HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

Học phần: ***Cơ sở an toàn thông tin***

Bài báo cáo:

Tìm hiểu về giải thuật mã hóa khóa bí mật AES: giải thuật sinh khóa phụ, mã hóa, giải mã, các điểm yếu, các dạng tấn công vào AES và phòng chống. Cài đặt thử nghiệm

Giảng viên hướng dẫn: *Hoàng Xuân Dậu*
Sinh viên thực hiện: *Nhóm 03*

Nguyễn Cảnh Hiếu	*B20DCAT059*
Nguyễn Đình Hinh	*B20DCAT065*
Đào Minh Hoàng	*B20DCAT067*
Hoàng Thạch Hùng	*B20DCAT074*
Lê Quang Huy	*B20DCAT077*
Nguyễn Quang Huy	*B20DCAT079*

Hà Nội 2022

MỤC LỤC

Lời mở đầu:

Cùng với sự phát triển của khoa học kỹ thuật và ứng dụng tin học trong mọi lĩnh vực của đời sống như an ninh quân sự, quốc phòng và trong các giao dịch thương mại điện tử…

Song hành cùng với sự phát triển như vũ bão của internets và các giao thức điện tử thì nhu cầu bảo vệ thông tin trong các hệ thống ứng dụng ngày càng được chú trọng vì vậy mà khoa học mật mã được sử dụng rộng rãi với mong muốn bảo vệ được nhưng thông tin cơ sở dữ liệu quan trọng mật mã mở ra nhiều hướng phát triển với những đặc trưng riêng. Trong chương trình an toàn và bảo mật hệ thống thông tin của sinh viên thì việc tự mã hóa và giải mã dữ liệu là khá quan trọng. Nó sẽ giúp sinh viên tìm hiểu sâu hơn về các thuật toán mã hóa và giải mã cũng như cách cung cấp các thuật toán mã hóa và giải mã trên hệ điều hành. Với những lý do như vậy nhóm sinh viên chúng em tìm hiểu sâu về mã hóa dựa trên thuật toán AES. AES là một trong nhưng thuật toán mã đối xứng đã được công nhận rộng rãi và mức độ an toàn.

❖ GIỚI THIỆU VỀ GIẢI THUẬT AES

1. Giới thiệu chung:

- AES là viết tắt của Advanced Encryption Standard là một chuẩn mã hóa dữ liệu được NIST công nhận năm 2001.
- AES được xây dựng dựa trên Rijndael cipher phát triển bởi 2 nhà mật mã học người Bỉ là Joan Deamen và Vincent Rijmen.
- Kích thước của khối dữ liệu của AES là 128 bít.
- Kích thước khóa có thể là 128, 192 hoặc 256bit (là bội của 32 và lớn nhất là 256 bít).
- AES thiết kế dựa trên mạng hoán vị- thay thế: có thể đạt tốc độ cao trên cả phần mềm và phần cứng.
- AES vận hành trên một matran 4x4, được gọi là state (trạng thái).
- Kích thước của khóa quyết định vào số vòng lặp chuyển đổi cần thực hiện để chuyển bản rõ thành bản mã:

  + 10 vòng lặp ứng với 128 bit.

  + 12 vòng lặp ứng với 192 bít.

  + 14 vòng lặp ứng với 256 bít.

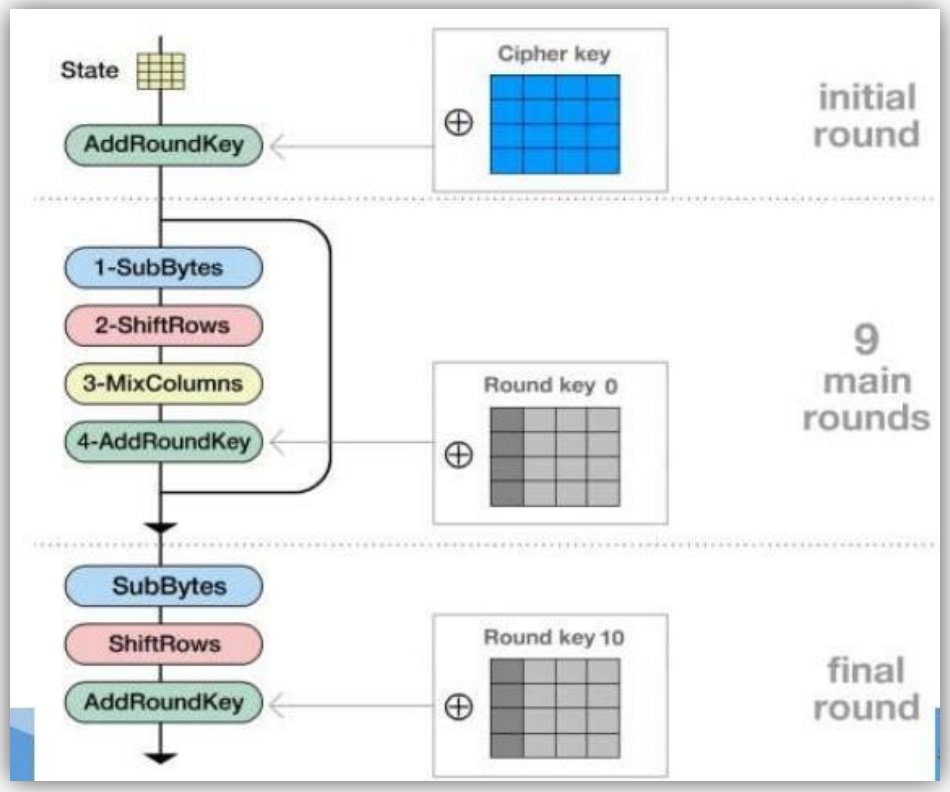
  Giải thích:

  - Đối với 128 bít thì kích thước khóa mở rộng là 44 từ. Mỗi khóa gồm 4 từ do đó 44 từ này được chia thành 11 cụm khóa con. Mỗi vòng giải mã/ mã hóa sử dụng một khóa con cộng với một khóa con ở phần bắt đầu của thuật toán AES. Như vậy số vòng = số khóa con – 1(mượn) => 128 bít ứng với 11-1 = 10 vòng.
  - Đối với 192 bit thì kích thước khóa mở rộng là 52 từ. Mỗi khóa gồm 4 từ, do đó 52 từ này được chia thành 13 cụm khóa con. Mỗi vòng mã hóa/ giải mã sử dụng một khóa con cộng với một khóa con ở phần bắt đầu của thuật tóa AES. Như vậy số vòng = số khóa con – 1 (mượn) => 192 bít sẽ tương ứng với 13 – 1 = 12 vòng.
  - Đối với 256 bit thì kích thước khóa mở rộng là 60 từ. Mỗi khóa gồm 4 từ, do đó 60 từ này được chia thành 15 cụm khóa con. Mỗi vòng mã hóa/ giải mã sử dụng một khóa con cộng với một khóa con ở phần bắt đầu của thuật tóa AES. Như vậy số vòng = số khóa con – 1 (mượn) => 192 bít sẽ tương ứng với 15 – 1 = 14 vòng.
- Mã hóa AES là một mã hóa theo mô hình 128 bit không sử dụng nguyên tắc

của hệ mã hóa Feistel mà sử dụng mô hình mạng SPN. AES dùng 4 phép biến đổi chính để mã hóa một khối: Add row key, Substitute bytes, Shift rows, Mix columns. Mỗi phép biến đổi nhận tham số đầu vào có kích thước 128 bít và cho kết quả đầu ra cũng có kích thước 128 bít. AES thực hiện 4 phép biến đổi trên nhiều lần tạo thành 10 vòng biến đổi.

