TCVN TIÊU CHUẨN QUỐC GIA

TCVN 7816 : 2007

Xuất bản lần 1

CÔNG NGHỆ THÔNG TIN - KỸ THUẬT MẬT MÃ - THUẬT TOÁN MÃ DỮ LIỆU AES

Cryptographic technique – Cryptographic algorithms – Data Encryption Algorithm AES

HÀ NỘI - 2007

Mục lục

Trang

Lời nói đầu

TCVN 7816 : 2007 được xây dựng trên cơ sở tham khảo tiêu chuẩn FIPS 197 và ISO /IEC18033-3..

TCVN 7816 : 2007 do Tiểu ban Kỹ thuật Tiêu chuẩn TCVN/JTC1/SC 27 "*Các kỹ thuật mật mã*" biên soạn, Ban cơ yếu Chính phủ đề nghị, Bộ Khoa học và Công nghệ công bố.

Xuất bản lần 1

Công nghệ thông tin – Kỹ thuật mật mã Thuật toán mã dữ liệu AES

Information technology - Cryptographic technique – Data encryption algorithm AES

1 Phạm vi áp dụng

Tiêu chuẩn áp dụng cho việc mã hóa dữ liệu trong các hoạt động giao dịch điện tử của các tổ chức, công dân Việt Nam và tổ chức, công dân nước ngoài có quan hệ kinh tế - xã hội với tổ chức, công dân Việt Nam.

2 Tài liệu viện dẫn

ISO/IEC 18033-3:2005, Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers (Công nghệ thông tin - Kỹ thuật mật mã – Các thuật toán mã hoá – Phần 3: Các khối mật mã).

FIPS PUB 197 – Announcing the Advance Encryption Standard (Thông báo về tiêu chuẩn mã hoá tiên tiến) (AES), NIST, 2001.

3 Thuật ngữ và định nghĩa

3.1 Thuật ngữ và các từ viết tắt

Tiêu chuẩn này sử dụng các khái niệm sau:

AES	Tiêu chuẩn mã hóa tiên tiến.
Bản mã	Dữ liệu đầu ra của Phép mã hóa hoặc dữ liệu đầu vào của Phép giải mã.
Bản rõ	Dữ liệu đầu vào của Phép mã hóa hoặc dữ liệu đầu ra của Phép giải mã.
Bít	Một chữ số nhị phân có giá trị 0 hoặc 1.
Byte	Một nhóm gồm 8 bít được xem như là một thực thể đơn lẻ hoặc một thực mảng 8 bít đơn lẻ.
Hộp-S	Một bảng thay thế phi tuyến, được sử dụng trong một số phép thay thế byte và trong quy trình Mở rộng khóa, nhằm thực hiện một phép thay thế một-một đối với giá trị mỗi byte.
Khóa mã	Một số được giữ bí mật dùng cho quy trình Mở rộng khóa nhằm tạo ra

	một tập các Khóa vòng.
Khóa vòng	Các khóa vòng là các giá trị sinh ra từ Khóa mã bằng quy trình Mở rộng khóa, chúng được áp dụng lên Trạng thái trong Phép mã hóa và Phép giải mã. Trong thuật toán AES, Khóa vòng có thể xem như là một mảng chữ nhật của các byte có 4 hàng và ***Nk*** cột.
Khối	Dãy liên tiếp các bít nhị phân có thể là đầu vào, đầu ra, Trạng thái và Khóa vòng. Độ dài của một dãy là số lượng bít chứa trong dãy đó. Các khối cũng còn được xem như là một mảng các byte.
Mảng	Tập hợp ở dạng liệt kê các thực thể đồng nhất (ví dụ như mảng các byte).
Mở rộng khóa	Phép được sử dụng để tạo ra một loạt các khóa vòng từ một Khóa mã.
Phép biến đổi Affine	Phép biến đổi gồm phép nhân với một ma trận và phép cộng với một véc-tơ.
Phép giải mã	Một loạt các phép biến đổi để biến đổi bản mã thành bản rõ sử dụng một Khóa mã.
Phép mã hóa	Loạt các phép biến đổi để biến đổi bản rõ thành bản mã sử dụng Khóa mã.
Rijndael	Thuật toán mật mã được mô tả trong Tiêu chuẩn này.
Trạng thái	Kết quả mã hóa trung gian. Trong AES Trạng thái được mô tả dưới dạng một mảng chữ nhật của các byte có 4 hàng và ***Nb*** cột.
Từ	Một nhóm gồm 32 bít được xem như là một thực thể đơn lẻ hoặc như là một mảng 4 byte.

3.2 Các tham số thuật toán, các ký hiệu và các hàm

Các tham số thuật toán, ký hiệu và hàm sau được sử dụng trong tiêu chuẩn này:

`AddRoundKey()`	Phép biến đổi trong Phép mã hóa và Phép giải mã. Trong đó, một Khóa vòng được cộng thêm vào Trạng thái bằng phép toán XOR. Độ dài của Khóa vòng bằng độ dài của Trạng thái (chẳng hạn, nếu ***Nb*** = 4 thì độ dài của Khóa vòng là 128 bít hay 16 byte).
`InvMixColumns()`	Phép biến đổi dùng trong Phép giải mã, là phép nghịch đảo của **`MixColumns()`**.
`InvShiftRows()`	Phép biến đổi dùng trong Phép giải mã là phép nghịch đảo của **`ShiftRows()`**.
`InvSubBytes()`	Phép biến đổi dùng trong Phép giải mã là phép nghịch đảo của **`SubBytes()`**.
K	Khóa mã.
`MixColumns()`	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.
Nb	Số các cột (các từ 32 bít) tạo nên Trạng thái. Trong tiêu chuẩn này, ***Nb*** = 4 (Xem thêm điều 6.3).
Nk	Số lượng các từ 32 bít trong Khóa mã. Trong tiêu chuẩn này, ***Nk*** = 4, 6 hoặc 8 (Xem thêm điều 6.3).
Nr	Số lượng vòng lặp, đó là một hàm của ***Nk*** và ***Nb*** (chúng là cố định). Trong tiêu chuẩn này, ***Nr*** = 10, 12 hoặc 14 (Xem thêm điều 6.3).
`Rcon[]`	Mảng từ hằng số vòng.
`RotWord()`	Hàm được sử dụng trong quy trình Mở rộng khóa bằng cách lấy một từ 4 byte và thực hiện một phép hoán vị vòng (quay vòng).
`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 các offset khác nhau.
`SubBytes()`	Phép biến đổi dùng trong Phép mã hóa áp dụng lên Trạng thái sử dụng một bảng thay thế byte phi tuyến (Hộp-S) trên mỗi byte Trạng thái một cách độc lập.
`SubWord()`	Hàm được sử dụng trong phép Mở rộng khóa, lấy một từ đầu vào gồm 4 byte và áp dụng Hộp-S vào mỗi byte để tạo đầu ra là một từ.
XOR	Phép toán HOẶC-loại trừ.
$\oplus$	Phép toán HOẶC-loại trừ.
$\otimes$	Phép nhân hai đa thức (có bậc < 4) theo modulo $x^4 + 1$.
•	Phép nhân trên trường hữu hạn.

4 Các ký pháp và Quy ước

4.1 Đầu vào và Đầu ra

Đầu vào và **đầu ra** của thuật toán AES là các **dãy nhị phân 128 bít** (chứa giá trị 0 hoặc 1). Các dãy bít này đôi khi được gọi là các **khối** và số lượng bít chứa trong dãy gọi là độ dài của dãy. **Khóa mã** dùng cho thuật toán AES là một **dãy nhị phân 128, 192 hoặc 256 bít**. Tiêu chuẩn này không cho phép sử dụng các đầu vào, đầu ra hoặc Khóa mã có độ dài khác với các kích thước trên .

Các bít trong các dãy nói trên sẽ được đánh số thứ tự từ không cho đến số cuối cùng. Số cuối cùng này là một số nhỏ hơn độ dài dãy (độ dài khối hay độ dài khóa) một đơn vị. Một số i ứng với một bít theo thứ tự sẽ có giá trị nằm trong khoảng $0 \le i < 128$, $0 \le i < 192$ hoặc $0 \le i < 256$ phụ thuộc vào độ dài của khối và độ dài khóa (như đã đề cập ở phần trên).

4.2 Byte

Đơn vị xử lý cơ bản trong thuật toán AES, đó là một dãy 8 bít được xem như là một thực thể đơn lẻ. Các dãy

đầu vào, đầu ra và Khóa mã đã nói ở điều 3.1 có thể biểu diễn theo các mảng byte bằng cách chia các dãy đó thành các nhóm 8 bít liên tiếp nhau (xem điều 3.3). Nếu đầu vào, đầu ra hoặc Khóa mã ký hiệu là *a* thì mảng byte thu được sẽ được tham chiếu đến theo một trong hai dạng: a_n hoặc $a[n]$, trong đó n nằm trong khoảng giới hạn sau:

Nếu độ dài Khóa = 128 bít thì $0 \le n < 16$; Độ dài Khối = 128 bít, $0 \le n < 16$;

Nếu độ dài Khóa = 192 bít thì $0 \le n < 24$;

Nếu độ dài Khóa = 256 bít thì $0 \le n < 32$;

Tất cả giá trị byte trong thuật toán AES sẽ được biểu diễn dưới dạng ghép các bít riêng lẻ (0 hoặc 1) giữa các dấu ngoặc theo thứ tự $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. Các byte này được được xem như là các phần tử trên trường hữu hạn bằng việc sử dụng biểu diễn đa thức:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_i x^i. \qquad \textbf{(3.1).}$$

Ví dụ, dãy bít {01100011} sẽ được biểu diễn thành phần tử trường hữu hạn là $x^6 + x^4 + x + 1$.

Cũng có thể biểu diễn các giá trị byte theo dạng thập lục phân (hexadecimal) bằng cách thay các nhóm 4 bít bằng một ký tự theo giá trị tương ứng cho bởi Hình 1.

Nhóm bít	Ký tự
0000	0
0001	1
0010	2
0011	3

Nhóm bít	Ký tự
0100	4
0101	5
0110	6
0111	7

Nhóm bít	Ký tự
1000	8
1001	9
1010	a
1011	b

Nhóm bít	Ký tự
1100	c
1101	d
1110	e
1111	f

Hình 1 - Ký hiệu thập lục phân tương ứng với các nhóm 4 bít

Chẳng hạn phần tử {01100011} được viết thành {63}, trong đó ký tự đại diện cho 4 bít cao sẽ được đặt bên trái.

Một số phép toán trên trường hữu hạn đòi hỏi thêm một bít (b_8) vào bên trái của một byte 8-bít. Khi có mặt bít mở rộng theo cách này người ta đặt trước byte 8-bít ký hiệu '{01}'. Chẳng hạn, một dãy 9 bít sẽ được biểu diễn dưới dạng {01}{1b}.

4.3 Mảng byte

Các mảng byte được biểu diễn theo dạng sau:

$a_0a_1a_2...a_{15}$

Cho một dãy 128 bít, nếu thứ tự các bít của dãy đầu vào này lần lượt là:

$input_0\ input_1\ input_2 \ldots input_{126}\ input_{127}$

thì mảng byte sẽ được xác định như sau:

$a_0 = \{input_0, input_1, \ldots, input_7\}$

$a_1 = \{input_8, input_9, \ldots, input_{15}\}$

Λ

a15 = {input120, input121, ..., input127}.

Cách này có thể áp dụng cho các dãy dài hơn (chẳng hạn cho dãy khóa 192 và 256 bít) một cách tổng quát như sau:

$a_n = \{input_{8n}, input_{8n+1}, \ldots, input_{8n+7}\}$. (3.2)

Đối chiếu điều 3.2 và điều 3.3, Hình 2 thể hiện cách các bít được đánh số thứ tự trong các byte:

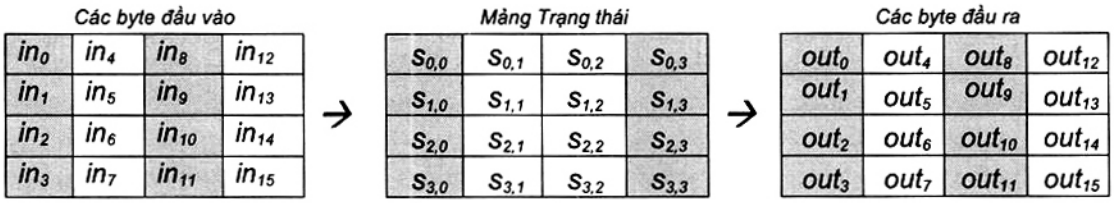
Dãy bít vào	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
Số hiệu byte	0								1								2								...
TT bít trong byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Hình 2 - Cách thể hiện thứ tự các byte và bít

4.4 Trạng thái

Các phép toán của thuật toán AES được thực hiện trên một mảng byte hai chiều gọi là **Trạng thái**. Trạng thái bao gồm 4 hàng byte, mỗi hàng chứa ***Nb*** byte, trong đó ***Nb*** bằng số lượng từ trong một khối. Trong một mảng Trạng thái ký hiệu là *s* thì mỗi byte riêng biệt có hai chỉ số, chỉ số hàng ký hiệu là *r*, $0 \le r < 4$, chỉ số cột ký hiệu là *c*, $0 \le c < Nb$. Điều này cho phép mỗi byte riêng biệt của Trạng thái được tham chiếu đến theo dạng $s_{r,c}$ hoặc *s*[*r*,*c*]. Trong tiêu chuẩn này, ***Nb*** = *4*, do đó $0 \le c < 4$ (xem thêm điều 6.3).

Tại thời điểm bắt đầu Phép mã hóa hoặc Phép giải mã như sẽ mô tả trong điều 5 thì đầu vào là một mảng byte $in_0, in_1, \ldots in_{15}$ sẽ được sao (copy) vào mảng Trạng thái như minh họa ở Hình 3. Các phép toán trên Phép mã hóa hoặc Phép giải mã tiếp theo được thao tác với mảng Trạng thái này, giá trị cuối cùng sẽ được copy đến đầu ra – đó là một mảng byte $out_0, out_1, \ldots out_{15}$.



Hình 3 - Đầu vào và đầu ra mảng Trạng thái

Như vậy, vào thời điểm khởi đầu Phép mã hóa hoặc Phép giải mã, mảng đầu vào *in* được copy vào mảng Trạng thái theo lược đồ:

$s[r,c] = in[r+4c]$ với $0 \le r < 4$ và $0 \le c < Nb$, (3.3)

và tại thời điểm kết thúc Phép mã hóa hoặc Phép giải mã, Trạng thái được copy vào mảng đầu ra *out* theo:

$out[r + 4c] = s[r, c]$ với $0 \le r < 4$ và $0 \le c < Nb$. (3.4)

4.5 Trạng thái được coi như một mảng các Cột

Bốn byte trên mỗi cột trong mảng Trạng thái tạo thành một từ 32 bít, trong đó số thứ tự của hàng *r* cho biết chỉ số của bốn byte trong mỗi từ. Vì thế có thể coi Trạng thái như là mảng một chiều chứa các từ 32 bít (các cột) $w_0 ... w_3$, trong đó số cột *c* cho biết chỉ mục trong mảng. Căn cứ vào Hình 3 thì Trạng thái có thể xem như là một mảng gồm bốn từ được xác định như sau:

$$w_0 = s_{0,0}\, s_{1,0}\, s_{2,0}\, s_{3,0} \qquad w_2 = s_{0,2}\, s_{1,2}\, s_{2,2}\, s_{3,2} \qquad (3.5)$$

$$w_1 = s_{0,1}\, s_{1,1}\, s_{2,1}\, s_{3,1} \qquad w_3 = s_{0,3}\, s_{1,3}\, s_{2,3}\, s_{3,3}\ .$$

5 Cơ sở toán học

5.1 Phép cộng

Phép cộng hai phần tử trên trường hữu hạn được thực hiện bằng cách "cộng" hệ số của các lũy thừa tương ứng trong các đa thức của hai phần tử. Phép cộng được thực hiện theo phép toán XOR (ký hiệu là $\oplus$). Tức là theo modulo 2 thì $1 \oplus 1 = 0$, $1 \oplus 0 = 1$ và $0 \oplus 0 = 0$. Do đó, phép trừ đa thức cũng là phép cộng đa thức.

