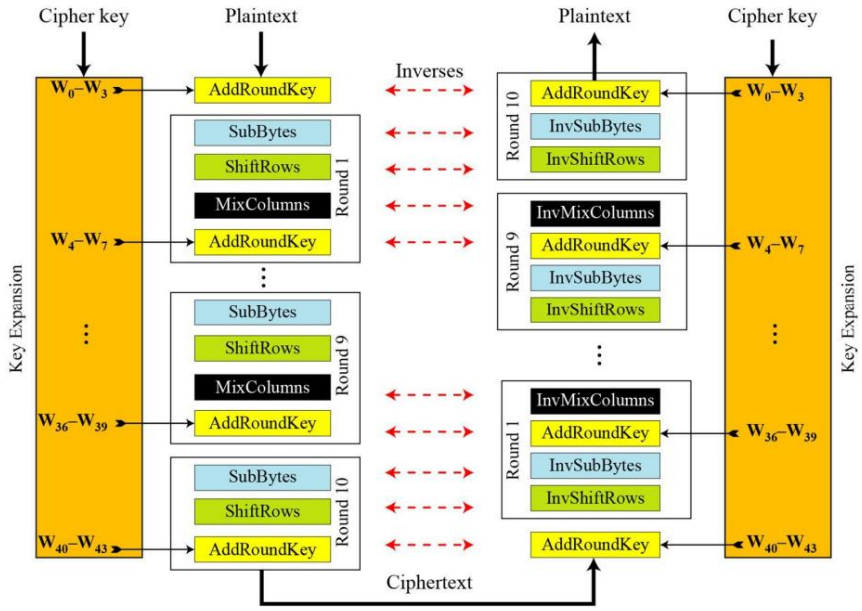
Mã cho phiên bản AES-128 của thiết kế này được thể hiện trong Thuật toán 7.6.

Algorithm 7.6 *Pseudocode for cipher in the original design*

```
Cipher (InBlock [16], OutBlock[16], w[0 … 43])
{
    BlockToState (InBlock, S)

    S ← AddRoundKey (S, w[0…3])
     for (round = 1 to 10)
    {
        S ← SubBytes (S)
        S ← ShiftRows (S)
         if (round ≠ 10)  S ← MixColumns (S)
         S ← AddRoundKey (S, w[4 × round, 4 × round + 3])
     }

    StateToBlock (S, OutBlock);
}
```