2. Mô tả khái quát giải thuật AES:

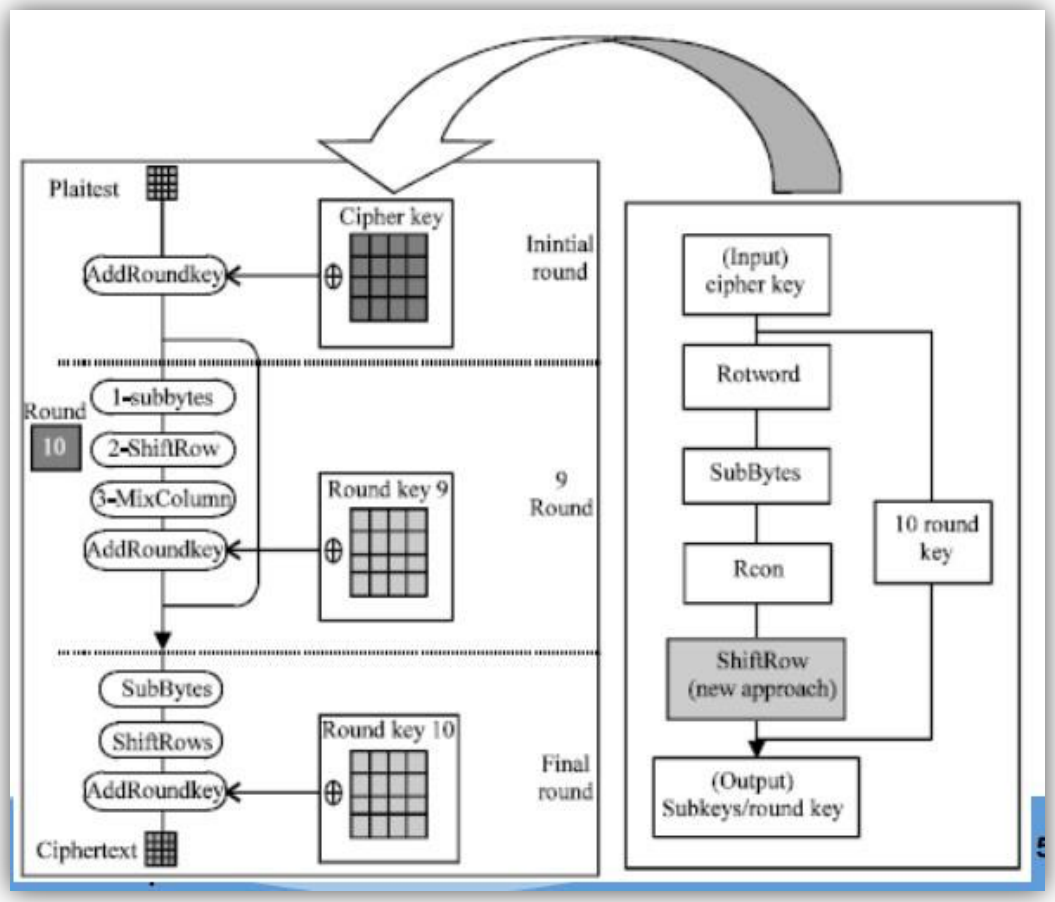
- Mở rộng khóa: Các khóa phụ dùng trong csác vòng lặp được sinh ra từ khóa chính AES sử dụng thủ tục sinh khóa Rijndael.
- Vòng khởi tạo:
  + AddRoundKey: mỗi byte trong state được kết hợp với khóa phụ sử dụng XOR.
- Vòng lặp chính:
  + SubBytes: bước thay thế phi tuyến tính, trong đó mỗi byte trong state được thay thế bằng một byte khác sử dụng bảng tham chiếu
  + ShiftRows: Bước đổi chỗ, trong đó mỗi dòng state được dịch một số bước theo chu kz.
  + MixColumns: trộn các cột trong state, kết hợp 4 bytes trong mỗi cột.
  + AddRoundKey
- Vòng cuối gồm 3 bước giống trên trừ bước MixColumns.



Sơ đồ các bước xử lý chính của AES.

Qua sơ đồ ta thấy được quá trình mã hóa của độ dài khối đầu vào 128 bít được thực hiện qua các vòng lặp, trong đó 9 vòng lặp chính được thực hiện qua 4 bước và một vòng lặp cuối cùng được thực hiện giống hệt nhưng không có giai đoạn MixColumns.

I. THUẬT TOÁN SINH KHÓA



Sơ đồ các bước xử lý chính của AES.

Nhìn vào sơ đồ trên ta thấy được việc sinh khóa phụ là bước đầu của giai đoạn mã hóa, Quá trình sinh khóa phụ được thực hiện ở vòng khởi tạo AddRoundKey. Bước thực hiện được chia là 4 bước:

+ Rotword: quay trái 8 bít.
+ SubBytes
+ Rcon: tính toán giá trị Rcon(i).
+ Shiftrow

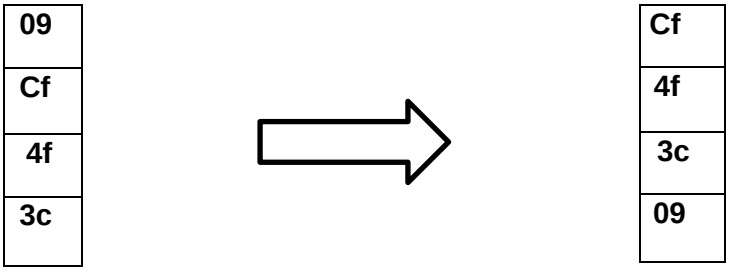
- 128 bít ban đầu được mở rộng thành 176 byte, được tổ chức thành 44 từ, mỗi từ 4 bytes, vừa đủ để tạo thành 10 khóa phụ cho 10 vòng mã hóa của thuật toán cộng với một khóa phụ cho thao tác cộng khóa ban đầu. Như vậy thuật toán sinh khóa phụ của AES thực chất là thuật toán mở rộng bốn từ khóa (128 bit) ban đầu thanh 44 từ. Thao tác cộng khóa được thực hiện qua các bước sau:

Giả sử ta có một matran như sau thực hiện sinh khóa:

2b	**28**	**Ab**	**09**
7e	**Ae**	**F7**	**Cf**
15	**D2**	**15**	**4f**
16	**A6**	**88**	**3c**

1. **<u>Rotword:</u>**

- Bốn từ khóa gốc được đưa trực tiếp vào phép cộng khóa ban đầu tức là w[0,3] = key. Lấy cột cuối cùng của matran thực hiện rotWord quay trái 8 bit:



2. SubBytes

- Thao tác thay thế byte, thao tác này có chức năng thay thế từng byte trong mảng trạng thái thành một byte khác sử dụng một matran kích thước 16 x16 (S-box). Nguyên tắc thay thế byte được hiểu: ứng với mỗi byte trong trạng thái hiện hành, bit bên trái được dùng để chọn một trong 16 dòng, bít bên phải được dùng để chọn một trong 16 cột. Giá trị của ô tương ứng với dòng và cột được chọn sẽ được thay thế cho byte hiện hành.
- **Giải thích về sự thiết lập nên bảng S-box:**

  + B1: Điền các con số từ 0-225 vào bảng matran 16x16 theo từng hàng. Với hệ Hex, do đó hàng 0 gồm các con số ,00-, ,01-, …,0F-. Hàng 1 gồm các con số: ,10-, ,11-,…, ,1F-. Điều này có nghĩa là tại hàng x cột y có giá trị {xy}

  +B2: thay thế mỗi byte trong bảng bằng giá trị nghịch đảo trong trường GF(28). Quy ước nghịch đảo của {00} cũng là {00}

  + B3: Đối với mỗi byte trong bảng, ký hiệu 8 bít là $b7b6b5b4b3b2b1b0$.
  Thay thế mỗi bít b_i bằng giá trị b'_i được tính sau:

$$b'_i = b_i \oplus b_{(i+4) mod\ 8} \oplus b_{(i+5) mod\ 8} \oplus b_{(i+6) mod\ 8} \oplus b_{(i+7) mod\ 8} \oplus c_i$$

  Với *ci* là bít chứ *i* của số{63}, tức $c_7c_6c_5c_4c_3c_2c_1c_0 = 01100011$. Việc tính toán trên tương đương với phép nhân ma trận sau trên GF(28) (B' = XB $\oplus$C):

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

  + Trong đó phép cộng thực hiện như phép XOR. Hình dưới trình bày nội dung bảng S-box sau khi tính toán.

  + **Ví dụ**: xét giá trị {95}, tại bước 1, giá trị tại dòng 9 cột 5 là {95}, sau bước 2 tính nghịch đảo giá trị của ô này là {8A} có dạng nhị phân là 10001010 thực hiện phép nhan ma trận

$$\begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

➡ Kết quả dưới dạng thập lục phân là {2A}

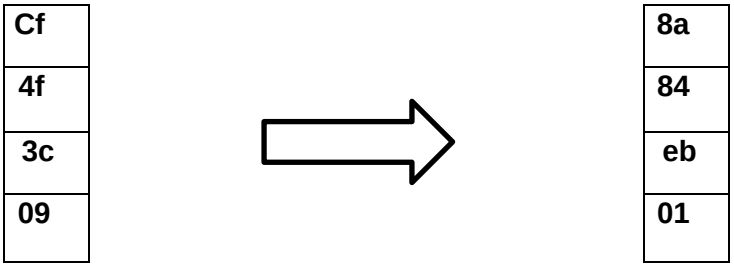
- Ở bước này chúng ta thực hiện bằng cách đem so sánh với bảng S-box.
  S-box được tạo ra bằng cách xác định nghịch đảo cho một số lượng nhất định $GF(2^8) = GF(2)[x] / (x^8 + x^4 + x^3 + x + 1)$, trường hữu hạn của Rijndael của Zero, mà không có nghịch đảo, được ánh xạ tới không. Sau đó nghịch đảo được biến đổi bằng cách sử dụng sau đây biến đổi afin:

$$\begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

  biến đổi afin này là tổng hợp của nhiều phép quay của các byte như là một vector, nơi Ngoài ra là các hoạt động XOR.

a- Ma trận S-Box

	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	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



3. Rcon

Rcon là một mảng số. Mảng này gồm 10 từ ứng với 10 vòng AES. Bốn byte của một phần tử Rcon[j] là (RC[j],0,0,0) với RC[j] là mảng 10 byte được thực hiện bằng cách tính theo công thức Rcon(i) = x^{i-1} mod $x^8 + x^4 + x^3 + x + 1$.

Cứ như thực hiện phép chia lấy phần dư ta được Rcon của từng vòng. Với i là số thứ tự vòng trong AES.

Ví dụ với i = 1 => phần dư khi thực hiện phép chia sẽ là 1.do đó Rcon*1+ = 1.

Ví dụ với i = 10 do đó ta thực hiện phép chia lấy dư của x^9 cho $x^8 + x^4 + x^3 + x + 1$ ta có:

$$X^9 \;\Big|\; x^8+x^4+x^3+x+1$$
$$X^9+x^5+x^4+x^2+x \;\Big|\; x$$
Dư: $x^5+x^4+x^2+x$

Do đó phần dư là $x^5+x^4+x^2+x$. Phép chia này được thực hiện trong trường GF(2^8) do đó phần dư sẽ được quy đổi về hệ Hexa 110110 = 36
Tương tự với i = 2,3,…, 9 ta được bảng các giá trị sau:

J	1	2	3	4	5	6	7	8	9	10
RC[j]	1	2	4	8	10	20	40	80	1B	36

4. Kết thúc:

- Sau khi có một cột đã mã hóa sử dụng phép toán XOR với một hằng số vòng (Rcon), hằng số này khác biệt với mỗi vòng lặp
- Sử dụng phép toán xor với cột đầu tiên của input và trước khi thực hiện tính toán, ta chuyển về dạng ASCII rồi chuyển qua hệ Hex để thực hiện tính tóa XOR

XOR	**8a**	**84**	**Eb**	**01**	**01**	**00**	**00**	**00**	**2b**	**7e**	**15**	**16**	**=**	**A0**	**fa**	**fe**	**17**

- Thực hiện phép toán XOR có nghĩa là nếu cả hai giá trị đều bằng 0 hoặc 1 thì sẽ cho kết quả là 0. Còn ngược lại
- Các cột còn lại sử dụng phép toán XOR với các cột tiếp theo của matran có nghĩa là cột 2 của matran ban đầu được XOR với cột vừa được tạo ở trên:

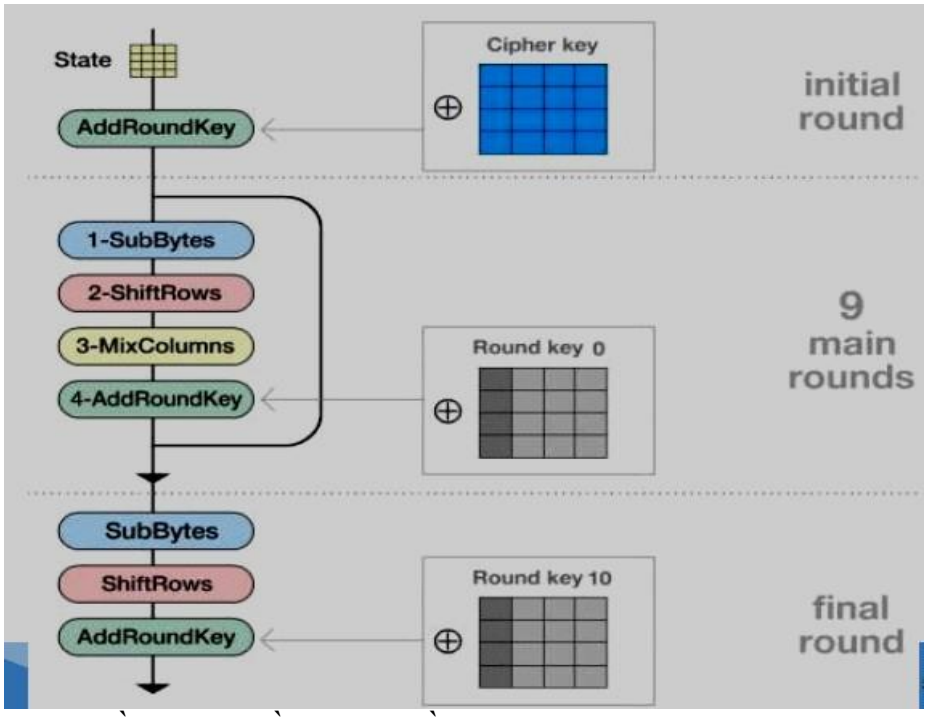
A0		28		88
Fa	XOR	Ae	=	54
Fe		D2		2c
17		A6		1b

Tương tự ta tìm được các cột tiếp theo sẽ có giá trị là

2b	**28**	**Ab**	**09**	**A0**	**88**	**23**	**2a**	F2	7a	23	73	.....	D0	C9	E1	B6
7e	**Ae**	**F7**	**Cf**	**fa**	**54**	**A3**	**6c**	C2	96	A3	59	.....	14	ee	3f	63
15	**D2**	**15**	**4f**	**fe**	**2c**	**39**	**76**	95	B9	39	F6	.....	F9	25	0c	0c
16	**A6**	**88**	**3c**	**17**	**B1**	**39**	**05**	F2	43	39	7f	.....	A8	89	C8	A 6
Cipher key				Round key 1				Round key 2					Round key 10			

II. MÃ HÓA

❖ Giới thiệu chung



- Nhìn vào sơ đồ trên thì đầu tiên ta cần có input là state và Cipher Key.
- Giả sử dữ liệu đầu vào là 2C:¿êZ0r11ŸóÓ7(BEL)4

  Key : +~NAKSNY(«Êª½,NAKêHT¤O<
- State được sinh ra từ nội dung ta cần mã hóa. Từ nội dung ta chuyển đổi nội dung đó thành các bit => ta có 1 mảng bit. Từ mảng bit đó thì mỗi 1/2 byte ta chuyển đổi chúng thành một kí hiệu hệ hexa => ta chuyển đổi chúng thành ma trận state
- Ví dụ mảng bit: 0011001000110011.....
- Ta sẽ tách chúng ra mỗi 1 byte khác nhau:

  + byte thứ nhất: 00110010 => 32(hexa)

  + byte thứ 2: 00110011=>33(hexa)

- Sau đó ta sẽ xếp vào ma trận state có dạng sau:

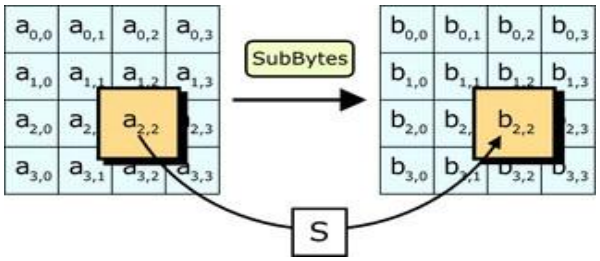
State			
32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

Tương tự cách trên với Cipher Key nhưng đó ko phải nội dung mà là key dc người dùng nhập vào:

Cipher Key			
2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

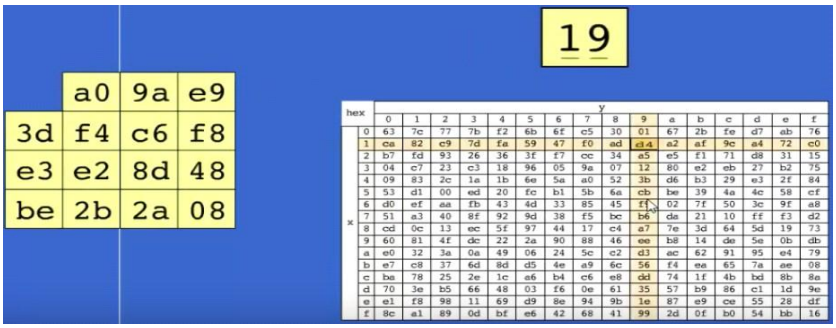
Ma trận này được đưa thêm vào để thực hiện mã hóa.

Khi có 2 input trên ta bắt đầu mã hóa: Nhìn trên hình vẽ thì ta sẽ có 4 phép biến đổi chính

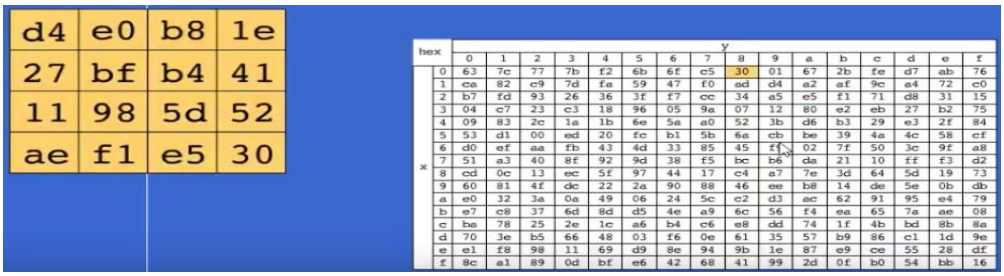


1. **SubBytes**

- Các byte được thế thông qua bảng tra S-box. Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box này được tạo ra từ một phép biến đổi khả nghịch trong trường hữu hạn GF (28) có tính chất phi tuyến. Để chống lại các tấn công dựa trên các đặc tính đại số, hộp S-box này được tạo nên bằng cách kết hợp phép nghịch đảo với một phép biến đổi affine khả nghịch. Hộp S-box này cũng được chọn để tránh các điểm bất động (fixed point).
- Giải thích ví dụ:

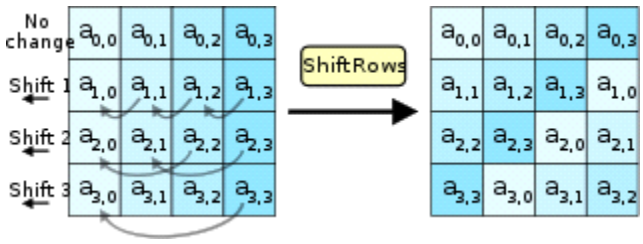


+ Lấy cái bảng state ra lấy vị trí của ma trận ví dụ a11 => so sánh với S box:
+ Lần lượt thay thế tất cả giá trị trong bảng Ss - Box => ta dc giá trị state tương ứng:s



- Nguyên tắc thay thế byte được hiểu: ứng với mỗi byte trong trạng thái hiện hành, bit bên trái được dùng để chọn một trong 16 dòng, bít bên phải được dùng để chọn một trong 16 cột. Giá trị của ô tương ứng với dòng và cột được chọn sẽ được thay thế cho byte hiện hành.