Việc cộng hai phần tử trên trường hữu hạn có thể được mô tả như phép cộng modulo 2 các bít tương ứng trong byte. Chẳng hạn, hai byte $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ và $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ sẽ có tổng là $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$, trong đó mỗi bít $c_i = a_i \oplus b_i$ (tức là, $c_7 = a_7 \oplus b_7$, $c_6 = a_6 \oplus b_6$, ... $c_0 = a_0 \oplus b_0$).

Lấy ví dụ, các cách viết sau là tương đương nhau:

$(x^6 + x^4 + x^2 + x + 1)$	$+ (x^7 + x + 1)$	$= x^7 + x^6 + x^4 + x^2$	(dạng đa thức)
{01010111}	$\oplus$ {10000011}	= {11010100}	(dạng nhị phân)
{57}	$\oplus$ {83}	= {d4}	(dạng thập lục phân)

5.2 Phép nhân

Trong cách biểu diễn đa thức, phép nhân trên trường GF(2^8) (ký hiệu là •) tương đương với phép nhân đa thức theo modulo của một **đa thức bất khả qui** bậc 8. Một đa thức được gọi là bất khả quy nếu nó chỉ có

ước là 1 và chính nó. **Đối với thuật toán AES, đa thức bất khả quy** này là:

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (4.1)$$

hoặc biểu diễn dưới dạng thập lục phân là {01}{1b}.

Ví dụ, {57} • {83} = {c1}, vì:

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x_2 + x + 1$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

và

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1.$$

Phép rút gọn theo modulo *m(x)* cho ta kết quả là một đa thức nhị phân có bậc thấp hơn 8 vì thế có thể trình bày ở dạng byte. Không giống như phép cộng, không tồn tại phép toán đơn giản ở mức byte cho phép nhân.

Phép nhân định nghĩa ở trên có tính kết hợp và phần tử {01} gọi là phần tử đơn vị. Đối với bất cứ đa thức nhị phân khác không $b(x)$ nào có bậc nhỏ hơn 8 thì phần tử nghịch đảo của $b(x)$ ký hiệu là $b^{-1}(x)$, có thể tìm được như sau: Thuật toán Ơcơlit [7] mở rộng được sử dụng để tính ra đa thức $a(x)$ và $c(x)$ thỏa mãn:

$$b(x)a(x) + m(x)c(x) = 1 \quad (4.2)$$

Bởi vì $a(x) \bullet b(x) \bmod m(x) = 1$ cho nên

$$b^{-1}(x) = a(x) \bmod m(x). \quad (4.3)$$

Hơn nữa, với bất kỳ $a(x)$, $b(x)$ và $c(x)$ trong trường, ta luôn có:

$$a(x) \bullet (b(x) + c(x)) = a(x) \bullet b(x) + a(x) \bullet c(x).$$

Từ đó suy ra rằng, tập hợp 256 giá trị byte có thể với phép toán XOR và phép nhân được định nghĩa như trên sẽ có cấu trúc của trường hữu hạn GF(2^8).

5.2.1 Phép nhân với *x*

Nhân một đa thức nhị phân được định nghĩa trong điều (3.1) với đa thức x sẽ cho kết quả là đa thức:

$$b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x. \quad (4.4)$$

Có thể tìm được kết quả của $x \bullet b(x)$ bằng cách rút gọn kết quả trên theo modulo $m(x)$ như định nghĩa ở (4.1). Nếu $b_7 = 0$ thì kết quả đã ở dạng rút gọn. Nếu $b_7 = 1$ thì sự rút gọn đạt được bằng cách trừ đi (tức là phép XOR) với đa thức $m(x)$. Từ đó suy ra, phép nhân với x (tức là x có giá trị {00000010} hoặc {02}) có thể được thực thi ở mức byte như là một phép dịch trái và một phép toán XOR có điều kiện ở mức bít với {1b}. Phép toán trên byte này ký hiệu là `xtime()`. Phép nhân với lũy thừa cao hơn của x có thể được thực hiện theo cách áp dụng lặp lại hàm `xtime()`. Bằng cách cộng các kết quả trung gian có thể thực hiện phép nhân với một byte bất kỳ.

Chẳng hạn, {57} • {13} = {fe} vì:

{57} • {02} = xtime({57}) = {ae}

{57} • {04} = xtime({ae}) = {47}

{57} • {08} = xtime({47}) = {8e}

{57} • {10} = xtime({8e}) = {07},

Do đó,

{57} • {13} = {57} • ({01} ⊕ {02} ⊕ {10})

= {57} ⊕ {ae} ⊕ {07}

= {fe}.

5.3 Đa thức với các hệ số trên trường GF(2^8)

Đa thức bốn hạng tử có thể được xác định với các hệ số là các phần tử trên trường hữu hạn có dạng:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \quad (4.5).$$

Đa thức này sẽ được ký hiệu như một từ có dạng [a_0 , a_1 , a_2 , a_3]. Chú ý rằng các đa thức trong phần này khác với các đa thức sử dụng khi định nghĩa về các phần tử trên trường hữu hạn, mặc dù cả hai dạng đa thức đều sử dụng cùng một biến x. Các hệ số của đa thức trong phần này bản thân đã là các phần tử trên trường hữu hạn, giá trị của chúng là các byte thay vì các bít. Ngoài ra, phép nhân các đa thức bốn hạng tử sử dụng một đa thức rút gọn khác sẽ được xác định dưới đây. Sự khác biệt này sẽ được làm sáng rõ trong trường hợp cụ thể.

Để minh họa cho các phép cộng và phép nhân đa thức, giả sử:

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \quad (4.6)$$

là một đa thức bốn hạng tử thứ hai. Phép cộng được thực hiện bằng cách cộng các hệ số trên trường hữu

hạn tương ứng cùng bậc với *x*. Phép cộng này tương đương với một phép toán XOR giữa các byte tương ứng trong mỗi từ, nói cách khác là một phép XOR của các giá trị từ đầy đủ.

Từ (4.5) và (4.6), ta có :

$$a(x)+b(x)=(a_3\oplus b_3)x^3+(a_2\oplus b_2)x^2+(a_1\oplus b_1)x+(a_0\oplus b_0) \qquad (4.7)$$

Phép nhân được thực hiện qua hai bước. Bước thứ nhất, thực hiện khai triển đại số. Sau đó, gộp các số hạng cùng bậc. Kết quả cho ta tích:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \qquad (4.8)$$

Trong đó:

$$c_0 = a_0 \bullet b_0 \qquad c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \qquad c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$

$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \qquad c6 = a3 \bullet b3 \qquad (4.9)$$

$$c3 = a3 \bullet b0 \oplus a2 \bullet b1 \oplus a1 \bullet b2 \oplus a0 \bullet b3$$

Kết quả thu được, *c(x)* không biểu diễn được bằng một từ 4 byte. Do đó, bước thứ hai là thực hiện phép nhân rút gọn *c*(*x*) theo modulo đa thức bậc 4. Kết quả có thể rút gọn thành một đa thức có bậc nhỏ hơn 4. **Đối với thuật toán AES, đa thức này là $x^4 + 1$,** vì thế:

$$x^i \bmod (x^4+1) = x^{i \bmod 4} \qquad (4.10)$$

Phép nhân modulo giữa *a*(*x*) và *b*(*x*) ký hiệu là $a(x) \otimes b(x)$ cho kết quả là một đa thức bốn hạng tử *d*(*x*):

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0 \qquad (4.11)$$

trong đó:

$$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$

$$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$

$$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \qquad (4.12)$$

$$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

Khi *a*(*x*) là một đa thức cố định thì phép toán chỉ ra trong biểu thức (4.11) có thể được viết dưới dạng ma trận

như sau:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0\ a_3\ a_2\ a_1 \\ a_1\ a_0\ a_3\ a_2 \\ a_2\ a_1\ a_0\ a_3 \\ a_3\ a_2\ a_1\ a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (4.13)$$

Do $x^4 + 1$ không phải là đa thức bất khả quy trên trường GF(2^8) nên phép nhân với một đa thức bất kỳ bốn hạng tử không nhất thiết phải có tính khả nghịch. Tuy nhiên, thuật toán AES sử dụng một đa thức cố định bốn hạng tử mà phép nhân với nó có tính khả nghịch (xem Phần 6.1.3 và Phần 6.3.3), đó là đa thức:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (4.14)$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}. \quad (4.15)$$

Một đa thức khác được sử dụng trong thuật toán AES (xem thêm về hàm **`RotWord()`** trong Phần 5.2) có $a_0 = a_1 = a_2 = \{00\}$ and $a_3 = \{01\}$ là đa thức x^3. Xem kỹ biểu thức (4.13) ta thấy rằng hiệu quả của nó là ở chỗ nó biến đổi một từ đầu vào thành một từ đầu ra bằng phép quay vòng byte. Nghĩa là $[b_0, b_1, b_2, b_3]$ được chuyển thành $[b_1, b_2, b_3, b_0]$.

6 Quy định thuật toán

Đối với thuật toán AES, **độ dài của khối đầu vào, khối đầu ra và Trạng thái đều là 128 bít**. Như vậy ***Nb*** = 4 là số lượng các từ 32 bít (số cột) của Trạng thái.

Trong thuật toán AES, **độ dài Khóa mã K có thể là 128, 192 hoặc 256 bít**. Độ dài khóa được biểu diễn bằng một số ***Nk*** = 4, 6, hoặc 8 thể hiện số lượng các từ 32 bít (số cột) của Khóa mã.

Đối với thuật toán AES, số vòng được thay đổi trong quá trình thực thi thuật toán phụ thuộc vào kích cỡ khóa. Số vòng này được ký hiệu là ***Nr***, trong AES, ***Nr*** = 10 khi ***Nk*** = 4, ***Nr*** = 12 khi ***Nk*** = 6 và ***Nr*** = 14 khi ***Nk*** = 8.

Các tổ hợp Khóa-Khối-Vòng phù hợp đối với tiêu chuẩn này cho bởi Hình 4. Việc thực thi cụ thể thuật toán có liên quan đến độ dài khóa, kích cỡ khối và số vòng nên xem thêm điều 6.3.

	Độ dài khóa *(Nk từ)*	Độ dài khối *(Nb từ)*	Số vòng *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Hình 4 - Các tổ hợp Khóa - Khối - Vòng

Đối với Phép mã hóa và Phép giải mã, thuật toán AES sử dụng một hàm vòng gồm bốn phép biến đổi byte: 1) phép thay thế byte sử dụng một bảng thay thế (Hộp-S), 2) phép dịch chuyển hàng của mảng Trạng thái

theo các offset khác nhau, 3) phép trộn dữ liệu trong mỗi cột của mảng Trạng thái, 4) phép cộng Khóa vòng vào Trạng thái. Các phép biến đổi này (cũng như phép nghịch đảo của chúng) được mô tả trong điều 5.1.1 – 5.1.4 và 5.3.1 – 5.3.4.

Phép mã hóa và Phép giải mã được mô tả ở điều 5.1 và điều 5.3, Lược đồ khóa sẽ được mô tả ở điều 5.2.

6.1 Phép mã hóa

Tại thời điểm bắt đầu Phép mã hóa, đầu vào được sao vào mảng Trạng thái sử dụng các quy ước đã mô tả ở điều 3.4. Sau phép cộng Khóa vòng khởi đầu, mảng Trạng thái được biến đổi bằng cách thực thi một hàm vòng liên tiếp với số lần vòng lặp 10, 12 hoặc 14 (phụ thuộc vào độ dài khóa), vòng cuối cùng khác biệt không đáng kể với ***Nr*** – 1 vòng đầu tiên. Trạng thái cuối cùng được chuyển thành đầu ra như đã mô tả ở điều 3.4.

Hàm vòng được tham số hóa bằng cách sử dụng một lược đồ khóa là mảng một chiều chứa các từ bốn byte nhận từ phép Mở rộng khóa mô tả ở điều 5.2.

