1. Ước số chung lớn nhất của a và b

- Kí hiệu: gcd (a, b) – Greatest Common Divisor
- Thuật toán Euclidean:

$\gcd(a, 0) = a$

$\gcd(a, b) = \gcd(b, a \ mod \ b)$

Ví dụ: $\gcd(36, 10) = \gcd(10, 6) = \gcd(6, 4) = \gcd(4, 2) = \gcd(2, 0) = 2$

- Thuật toán Euclidean mở rộng:

Bước 1. Khởi tạo $r_1 = a; r_2 = b; t_1 = 0; t_2 = 1; s_1 = 1; s_2 = 0$

Bước 2. $q = \lfloor r_1/r_2 \rfloor; r = r_1 \ mod \ r_2; t = t_1 - q \times t_2$

Bước 3. Lặp lại Bước 2 cho đến khi tìm s_1 và t_1 thoả mãn $\gcd(a, b) = s \times a + t \times b$

Ví dụ: Tìm $\gcd(161,28)$

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

$\gcd(161, 28) = (-1) \times 161 + 6 \times 28 = 7$

2. Toán học modular

$$Zn = \{0; 1; 2; \dots; (n-1)\}$$

2.1 Toán học modular cho số nguyên

a. Hai số a và b được gọi là nghịch đảo cộng của nhau nếu $(a + b)$ đồng dư $0\ (mod\ n)$

Ví dụ: trong tập Z_{10} có 6 cặp nghịch đảo cộng: (0,0) (1,9) (2,8) (3,7) (4,6) (5,5)

b. Hai số a và b được gọi là nghịch đảo nhân của nhau nếu $a \times b$ đồng dư 1 $(mod\ n)$
Điều kiện để một số nguyên a trong tập Z_n có nghịch đảo nhân: $gcd(n, a) = 1$
Note: Không phải phần tử nào trong tập Z_n cũng có nghịch đảo nhân.
Vd: $gcd(8,10) = 2$ khác 1 ⇒ Không tồn tại nghịch đảo nhân của 8 trong tập Z_{10}

c. Sử dụng thuật toán Euclidean mở rộng để tìm nghịch đảo nhân
Ví dụ: Tìm nghịch đảo nhân (multiplicative inverse) của 11 trong tập Z_{26}

Bước 1: $\gcd(26,11) = \cdots = 1$ ⇒ Tồn tại MI của 11 trong tập Z_{26}

Bước 2: Lập bảng

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26
	1	0		-7	26	

Sau khi lập bảng ta tìm được $t_1 = -7$ nhưng -7 không thuộc tập Z_{26} nên ta phải lấy đồng dư của -7 trong tập Z_{26} là 19.

Bước 3: Thử lại

$$(11 \times 19)\ mod\ 26 = 209\ mod\ 26 = 1 \text{ (thỏa mãn)}$$

d. Nhận xét
Trong tập Z_n với n là số nguyên tố: Mỗi phần tử của Z_n đều có nghịch đảo cộng, mỗi phần tử trừ 0 đều có nghịch đảo nhân

2.2 Toán học modular cho ma trận

– Ma trận A và ma trận B là nghịch đảo cộng của nhau nếu $A + B = 0$

– Ma trận A và ma trận B là nghịch đảo nhân của nhau nếu $A.B = B.A = I\ (mod\ n)$

– Điều kiện để một ma trận có ma trận nghịch đảo nhân là $gcd[det(A), n] = 1$

Ví dụ: Tìm ma trận nghịch đảo của $A = \begin{pmatrix} 3 & 9 \\ 2 & 7 \end{pmatrix}$ trong Z_{17}

- Bước 1: Tìm $det(A) = 21 - 18 = 3$
- Bước 2: Tìm $gcd[det(A),17] = 1$ ⇒ Tồn tại MI của A trong Z_{17}
- Bước 3: Tìm MI của 3 trong tập Z_{17} sử dụng thuật toán Euclidean mở rộng

  ⇒ Kết quả: 6

- Bước 4: Tìm ma trận B từ ma trận A với:

  $B_{ij} = (-1)^{i+j} \times D_{ij}$

  ⇒ Kết quả: $B = \begin{pmatrix} 7 & -2 \\ -9 & 3 \end{pmatrix}$
- Bước 5: MI của $A = A^{-1} = MI(det(A)) \times B^T$
- Bước 6: Thử lại $A.A^{-1} = I$ ⇒ Kết quả đúng

3. Bộ mật cổ điển

Sử dụng bảng chữ cái tiếng anh với 26 kí tự từ a đến z và được đánh số tương ứng từ 0 đến 25

a	b	c	d	e	f	g	h	i
0	1	2	3	4	5	6	7	8

j	k	l	m	n	o	p	q	r
9	10	11	12	13	14	15	16	17

s	t	u	v	w	x	y	z
18	19	20	21	22	23	24	25

3.1 Ceasar cipher (hay còn gọi là Additive cipher hay Shift cipher)

Encrypt: $C = (P + K)\ mod\ 26$

Decrypt: $P = (C - K)\ mod\ 26$

3.2 Affine cipher

Sử dụng 2 khóa K1, K2

Encrypt: $C = (P * K_1 + K_2)\ mod\ 26$

Decrypt: $P = ((C - K_2) * K_1^{-1})\ mod\ 26$

3.3 Autokey cipher

Plain text $= P_1P_2P_3 \ldots.$

Cipher text $= C_1C_2C_3 \ldots$

Key $= K_1P_1P_2 \ldots$

Encrypt: $C_i = (P_i + K_i)\ mod\ 26$

Decrypt: $P_i = (C_i - K_i)\ mod\ 26$

3.4 Vigenere cipher

Plain text $= P_1P_2P_3 \ldots$

Cipher text $= C_1C_2C_3 \ldots$

Key $= [(K_1, K_2, .. K_m), (K_1, K_2, \ldots K_m) \ldots]$ (Key stream)

Encrypt: $C_i = (P_i + K_i)\ mod\ 26$

Decrypt: $P_i = (C_i - K_i)\ mod\ 26$

Ví dụ:

Plaintext	S	H	E	I	S	L	I
P's value	18	7	4	8	18	11	8
Key word	P	A	S	C	A	L	P
Key stream	15	00	18	02	00	11	15
C's value	7	7	22	10	18	22	23
Ciphertext	H	H	W	K	S	W	X

Note: Cách làm khác nhanh hơn ⇒ tra bảng Vigenere Table với trục ngang là P trục đứng là Key

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

3.5 Hill cipher

Encrypt: $C = (P \times K)\ mod\ 26$

Decrypt: $P = (C \times K^{-1})\ mod\ 26$

Note: Khóa K là một ma trận vuông bậc m và tồn tại ma trận nghịch đảo nhân K^{-1}

Chuyển P hoặc C (với bài toán mật mã hóa hoặc giải mật mã hóa) thành ma trận có m cột để tương ứng với ma trận K