2. ShiftRows



- Phép biến đổi dùng trong phép mã hóa áp dụng lên trạng thái bằng cách chuyển dịch vòng ba hàng cuối của trạng thái theo số lượng byte các offset khác nhau.
- Cách thực hiện:
- Các hàng được dịch vòng một số bước nhất định. Đối với AES, hàng đầu được giữ nguyên. Mỗi byte của hàng thứ 2 được dịch vòng trái một vị trí. Tương tự, các hàng thứ 3 và 4 được dịch vòng 2 và 3 vị trí. Do vậy, mỗi cột khối đầu ra của bước này sẽ bao gồm các byte ở đủ 4 cột khối đầu vào. Đối với Rijndael với độ dài khối khác nhau thì số vị trí dịch chuyển cũng khác nhau.
- Giải thích ví dụ:

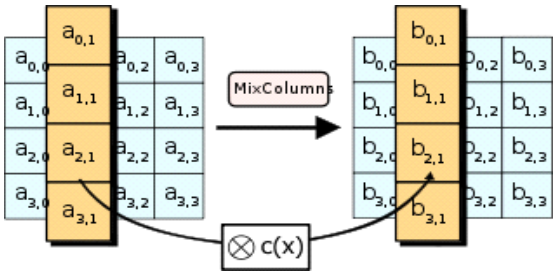
+ Hàng 2 dịch 1 byte

+ Hàng 3 dịch 2 byte

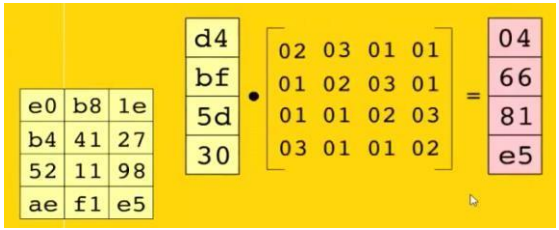
+ Hàng 4 dịch 3 byte, do đó ta được matran:

3. MixColumns

- Dùng để: Phép biến đổi trong phép mã hóa thực hiện bằng cách lấy tất cả các cột trạng thái trộn với dữ liệu của chúng (một cách độc lập nhau) để tạo ra các cột mới.
- Cách thực hiện:
- Bốn byte trong từng cột được kết hợp lại theo một phép biến đổituyến tính khả nghịch. Mỗi khối bốn byte đầu vào sẽ cho một khối bôn byte ở dầu ra với tính chất là mỗi byte ở đàu vào đều ảnh hưởng tới cả bốn byte dầu ra cùng với bước ShiftRows, MixColumns đã tạo ra tính chất khuếch tán cho thuật toán. Mỗi cột được xem như một đa thức trong trường hữu hạn và được nhân với đa thức {\displaystylec c(x)=3x^{3}+x^{2}+x+2} {\displaystylec c(x)=3x^{3}+x^{2}+x+2} (modulo {\displaystyle x^{4}+1} {\displaystyle x^{4}+1}) Vì vậy bước này có thể xem là phép nhân ma trận trong trường hữu hạn



- Giải thích ví dụ:



+ Việc nhân matran được thực hiện bằng cách
+ Ta sẽ lấy cột nhân với hàng theo quy tắc sau:
(D4 * 02) XOR (BF * 03) XOR (5D * 01) XOR (30 * 01)
+ Biến đổi về nhị phân D4 = 11010100
+ Phép nhân trong trường GF(2^8) có hai trường hợp nếu trường hợp b7= 0 thì chúng ta thực hiện bình thường đa thức với đa thức, còn trường hợp b7 = 1 có thể thực hiện ở mức độ byte bằng một phép dịch trái (thêm số 0 cuối dãy)và sau đó thực hiện phép toán XOR. Quay lại ví dụ việc D4 có b7 = 1 do đó ta thực hiện phép XOR với dãy số

10101000
XOR 00011011 (môi trường GF(2^8)) 10110011

+ Tiếp theo chúng ta thực hiện phép tính (BF*03), BF= 10111111
Tách 03 = 02 + 01. Do đó (BF*03) = (BF*02) XOR (BF*01). Với (BF*02) ta tính tương tự như bước trên

01111110
XOR 00011011 (môi trường GF(2^8))

01100101

Với (BF*01) = BF = 10111111
Do đó kết quả của phép tính (D4 * 02) XOR (BF * 03) XOR (5D * 01) XOR (30 * 01) là thực hiện phép XOR của các biểu thức sau:

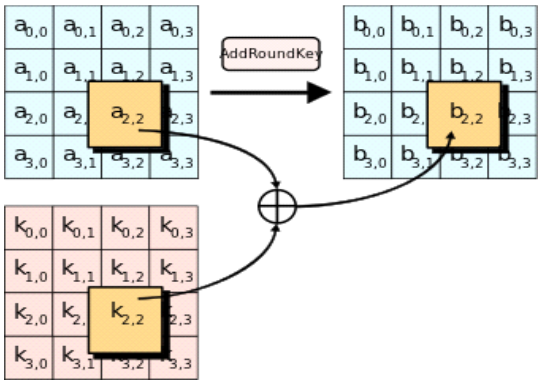
	10110011
Xor	01100101
Xor	10111111
Xor	01011101
Xor	00110000
	00000100 = 04 (Hex)

- Làm tương tự với việc nhân các cột khác ta được matran cuối cùng sẽ có giá trị như sau:

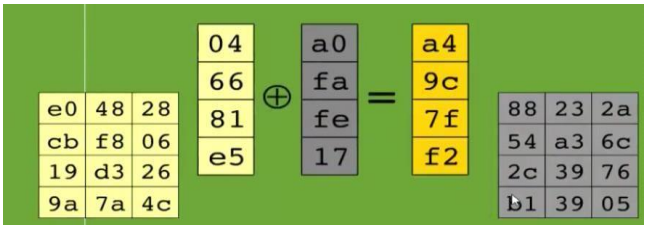
04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

4. AddRoundKey

- Phép biến đổi trong phép mã hóa và phép giải mã. Trong đó, một khóa vòng (các giá trị sinh ra từ khóa mã bằng quy trình mở rộng khóa) được cộng thêm vào trạng thái bằng phép toán XOR (phép toán hoặc và loại trừ). Độ dài của khóa vòng bằng độ dài của trạng thái.
- Cách thực hiện: Khóa con được kết hợp với các khối. Khóa con trong mỗi chu trình được tạo ra từ khóa chính với quá trình tạo khóa con Rijndael; mỗi khóa con có độ dài giống như các khối. Quá trình kết hợp được thực hiện bằng cách XOR từng bít của khóa con với khối dữ liệu.