Phép mã hóa được mô tả theo dạng tựa mã ở Hình 5. Các phép biến đổi cụ thể - **`SubBytes()`**, **`ShiftRows()`**, **`MixColumns()`** và **`AddRoundKey()`** – dùng để xử lý Trạng thái được mô tả ở các mục tiếp theo. Ở Hình 5, mảng **`w[]`** có chứa một lược đồ khóa sẽ được mô tả ở điều 5.2.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]
   state = in
   AddRoundKey(state, w[0,Nb-1])                          //Xem phần 5.1.4
   for round = 1 step 1 to Nr-1
      SubBytes(state)                                     //Xem phần 5.1.1
      ShiftRows(state)                                    //Xem phần 5.1.2
      MixColumns(state)                                   //Xem phần 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for
   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
   out = state
 end
```

Hình 5 - Đoạn tựa mã cho Phép mã hóa[1]

Như chúng ta đã thấy ở Hình 5, tất cả ***Nr*** vòng là giống hệt nhau chỉ có ngoại lệ ở vòng cuối cùng, vòng này không có phép biến đổi **`MixColumns()`**.

Phụ lục B trình bày một ví dụ về Phép mã hóa, chỉ ra cụ thể các giá trị của mảng Trạng thái tại thời điểm bắt đầu mỗi vòng và sau khi áp dụng bốn phép biến đổi.

[1] Một số phép biến đổi (vd: SubBytes(), ShiftRows(),...) thao tác trên mảng Trạng thái được tham chiếu bởi con trỏ "state". Phép biến đổi AddRoundKey() sử dụng một con trỏ khác để tham chiếu đến Khóa vòng.

6.1.1 Phép biến đổi `SubBytes()`

Phép biến đổi **`SubBytes()`** là một phép thay thế phi tuyến được thực hiện độc lập trên mỗi byte của Trạng thái sử dụng một bảng thay thế (Hộp-S). Hộp-S này có tính khả nghịch được tạo bởi hai Phép biến đổi (Hình 7) sau:

1. Phép nghịch đảo trên trường hữu hạn GF(2^8), mô tả ở điều 4.2, trong đó phần tử {00} được ánh xạ vào chính nó.

2. Áp dụng phép biến đổi affine (trên trường GF(2)) sau:

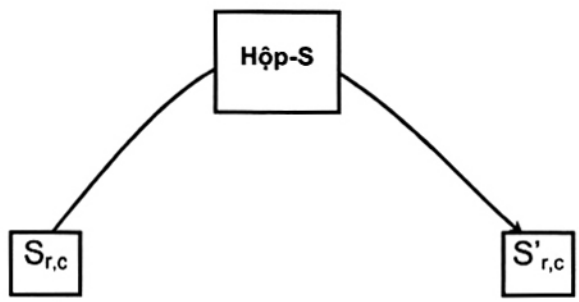
$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad (6.1)$$

với $0 \le i < 8$, trong đó b_i là bít thứ i trong byte và c_i là bít thứ i của một byte c có giá trị là {63} hoặc {01100011}. Từ đây về sau, dấu phẩy trên các biến (ví dụ b') có nghĩa là biến đã được cập nhật một giá trị nằm phía bên phải.

Ở dạng ma trận, phần tử của phép biến đổi affine trong Hộp-S có thể biểu diễn như sau:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad (6.2)$$

Hình 6 minh họa tác động của Phép biến đổi **`SubBytes()`** lên Trạng thái



Hình 6

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$s'_{0,0}$	$s'_{0,1}$	$s'_{0,2}$	$s'_{0,3}$
$s_{1,0}$	$s'_{1,1}$	$s'_{1,2}$	$s'_{1,3}$
$s'_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s'_{3,0}$	$s'_{3,1}$	$s'_{3,2}$	$s'_{3,3}$

- Phép `SubBytes()` áp dụng Hộp-S vào mỗi byte của Trạng thái

Hộp-S sử dụng trong Phép biến đổi **`SubBytes()`** được trình bày ở dạng thập lục phân như trong Hình 7. Ví dụ, nếu $s_{1,1} = \{53\}$, thì giá trị thay thế được xác định bởi vị trí giao của hàng số "5" và cột số "3" trong Hình 7. Điều này cho kết quả $s'_{1,1}$ là {ed

		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

Hình 7 - Hộp-S: các giá trị thay thế cho byte {xy} (theo dạng thập lục phân)

6.1.2 Phép biến đổi `ShiftRows()`

Trong phép biến đổi **`ShiftRows()`**, các byte ở ba hàng cuối của Trạng thái được dịch vòng theo số lượng byte khác nhau (các offset). Hàng đầu tiên, $r = 0$, không được dịch chuyển.

Cụ thể, Phép biến đổi **`ShiftRows()`** thực hiện như sau:

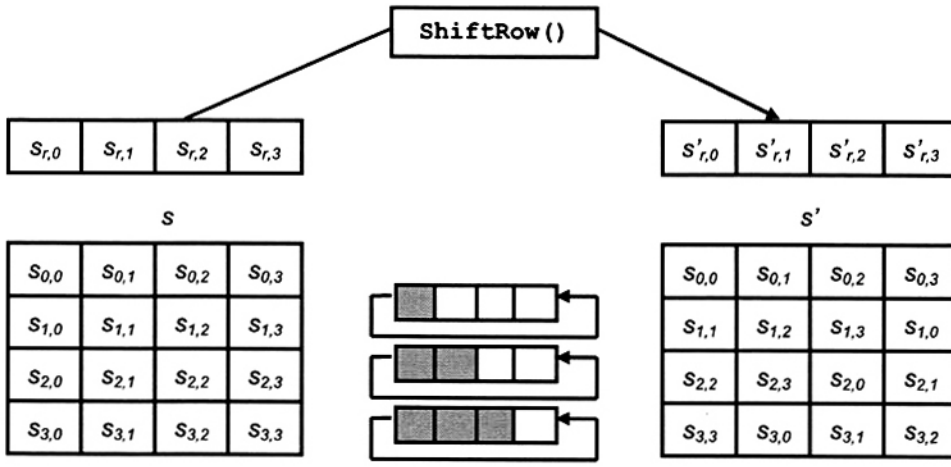
$$s'_{r,c} = s_{r,(c + shift(r,Nb)) \bmod Nb} \text{ với } 0 < r < 4 \text{ và } 0 \le c < \mathbf{Nb} \quad (6.3)$$

trong đó giá trị dịch chuyển *shift*(*r*,*Nb*) phụ thuộc vào số lượng hàng *r*, như sau (chú ý rằng ***Nb*** = 4):

$$shift(1,4) = 1 ; shift(2,4) = 2 ; shift(3,4) = 3 \quad (5.4)$$

Hệ quả của việc này là di chuyển các byte từ vị trí thấp của hàng (là các giá trị thấp của *c* trong một hàng đã cho), trong khi các byte "thấp nhất" được đưa lên trên cùng của hàng (các giá trị cao của *c* trong một hàng đã cho).

Hình 8 minh họa Phép biến đổi **`ShiftRows()`**.



Hình 8 - Các phép dịch vòng trong `ShiftRows()` đối với ba hàng cuối của mã Trạng thái

6.1.3 Phép biến đổi `MixColumns()`

Phép biến đổi **`MixColumns()`** trên Trạng thái được thực hiện theo từng cột, nghĩa là mỗi cột được xem như là một đa thức 4 hạng tử được mô tả ở điều 4.3. Các cột được coi là các đa thức trên trường $GF(2^8)$ và được nhân theo modulo $x^4 + 1$ với một đa thức cố định $a(x)$ sau đây:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (5.5)$$

Như đã mô tả ở điều 4.3, điều này có thể biểu diễn dưới dạng một phép nhân ma trận.

Giả sử $s'(x) = a(x) \otimes s(x)$:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s_{3,c} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{với } 0 \le c < \mathbf{Nb}. \qquad (5.6)$$

Ở kết quả của phép nhân này, bốn byte trong một cột được thay thế như sau:

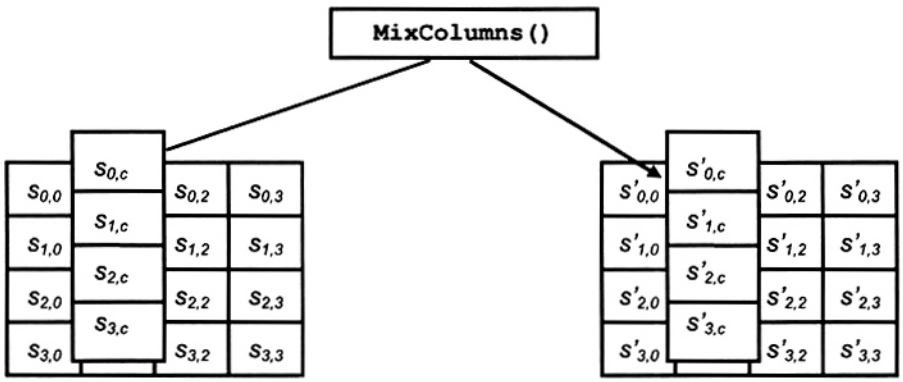
$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$

$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$

$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$

$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$.

Hình 9 minh họa Phép biến đổi **`MixColumns()`**.



Hình 9 - Phép `MixColumns()` thao tác trên Trạng thái theo cách cột-cột

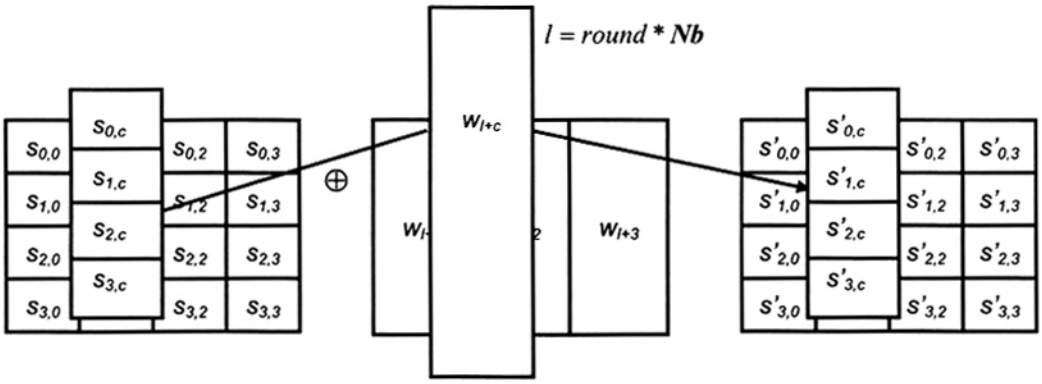
6.1.4 Phép biến đổi `AddRoundKey()`

Trong Phép biến đổi **`AddRoundKey()`**, một Khóa vòng được cộng với Trạng thái bằng một phép toán XOR đơn giản trên bit. Mỗi Khóa vòng bao gồm ***Nb*** từ nhận được từ lược đồ khóa (mô tả ở điều 5.2). ***Nb*** từ này được cộng với các cột của Trạng thái sao cho:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}] \text{ với } 0 \le c < \mathbf{Nb} \qquad (5.7)$$

Trong đó, $[w_i]$ là các từ của lược đồ khóa mô tả ở điều 5.2, *round* là một giá trị nằm trong $0 \le$ *round* $\le$ ***Nr***. Đối với Phép mã hóa, Phép cộng Khóa vòng ban đầu xảy ra khi *round* = 0, trước khi áp dụng lần đầu hàm vòng (xem Hình 5). Việc áp dụng Phép biến đổi **`AddRoundKey()`** theo ***Nr*** vòng của Phép mã hóa xảy ra khi $1 \le$ *round* $\le$ ***Nr***.

Các thao tác của Phép biến đổi này được minh họa ở Hình 10, trong đó *l* = *round* * ***Nb***. Địa chỉ byte trong các từ của lược đồ khóa được mô tả ở điều 3.1.



Hình 10 - Phép `AddRoundKey()` thực hiện XOR mỗi cột của Trạng thái với một từ của lược đồ khóa

6.2 Mở rộng khóa

Thuật toán AES nhận vào một Khóa mã **K** và thực hiện phép Mở rộng khóa để tạo ra một lược đồ khóa. Phép Mở rộng khóa tạo ra tổng số ***Nb*** (***Nr*** + 1) từ: thuật toán yêu cầu một tập khởi tạo gồm ***Nb*** từ và mỗi trong số ***Nr*** vòng đòi hỏi ***Nb*** từ làm dữ liệu khóa đầu vào. Lược đồ khóa cuối cùng là một mảng tuyến tính các từ 4 byte ký hiệu là $[w_i]$, với i nằm trong khoảng $0 \le i < Nb(Nr + 1)$.

Quá trình mở rộng khóa đầu vào thành lược đồ khóa được biểu diễn ở dạng tựa mã như Hình 11.

`SubWord()` là một hàm nhận một từ 4 byte làm đầu vào và áp dụng Hộp-S (Phần 5.1.1 Hình 7) đối với mỗi byte để tạo thành đầu ra là một từ. Hàm `RotWord()` nhận một từ $[a_0,a_1,a_2,a_3]$ làm đầu vào, thực hiện phép hoán vị tuần hoàn và cho kết quả đầu ra là một từ $[a_1,a_2,a_3,a_0]$. Mảng từ hằng số vòng `Rcon[i]` chứa các giá trị được cho bởi $[x^{i-1},\{00\},\{00\},\{00\}]$, với x^{i-1} là lũy thừa của x (x được ký hiệu là $\{02\}$) trong trường $GF(2^8)$ như đã mô tả ở điều 4.2 (chú ý i bắt đầu từ 1 chứ không phải từ 0).

Hình 11 cho thấy rằng ***Nk*** từ đầu tiên của khóa mở rộng được lấp đầy bởi Khóa mã. Mỗi từ tiếp sau đó (`w[i]`) đều tương đương với phép XOR của từ trước đó (`w[i-1]`) với từ trước đó ***Nk*** vị trí (`w[i-Nk]`). Đối với các từ ở các vị trí là bội số của ***Nk*** thì một phép biến đổi được áp dụng cho `w[i-1]` trước khi được XOR, tiếp đó một phép XOR với hằng của vòng `Rcon[i]` được áp dụng. Phép biến đổi này bao gồm một phép dịch vòng của các byte trong một từ (`RotWord()`), tiếp đến áp dụng một phép tra bảng cho cả bốn byte của từ (`SubWord()`).

Một chú ý quan trọng là phép Mở rộng khóa cho các Khóa mã 256 bít (***Nk*** = 8) có sự khác biệt không đáng kể so với phép Mở rộng khóa cho các Khóa mã 128 và 192 bít. Nếu ***Nk*** = 8 và `i-4` là bội số của ***Nk*** thì hàm `SubWord()` được áp dụng cho `w[i-1]` trước khi áp dụng XOR.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   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)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Chú ý: Không phải tất cả giá trị *Nk* = 4, 6 và 8 đều được thực thi, chúng được đưa vào câu lệnh điều kiện ở trên cho gọn. Các yêu cầu thực thi cụ thể đối với Khóa mã được trình bày ở Phần 6.1.

Hình 11 - Tựa mã cho phép Mở rộng khóa[2]

Phụ lục A sẽ trình bày các ví dụ về phép Mở rộng khóa.

6.3 Phép giải mã

Các phép biến đổi trong Phép mã hóa ở điều 5.1 có thể được đảo ngược và sau đó thực thi theo chiều ngược lại nhằm tạo ra Phép giải mã trực tiếp của thuật toán AES. Các phép biến đổi sử dụng trong Phép giải mã là **`InvShiftRows()`**, **`InvSubBytes()`**, **`InvMixColumns()`** và **`AddRoundKey()`**, chúng sẽ thao tác với Trạng thái và được mô tả ở các mục tiếp sau đây.

Phép giải mã được mô tả ở dạng tựa mã như Hình 12. Trong hình này, mảng **`w[]`** chứa lược đồ khóa như đã mô tả ở điều 5.2 trước đây.

2 Các hàm (SubWord() và RotWord()) trả về kết quả là một biến đổi của hàm đầu vào, trong khi các phép biến đổi trong Phép mã hóa và Phép giải mã (ShiftRow(), SubBytes(),...) lại iến đổi mảng Trạng thái được tham chiếu bởi con trỏ "state".

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]
   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])                    // Xem phần 5.1.4
   for round = Nr-1 step -1 downto 1
      InvShiftRows(state)                                        // Xem phần  5.3.1
      InvSubBytes(state)                                         // Xem phần  5.3.2
      AddRoundKey(state,w[round*Nb,(round+1)*Nb-1])
      InvMixColumns(state)                                       //Xem phần5.3.3
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

Hình 12 - Tựa mã về Phép giải mã[3]

6.3.1 Phép biến đổi `InvShiftRows()`

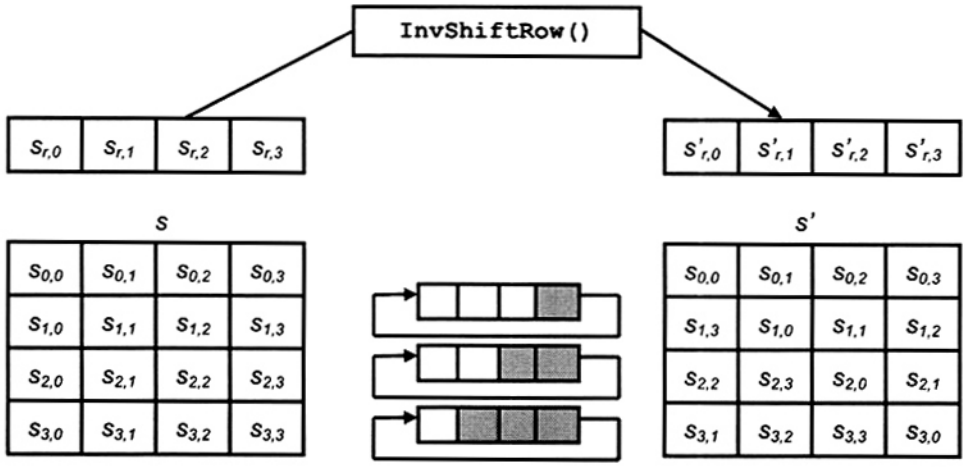
Phép biến đổi **`InvShiftRows()`** là phép biến đổi ngược của **`ShiftRows()`**. Các byte trong ba từ cuối của Trạng thái được dịch vòng theo số byte khác nhau (offset). Ở hàng đầu tiên (r =0) không thực hiện phép chuyển dịch. Ba hàng dưới cùng được dịch vòng ***Nb*** – *shift*(r, *Nb*) byte, trong đó *shift(r,Nb)* phụ thuộc vào số dòng nhận được từ hệ thức (5.4) (xem điều 6.1.2).

Cụ thể, phép biến đổi **`InvShiftRows()`** được thực hiện như sau:

$$s_{r,(c+shift(r,Nb)) \bmod Nb} = s'_{r,c} \text{ với } 0 < r < 4 \text{ và } 0 \le c < \mathbf{Nb} \qquad (6.8)$$

Hình 13 minh họa phép biến đổi **`InvShiftRows()`**.

3 Một số phép biến đổi (vd: InvSubBytes(), InvShiftRows(),...) thao tác trên mảng Trạng thái được tham chiếu bởi con trỏ "state". Phép biến đổi AddRoundKey() sử dụng một con trỏ khác để tham chiếu đến Khóa vòng.



Hình 13 - Hàm `InvShiftRows()` chuyển dịch vòng ba hàng cuối của mã Trạng thái

6.3.2 Phép biến đổi `InvSubBytes()`

Phép biến đổi **`InvSubBytes()`** là nghịch đảo của phép thay thế theo byte **`SubBytes()`**, trong đó sử dụng một Hộp-S nghịch áp dụng cho mỗi byte của Trạng thái. Điều này đạt được bằng cách áp dụng phép ngược của phép biến đổi affine (5.1) sau khi thực hiện phép nghịch đảo trên trường GF(2^8) cho bởi bảng dưới.

Hộp-S nghịch sử dụng trong phép biến đổi **`InvSubBytes()`** được trình bày ở Hình 14.

x \ y	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

Hình 14 - Hộp-S nghịch: thay thế các giá trị theo byte {xy} (dạng thập lục phân)

6.3.3 Phép biến đổi `InvMixColumns()`

Phép biến đổi **`InvMixColumns()`** là phép biến đổi ngược của **`MixColumns()`**. **`InvMixColumns()`** thao tác theo từng cột của Trạng thái, xem mỗi cột như một đa thức bốn hạng tử đã nói ở điều 4.3. Các cột được coi như các đa thức trên trường GF(2^8) và được nhân theo

modulo $x^4 + 1$ với đa thức nghịch đảo của $a(x)$ là $a^{-1}(x)$, tức là:

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (5.9)$$

Như đã mô tả ở điều 4.3, nó có thể được biểu diễn như một phép nhân ma trận $s'(x) = a^{-1}(x) \otimes s(x)$ như sau:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{với } 0 \le c < \mathbf{Nb}. \quad (5.10)$$

ở kết quả của phép nhân, bốn byte trong một cột được thay thế bởi:

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

6.3.4 Phép biến đổi nghịch của `AddRoundKey()`

Phép biến đổi **`AddRoundKey()`** đã được đề cập ở điều 5.1.4 là một phép biến đổi thuận nghịch vì nó chỉ áp dụng một phép toán XOR nên nó được thực hiện như nhau ở Phép mã hóa và Phép giải mã.

6.3.5 Phép giải mã tương đương

Trong Phép giải mã trình bày ở điều 5.3 và Hình 12, thứ tự của các phép biến đổi khác với thứ tự của các phép biến đổi của Phép mã hóa, trong khi đó khuôn dạng lược đồ khóa cho quá trình mã hóa và giải mã không thay đổi. Tuy nhiên, một số tính chất của thuật toán AES cho phép thực hiện một Phép giải mã tương đương có cùng thứ tự các phép biến đổi như trong Phép mã hóa (các biến đổi được thay bằng các phép biến đổi ngược). Có thể thực hiện điều này là nhờ một thay đổi trong lược đồ khóa.

Hai tính chất cấu thành một Phép giải mã tương đương là:

Tính giao hoán giữa hai phép biến đổi **`SubBytes()`** và **`ShiftRows()`**. Tính chất này nói lên rằng việc thực hiện phép biến đổi **`SubBytes()`** trước rồi thực hiện **`ShiftRows()`** cũng tương đương với việc thực hiện **`ShiftRows()`** trước rồi thực hiện **`SubBytes()`**. Điều này cũng đúng với phép nghịch đảo của chúng **`InvSubBytes()`** và **`InvShiftRows`**.

Các phép toán trộn cột **`MixColumns()`** và **`InvMixColumns()`** là tuyến tính đối với đầu vào cột, nghĩa là:

```
InvMixColumns(state XOR Round Key) =
                    InvMixColumns(state) XOR InvMixColumns(Round Key).
```

Các tính chất này cho phép đảo ngược thứ tự của các phép biến đổi **InvSubBytes()** và **InvShiftRows()**. Thứ tự của các phép biến đổi **AddRoundKey()** và **InvMixColumns()** cũng có thể đảo ngược với điều kiện đảm bảo rằng các cột (từ) của lược đồ khóa giải mã được sửa đổi bằng cách sử dụng phép biến đổi **InvMixColumns()**.

Phép giải mã tương đương được xác định bằng cách đảo thứ tự của phép biến đổi **InvSubBytes()** và **InvShiftRows()** cho nhau như Hình 12 và đảo thứ tự của phép biến đổi **AddRoundKey()** và **InvMixColumns()** được sử dụng trong "vòng lặp" **for *round* =1 to *Nr*-1** sau khi sửa đổi lần đầu lược đồ khóa mã sử dụng phép biến đổi **InvMixColumns()**. Theo phương pháp này, ***Nb*** từ đầu tiên và cuối cùng trong lược đồ khóa mã sẽ không bị sửa đổi.

Dựa trên các thay đổi này, Phép giải mã tương đương nhận được đưa ra một cấu trúc hiệu quả hơn Phép giải mã đã mô tả ở điều 5.3 và Hình 12. Tựa mã của Phép giải mã tương đương cho bởi Hình 15 (mảng từ **dw[]** chứa lược đồ khóa mã đã sửa đổi. Việc sửa đổi phép Mở rộng khóa cũng được cho bởi Hình 15).

```
EqInvCipher(byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, dw[Nr*Nb, (Nr+1)*Nb-1])

   for round = Nr-1 step -1 downto 1
      InvSubBytes(state)
      InvShiftRows(state)
      InvMixColumns(state)
      AddRoundKey(state, dw[round*Nb, (round+1)*Nb-1])
   end for

   InvSubBytes(state)
   InvShiftRows(state)
   AddRoundKey(state, dw[0, Nb-1])

   out = state end

/* Đối với Phép giải mã tương đương, đoạn giả mã sau được thêm vào phần cuối của
phép Mở rộng khóa (điều 5.2) */
   for i = 0 step 1 to (Nr+1)*Nb-1
      dw[i] = w[i]
   end for

   for round = 1 step 1 to Nr-1
      InvMixColumns(dw[round*Nb, (round+1)*Nb-1])
      //chú ý sự thay đổi kiểu
   end for

Chú ý rằng, vì InvMixColumns thao tác trên một mảng byte hai chiều trong khi các
Khóa vòng lại được giữ trong mảng các từ, lời gọi InvMixC olumns() ở đoạn mã này
bao gồm cả việc thay đổi kiểu (tức là, đầu vào của InvMixColumns thường là mảng
Trạng thái, nó được coi như một mảng byte hai chiều trong khi đầu vào ở đây lại là
Khóa vòng được tính như một mảng của các từ một chiều).
```

Hình 15 - Tựa mã của Phép giải mã tương đương

7 Các vấn đề thực thi

7.1 Yêu cầu về độ dài khóa

Mỗi thực thi cụ thể của thuật toán AES sẽ hỗ trợ ít nhất một trong ba độ dài khóa chỉ ra trong điều 5: 128, 192 hoặc 256 bít (như vậy, ***Nk*** = 4, 6 hoặc 8). Các cài đặt có thể tùy chọn hỗ trợ hai hoặc ba độ dài khóa, chúng có thể tăng thêm tính tương tác cho các cài đặt thuật toán.

7.2 Các hạn chế về khóa

Chưa phát hiện ra khóa yếu hoặc khóa bán yếu đối với thuật toán AES và không có hạn chế nào đối với việc lựa chọn khóa.

7.3 Tham số hóa Độ dài khóa, Kích cỡ khối và Số vòng lặp

Tiêu chuẩn này định ra cụ thể các giá trị được phép dùng cho chiều dài khóa (***Nk***), kích cỡ khối (***Nb***) và số lượng vòng lặp (***Nr***) như Hình 4. Tuy nhiên, các xác nhận lại trong tương lai của tiêu chuẩn này có thể bao gồm những thay đổi hoặc bổ sung đối với các giá trị cho phép của tham số này. Bởi vậy, người cài đặt có thể lựa chọn để thiết kế cài đặt AES của họ tính đến sự mềm dẻo trong tương lai. Vì thế, trong quá trình thực thi thuật toán AES nên lựa chọn cách thiết kế có tính linh hoạt mong muốn.

7.4 Các đề xuất thực thi đối với các nền khác nhau

Trong nhiều trường hợp, rất có khả năng là sự khác nhau trong thực thi thuật toán ảnh hưởng đến hiệu năng hoặc các ưu điểm khác của thuật toán. Cho trước cùng một khóa và dữ liệu đầu vào (bản rõ hoặc bản mã), một cài đặt nào đó sinh ra cùng một đầu ra (bản mã hoặc rõ) như thuật toán đã mô tả trong chuẩn này đều được coi là một thực thi chấp nhận được của AES.

Các đề xuất về việc làm thế nào để thực thi hiệu quả thuật toán AES trên các nền khác nhau được trình bày trong tài liệu tham khảo [3] và một số bài báo trong tài liệu tham khảo [1].

7.5 Một số chỉ dẫn để thực thi thuật toán

7.5.1 Các chế độ hoạt động của AES

Khi cài đặt thuật toán mã AES người ta thường không sử dụng ở dạng nguyên gốc. AES thường hoạt động ở bốn chế độ cơ bản của mã khối n-bit (ECB, CBC, CFB và OFB) được quy định bởi tiêu chuẩn ISO/IEC 10116 :1997, *Information technology – Security techniques – Modes of operation for an n-bit cipher(Công nghệ thông tin - Kỹ thuật mật mã – Các thuật toán mã hoá – Phần 3 : Các khối mật mã)*. Trên cơ sở bốn chế độ cơ bản ban đầu này người ta đã phát triển thêm một số chế độ khác (Có thể trong tương lai ISO/IEC sẽ công bố thêm một số chế độ hoạt động khác nữa cho mã khối. Tuy nhiên, hiện tại thì ISO/IEC vẫn mới quy định bốn chế độ cơ bản cho mã khối nói trên). Sau đây là những nét sơ lược của bốn chế độ này.

Chế độ sách mã điện tử ECB (Electronic Code Book): Trong chế độ ECB các khối rõ được mã hoá độc lập

nhau và khối mã được giải mã độc lập: $C_i = E_k(M_i)$; $M_i = D_k(C_i)$, trong đó E_k và D_k là các phép mã hoá và giải mã theo khoá mật K.

Chế độ xích liên kết khối mã CBC (Cipher block Chaining): Trong chế độ này, đầu tiên người ta tạo ra một xâu nhị phân 64 bít được gọi là véc-tơ khởi đầu và thông báo cho nhau. Trong bước đầu tiên khối dữ liệu rõ M_1 được cộng với véc-tơ khởi đầu theo phép cộng bít, kết quả nhận được sẽ được biến đổi qua các phép mã hóa để được đầu ra là khối mã C_1. Ở các bước sau, mỗi khối M_i của bản rõ được cộng theo modulo 2 với bản mã trước đó C_{i-1} và được mã hoá:

$C_i = E_k(M_i \oplus C_{i-1})$

$M_i = D_k(C_i) \oplus C_{i-1}$

Chế độ mã liên kết ngược CFB (Cipher Feed Back): Chế độ này này khác với chế độ CBC, tại bước đầu tiên véc-tơ khởi đầu được mã hóa bằng E_k rồi cộng theo modulo 2 với khối rõ. Kết quả thu được lại làm véc-tơ khởi đầu cho bước tiếp theo, rồi lại thực hiện tương tự chế độ CBC:

$C_i = M_i \oplus E_k(C_{i-1})$

$M_i = C_i \oplus D_k(C_{i-1})$

Chế độ đầu ra liên kết ngược OFB (Output Feedback): Thực chất của chế độ OFB là tạo ra khóa dòng rồi cộng theo modulo 2 với bản rõ. Khóa dòng được tạo như sau: Đầu tiên lấy véc-tơ khởi đầu s_0 rồi mã hóa qua phép mã khối E_k được s_1. Tiếp đó, s_1 lại được mã hóa qua E_k để được s_2,..và cứ thế thực hiện cho đến khi tạo được khóa dòng có độ dài bằng dữ liệu cần mã.

Mỗi chế độ sử dụng mã khối trên đây đều có ưu điểm và nhược điểm riêng. Tùy từng trường hợp cụ thể mà người ta lựa chọn một chế độ sử dụng phù hợp đáp ứng yêu cầu bảo mật đặt ra.

7.5.2 Thực thi thuật toán AES bằng phần mềm

Thuật toán AES cho phép thực thi hiệu quả bằng cả phần mềm và phần cứng. Thông thường, với những ứng dụng không yêu cầu hiệu năng và tốc độ cao thì thuật toán thường được thực thi ở dạng phần mềm. Ngược lại, có những thiết bị phần cứng chuyên dụng cho phép thực thi thuật toán AES với tốt độ cao và khả năng vận hành bền vững.

Nếu xét về ngôn ngữ lập trình thì hiện có khá nhiều mã nguồn thực thi thuật toán AES được viết bằng nhiều ngôn ngữ lập trình (Assembler, C/C++, Visual Basic, Java, C#,…) có thể vận hành trên nhiều nền hệ điều hành (Windows, Linux/Unix, Solaris,…) chạy các vi xử lý dòng Intel, AMD,... Việc lựa chọn ngôn ngữ lập trình nào để thể hiện thuật toán AES tùy thuộc vào ý thích và khả năng của người lập trình cũng như mục đích, nền tảng phần cứng cho phép. Chẳng hạn, nếu với mục đích cứng hóa thuật toán thì người lập trình thường chọn các dạng ngôn ngữ ở thấp như hợp ngữ Assembler. Nếu hướng tới việc xây dựng các thư viện mật mã để thực thi trong nhân hệ điều hành hoặc sử dụng bởi ứng dụng khác thì người lập trình lại thường chọn ngôn ngữ C/C++. Nếu thực thi thuật toán trong các ứng dụng đồ họa thì người lập trình lại thường chọn các

ngôn ngữ bậc cao như Visual Basic, Visual C++, Java, C#,….Nếu viết để thực thi thuật toán AES trên nền web thì người lập trình thường chọn các ngôn ngữ hướng web như Java, VBScript, JavaScript, C#, …Hiện nay, trên Internet có rất nhiều mã nguồn thực thi thuật toán AES do nhiều tác giả viết bằng nhiều dạng ngôn ngữ lập trình khác nhau. Tuy nhiên, chúng tôi khuyến cáo chỉ nên tham khảo một số mã nguồn thuật toán được cung cấp bởi các tác giả đáng tin cậy. Nên tham khảo web-site hỗ trợ nhiều thông tin về AES trên các web-site:

- http://csrc.nist.gov/archive/aes/.
- http://www.iaik.tu – graz.ac.at/research/krypto/aes/old/~rijment/rijndael/.
- http://fp.gladman.plus.com/index.htm
- http://www.openssl.org
- http://tcmm.bcy.gov.vn

7.5.3 Thực thi thuật toán AES bằng phần cứng

Các thiết bị phần cứng thực thi thuật toán AES được chia làm hai dòng. Dòng thiết bị thứ nhất dựa vào một hệ vi xử lý kết phụ với hệ vi xử lý chính của máy tính (co-processor). Thông thường thiết bị thuộc dòng này được thiết kế ở dạng card chuyên dụng kết nối qua giao diện ghép nối với máy tính (ví dụ qua PCI). Dòng thiết bị thứ hai thường được thiết kế ở dạng thẻ thông minh (Smart Card) hoặc các thiết bị cắm qua cổng USB (USB Devices). Các thiết bị thuộc dòng thứ hai thường có một hệ vi xử lý (CPU) và bộ nhớ (ROM/RAM) riêng để thực thi thuật toán độc lập so với máy tính.

Công nghệ cứng hóa thuật toán hiện cũng tồn tại hai dòng. Dòng công nghệ thứ nhất sử dụng một kỹ thuật chuyên dụng chẳng hạn như **ASIC** (Application Specific Integrated Circuit). Công nghệ ASIC cho phép thực thi thuật toán rất nhanh và hiệu quả với năng lượng tiêu tốn rất ít. Tuy nhiên, công nghệ này có một số hạn chế như không cho phép sửa đổi thuật toán sau khi đã tạo thành thiết bị. Dòng công nghệ thứ hai sử dụng một bộ mạch tích hợp chứa hệ vi xử lý cho phép lập trình bằng phần mềm. Với công nghệ này, thuật toán có thể được cấu hình lại theo ý đồ của người lập trình nhưng vẫn cho phép thực thi thuật toán với tốc độ và hiệu quả cao. Điển hình cho dòng công nghệ này là các thiết bị sử dụng công nghệ **FPGA** (Field Programmable Gate Arrays). Hiện nay, công nghệ FPGA được coi là công nghệ có nhiều ưu điểm và được sử dụng trong hầu hết các thiết bị phần cứng thực thi thuật toán AES. Có thể tham khảo trang http://www.iaik.tu-graz.ac.at/research/krypto/AES/ để có thông tin đầy đủ hơn về các vấn đề thực thi thuật toán AES bằng phần cứng.

Phụ lục A
(tham khảo)

Các ví dụ về mở rộng khóa

Phụ lục này trình bày việc thiết lập lược đồ khóa cho các kích cỡ khóa khác nhau. Chú ý là các giá trị chứa nhiều byte được trình bày theo cách viết mô tả ở điều 3. Các giá trị trung gian tạo ra trong quá trình phát triển lược đồ khóa (xem điều 5.2) cho bởi bảng sau (tất cả giá trị đều ở dạng thập lục phân, ngoại trừ cột số thứ tự (cột ký hiệu là *i*)).

A.1 Ví dụ về mở rộng một Khóa mã 128 bít

Cho một khóa mã dài 128 bít sau đây:

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

Với ***Nk*** = 4, các từ được xác định như sau:

w_0 = **2b7e1516** w_1 = **28aed2a6** w_2 = **abf71588** w_3 = **09cf4f3c**

Quá trình mở rộng Khóa được cho bởi bảng sau:

i	temp	Sau phép RotWord()	Sau phép SubWord()	Rcon[i/Nk]	Sau phép XOR với Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafe17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d
13	3d80477d					7a96b943	4716fe3e
14	4716fe3e					5935807a	1e237e44
15	1e237e44					7359f67f	6d7a883b
16	6d7a883b	7a883b6d	dac4e23c	08000000	d2c4e23c	3d80477d	ef44a541
17	ef44a541					4716fe3e	a8525b7f
18	a8525b7f					1e237e44	b671253b
19	b671253b					6d7a883b	db0bad00

20	db0bad00	0bad00db	2b9563b9	10000000	3b9563b9	ef44a541	d4d1c6f8
21	d4d1c6f8					a8525b7f	7c839d87
22	7c839d87					b671253b	caf2b8bc
23	caf2b8bc					db0bad00	11f915bc
24	11f915bc	f915bc11	99596582	20000000	b9596582	d4d1c6f8	6d88a37a
25	6d88a37a					7c839d87	110b3efd
26	110b3efd					caf2b8bc	dbf98641
27	dbf98641					11f915bc	ca0093fd
28	ca0093fd	0093fdca	63dc5474	40000000	23dc5474	6d88a37a	4e54f70e
29	4e54f70e					110b3efd	5f5fc9f3
30	5f5fc9f3					dbf98641	84a64fb2
31	84a64fb2					ca0093fd	4ea6dc4f
32	4ea6dc4f	a6dc4f4e	2486842f	80000000	a486842f	4e54f70e	ead27321
33	ead27321					5f5fc9f3	b58dbad2
34	b58dbad2					84a64fb2	312bf560
35	312bf560					4ea6dc4f	7f8d292f
36	7f8d292f	8d292f7f	5da515d2	1b000000	46a515d2	ead27321	ac7766f3
37	ac7766f3					b58dbad2	19fadc21
38	19fadc21					312bf560	28d12941
39	28d12941					7f8d292f	575c006e
40	575c006e	5c006e57	4a639f5b	36000000	7c639f5b	ac7766f3	d014f9a8
41	d014f9a8					19fadc21	c9ee2589
42	c9ee2589					28d12941	e13f0cc8
43	e13f0cc8					575c006e	b6630ca6

A.2 Ví dụ về mở rộng một Khóa mã 192 bít

Cho một khóa mã dài 192 bít sau đây:

```
Cipher Key =      8e 73 b0 f7 da 0e 64 52 c8 10 f3 2b
                  80 90 79 e5 62 f8 ea d2 52 2c 6b 7b
```

Với ***Nk*** = 6, các từ được xác định như sau:

w_0 = 8e73b0f7 w_1 = da0e6452 w_2 = c810f32b w_3 = 809079e5

w_4 = 62f8ead2 w_5 = 522c6b7b

Quá trình mở rộng Khóa được cho bởi bảng sau:

i	temp	Sau phép RotWord()	Sau phép SubWord()	Rcon[i/Nk]	Sau phépXOR với Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
6	522c6b7b	2c6b7b52	717f2100	01000000	707f2100	8e73b0f7	fe0c91f7
7	fe0c91f7					da0e6452	2402f5a5
8	2402f5a5					c810f32b	ec12068e
9	ec12068e					809079e5	6c827f6b
10	6c827f6b					62f8ead2	0e7a95b9
11	0e7a95b9					522c6b7b	5c56fec2
12	5c56fec2	56fec25c	b1bb254a	02000000	b3bb254a	fe0c91f7	4db7b4bd
13	4db7b4bd					2402f5a5	69b54118
14	69b54118					ec12068e	85a74796
15	85a74796					6c827f6b	e92538fd
16	e92538fd					0e7a95b9	e75fad44
17	e75fad44					5c56fec2	bb095386
18	bb095386	095386bb	01ed44ea	04000000	05ed44ea	4db7b4bd	485af057
19	485af057					69b54118	21efb14f
20	21efb14f					85a74796	a448f6d9
21	a448f6d9					e92538fd	4d6dce24
22	4d6dce24					e75fad44	aa326360
23	aa326360					bb095386	113b30e6
24	113b30e6	3b30e611	e2048e82	08000000	ea048e82	485af057	a25e7ed5
25	a25e7ed5					21efb14f	83b1cf9a
26	83b1cf9a					a448f6d9	27f93943
27	27f93943					4d6dce24	6a94f767
28	6a94f767					aa326360	c0a69407
29	c0a69407					113b30e6	d19da4e1
30	d19da4e1	9da4e1d1	5e49f83e	10000000	4e49f83e	a25e7ed5	ec1786eb
31	ec1786eb					83b1cf9a	6fa64971
32	6fa64971					27f93943	485f7032
33	485f7032					6a94f767	22cb8755
34	22cb8755					c0a69407	e26d1352
35	e26d1352					d19da4e1	33f0b7b3
36	33f0b7b3	f0b7b333	8ca96dc3	20000000	aca96dc3	ec1786eb	40beeb28
37	40beeb28					6fa64971	2f18a259
38	2f18a259					485f7032	6747d26b
39	6747d26b					22cb8755	458c553e
40	458c553e					e26d1352	a7e1466c
41	a7e1466c					33f0b7b3	9411f1df

42	9411f1df	11f1df94	82a19e22	40000000	c2a19e22	40beeb28	821f750a
43	821f750a					2f18a259	ad07d753
44	ad07d753					6747d26b	ca400538
45	ca400538					458c553e	8fcc5006
46	8fcc5006					a7e1466c	282d166a
47	282d166a					9411f1df	bc3ce7b5
48	bc3ce7b5	3ce7b5bc	eb94d565	80000000	6b94d565	821f750a	e98ba06f
49	e98ba06f					ad07d753	448c773c
50	448c773c					ca400538	8ecc7204
51	8ecc7204					8fcc5006	01002202

A.3 Ví dụ về mở rộng một Khóa mã 256 bít

Cho một khóa mã dài 256 bít sau đây:

```
Cipher Key =      60 3d eb 10 15 ca 71 be 2b 73 ae f0 85 7d 77 81
                  1f 35 2c 07 3b 61 08 d7 2d 98 10 a3 09 14 df f4
```

Với ***Nk*** = 8, các từ được xác định như sau:

w_0 = 603deb10 w_1 = 15ca71be w_2 = 2b73aef0 w_3 = 857d7781

w_4 = 1f352c07 w_5 = 3b6108d7 w_6 = 2d9810a3 w_7 = 0914dff4

Quá trình mở rộng Khóa được cho bởi bảng sau:

i	temp	Sau phép RotWord()	Sau phép SubWord()	Rcon[i/Nk]	Sau phép XOR với Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
8	0914dff4	14dff409	fa9ebf01	01000000	fb9ebf01	603deb10	9ba35411
9	9ba35411					15ca71be	8e6925af
10	8e6925af					2b73aef0	a51a8b5f
11	a51a8b5f					857d7781	2067fcde
12	2067fcde		b785b01d			1f352c07	a8b09c1a
13	a8b09c1a					3b6108d7	93d194cd
14	93d194cd					2d9810a3	be49846e
15	be49846e					0914dff4	b75d5b9a
16	b75d5b9a	5d5b9ab7	4c39b8a9	02000000	4e39b8a9	9ba35411	d59aecb8
17	d59aecb8					8e6925af	5bf3c917
18	5bf3c917					a51a8b5f	fee94248
19	fee94248					2067fcde	de8ebe96
20	de8ebe96		1d19ae90			a8b09c1a	b5a9328a
21	b5a9328a					93d194cd	2678a647
22	2678a647					be49846e	98312229

23	98312229					b75d5b9a	2f6c79b3
24	2f6c79b3	6c79b32f	50b66d15	04000000	54b66d15	d59aecb8	812c81ad
25	812c81ad					5bf3c917	dadf48ba
26	dadf48ba					fee94248	24360af2
27	24360af2					de8ebe96	fab8b464
28	fab8b464		2d6c8d43			b5a9328a	98c5bfc9
29	98c5bfc9					2678a647	bebd198e
30	bebd198e					98312229	268c3ba7
31	268c3ba7					2f6c79b3	09e04214
32	09e04214	e0421409	e12cfa01	08000000	e92cfa01	812c81ad	68007bac
33	68007bac					dadf48ba	b2df3316
34	b2df3316					24360af2	96e939e4
35	96e939e4					fab8b464	6c518d80
36	6c518d80		50d15dcd			98c5bfc9	c814e204
37	c814e204					bebd198e	76a9fb8a
38	76a9fb8a					268c3ba7	5025c02d
39	5025c02d					09e04214	59c58239
40	59c58239	c5823959	a61312cb	10000000	b61312cb	68007bac	de136967
41	de136967					b2df3316	6ccc5a71
42	6ccc5a71					96e939e4	fa256395
43	fa256395					6c518d80	9674ee15
44	9674ee15		90922859			c814e204	5886ca5d
45	5886ca5d					76a9fb8a	2e2f31d7
46	2e2f31d7					5025c02d	7e0af1fa
47	7e0af1fa					59c58239	27cf73c3
48	27cf73c3	cf73c327	8a8f2ecc	20000000	aa8f2ecc	de136967	749c47ab
49	749c47ab					6ccc5a71	18501dda
50	18501dda					fa256395	e2757e4f
51	e2757e4f					9674ee15	7401905a
52	7401905a		927c60be			5886ca5d	cafaaae3
53	cafaaae3					2e2f31d7	e4d59b34
54	e4d59b34					7e0af1fa	9adf6ace
55	9adf6ace					27cf73c3	bd10190d
56	bd10190d	10190dbd	cad4d77a	40000000	8ad4d77a	749c47ab	fe4890d1
57	fe4890d1					18501dda	e6188d0b
58	e6188d0b					e2757e4f	046df344
59	046df344					7401905a	706c631e

Phụ lục B
(tham khảo)

Ví dụ về Phép mã hóa

Sơ đồ sau trình bày các giá trị trong mảng Trạng thái của Phép mã hóa với độ dài khối và độ dài khóa mã là 16 byte (tức là ***Nb*** = 4 và ***Nk*** = 4).

Đầu vào:

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

Khóa mã:

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

Các giá trị của Khóa vòng nhận được từ phép Mở rộng khóa đã mô tả ở Phụ lục A. Toàn bộ quá trình mã hóa được mô tả chi tiết như sau:

Vòng	Bắt đầu vòng	Sau phép SubBytes	Sau phép ShiftRows	Sau phép MixColumns	Giá trị khóa vòng	
in	32 88 31 e0 43 5a 31 37 f6 30 98 07 a8 8d a2 34				⊕ 2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c	=
1	19 a0 9a e9 3d f4 c6 f8 e3 e2 8d 48 be 2b 2a 08	d4 e0 b8 1e 27 bf b4 41 11 98 5d 52 ae f1 e5 30	d4 e0 b8 1e bf b4 41 27 5d 52 11 98 30 ae f1 e5	04 e0 48 28 66 cb f8 06 81 19 d3 26 e5 9a 7a 4c	⊕ a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05	=
2	a4 68 6b 02 9c 9f 5b 6a 7f 35 ea 50 f2 2b 43 49	49 45 7f 77 de db 39 02 d2 96 87 53 89 f1 1a 3b	49 45 7f 77 db 39 02 de 87 53 d2 96 3b 89 f1 1a	58 1b db 1b 4d 4b e7 6b ca 5a ca b0 f1 ac a8 e5	⊕ f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f	=
3	aa 61 82 68 8f dd d2 32 5f e3 4a 46 03 ef d2 9a	ac ef 13 45 73 c1 b5 23 cf 11 d6 5a 7b df b5 b8	ac ef 13 45 c1 b5 23 73 d6 5a cf 11 b8 7b df b5	75 20 53 bb ec 0b c0 25 09 63 cf d0 93 33 7c dc	⊕ 3d 47 1e 6d 80 16 23 7a 47 fe 7e 88 7d 3e 44 3b	=
4	48 67 4d d6 6c 1d e3 5f 4e 9d b1 58	52 85 e3 f6 50 a4 11 cf 2f 5e c8 6a	52 85 e3 f6 a4 11 cf 50 c8 6a 2f 5e	0f 60 6f 5e d6 31 c0 b3 da 38 10 13	⊕ ef a8 b6 db 44 52 71 0b a5 5b 25 ad	=

	ee	0d	38	e7	28	d7	07	94	94	28	d7	07	a9	bf	6b	01	41	7f	3b	00
5	e0	c8	d9	85	e1	e8	35	97	e1	e8	35	97	25	bd	b6	4c	d4	7c	ca	11
	92	63	b1	b8	4f	fb	c8	6c	fb	c8	6c	4f	d1	11	3a	4c	d1	83	f2	f9
	7f	63	35	be	d2	fb	96	ae	96	ae	d2	fb	a9	d1	33	c0	c6	9d	b8	15
	e8	c0	50	01	9b	ba	53	7c	7c	9b	ba	53	ad	68	8e	b0	f8	87	bc	bc
6	f1	c1	7c	5d	a1	78	10	4c	a1	78	10	4c	4b	2c	33	37	6d	11	db	ca
	00	92	c8	b5	63	4f	e8	d5	4f	e8	d5	63	86	4a	9d	d2	88	0b	f9	00
	6f	4c	8b	d5	a8	29	3d	03	3d	03	a8	29	8d	89	f4	18	a3	3e	86	93
	55	ef	32	0c	fc	df	23	fe	fe	fc	df	23	6d	80	e8	d8	7a	fd	41	fd
7	26	3d	e8	fd	f7	27	9b	54	f7	27	9b	54	14	46	27	34	4e	5f	84	4e
	0e	41	64	d2	ab	83	43	b5	83	43	b5	ab	15	16	46	2a	54	5f	a6	a6
	2e	b7	72	8b	31	a9	40	3d	40	3d	31	a9	b5	15	56	d8	f7	c9	4f	dc
	17	7d	a9	25	f0	ff	d3	3f	3f	f0	ff	d3	bf	ec	d7	43	0e	f3	b2	4f
8	5a	19	a3	7a	be	d4	0a	da	be	d4	0a	da	00	b1	54	fa	ea	b5	31	7f
	41	49	e0	8c	83	3b	e1	64	3b	e1	64	83	51	c8	76	1b	d2	8d	2b	8d
	42	dc	19	04	2c	86	d4	f2	d4	f2	2c	86	2f	89	6d	99	73	ba	f5	29
	b1	1f	65	0c	c8	c0	4d	fe	fe	c8	c0	4d	d1	ff	cd	ea	21	d2	60	2f
9	ea	04	65	85	87	f2	4d	97	87	f2	4d	97	47	40	a3	4c	ac	19	28	57
	83	45	5d	96	ec	6e	4c	90	6e	4c	90	ec	37	d4	70	9f	77	fa	d1	5c
	5c	33	98	b0	4a	c3	46	e7	46	e7	4a	c3	94	e4	3a	42	66	dc	29	00
	f0	2d	ad	c5	8c	d8	95	a6	a6	8c	d8	95	ed	a5	a6	bc	f3	21	41	6e
10	eb	59	8b	1b	e9	cb	3d	af	e9	cb	3d	af					d0	c9	e1	b6
	40	2e	a1	c3	09	31	32	2e	31	32	2e	09					14	ee	3f	63
	f2	38	13	42	89	07	7d	2c	7d	2c	89	07					f9	25	0c	0c
	1e	84	e7	d2	72	5f	94	b5	b5	72	5f	94					a8	89	c8	a6
out	39	02	dc	19																
	25	dc	11	6a																
	84	09	85	0b																
	1d	fb	97	32																

Phụ lục C
(tham khảo)

Các Véc-tơ ví dụ

Phụ lục này trình bày các véc-tơ ví dụ (chứa các giá trị trung gian) đối với cả ba trường hợp độ dài khóa (***Nk*** =4,6 và 8) cho Phép mã hóa, Phép giải mã và Phép giải mã tương đương như đã mô tả ở điều 5.1, 5.3 và 5.3.5. Có thể xem thêm các tài liệu tham khảo thêm [1] và [5].

Tất cả véc-tơ đều ở dạng thập lục phân, mỗi cặp ký tự mang giá trị của một byte trong đó ký tự bên trái là giá trị của 4 bít cao, còn ký tự bên phải mang giá trị của 4 bít thấp như đã đề cập ở d 3.2. Chỉ số mảng của tất cả các byte (gồm hai số thập lục phân) trong véc-tơ kiểm tra này bắt đầu từ giá trị không và tăng dần từ trái qua phải.