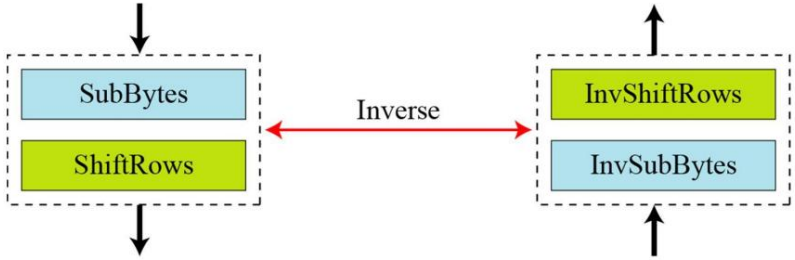

7.4.1 Thiết kế ban đầu

Hình 7.17 Mật mã và mật mã nghịch đảo của thiết kế ban đầu

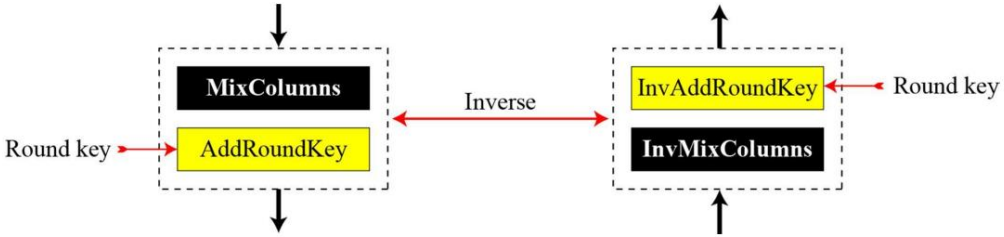


7.4.2 Thiết kế thay thế

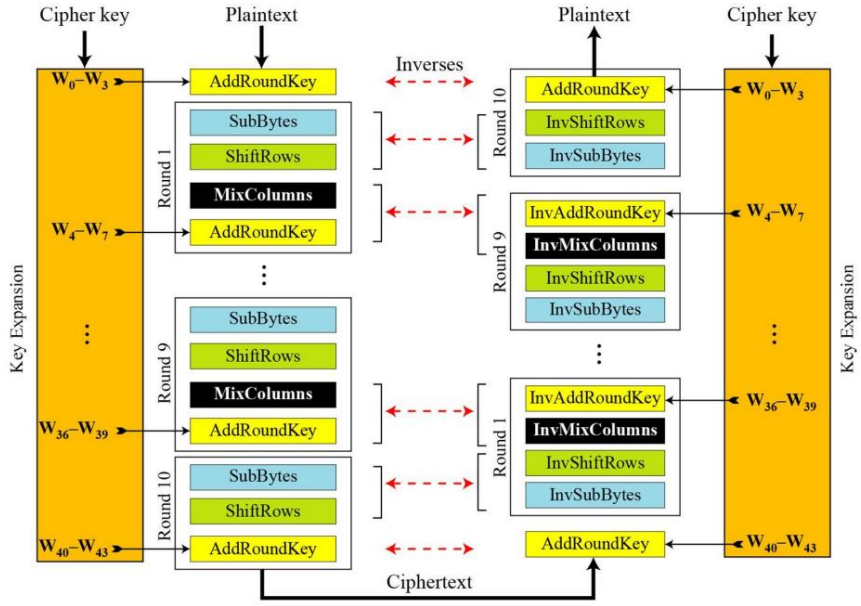
Hình 7.18 Khả năng đảo ngược của sự kết hợp SubBytes và ShiftRows



Hình 7.19 Khả năng đảo ngược của tổ hợp MixColumns và AddRoundKey



Hình 7.20 Mật mã và mật mã ngược trong thiết kế thay thế



Thay đổi thuật toán mở rộng khóa Thay vì sử dụng phép biến đổi InvRoundKey trong mật mã ngược, thuật toán mở rộng khóa có thể được thay đổi để tạo ra một bộ khóa tròn khác cho mật mã nghịch đảo.

7-5 Ví dụ

Trong phần này, một số ví dụ về mã hóa/giải mã và tạo khóa được đưa ra để nhấn mạnh một số điểm đã được thảo luận trong hai phần trước.

Ví dụ 7.10

Phần sau đây cho thấy khối văn bản mã hóa được tạo từ khối văn bản gốc bằng cách sử dụng khóa mật mã được chọn ngẫu nhiên.

Plaintext:	00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19
Cipher Key:	24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87
Ciphertext:	BC 02 8B D3 E0 E3 B1 95 55 0D 6D FB E6 F1 82 41

7-5 Tiếp theo

Ví dụ 7.10 Tiếp theo

Table 7.7 *Example of encryption*

Round	*Input State*	*Output State*	*Round Key*
Pre-round	00 12 0C 08 04 04 00 23 12 12 13 19 14 00 11 19	24 26 3D 1B 71 71 E2 89 B0 44 01 4D A7 88 11 9E	24 34 31 13 75 75 E2 AA A2 56 12 54 B3 88 00 87
1	24 26 3D 1B 71 71 E2 89 B0 44 01 4D A7 88 11 9E	6C 44 13 BD B1 9E 46 35 C5 B5 F3 02 5D 87 FC 8C	89 BD 8C 9F 55 20 C2 68 B5 E3 F1 A5 CE 46 46 C1
2	6C 44 13 BD B1 9E 46 35 C5 B5 F3 02 5D 87 FC 8C	1A 90 15 B2 66 09 1D FC 20 55 5A B2 2B CB 8C 3C	CE 73 FF 60 53 73 B1 D9 CD 2E DF 7A 15 53 15 D4

7-5 Tiếp tục

Ví dụ 7.10 Tiếp theo

3	1A 90 15 B2 66 09 1D FC 20 55 5A B2 2B CB 8C 3C	F6 7D A2 B0 1B 61 B4 B8 67 09 C9 45 4A 5C 51 09	FF 8C 73 13 89 FA 4B 92 85 AB 74 0E C5 96 83 57
4	F6 7D A2 B0 1B 61 B4 B8 67 09 C9 45 4A 5C 51 09	CA E5 48 BB D8 42 AF 71 D1 BA 98 2D 4E 60 9E DF	B8 34 47 54 22 D8 93 01 DE 75 01 0F B8 2E AD FA
5	CA E5 48 BB D8 42 AF 71 D1 BA 98 2D 4E 60 9E DF	90 35 13 60 2C FB 82 3A 9E FC 61 ED 49 39 CB 47	D4 E0 A7 F3 54 8C 1F 1E F3 86 87 88 98 B6 1B E1
6	90 35 13 60 2C FB 82 3A 9E FC 61 ED 49 39 CB 47	18 0A B9 B5 64 68 6A FB 5A EF D7 79 8E B2 10 4D	86 66 C1 32 90 1C 03 1D 0B 8D 0A 82 95 23 38 D9