3.6 Rain fence cipher (zigzag cipher) (Chuyển vị không khóa)

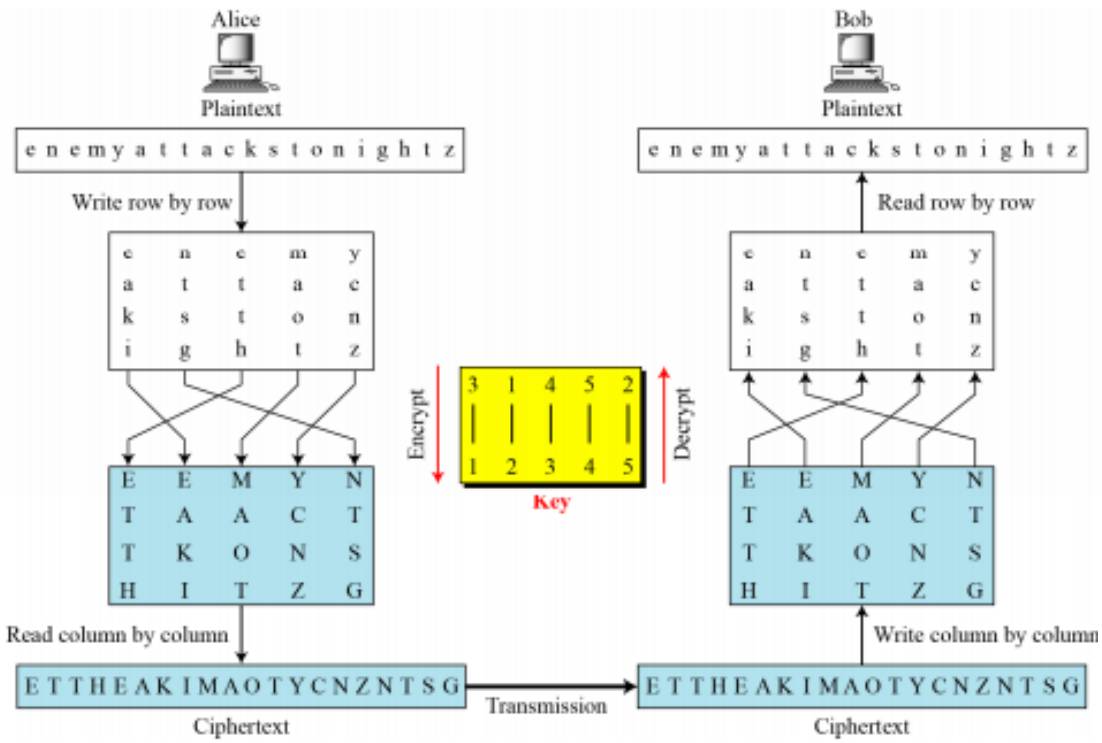
Sắp xếp Plaintext theo hình chữ V và đọc theo từng hang để suy ra ciphertext.

3.7 Chuyển vị có khóa

Ví dụ: Plaintext:

Viết Plaintext theo hàng ngang với số cột cố định để tạo thành các khối (nếu không đủ thì tự thêm kí tự)

Thông qua quy luật khoá ⇒ Khối mới ⇒ Đọc theo cột để suy ra cipher text

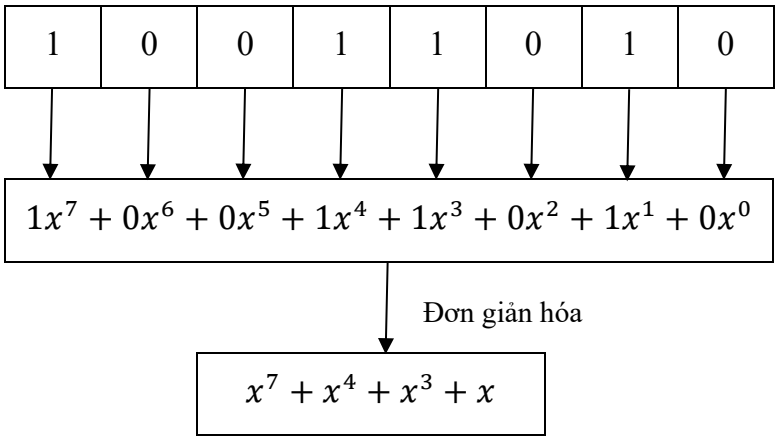


4. Trường Galois (Galois Field – GF)

- Định nghĩa: Một trường Galois p^n là một trường hữu hạn bao gồm p^n phần tử.
- Ký hiệu: $GF(p^n)$. Các phép toán trong trường Galois: cộng, trừ, nhân, chia
- $GF(2^n)$ là một tập 2^n phần tử, mỗi phần tử gồm n bit (Giá trị 2 tương đương với hai bit nhị phân 1 và 0, với n là độ dài của từ mã).
- $GF(2^n)$ có thể được biểu diễn dưới dạng đa thức bậc $(n-1)$

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x^1 + a_0x^0$$

- Ví dụ: Cách biểu diễn một từ 8-bit sử dụng đa thức:



- Để đảm bảo các giá trị của các phép tính đa thức vẫn trong giới hạn bậc $(n-1)$, trong một số trường hợp ta phải tiến hành giảm bậc. Việc giảm bậc các đa thức thực chất là tiến hành chia lấy dư cho đa thức tối giản.