```
Ký hiệu dùng trong PHÉP MÃ HÓA (Số vòng r = 0 đến 10, 12 hoặc 14):
   input:   đầu vào Phép mã hóa
   start:   trạng thái bắt đầu của round[r]
   s_box:   trạng thái sau tác động của SubBytes()
   s_row:   trạng thái sau tác động của ShiftRows()
   m_col:   trạng thái sau tác động của MixColumns()
   k_sch:   giá trị lược đồ khóa cho round[r]
   output:  đầu ra Phép mã hóa

Ký hiệu dùng trong PHÉP GIẢI MÃ (Số vòng r = 0 đến 10, 12 hoặc 14):
   iinput:  đầu vào của Phép giải mã
   istart:  trạng thái tại điểm bắt đầu của round[r]
   is_box:  trạng thái sau tác động của InvSubBytes()
   is_row:  trạng thái sau tác động của InvShiftRows()
   ik_sch:  giá trị lược đồ khóa cho round[r]
   ik_add:  trạng thái sau tác động của AddRoundKey()
   ioutput: đầu ra Phép giải mã

Ký hiệu dùng trong PHÉP GIẢI MÃ TƯƠNG ĐƯƠNG (GIẢI MÃ) (Số vòng r = 0 đến 10, 12
   hoặc 14):

   iinput:  đầu vào Phép giải mã
   istart:  trạng thái tại điểm bắt đầu của round[r]
   is_box:  trạng thái sau tác động của InvSubBytes()
   is_row:  trạng thái sau tác động của InvShiftRows()
   im_col:  trạng thái sau tác động của InvMixColumns()
   ik_sch:  giá trị lược đồ tạo khóa của round[r]
   ioutput: đầu ra mã nghịch
   round:   biểu diễn thứ tự vòng
```

C.1 AES-128 (*Nk*=4, *Nr*=10)

```
BẢN RÕ:              00112233445566778899aabbccddeeff
KHÓA:                000102030405060708090a0b0c0d0e0f

MÃ THUẬN (MÃ HÓA):
round[ 0].input      00112233445566778899aabbccddeeff
round[ 0].k_sch      000102030405060708090a0b0c0d0e0f
round[ 1].start      00102030405060708090a0b0c0d0e0f0
round[ 1].s_box      63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row      6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col      5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch      d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 2].start      89d810e8855ace682d1843d8cb128fe4
round[ 2].s_box      a761ca9b97be8b45d8ad1a611fc97369
round[ 2].s_row      a7be1a6997ad739bd8c9ca451f618b61
round[ 2].m_col      ff87968431d86a51645151fa773ad009
round[ 2].k_sch      b692cf0b643dbdf1be9bc5006830b3fe
round[ 3].start      4915598f55e5d7a0daca94fa1f0a63f7
round[ 3].s_box      3b59cb73fcd90ee05774222dc067fb68
round[ 3].s_row      3bd92268fc74fb735767cbe0c0590e2d
round[ 3].m_col      4c9c1e66f771f0762c3f868e534df256
round[ 3].k_sch      b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 4].start      fa636a2825b339c940668a3157244d17
round[ 4].s_box      2dfb02343f6d12dd09337ec75b36e3f0
round[ 4].s_row      2d6d7ef03f33e334093602dd5bfb12c7
round[ 4].m_col      6385b79ffc538df997be478e7547d691
round[ 4].k_sch      47f7f7bc95353e03f96c32bcfd058dfd
round[ 5].start      247240236966b3fa6ed2753288425b6c
round[ 5].s_box      36400926f9336d2d9fb59d23c42c3950
round[ 5].s_row      36339d50f9b539269f2c092dc4406d23
round[ 5].m_col      f4bcd45432e554d075f1d6c51dd03b3c
round[ 5].k_sch      3caaa3e8a99f9deb50f3af57adf622aa
round[ 6].start      c81677bc9b7ac93b25027992b0261996
round[ 6].s_box      e847f56514dadde23f77b64fe7f7d490
round[ 6].s_row      e8dab6901477d4653ff7f5e2e747dd4f
round[ 6].m_col      9816ee7400f87f556b2c049c8e5ad036
round[ 6].k_sch      5e390f7df7a69296a7553dc10aa31f6b
round[ 7].start      c62fe109f75eedc3cc79395d84f9cf5d
round[ 7].s_box      b415f8016858552e4bb6124c5f998a4c
round[ 7].s_row      b458124c68b68a014b99f82e5f15554c
round[ 7].m_col      c57e1c159a9bd286f05f4be098c63439
round[ 7].k_sch      14f9701ae35fe28c440adf4d4ea9c026
round[ 8].start      d1876c0f79c4300ab45594add66ff41f
round[ 8].s_box      3e175076b61c04678dfc2295f6a8bfc0
round[ 8].s_row      3e1c22c0b6fcbf768da85067f6170495
```

```
round[ 8].m_col     baa03de7a1f9b56ed5512cba5f414d23
round[ 8].k_sch     47438735a41c65b9e016baf4aebf7ad2
round[ 9].start     fde3bad205e5d0d73547964ef1fe37f1
round[ 9].s_box     5411f4b56bd9700e96a0902fa1bb9aa1
round[ 9].s_row     54d990a16ba09ab596bbf40ea111702f
round[ 9].m_col     e9f74eec023020f61bf2ccf2353c21c7
round[ 9].k_sch     549932d1f08557681093ed9cbe2c974e
round[10].start     bd6e7c3df2b5779e0b61216e8b10b689
round[10].s_box     7a9f102789d5f50b2beffd9f3dca4ea7
round[10].s_row     7ad5fda789ef4e272bca100b3d9ff59f
round[10].k_sch     13111d7fe3944a17f307a78b4d2b30c5
round[10].output    69c4e0d86a7b0430d8cdb78070b4c55a

MÃ NGHỊCH (GIẢI MÃ):
round[ 0].iinput    69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0].ik_sch    13111d7fe3944a17f307a78b4d2b30c5
round[ 1].istart    7ad5fda789ef4e272bca100b3d9ff59f
round[ 1].is_row    7a9f102789d5f50b2beffd9f3dca4ea7
round[ 1].is_box    bd6e7c3df2b5779e0b61216e8b10b689
round[ 1].ik_sch    549932d1f08557681093ed9cbe2c974e
round[ 1].ik_add    e9f74eec023020f61bf2ccf2353c21c7
round[ 2].istart    54d990a16ba09ab596bbf40ea111702f
round[ 2].is_row    5411f4b56bd9700e96a0902fa1bb9aa1
round[ 2].is_box    fde3bad205e5d0d73547964ef1fe37f1
round[ 2].ik_sch    47438735a41c65b9e016baf4aebf7ad2
round[ 2].ik_add    baa03de7a1f9b56ed5512cba5f414d23
round[ 3].istart    3e1c22c0b6fcbf768da85067f6170495
round[ 3].is_row    3e175076b61c04678dfc2295f6a8bfc0
round[ 3].is_box    d1876c0f79c4300ab45594add66ff41f
round[ 3].ik_sch    14f9701ae35fe28c440adf4d4ea9c026
round[ 3].ik_add    c57e1c159a9bd286f05f4be098c63439
round[ 4].istart    b458124c68b68a014b99f82e5f15554c
round[ 4].is_row    b415f8016858552e4bb6124c5f998a4c
round[ 4].is_box    c62fe109f75eedc3cc79395d84f9cf5d
round[ 4].ik_sch    5e390f7df7a69296a7553dc10aa31f6b
round[ 4].ik_add    9816ee7400f87f556b2c049c8e5ad036
round[ 5].istart    e8dab6901477d4653ff7f5e2e747dd4f
round[ 5].is_row    e847f56514dadde23f77b64fe7f7d490
round[ 5].is_box    c81677bc9b7ac93b25027992b0261996
round[ 5].ik_sch    3caaa3e8a99f9deb50f3af57adf622aa
round[ 5].ik_add    f4bcd45432e554d075f1d6c51dd03b3c
round[ 6].istart    36339d50f9b539269f2c092dc4406d23
round[ 6].is_row    36400926f9336d2d9fb59d23c42c3950
round[ 6].is_box    247240236966b3fa6ed2753288425b6c
round[ 6].ik_sch    47f7f7bc95353e03f96c32bcfd058dfd
round[ 6].ik_add    6385b79ffc538df997be478e7547d691
round[ 7].istart    2d6d7ef03f33e334093602dd5bfb12c7
round[ 7].is_row    2dfb02343f6d12dd09337ec75b36e3f0
```

```
round[ 7].is_box    fa636a2825b339c940668a3157244d17
round[ 7].ik_sch    b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 7].ik_add    4c9c1e66f771f0762c3f868e534df256
round[ 8].istart    3bd92268fc74fb735767cbe0c0590e2d
round[ 8].is_row    3b59cb73fcd90ee05774222dc067fb68
round[ 8].is_box    4915598f55e5d7a0daca94fa1f0a63f7
round[ 8].ik_sch    b692cf0b643dbdf1be9bc5006830b3fe
round[ 8].ik_add    ff87968431d86a51645151fa773ad009
round[ 9].istart    a7be1a6997ad739bd8c9ca451f618b61
round[ 9].is_row    a761ca9b97be8b45d8ad1a611fc97369
round[ 9].is_box    89d810e8855ace682d1843d8cb128fe4
round[ 9].ik_sch    d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 9].ik_add    5f72641557f5bc92f7be3b291db9f91a
round[10].istart    6353e08c0960e104cd70b751bacad0e7
round[10].is_row    63cab7040953d051cd60e0e7ba70e18c
round[10].is_box    000102030405060708090a0b0c0d0e0f0
round[10].ik_sch    00010203040506070809a0b0c0d0e0f
round[10].ioutput   00112233445566778899aabbccddeeff

GIẢI MÃ TƯƠNG ĐƯƠNG:
round[ 0].iinput    69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0].ik_sch    13111d7fe3944a17f307a78b4d2b30c5
round[ 1].istart    7ad5fda789ef4e272bca100b3d9ff59f
round[ 1].is_box    bdb52189f261b63d0b107c9e8b6e776e
round[ 1].is_row    bd6e7c3df2b5779e0b61216e8b10b689
round[ 1].im_col    4773b91ff72f354361cb018ea1e6cf2c
round[ 1].ik_sch    13aa29be9c8faff6f770f58000f7bf03
round[ 2].istart    54d990a16ba09ab596bbf40ea111702f
round[ 2].is_box    fde596f1054737d235febad7f1e3d04e
round[ 2].is_row    fde3bad205e5d0d73547964ef1fe37f1
round[ 2].im_col    2d7e86a339d9393ee6570a1101904e16
round[ 2].ik_sch    1362a4638f2586486bff5a76f7874a83
round[ 3].istart    3e1c22c0b6fcbf768da85067f6170495
round[ 3].is_box    d1c4941f7955f40fb46f6c0ad68730ad
round[ 3].is_row    d1876c0f79c4300ab45594add66ff41f
round[ 3].im_col    39daee38f4f1a82aaf432410c36d45b9
round[ 3].ik_sch    8d82fc749c47222be4dadc3e9c7810f5
round[ 4].istart    b458124c68b68a014b99f82e5f15554c
round[ 4].is_box    c65e395df779cf09ccf9e1c3842fed5d
round[ 4].is_row    c62fe109f75eedc3cc79395d84f9cf5d
round[ 4].im_col    9a39bf1d05b20a3a476a0bf79fe51184
round[ 4].ik_sch    72e3098d11c5de5f789dfe1578a2cccb
round[ 5].istart    e8dab6901477d4653ff7f5e2e747dd4f
round[ 5].is_box    c87a79969b0219bc2526773bb016c992
round[ 5].is_row    c81677bc9b7ac93b25027992b0261996
round[ 5].im_col    18f78d779a93eef4f6742967c47f5ffd
round[ 5].ik_sch    2ec410276326d7d26958204a003f32de
round[ 6].istart    36339d50f9b539269f2c092dc4406d23
```

```
round[ 6].is_box    2466756c69d25b236e4240fa8872b332
round[ 6].is_row    247240236966b3fa6ed2753288425b6c
round[ 6].im_col    85cf8bf472d124c10348f545329c0053
round[ 6].ik_sch    a8a2f5044de2c7f50a7ef79869671294
round[ 7].istart    2d6d7ef03f33e334093602dd5bfb12c7
round[ 7].is_box    fab38a1725664d2840246ac957633931
round[ 7].is_row    fa636a2825b339c940668a3157244d17
round[ 7].im_col    fc1fc1f91934c98210fbfb8da340eb21
round[ 7].ik_sch    c7c6e391e54032f1479c306d6319e50c
round[ 8].istart    3bd92268fc74fb735767cbe0c0590e2d
round[ 8].is_box    49e594f755ca638fda0a59a01f15d7fa
round[ 8].is_row    4915598f55e5d7a0daca94fa1f0a63f7
round[ 8].im_col    076518f0b52ba2fb7a15c8d93be45e00
round[ 8].ik_sch    a0db02992286d160a2dc029c2485d561
round[ 9].istart    a7be1a6997ad739bd8c9ca451f618b61
round[ 9].is_box    895a43e485188fe82d121068cbd8ced8
round[ 9].is_row    89d810e8855ace682d1843d8cb128fe4
round[ 9].im_col    ef053f7c8b3d32fd4d2a64ad3c93071a
round[ 9].ik_sch    8c56dff0825dd3f9805ad3fc8659d7fd
round[10].istart    6353e08c0960e104cd70b751bacad0e7
round[10].is_box    0050a0f04090e03080d02070c01060b0
round[10].is_row    00102030405060708090a0b0c0d0e0f0
round[10].ik_sch    000102030405060708090a0b0c0d0e0f
round[10].ioutput   00112233445566778899aabbccddeeff
```

C.2 AES-192 (*Nk*=6, *Nr*=12)

```
BẢN RÕ:      00112233445566778899aabbccddeeff
KHÓA:        000102030405060708090a0b0c0d0e0f1011121314151617

MÃ THUẬN (MÃ HÓA):
round[ 0].input     00112233445566778899aabbccddeeff
round[ 0].k_sch     000102030405060708090a0b0c0d0e0f
round[ 1].start     00102030405060708090a0b0c0d0e0f0
round[ 1].s_box     63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row     6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col     5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch     101112131415161758462f95c43f4fe
round[ 2].start     4f63760643e0aa85aff8c9d041fa0de4
round[ 2].s_box     84fb386f1ae1ac977941dd70832dd769
round[ 2].s_row     84e1dd691a41d76f792d389783fbac70
round[ 2].m_col     9f487f794f955f662afc86abd7f1ab29
round[ 2].k_sch     544afef55847f0fa4856e2e95c43f4fe
round[ 3].start     cb02818c17d2af9c62aa64428bb25fd7
round[ 3].s_box     1f770c64f0b579deaaac432c3d37cf0e
round[ 3].s_row     1fb5430ef0accf64aa370cde3d77792c
round[ 3].m_col     b7a53ecbbf9d75a0c40efc79b674cc11
```