7-5 Tiếp theo

Ví dụ 7.10 Tiếp theo

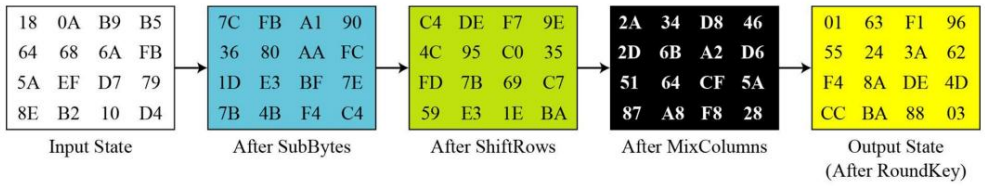
7	18 0A B9 B5 64 68 6A FB 5A EF D7 79 8E B2 10 4D	01 63 F1 96 55 24 3A 62 F4 8A DE 4D CC BA 88 03	62 04 C5 F7 83 9F 9C 81 3E B3 B9 3B B6 95 AD 74
8	01 63 F1 96 55 24 3A 62 F4 8A DE 4D CC BA 88 03	2A 34 D8 46 2D 6B A2 D6 51 64 CF 5A 87 A8 F8 28	EE EA 2F D8 61 FE 62 E3 AC 1F A6 9D DE 4B E6 92
9	2A 34 D8 46 2D 6B A2 D6 51 64 CF 5A 87 A8 F8 28	0A D9 F1 3C 95 63 9F 35 2A 80 29 00 16 76 09 77	E4 0E 21 F9 3F C1 A3 40 E3 FC 5A C7 BF F4 12 80
10	0A D9 F1 3C 95 63 9F 35 2A 80 29 00 16 76 09 77	BC E0 55 E6 02 E3 0D F1 8B B1 6D 82 D3 95 F8 41	DB D5 F4 0D F9 38 9B DB 2E D2 88 4F 26 D2 C0 40

7-5 Tiếp theo

Ví dụ 7.11

Hình 7.21 hiển thị các mục trạng thái trong một vòng, vòng 7, trong Ví dụ 7.10.

Hình 7.21 Các trạng thái trong một vòng



7-5 Tiếp theo

Ví dụ 7.12

Người ta có thể tò mò muốn xem kết quả của việc mã hóa khi bản rõ được tạo thành từ tất cả các số 0. Sử dụng khóa mật mã trong Ví dụ 7.10 sẽ thu được bản mã.

```
Plaintext:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Cipher Key: 24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87
Ciphertext: 63 2C D4 5E 5D 56 ED B5 62 04 01 A0 AA 9C 2D 8D
```


7-5 Tiếp theo

Ví dụ 7.13

Chúng ta hãy kiểm tra hiệu ứng tuyết lở mà chúng ta đã thảo luận trong Chương 6.

Chúng ta chỉ thay đổi một bit trong bản rõ và so sánh kết quả. Chúng tôi chỉ thay đổi một bit trong byte cuối cùng.

Kết quả cho thấy rõ tác dụng khuếch tán và nhầm lẫn.

Việc thay đổi một bit trong bản rõ sẽ ảnh hưởng đến nhiều bit trong bản mã.

```
Plaintext 1:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Plaintext 2:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
Ciphertext 1: 63 2C D4 5E 5D 56 ED B5 62 04 01 A0 AA 9C 2D 8D
Ciphertext 2: 26 F3 9B BC A1 9C 0F B7 C7 2E 7E 30 63 92 73 13
```


7-5 Tiếp tục

Ví dụ 7.14

Phần sau đây cho thấy tác dụng của việc sử dụng khóa mật mã trong đó tất cả các bit là 0.

```
Plaintext:  00 04 12 14 12 04 12 00 0c 00 13 11 08 23 19 19
Cipher Key: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Ciphertext: 5A 6F 4B 67 57 B7 A5 D2 C4 30 91 ED 64 9A 42 72
```


7-6 PHÂN TÍCH AES

Phần này là một đánh giá ngắn gọn về ba đặc điểm của AES.

Các chủ đề được thảo luận trong phần này: 7.6.1 Bảo mật 7.6.2 Triển khai 7.6.3 Đơn giản và Chi phí

7.6.1 Bảo mật

AES được thiết kế sau DES. Hầu hết các cuộc tấn công DES đã biết đều đã được thử nghiệm trên AES.

Tấn công vũ phu

AES chắc chắn an toàn hơn DES do khóa có kích thước lớn hơn.

Tấn công thống kê

Nhiều thử nghiệm đã thất bại trong việc phân tích thống kê văn bản mã hóa.

Các cuộc tấn công vi sai và tuyến tính

Hiện chưa có các cuộc tấn công vi sai và tuyến tính nào trên AES.

7.6.1 Tiếp tục

Tấn công thống kê

Nhiều thử nghiệm đã thất bại trong việc phân tích thống kê văn bản mã hóa.

Các cuộc tấn công vi sai và tuyến tính

Hiện chưa có các cuộc tấn công vi sai và tuyến tính nào trên AES.

7.6.2 Thực hiện

AES có thể được triển khai trong phần mềm, phần cứng và phần sụn. Việc triển khai có thể sử dụng quy trình tra cứu bảng hoặc các thủ tục sử dụng cấu trúc đại số được xác định rõ.

7.6.3 Đơn giản và chi phí

Các thuật toán được sử dụng trong AES đơn giản đến mức chúng có thể được thực hiện dễ dàng bằng cách sử dụng bộ xử lý giá rẻ và lượng bộ nhớ tối thiểu.