Bậc	**Đa thức tối giản**
1	$(x+1), (x)$
2	$(x^2 + x + 1)$
3	$(x^3 + x^2 + 1), (x^3 + x + 1)$
4	$(x^4 + x + 1), (x^4 + x^3 + x^2 + x + 1), (x^4 + x^3 + 1)$
5	$(x^5 + x^2 + 1), (x^5 + x^3 + x^2 + x + 1), (x^3 + x^2 + 1), (x^5 + x^3 + 1),$ $(x^5 + x^4 + x^3 + x + 1), (x^5 + x^4 + x^3 + x^2 + x + 1), (x^5 + x^4 + x^2 + x + 1)$
.....	
8	$(x^8 + x^4 + x^3 + x + 1)$

- Lưu ý:
    - Thường là $GF(2^8)$ ⇒ Nhớ đa thức $(x^8 + x^4 + x^3 + x + 1)$
    - Phép tính cộng trong trường GF là phép cộng modulo 2.
    - Phép nhân: thường phải giảm bậc.
- Xét ví dụ: $(21)_{16} \times (FB)_{16}$ trong trường $GF(2^8)$

  $21_{16} = 0010\ 0001 = x^5 + 1$

  $\mathrm{FB}_{16} = 1110\ 1011 = x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$

  $21_{16} \times \mathrm{FB}_{16}$

  $= x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$

  $= x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^4 + x^3 + x + 1$

    - Đa thức trên có bậc $12 > 7$ ⇒ Tiến hành giảm bậc bằng cách lấy $f(x)$ chia lấy dư cho đa thức $(x^8 + x^4 + x^3 + x + 1)$ ⇒ Kết quả là $(x^7 + x^5 + x^3 + 1) = 1010\ 1001 = A9$
    - Vậy $21_{16} \times \mathrm{FB}_{16} = A9_{16}$

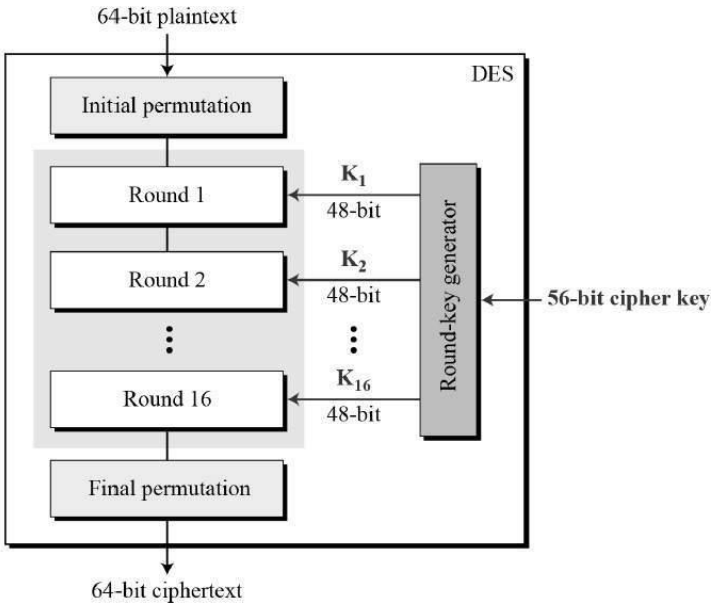
5. Bộ mật hiện đại

Tiêu chí thiết kế mật mã khối:

– Độ hoàn chỉnh: Mỗi bit của khối đầu ra sẽ chỉ phụ thuộc vào 1 bit của khối đầu vào và mỗi bit của khoá
– Tính thác lũ: Việc thay đổi 1 bit đầu vào của bộ mật sẽ kéo theo sự thay đổi của xấp xỉ một nửa độ dài bit của khối đầu ra.
– Tính thống kê

DES

– Là hệ mật hiện đại đầu tiên, điển hình
– Là bộ mật mã hoá khối với khoá đối xứng (Khoá phía thu với phía phát là giống nhau)
– Cipher Key = 56 bit
– Block size = 64 bit
– Rounds = 16
– Cấu trúc mật mã hóa