```
round[ 3].k_sch     40f949b31cbabd4d48f043b810b7b342
round[ 4].start     f75c7778a327c8ed8cfebfc1a6c37f53
round[ 4].s_box     684af5bc0acce85564bb0878242ed2ed
round[ 4].s_row     68cc08ed0abbd2bc642ef555244ae878
round[ 4].m_col     7a1e98bdacb6d1141a6944dd06eb2d3e
round[ 4].k_sch     58e151ab04a2a5557effb5416245080c
round[ 5].start     22ffc916a81474416496f19c64ae2532
round[ 5].s_box     9316dd47c2fa92834390a1de43e43f23
round[ 5].s_row     93faa123c2903f4743e4dd83431692de
round[ 5].m_col     aaa755b34cffe57cef6f98e1f01c13e6
round[ 5].k_sch     2ab54bb43a02f8f662e3a95d66410c08
round[ 6].start     80121e0776fd1d8a8d8c31bc965d1fee
round[ 6].s_box     cdc972c53854a47e5d64c765904cc028
round[ 6].s_row     cd54c7283864c0c55d4c727e90c9a465
round[ 6].m_col     921f748fd96e937d622d7725ba8ba50c
round[ 6].k_sch     f501857297448d7ebdf1c6ca87f33e3c
round[ 7].start     671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 7].s_box     8572a1542fe5727b9e86c8df27bc1404
round[ 7].s_row     85e5c8042f8614549ebca17b277272df
round[ 7].m_col     e913e7b18f507d4b227ef652758acbcc
round[ 7].k_sch     e510976183519b6934157c9ea351f1e0
round[ 8].start     0c0370d00c01e622166b8accd6db3a2c
round[ 8].s_box     fe7b5170fe7c8e93477f7e4bf6b98071
round[ 8].s_row     fe7c7e71fe7f807047b95193f67b8e4b
round[ 8].m_col     6cf5edf996eb0a069c4ef21cbfc25762
round[ 8].k_sch     1ea0372a995309167c439e77ff12051e
round[ 9].start     7255dad30fb80310e00d6c6b40d0527c
round[ 9].s_box     40fc5766766c7bcae1d7507f09700010
round[ 9].s_row     406c501076d70066e17057ca09fc7b7f
round[ 9].m_col     7478bcdce8a50b81d4327a9009188262
round[ 9].k_sch     dd7e0e887e2fff68608fc842f9dcc154
round[10].start     a906b254968af4e9b4bdb2d2f0c44336
round[10].s_box     d36f3720907ebf1e8d7a37b58c1c1a05
round[10].s_row     d37e3705907a1a208d1c371e8c6fbfb5
round[10].m_col     0d73cc2d8f6abe8b0cf2dd9bb83d422e
round[10].k_sch     859f5f237a8d5a3dc0c02952beefd63a
round[11].start     88ec930ef5e7e4b6cc32f4c906d29414
round[11].s_box     c4cedcabe694694e4b23bfdd6fb522fa
round[11].s_row     c494bffae62322ab4bb5dc4e6fce69dd
round[11].m_col     71d720933b6d677dc00b8f28238e0fb7
round[11].k_sch     de601e7827bcdf2ca223800fd8aeda32
round[12].start     afb73eeb1cd1b85162280f27fb20d585
round[12].s_box     79a9b2e99c3e6cd1aa3476cc0fb70397
round[12].s_row     793e76979c3403e9aab7b2d10fa96ccc
round[12].k_sch     a4970a331a78dc09c418c271e3a41d5d
round[12].output    dda97ca4864cdfe06eaf70a0ec0d7191
```

MÃ NGHỊCH (GIẢI MÃ):

```
round[ 0].iinput    dda97ca4864cdfe06eaf70a0ec0d7191
round[ 0].ik_sch    a4970a331a78dc09c418c271e3a41d5d
round[ 1].istart    793e76979c3403e9aab7b2d10fa96ccc
round[ 1].is_row    79a9b2e99c3e6cd1aa3476cc0fb70397
round[ 1].is_box    afb73eeb1cd1b85162280f27fb20d585
round[ 1].ik_sch    de601e7827bcdf2ca223800fd8aeda32
round[ 1].ik_add    71d720933b6d677dc00b8f28238e0fb7
round[ 2].istart    c494bffae62322ab4bb5dc4e6fce69dd
round[ 2].is_row    c4cedcabe694694e4b23bfdd6fb522fa
round[ 2].is_box    88ec930ef5e7e4b6cc32f4c906d29414
round[ 2].ik_sch    859f5f237a8d5a3dc0c02952beefd63a
round[ 2].ik_add    0d73cc2d8f6abe8b0cf2dd9bb83d422e
round[ 3].istart    d37e3705907a1a208d1c371e8c6fbfb5
round[ 3].is_row    d36f3720907ebf1e8d7a37b58c1c1a05
round[ 3].is_box    a906b254968af4e9b4bdb2d2f0c44336
round[ 3].ik_sch    dd7e0e887e2fff68608fc842f9dcc154
round[ 3].ik_add    7478bcdce8a50b81d4327a9009188262
round[ 4].istart    406c501076d70066e17057ca09fc7b7f
round[ 4].is_row    40fc5766766c7bcae1d7507f09700010
round[ 4].is_box    7255dad30fb80310e00d6c6b40d0527c
round[ 4].ik_sch    1ea0372a995309167c439e77ff12051e
round[ 4].ik_add    6cf5edf996eb0a069c4ef21cbfc25762
round[ 5].istart    fe7c7e71fe7f807047b95193f67b8e4b
round[ 5].is_row    fe7b5170fe7c8e93477f7e4bf6b98071
round[ 5].is_box    0c0370d00c01e622166b8accd6db3a2c
round[ 5].ik_sch    e510976183519b6934157c9ea351f1e0
round[ 5].ik_add    e913e7b18f507d4b227ef652758acbcc
round[ 6].istart    85e5c8042f8614549ebca17b277272df
round[ 6].is_row    8572a1542fe5727b9e86c8df27bc1404
round[ 6].is_box    671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 6].ik_sch    f501857297448d7ebdf1c6ca87f33e3c
round[ 6].ik_add    921f748fd96e937d622d7725ba8ba50c
round[ 7].istart    cd54c7283864c0c55d4c727e90c9a465
round[ 7].is_row    cdc972c53854a47e5d64c765904cc028
round[ 7].is_box    80121e0776fd1d8a8d8c31bc965d1fee
round[ 7].ik_sch    2ab54bb43a02f8f662e3a95d66410c08
round[ 7].ik_add    aaa755b34cffe57cef6f98e1f01c13e6
round[ 8].istart    93faa123c2903f4743e4dd83431692de
round[ 8].is_row    9316dd47c2fa92834390a1de43e43f23
round[ 8].is_box    22ffc916a81474416496f19c64ae2532
round[ 8].ik_sch    58e151ab04a2a5557effb5416245080c
round[ 8].ik_add    7a1e98bdacb6d1141a6944dd06eb2d3e
round[ 9].istart    68cc08ed0abbd2bc642ef555244ae878
round[ 9].is_row    684af5bc0acce85564bb0878242ed2ed
round[ 9].is_box    f75c7778a327c8ed8cfebfc1a6c37f53
round[ 9].ik_sch    40f949b31cbabd4d48f043b810b7b342
round[ 9].ik_add    b7a53ecbbf9d75a0c40efc79b674cc11
```

```
round[10].istart   1fb5430ef0accf64aa370cde3d77792c
round[10].is_row   1f770c64f0b579deaaac432c3d37cf0e
round[10].is_box   cb02818c17d2af9c62aa64428bb25fd7
round[10].ik_sch   544afef55847f0fa4856e2e95c43f4fe
round[10].ik_add   9f487f794f955f662afc86abd7f1ab29
round[11].istart   84e1dd691a41d76f792d389783fbac70
round[11].is_row   84fb386f1ae1ac977941dd70832dd769
round[11].is_box   4f63760643e0aa85aff8c9d041fa0de4
round[11].ik_sch   101112131415161758 46f2f95c43f4fe
round[11].ik_add   5f72641557f5bc92f7be3b291db9f91a
round[12].istart   6353e08c0960e104cd70b751bacad0e7
round[12].is_row   63cab7040953d051cd60e0e7ba70e18c
round[12].is_box   000102030405060708090a0b0c0d0e0f0
round[12].ik_sch   00010203040506070809 0a0b0c0d0e0f
round[12].ioutput  00112233445566778899aabbccddeeff

GIẢI MÃ TƯƠNG ĐƯƠNG:
round[ 0].iinput   dda97ca4864cdfe06eaf70a0ec0d7191
round[ 0].ik_sch   a4970a331a78dc09c418c271e3a41d5d
round[ 1].istart   793e76979c3403e9aab7b2d10fa96ccc
round[ 1].is_box   afd10f851c28d5eb62203e51fbb7b827
round[ 1].is_row   afb73eeb1cd1b85162280f27fb20d585
round[ 1].im_col   122a02f7242ac8e20605afce51cc7264
round[ 1].ik_sch   d6bebd0dc209ea494db073803e021bb9
round[ 2].istart   c494bffae62322ab4bb5dc4e6fce69dd
round[ 2].is_box   88e7f414f532940eccd293b606ece4c9
round[ 2].is_row   88ec930ef5e7e4b6cc32f4c906d29414
round[ 2].im_col   5cc7aecce3c872194ae5ef8309a933c7
round[ 2].ik_sch   8fb999c973b26839c7f9d89d85c68c72
round[ 3].istart   d37e3705907a1a208d1c371e8c6fbfb5
round[ 3].is_box   a98ab23696bd4354b4c4b2e9f006f4d2
round[ 3].is_row   a906b254968af4e9b4bdb2d2f0c44336
round[ 3].im_col   b7113ed134e85489b20866b51d4b2c3b
round[ 3].ik_sch   f77d6ec1423f54ef5378317f14b75744
round[ 4].istart   406c501076d70066e17057ca09fc7b7f
round[ 4].is_box   72b86c7c0f0d52d3e0d0da104055036b
round[ 4].is_row   7255dad30fb80310e00d6c6b40d0527c
round[ 4].im_col   ef3b1be1b9b0e64bdcb79f1e0a707fbb
round[ 4].ik_sch   1147659047cf663b9b0ece8dfc0bf1f0
round[ 5].istart   fe7c7e71fe7f807047b95193f67b8e4b
round[ 5].is_box   0c018a2c0c6b3ad016db7022d603e6cc
round[ 5].is_row   0c0370d00c01e622166b8accd6db3a2c
round[ 5].im_col   592460b248832b2952e0b831923048f1
round[ 5].ik_sch   dcc1a8b667053f7dcc5c194ab5423a2e
round[ 6].istart   85e5c8042f8614549ebca17b277272df
round[ 6].is_box   672ab1304edc9bfddf78f1033d1e1eef
round[ 6].is_row   671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 6].im_col   0b8a7783417ae3a1f9492dc0c641a7ce
```

```
round[ 6].ik_sch    c6deb0ab791e2364a4055fbe568803ab
round[ 7].istart    cd54c7283864c0c55d4c727e90c9a465
round[ 7].is_box    80fd31ee768c1f078d5d1e8a96121dbc
round[ 7].is_row    80121e0776fd1d8a8d8c31bc965d1fee
round[ 7].im_col    4ee1ddf9301d6352c9ad769ef8d20515
round[ 7].ik_sch    dd1b7cdaf28d5c158a49ab1dbbc497cb
round[ 8].istart    93faa123c2903f4743e4dd83431692de
round[ 8].is_box    2214f132a896251664aec94164ff749c
round[ 8].is_row    22ffc916a81474416496f19c64ae2532
round[ 8].im_col    1008ffe53b36ee6af27b42549b8a7bb7
round[ 8].ik_sch    78c4f708318d3cd69655b701bfc093cf
round[ 9].istart    68cc08ed0abbd2bc642ef555244ae878
round[ 9].is_box    f727bf53a3fe7f788cc377eda65cc8c1
round[ 9].is_row    f75c7778a327c8ed8cfebfc1a6c37f53
round[ 9].im_col    7f69ac1ed939ebaac8ece3cb12e159e3
round[ 9].ik_sch    60dcef10299524ce62dbef152f9620cf
round[10].istart    1fb5430ef0accf64aa370cde3d77792c
round[10].is_box    cbd264d717aa5f8c62b2819c8b02af42
round[10].is_row    cb02818c17d2af9c62aa64428bb25fd7
round[10].im_col    cfaf16b2570c18b52e7fef50cab267ae
round[10].ik_sch    4b4ecbdb4d4dcfda5752d7c74949cbde
round[11].istart    84e1dd691a41d76f792d389783fbac70
round[11].is_box    4fe0c9e443f80d06affa76854163aad0
round[11].is_row    4f63760643e0aa85aff8c9d041fa0de4
round[11].im_col    794cf891177bfd1d8a327086f3831b39
round[11].ik_sch    1a1f181d1e1b1c194742c7d74949cbde
round[12].istart    6353e08c0960e104cd70b751bacad0e7
round[12].is_box    0050a0f04090e03080d02070c01060b0
round[12].is_row    00102030405060708090a0b0c0d0e0f0
round[12].ik_sch    000102030405060708090a0b0c0d0e0f
round[12].ioutput   00112233445566778899aabbccddeeff
```

C.3 AES-256 (*Nk*=8, *Nr*=14)

```
BẢN RÕ:      00112233445566778899aabbccddeeff
KHÓA:        000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f

MÃ THUẬN (MÃ HÓA):
round[ 0].input     00112233445566778899aabbccddeeff
round[ 0].k_sch     000102030405060708090a0b0c0d0e0f
round[ 1].start     00102030405060708090a0b0c0d0e0f0
round[ 1].s_box     63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row     6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col     5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch     101112131415161718191a1b1c1d1e1f
round[ 2].start     4f63760643e0aa85efa7213201a4e705
round[ 2].s_box     84fb386f1ae1ac97df5cfd237c49946b
```