- Giải thích ví dụ:
  Thực hiện phép XOR với một key được sinh trên phần sinh khóa phụ. Đưa các con số về hệ nhị phân rồi thực hiện phép XOR , do đó ta được kết quả

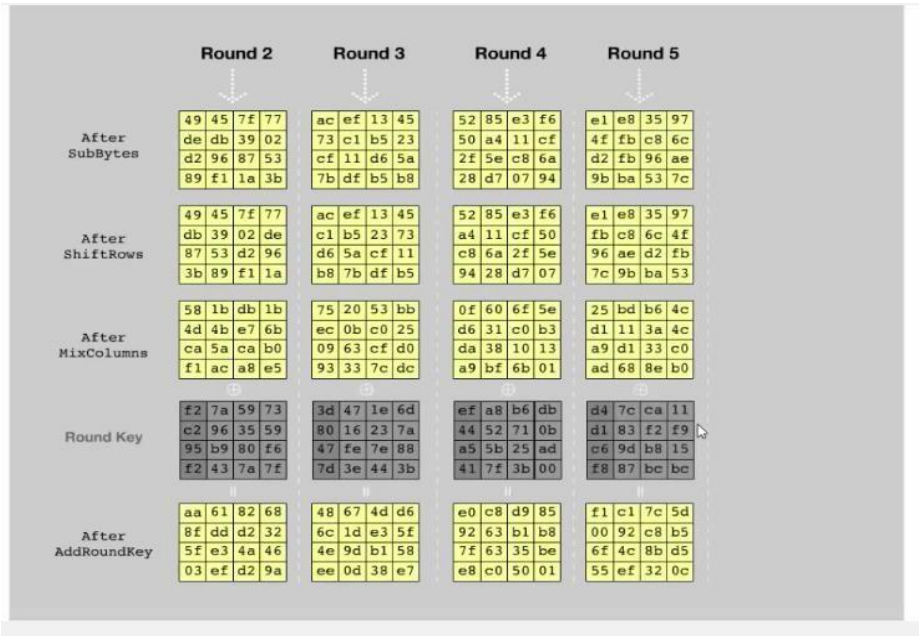


Ví dụ như 04 =00000100

A0 = 10100000

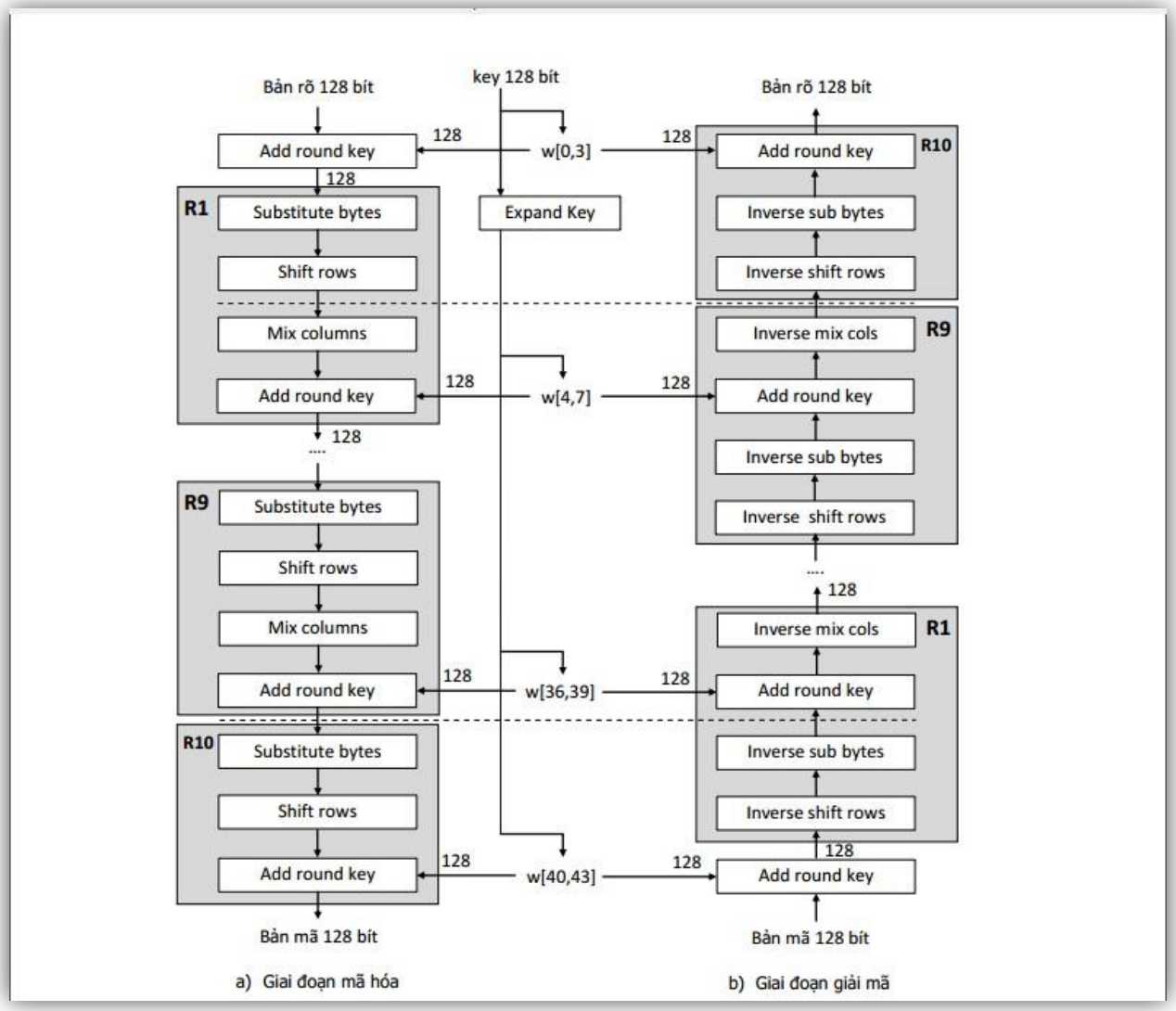
} XOR 10100100 = a4

Kết luận: Những phép biến đổi này được sử dụng trong 9 vòng tiếp theo. àm tương tự như các vòng còn lại. Riêng vòng lặp cuối ko dùng MixColumns. Và kết quả cuối cùng của mã hóa sẽ đưa ra:


Round 2
Round 3
Round 4
Round 5
After SubBytes
49 45 7f 77
de db 39 02
d2 96 87 53
89 f1 1a 3b
ac ef 13 45
73 c1 b5 23
cf 11 d6 5a
7b df b5 b8
52 85 e3 f6
50 a4 11 cf
2f 5e c8 6a
28 d7 07 94
e1 e8 35 97
4f fb c8 6c
d2 fb 96 ae
9b ba 53 7c
After ShiftRows
49 45 7f 77
db 39 02 de
87 53 d2 96
3b 89 f1 1a
ac ef 13 45
c1 b5 23 73
d6 5a cf 11
b8 7b df b5
52 85 e3 f6
a4 11 cf 50
c8 6a 2f 5e
94 28 d7 07
e1 e8 35 97
fb c8 6c 4f
96 ae d2 fb
7c 9b ba 53
After MixColumns
58 1b db 1b
4d 4b e7 6b
ca 5a ca b0
f1 ac a8 e5
75 20 53 bb
ec 0b c0 25
09 63 cf d0
93 33 7c dc
0f 60 6f 5e
d6 31 c0 b3
da 38 10 13
a9 bf 6b 01
25 bd b6 4c
d1 11 3a 4c
a9 d1 33 c0
ad 68 8e b0
Round Key
f2 7a 59 73
c2 96 35 59
95 b9 80 f6
f2 43 7a 7f
3d 47 1e 6d
80 16 23 7a
47 fe 7e 88
7d 3e 44 3b
ef a8 b6 db
44 52 71 0b
a5 5b 25 ad
41 7f 3b 00
d4 7c ca 11
d1 83 f2 f9
c6 9d b8 15
f8 87 bc bc
After AddRoundKey
aa 61 82 68
8f dd d2 32
5f e3 4a 46
03 ef d2 9a
48 67 4d d6
6c 1d e3 5f
4e 9d b1 58
ee 0d 38 e7
e0 c8 d9 85
92 63 b1 b8
7f 63 35 be
e8 c0 50 01
f1 c1 7c 5d
00 92 c8 b5
6f 4c 8b d5
55 ef 32 0c

III. GIẢI MÃ

1. Giới thiệu chung



- Thực hiện quy trình giải mã qua các giai đoạn:
  + Điều đầu tiên là cần thực hiện thao tác AddRoundKey trước khi thực hiện các chu kì giải mã.
  + Trong mỗi chu kì sẽ bao gồm 4 bước biến đổi liên tiếp: AddRoundKey, InvShiftRows, InvSubBtes, AddRoundKey, Inverse mix cols.

  + Thực hiện chu kì giải mã cuối cùng tuy nhiên ở chu kì này bước biến đổi "InvMixColumns" được bỏ qua.

  + Việc thực giải mã gồm 10 vòng đối với dữ liệu đầu vào 128 bít.

+ Quá trình giải mã là ngược lại của mã hóa.

2. AddRoundKey

- Bản khóa :

39	02	dc	19
25	dc	11	6a
84	09	85	0b
1d	Fd	97	32

- Round key 10:

D0	C9	E1	B6
14	Ee	3f	63
F9	25	0c	0c
A8	89	C8	A6

- Kết quả: được đưa ra bằng cách chuyển sang hệ nhị phân rồi XOR sau đó lại chuyển về hệ Hex

$39_{16} = 00111001_2$

$D0_{16} = 11010000_2$

XOR

=> 11101001 = E9

- Làm tương tự ta được matran kết quả:

E9	Cb	3d	Af
31	32	2e	09
7d	2c	89	07
B5	72	5f	94

3. InvShiftRows

Tương tự biến đổi ShiftRows thay vì dịch trái thì trong biến đổi này là dịch phải
Ban đầu:

E9	Cb	3d	Af
31	32	2c	09
7d	2c	89	07
B5	72	5f	94

Dòng thứ nhất vẫn dữ nguyên, dòng thứ 2 xoay phải 1 byte, dòng thứ 3 xoay phải 2 byte, dòng thứ 4 xoay phải 3 bit:

E9	Cb	3d	Af
09	31	32	2c
89	07	7d	2c
72	5f	94	B5

4. InvSubBtes

- Quá trình Process InvSubBytes tương tự với SubBytes, nhưng các bảng được sử dụng khác nhau. Bảng được sử dụng là các bảng S-Box nghịch đảo:

		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

- Sau khi đối chiếu so sánh bảng ta được matran mới là:

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1C	84	E7	D2

5. **AddRoundKey**

Bước này thực hiện phép XOR với key 9:

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1C	84	E7	D2

Kết quả:

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

6. **Inverse mix cols**

- Mỗi cột của ma trận state được nhân với đa thức

$$b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$$

và modulo cho đa thức

$$x^4 + 1.$$

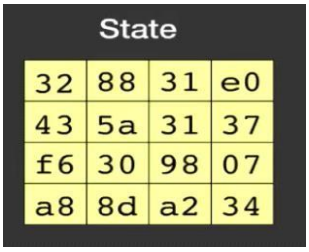
Hay viết dưới dạng ma trận:

$$\begin{bmatrix} s'_{01} \\ s'_{11} \\ s'_{21} \\ s'_{31} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{01} \\ s_{11} \\ s_{21} \\ s_{31} \end{bmatrix}$$

Nhân từng cột với ma trận: Việc nhân giống hệt như mã hóa ta được kết quả khi nhân phép tính là

87	F2	4d	97
6e	4c	90	ec
46	E7	4a	C3
A6	8c	D8	95

Làm tương tự với các vòng lặp theo sơ đồ mã hóa và giải mã, kết quả cuối cùng sau vòng lặp thứ 10 là



IV. CÁC ĐIỂM MẠNH VÀ CÁC ĐIỂM YẾU

- **Ưu điểm:**

- Thuật toán AES thực hiện việc xử lý rất nhanh.
- Mã chương trình ngắn gọn, thao tác xử lý sử dụng ít bộ nhớ.
- Tất cả các bước xử lý của việc mã hóa và giải mã đều được thiết kế thích hợp với cơ chế xử lý song song
- Yêu cầu đơn giản trong việc thiết kế cùng tính linh hoạt trong xử lý luôn được đặt ra và đã được đáp ứng.
- Độ lớn của khối dữ liệu cũng như của mã khóa chính có thể tùy biến linh hoạt từ 128 đến 256-bit với điều kiện là chia hết cho 32. Số lượng chu kz có thể được thay đổi tùy thuộc vào yêu cầu riêng được đặt ra cho từng ứng dụng và hệ thống cụ thể.

- **Nhược điểm:**

- Mã chương trình cũng như thời gian xử lý của việc giải mã tương đối lớn hơn việc mã hóa.
- Không thể tận dụng lại toàn bộ đoạn chương trình mã hóa cũng như các bảng tra cứu cho việc giải mã.
- Khi cài đặt trên phần cứng, thì việc giải mã chỉ sử dụng lại một phần các mạch điện tử sử dụng trong việc mã hóa và với trình tự sử dụng cũng khác nhau.
- Mô tả toán học khá là đơn giản.
- AES không đủ an toàn đối với dạng tấn công Side channel attack.

V. CÁC DẠNG TẤN CÔNG VÀO AES VÀ CÁCH PHÒNG CHỐNG

Tấn công bên là một trong những dạng tấn công Tấn công kênh bên không tấn công trực tiếp vào thuật toán mã hóa mà thay vào đó, tấn công lên các hệ thống thực hiện thuật toán có sơ hở làm lộ dữ liệu.

Tháng 4 năm 2005 Daniel.J.Bernstin ông bố một tấn công lên hệ thống mã hóa AES trong OpenSL. Một máy chủ được thiết kế để đưa ra tối đa thông tin về thời gian có thể thu được và cuộc tấn công cần tới 200 triệu bản rõ lựa chọn. Một số người cho rằng tấn công không thể thực hiện được trên Internet với khoảng cách vài điểm mạng.

Tháng 10 năm 2005, Adi Shamir và 2 nhà nghiên cứu khác có một bài nghiên cứu minh họa một vài dạng khác. Trong đó, một tấn công có thể lấy được khóa AES với 800 lần ghi trong 65 mili giây. Tấn công này yêu cầu kẻ tấn công có khả năng chạy chương trình trên chính hệ thống thực hiện mã hóa. Có 3 phương pháp tấn công bên:

1. Tấn công thời gian:

Quá trình triển khai các thuật toán mã hóa thường thực hiện tính toán trong khoảng thời gian không đổi, để tối ưu hóa hiệu suất. Nếu hoạt động đó liên quan đến các thông số bí mật, từ các biến thời gian có thể rò rỉ một số thông tin và cung cấp thông tin về quá trình triển khai, một phân tích thống kê cụ thể có thể thu được các thông số bí mật. Về cơ bản, tấn công thời gian là một hình thức lấy thông tin cá nhân của người dùng bằng cách đo thời gian khi người đó thực hiện mã hóa. Nguyên tắc của tấn công này rất đơn giản: phải khai thác đúng thời điểm đang thực hiện

2. Tấn công dựa vào lỗi:

Hầu hết các thiết bị thực hiện các quá trình mã hóa khác nhau thường đáng tin cậy hơn. Lỗi phần cứng và các lỗi xảy ra trong quá trình triển khai của modul mã hóa trên thực tế đã chứng minh những ảnh hưởng nghiêm trọng của nó đến bảo mật. Những hành vi hoặc đầu ra bị lỗi có thể trở thành các kênh kề quan trọng, đôi khi sẽ là nguyên nhân làm tăng các lỗ hổng bảo mật trong mã hóa.

Có hai loại tấn công kênh kề dựa vào lỗi. Loại thứ nhất là các kênh sinh ra từ các lỗi tính toán trong quá trình tính toán mã hóa trong một modul bị tấn công. Những lỗi này có thể là ngẫu nhiên hay cố ý, gây ra.

Loại thứ hai của tấn công kênh kề dựa trên lỗi, bằng cách cố ý gửi các dữ liệu đầu vào bị lỗi đến modul bị tấn công. Modul sẽ gửi một thông báo lỗi đến người sử dụng, quá trình tính toán sẽ bị ngừng lại.

Tóm lại, các cuộc tấn công dựa vào lỗi thực hiện qua hai bước: chèn lỗi (fault injection) và khai thác lỗi.

Bước đầu tiên bao gồm chèn một lỗi tại một thời điểm thích hợp trong quá trình xử lý. Việc chèn lỗi phụ thuộc vào phần cứng thiết bị. Lỗi có thể được phát sinh trong thẻ thông minh bằng các tác động vào môi trường của nó và đặt nó trong điều kiện bình thường. Một số lỗi là bất thường và điện áp cao hoặc thấp bất thường, đồng hồ, nhiệt độ, bức xạ, ánh sáng.

Bước thứ hai bao gồm khai thác các kết quả sai hoặc các hành vi bất thường. Việc khai thác lỗi phụ thuộc vào quá trình triển khai và thiết kế phần mềm. Trong trường hợp là một thuật toán thì nó sẽ phụ thuộc vào đặc điểm kỹ thuật của thuật toán đó.

Tùy thuộc vào loại phân tích được áp dụng, việc chèn lỗi phải được thực hiện ngay lập tức hoặc trong một khoảng thời gian nhất định.

3. Tấn công phân tích năng lượng:

Ngoài thời gian hoạt động và lỗi, năng lượng tiêu thụ của một thiết bị mật mã có thể cung cấp nhiều thông tin về các hoạt động và các thông số của hệ thống. Tấn công phân tích năng lượng chỉ có thể áp dụng vào triển khai phần cứng của hệ thống mật mã. Loại tấn công này thực sự hiệu quả và đã được chứng minh khi tấn công thành công thẻ thông minh hoặc các hệ thống chuyên dụng lưu trữ khóa bí mật.

Về cơ bản, tấn công phân tích năng lượng có thể chia thành Tấn công phân tích năng lượng đơn giản (Simple Power Analysis SPA) và Tấn công phân tích năng lượng vi sai (Differential Power Analysis DPA). Trong tấn công SPA, dựa vào các dấu vết về năng lượng tiêu thụ để đoán thời gian thực thi dữ liệu, giá trị của đầu vào, đầu ra. Tấn công DPA sử dụng phương pháp thống kê trong quá trình xử lý.

4. Tấn công điện từ:

Như các thiết bị điện, các thành phần của 1 máy tính thường tạo ra các bức xạ điện từ, kẻ tấn công sẽ quan sát những bức xạ điện từ phát ra và có thể hiểu được mối quan hệ giữa quá trình tính toán và dữ liệu, từ đó có thể suy ra thông tin về tính toán và dữ liệu. Tấn công phân tích điện từ (ElectroMagnetic Analysis EMA) được phân thành 2 loại chính: Phân tích điện từ đơn giản (Simple ElectroMagnetic Analysis SEMA) và Phân tích điện từ vi sai (Differential ElectroMagnetic Analysis DEMA).

Có 2 phương pháp chống lại các cuộc tấn công phân tích điện từ (EM attack): giảm cường độ tín hiệu và giảm thông tin tín hiệu. Kỹ thuật giảm cường độ tín hiệu bao gồm thiết kế lại mạch để giảm những phát sinh ngoài ra muốn và thiết lập vùng bảo mật để giảm.

VI. TÀI LIỆU THAM KHẢO

1. Tổng quan về an toàn bảo mật HTTT_ thầy Hoàng Xuân Dậu
2. Luận Văn sinh viên nghiên cứu AES http://luanvan.co/luan-van/tim-hieu-ve-thuat-toan-aes- 44835/
3. Bài giảng an toàn bảo mật trường đại học Nha Trang