```
round[ 2].s_row    84e1fd6b1a5c946fdf4938977cfbac23
round[ 2].m_col    bd2a395d2b6ac438d192443e615da195
round[ 2].k_sch    a573c29fa176c498a97fce93a572c09c
round[ 3].start    1859fbc28a1c00a078ed8aadc42f6109
round[ 3].s_box    adcb0f257e9c63e0bc557e951c15ef01
round[ 3].s_row    ad9c7e017e55ef25bc150fe01ccb6395
round[ 3].m_col    810dce0cc9db8172b3678c1e88a1b5bd
round[ 3].k_sch    1651a8cd0244beda1a5da4c10640bade
round[ 4].start    975c66c1cb9f3fa8a93a28df8ee10f63
round[ 4].s_box    884a33781fdb75c2d380349e19f876fb
round[ 4].s_row    88db34fb1f807678d3f833c2194a759e
round[ 4].m_col    b2822d81abe6fb275faf103a078c0033
round[ 4].k_sch    ae87dff00ff11b68a68ed5fb03fc1567
round[ 5].start    1c05f271a417e04ff921c5c104701554
round[ 5].s_box    9c6b89a349f0e18499fda678f2515920
round[ 5].s_row    9cf0a62049fd59a399518984f26be178
round[ 5].m_col    aeb65ba974e0f822d73f567bdb64c877
round[ 5].k_sch    6de1f1486fa54f9275f8eb5373b8518d
round[ 6].start    c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 6].s_box    2e5bacf8af6ea9e73ac67a34c286ee2d
round[ 6].s_row    2e6e7a2dafc6eef83a86ace7c25ba934
round[ 6].m_col    b951c33c02e9bd29ae25cdb1efa08cc7
round[ 6].k_sch    c656827fc9a799176f294cec6cd5598b
round[ 7].start    7f074143cb4e243ec10c815d8375d54c
round[ 7].s_box    d2c5831a1f2f36b278fe0c4cec9d0329
round[ 7].s_row    d22f0c291ffe031a789d83b2ecc5364c
round[ 7].m_col    ebb19e1c3ee7c9e87d7535e9ed6b9144
round[ 7].k_sch    3de23a75524775e727bf9eb45407cf39
round[ 8].start    d653a4696ca0bc0f5acaab5db96c5e7d
round[ 8].s_box    f6ed49f950e06576be74624c565058ff
round[ 8].s_row    f6e062ff507458f9be50497656ed654c
round[ 8].m_col    5174c8669da98435a8b3e62ca974a5ea
round[ 8].k_sch    0bdc905fc27b0948ad5245a4c1871c2f
round[ 9].start    5aa858395fd28d7d05e1a38868f3b9c5
round[ 9].s_box    bec26a12cfb55dff6bf80ac4450d56a6
round[ 9].s_row    beb50aa6cff856126b0d6aff45c25dc4
round[ 9].m_col    0f77ee31d2ccadc05430a83f4ef96ac3
round[ 9].k_sch    45f5a66017b2d387300d4d33640a820a
round[10].start    4a824851c57e7e47643de50c2af3e8c9
round[10].s_box    d61352d1a6f3f3a04327d9fee50d9bdd
round[10].s_row    d6f3d9dda6279bd1430d52a0e513f3fe
round[10].m_col    bd86f0ea748fc4f4630f11c1e9331233
round[10].k_sch    7ccff71cbeb4fe5413e6bbf0d261a7df
round[11].start    c14907f6ca3b3aa070e9aa313b52b5ec
round[11].s_box    783bc54274e280e0511eacc7e200d5ce
round[11].s_row    78e2acce741ed5425100c5e0e23b80c7
round[11].m_col    af8690415d6e1dd387e5fbedd5c89013
round[11].k_sch    f01afafee7a82979d7a5644ab3afe640
```

```
round[12].start     5f9c6abfbac634aa50409fa766677653
round[12].s_box     cfde0208f4b418ac5309db5c338538ed
round[12].s_row     cfb4dbedf4093808538502ac33de185c
round[12].m_col     7427fae4d8a695269ce83d315be0392b
round[12].k_sch     2541fe719bf500258813bbd55a721c0a
round[13].start     516604954353950314fb86e401922521
round[13].s_box     d133f22a1aed2a7bfa0f44697c4f3ffd
round[13].s_row     d1ed44fd1a0f3f2afa4ff27b7c332a69
round[13].m_col     2c21a820306f154ab712c75eee0da04f
round[13].k_sch     4e5a6699a9f24fe07e572baacdf8cdea
round[14].start     627bceb9999d5aaac945ecf423f56da5
round[14].s_box     aa218b56ee5ebeacdd6ecebf26e63c06
round[14].s_row     aa5ece06ee6e3c56dde68bac2621bebf
round[14].k_sch     24fc79ccbf0979e9371ac23c6d68de36
round[14].output    8ea2b7ca516745bfeafc49904b496089

MÃ NGHỊCH (GIẢI MÃ):
round[ 0].iinput    8ea2b7ca516745bfeafc49904b496089
round[ 0].ik_sch    24fc79ccbf0979e9371ac23c6d68de36
round[ 1].istart    aa5ece06ee6e3c56dde68bac2621bebf
round[ 1].is_row    aa218b56ee5ebeacdd6ecebf26e63c06
round[ 1].is_box    627bceb9999d5aaac945ecf423f56da5
round[ 1].ik_sch    4e5a6699a9f24fe07e572baacdf8cdea
round[ 1].ik_add    2c21a820306f154ab712c75eee0da04f
round[ 2].istart    d1ed44fd1a0f3f2afa4ff27b7c332a69
round[ 2].is_row    d133f22a1aed2a7bfa0f44697c4f3ffd
round[ 2].is_box    516604954353950314fb86e401922521
round[ 2].ik_sch    2541fe719bf500258813bbd55a721c0a
round[ 2].ik_add    7427fae4d8a695269ce83d315be0392b
round[ 3].istart    cfb4dbedf4093808538502ac33de185c
round[ 3].is_row    cfde0208f4b418ac5309db5c338538ed
round[ 3].is_box    5f9c6abfbac634aa50409fa766677653
round[ 3].ik_sch    f01afafee7a82979d7a5644ab3afe640
round[ 3].ik_add    af8690415d6e1dd387e5fbedd5c89013
round[ 4].istart    78e2acce741ed5425100c5e0e23b80c7
round[ 4].is_row    783bc54274e280e0511eacc7e200d5ce
round[ 4].is_box    c14907f6ca3b3aa070e9aa313b52b5ec
round[ 4].ik_sch    7ccff71cbeb4fe5413e6bbf0d261a7df
round[ 4].ik_add    bd86f0ea748fc4f4630f11c1e9331233
round[ 5].istart    d6f3d9dda6279bd1430d52a0e513f3fe
round[ 5].is_row    d61352d1a6f3f3a04327d9fee50d9bdd
round[ 5].is_box    4a824851c57e7e47643de50c2af3e8c9
round[ 5].ik_sch    45f5a66017b2d387300d4d33640a820a
round[ 5].ik_add    0f77ee31d2ccadc05430a83f4ef96ac3
round[ 6].istart    beb50aa6cff856126b0d6aff45c25dc4
round[ 6].is_row    bec26a12cfb55dff6bf80ac4450d56a6
round[ 6].is_box    5aa858395fd28d7d05e1a38868f3b9c5
round[ 6].ik_sch    0bdc905fc27b0948ad5245a4c1871c2f
```

```
round[ 6].ik_add    5174c8669da98435a8b3e62ca974a5ea
round[ 7].istart    f6e062ff507458f9be50497656ed654c
round[ 7].is_row    f6ed49f950e06576be74624c565058ff
round[ 7].is_box    d653a4696ca0bc0f5acaab5db96c5e7d
round[ 7].ik_sch    3de23a75524775e727bf9eb45407cf39
round[ 7].ik_add    ebb19e1c3ee7c9e87d7535e9ed6b9144
round[ 8].istart    d22f0c291ffe031a789d83b2ecc5364c
round[ 8].is_row    d2c5831a1f2f36b278fe0c4cec9d0329
round[ 8].is_box    7f074143cb4e243ec10c815d8375d54c
round[ 8].ik_sch    c656827fc9a799176f294cec6cd5598b
round[ 8].ik_add    b951c33c02e9bd29ae25cdb1efa08cc7
round[ 9].istart    2e6e7a2dafc6eef83a86ace7c25ba934
round[ 9].is_row    2e5bacf8af6ea9e73ac67a34c286ee2d
round[ 9].is_box    c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 9].ik_sch    6de1f1486fa54f9275f8eb5373b8518d
round[ 9].ik_add    aeb65ba974e0f822d73f567bdb64c877
round[10].istart    9cf0a62049fd59a399518984f26be178
round[10].is_row    9c6b89a349f0e18499fda678f2515920
round[10].is_box    1c05f271a417e04ff921c5c104701554
round[10].ik_sch    ae87dff00ff11b68a68ed5fb03fc1567
round[10].ik_add    b2822d81abe6fb275faf103a078c0033
round[11].istart    88db34fb1f807678d3f833c2194a759e
round[11].is_row    884a33781fdb75c2d380349e19f876fb
round[11].is_box    975c66c1cb9f3fa8a93a28df8ee10f63
round[11].ik_sch    1651a8cd0244beda1a5da4c10640bade
round[11].ik_add    810dce0cc9db8172b3678c1e88a1b5bd
round[12].istart    ad9c7e017e55ef25bc150fe01ccb6395
round[12].is_row    adcb0f257e9c63e0bc557e951c15ef01
round[12].is_box    1859fbc28a1c00a078ed8aadc42f6109
round[12].ik_sch    a573c29fa176c498a97fce93a572c09c
round[12].ik_add    bd2a395d2b6ac438d192443e615da195
round[13].istart    84e1fd6b1a5c946fdf4938977cfbac23
round[13].is_row    84fb386f1ae1ac97df5cfd237c49946b
round[13].is_box    4f63760643e0aa85efa7213201a4e705
round[13].ik_sch    101112131415161718191a1b1c1d1e1f
round[13].ik_add    5f72641557f5bc92f7be3b291db9f91a
round[14].istart    6353e08c0960e104cd70b751bacad0e7
round[14].is_row    63cab7040953d051cd60e0e7ba70e18c
round[14].is_box    00102030405060708090a0b0c0d0e0f0
round[14].ik_sch    000102030405060708090a0b0c0d0e0f
round[14].ioutput   00112233445566778899aabbccddeeff

GIẢI MÃ TƯƠNG ĐƯƠNG:
round[ 0].iinput    8ea2b7ca516745bfeafc49904b496089
round[ 0].ik_sch    24fc79ccbf0979e9371ac23c6d68de36
round[ 1].istart    aa5ece06ee6e3c56dde68bac2621bebf
round[ 1].is_box    629deca599456db9c9f5ceaa237b5af4
round[ 1].is_row    627bceb9999d5aaac945ecf423f56da5
```

```
round[ 1].im_col    e51c9502a5c1950506a61024596b2b07
round[ 1].ik_sch    34f1d1ffbfceaa2ffce9e25f2558016e
round[ 2].istart    d1ed44fd1a0f3f2afa4ff27b7c332a69
round[ 2].is_box    5153862143fb259514920403016695e4
round[ 2].is_row    516604954353950314fb86e401922521
round[ 2].im_col    91a29306cc450d0226f4b5eaef5efed8
round[ 2].ik_sch    5e1648eb384c350a7571b746dc80e684
round[ 3].istart    cfb4dbedf4093808538502ac33de185c
round[ 3].is_box    5fc69f53ba4076bf50676aaa669c34a7
round[ 3].is_row    5f9c6abfbac634aa50409fa766677653
round[ 3].im_col    b041a94eff21ae9212278d903b8a63f6
round[ 3].ik_sch    c8a305808b3f7bd043274870d9b1e331
round[ 4].istart    78e2acce741ed5425100c5e0e23b80c7
round[ 4].is_box    c13baaeccae9b5f6705207a03b493a31
round[ 4].is_row    c14907f6ca3b3aa070e9aa313b52b5ec
round[ 4].im_col    638357cec07de6300e30d0ec4ce2a23c
round[ 4].ik_sch    b5708e13665a7de14d3d824ca9f151c2
round[ 5].istart    d6f3d9dda6279bd1430d52a0e513f3fe
round[ 5].is_box    4a7ee5c9c53de85164f348472a827e0c
round[ 5].is_row    4a824851c57e7e47643de50c2af3e8c9
round[ 5].im_col    ca6f71058c642842a315595fdf54f685
round[ 5].ik_sch    74da7ba3439c7e50c81833a09a96ab41
round[ 6].istart    beb50aa6cff856126b0d6aff45c25dc4
round[ 6].is_box    5ad2a3c55fe1b93905f3587d68a88d88
round[ 6].is_row    5aa858395fd28d7d05e1a38868f3b9c5
round[ 6].im_col    ca46f5ea835eab0b9537b6dbb221b6c2
round[ 6].ik_sch    3ca69715d32af3f22b67ffade4ccd38e
round[ 7].istart    f6e062ff507458f9be50497656ed654c
round[ 7].is_box    d6a0ab7d6cca5e695a6ca40fb953bc5d
round[ 7].is_row    d653a4696ca0bc0f5acaab5db96c5e7d
round[ 7].im_col    2a70c8da28b806e9f319ce42be4baead
round[ 7].ik_sch    f85fc4f3374605f38b844df0528e98e1
round[ 8].istart    d22f0c291ffe031a789d83b2ecc5364c
round[ 8].is_box    7f4e814ccb0cd543c175413e8307245d
round[ 8].is_row    7f074143cb4e243ec10c815d8375d54c
round[ 8].im_col    f0073ab7404a8a1fc2cba0b80df08517
round[ 8].ik_sch    de69409aef8c64e7f84d0c5fcfab2c23
round[ 9].istart    2e6e7a2dafc6eef83a86ace7c25ba934
round[ 9].is_box    c345bdfa1bc799e1a2dcaab0a857b728
round[ 9].is_row    c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 9].im_col    3225fe3686e498a32593c1872b613469
round[ 9].ik_sch    aed55816cf19c100bcc24803d90ad511
round[10].istart    9cf0a62049fd59a399518984f26be178
round[10].is_box    1c17c554a4211571f970f24f0405e0c1
round[10].is_row    1c05f271a417e04ff921c5c104701554
round[10].im_col    9d1d5c462e655205c4395b7a2eac55e2
round[10].ik_sch    15c668bd31e5247d17c168b837e6207c
round[11].istart    88db34fb1f807678d3f833c2194a759e
```

```
round[11].is_box    979f2863cb3a0fc1a9e166a88e5c3fdf
round[11].is_row    975c66c1cb9f3fa8a93a28df8ee10f63
round[11].im_col    d24bfb0e1f997633cfce86e37903fe87
round[11].ik_sch    7fd7850f61cc991673db890365c89d12
round[12].istart    ad9c7e017e55ef25bc150fe01ccb6395
round[12].is_box    181c8a098aed61c2782ffba0c45900ad
round[12].is_row    1859fbc28a1c00a078ed8aadc42f6109
round[12].im_col    aec9bda23e7fd8aff96d74525cdce4e7
round[12].ik_sch    2a2840c924234cc026244cc5202748c4
round[13].istart    84e1fd6b1a5c946fdf4938977cfbac23
round[13].is_box    4fe0210543a7e706efa476850163aa32
round[13].is_row    4f63760643e0aa85efa7213201a4e705
round[13].im_col    794cf891177bfd1ddf67a744acd9c4f6
round[13].ik_sch    1a1f181d1e1b1c191217101516131411
round[14].istart    6353e08c0960e104cd70b751bacad0e7
round[14].is_box    0050a0f04090e03080d02070c01060b0
round[14].is_row    00102030405060708090a0b0c0d0e0f0
round[14].ik_sch    0001020304050607080 90a0b0c0d0e0f
round[14].ioutput   00112233445566778899aabbccddeeff
```

Tài liệu tham khảo

[1] Trang thông tin về AES của NIST http://www.nist.gov/CryptoToolkit.[4]

[2] Danh mục về các đối tượng an toàn máy tính (CSOR - Computer Security Objects Register) tại địa chỉ của NIST: http://csrc.nist.gov/csor/.

[3] J. Daemen và V. Rijmen, *AES Proposal: Rijndael*, AES Algorithm Submission (*Đề xuất thuật toán AES*, Bản đệ trình về thuật toán AES), 03/12/1999 có thể xem tại [1].

[4] J. Daemen and V. Rijmen, *The block cipher Rijdael*, Smart Card research and Applications (*Hệ mã khối Rijndael*, Nghiên cứu và ứng dụng bằng Thẻ thông minh), LNCS 1820, Springer-Verlag, trang 288-296.

[5] Trang web liên quan đến AES của B. Gladman http://fp.gladman.plus.com/cryptography_technology/.

[6] A. Lee, NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, National Institute if Standard and Technology (Bản công bố đặc tả của NIST 800-21, *Chỉ dẫn thực thi mật mã đối với cơ quan Chính phủ Liên bang,* Viện Tiêu chuẩn và Công nghệ quốc gia Hoa Kỳ), 1999.

[7] A. Menezes, P. van Oorschot, và S. Vanstone, *Handbook of Applied Cryptography* (*Sổ tay ứng dụng mật mã)*, CRC Press, New York, 1997, trang 81-83.

[8] J. Nechvatal, et. al., *Report on the Development of the Advanced Encryption Standard (AES),* National Institute if Standard and Technology *(Báo cáo về việc Phát triển Chuẩn mã hóa tiên tiến (AES)*, Viện tiêu chuẩn và công nghệ quốc gia Hoa Kỳ), 02/10/2000, có tại [1].

---

[4] Một tập hoàn chỉnh tài liệu về các nỗ lực phát triển AES – bao gồm các thông báo, ý kiến công khai, bài báo phân tích, hội nghị,... có tại web - site này.