Encrypt DES

Xét ví dụ: Plaintext $= 0123456789ABCDEF$ với key $= 133457799BBCDFF1$

Bước 1: Round-Key Generator

– Từ key ban đầu là 64 bit ta bỏ đi 8 bit chẵn lẻ để tạo thành cipher key 56 bit thông qua bảng PC-1:

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

Key:

0001 0011 0011 0100 0101 0111 0111 1001 1001 1011 1011 1100 1101 1111 1111 0001

⇒ Cipher key:

1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

– Chia cipher key 56 bit thành 2 nửa C và D mỗi nửa 28 bit

⇒ C_0: 1111000 0110011 0010101 0101111

D_0: 0101010 1011001 1001111 0001111

– Từ C_0 và D_0 ta tạo ra các C_n và D_n với $1 \leq n \leq 16$ với quy tắc dịch trái như bảng:

Shifting

Rounds	Shift
1, 2, 9, 16	One bit
Others	Two bits

Từ quy tắc trên ta có các C_n và D_n như sau:

$\boldsymbol{C_1} = 11100001100110010101011111$
$\boldsymbol{D_1} = 10101010110011001111100011110$

$\boldsymbol{C_2} = 11000011001100101010111111$
$\boldsymbol{D_2} = 01010101100110011111000111101$

$\boldsymbol{C_3} = 00001100110010101011111111$
$\boldsymbol{D_3} = 01010110011001111100011110101$

$\boldsymbol{C_4} = 00110011001010101011111111100$
$\boldsymbol{D_4} = 01011001100111100011110101$

$\boldsymbol{C_5} = 11001100101010101111111110000$
$\boldsymbol{D_5} = 01100110011110001111010101$

$\boldsymbol{C_6} = 00110010101010111111111000011$
$\boldsymbol{D_6} = 10011001111000111101010101$

$\boldsymbol{C_7} = 11001010101011111111100001100$
$\boldsymbol{D_7} = 01100111100011110101010110$

$\boldsymbol{C_8} = 00101010101111111110000110011$
$\boldsymbol{D_8} = 10011110001111010101011001$

$\boldsymbol{C_9} = 01010101011111111100001100110$
$\boldsymbol{D_9} = 00111100011110101010110011$

$\boldsymbol{C_{10}} = 01010101111111100001100110011$
$\boldsymbol{D_{10}} = 11110001111010101011001100$

$\boldsymbol{C_{11}} = 01010111111110000110011001101$
$\boldsymbol{D_{11}} = 11000111101010101100110011$

$\boldsymbol{C_{12}} = 010111111110000110011001010 1$
$\boldsymbol{D_{12}} = 000111101010101011001100 1111$

$\boldsymbol{C_{13}} = 011111111000011001100101010 1$
$\boldsymbol{D_{13}} = 011110101010101100110011110 0$

$\boldsymbol{C_{14}} = 111111100001100110010101010 1$
$\boldsymbol{D_{14}} = 111010101010110011001111000 1$

$\boldsymbol{C_{15}} = 111110000110011001010101011 1$
$\boldsymbol{D_{15}} = 101010101011001100111100011 1$

$\boldsymbol{C_{16}} = 111100001100110010101010111 1$
$\boldsymbol{D_{16}} = 010101010110011001111000111 1$

– Ta ghép C_n và D_nvới $1 \leq n \leq 16$ thành một dãy 56 bit sau đó bỏ đi 8 bit để tạo thành K_n 48 bit thông qua bảng PC-2 sau:

PC-2

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

Kết quả thu được như sau:

$\boldsymbol{K1} = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$
$\boldsymbol{K2} = 011110\ 011010\ 111011\ 011001\ 110110\ 111100\ 100111\ 100101$
$\boldsymbol{K3} = 010101\ 011111\ 110010\ 001010\ 010000\ 101100\ 111110\ 011001$
$\boldsymbol{K4} = 011100\ 101010\ 110111\ 010110\ 110110\ 110011\ 010100\ 011101$
$\boldsymbol{K5} = 011111\ 001110\ 110000\ 000111\ 111010\ 110101\ 001110\ 101000$
$\boldsymbol{K6} = 011000\ 111010\ 010100\ 111110\ 010100\ 000111\ 101100\ 101111$
$\boldsymbol{K7} = 111011\ 001000\ 010010\ 110111\ 111101\ 100001\ 100010\ 111100$
$\boldsymbol{K8} = 111101\ 111000\ 101000\ 111010\ 110000\ 010011\ 101111\ 111011$
$\boldsymbol{K9} = 111000\ 001101\ 101111\ 101011\ 111011\ 011110\ 011110\ 000001$
$\boldsymbol{K10} = 101100\ 011111\ 001101\ 000111\ 101110\ 100100\ 011001\ 001111$
$\boldsymbol{K11} = 001000\ 010101\ 111111\ 010011\ 110111\ 101101\ 001110\ 000110$
$\boldsymbol{K12} = 011101\ 010111\ 000111\ 110101\ 100101\ 000110\ 011111\ 101001$

$\boldsymbol{K13}$ = 100101 111100 010111 010001 111110 101011 101001 000001

$\boldsymbol{K14}$ = 010111 110100 001110 110111 111100 101110 011100 111010

$\boldsymbol{K15}$ = 101111 111001 000110 001101 001111 010011 111100 001010

$\boldsymbol{K16}$ = 110010 110011 110110 001011 000011 100001 011111 110101

Bước 2: Hoán vị khởi tạo

64 bit đầu vào được hoán đổi vị trí để tạo thành dãy 64 bit mới thông qua bảng hoán vị khởi tạo sau:

The Initial Permutation: IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Plaintext =

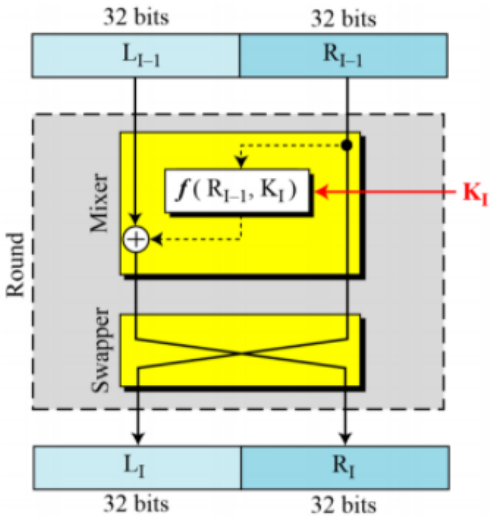
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

$\Rightarrow IP =$

1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Bước 3: Các rounds

64 bit sau bước hoán vị khởi tạo được đưa vào lần lượt 16 rounds. Mỗi round là một bộ mật Feistel (Feistel cipher) đươc biểu diễn như sơ đồ:



Nhìn vào sơ đồ ta có thể thấy mỗi round gồm 2 phần: Mixer (trộn) và Swapper (đổi chỗ).

64 bit được chia thành hai nửa trái phải, mỗi nửa 32 bit. Nửa bên phải được đưa qua hàm f gọi là hàm DES (DES function) sau đó cộng modulo với nửa trái $\Rightarrow$ đầu ra của Mixer. Đầu ra của Mixer và nửa phải ban đầu được Swap cho nhau tạo thành hai nửa trái phải mới, mỗi nửa 32 bit.

Các vòng được thực hiện giống nhau tuy nhiên: Round cuối cùng (round 16) chỉ có Mixer mà không có Swapper.

DES function: $f(R_{i-1}, K_i)$

Hàm DES có hai biến đầu vào là nửa phải R_{i-1} 32 bit và khóa khởi tạo K_i tương ứng 48 bit. Kết quả trả về của hàm DES là dãy 32 bit mới. Cấu trúc hàm DES:

Nửa phải 32 bit được đưa qua khối Expansion P-box để tạo thành dãy 48 bit mới thông qua bảng sau:

E BIT-SELECTION TABLE

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

Kết quả sau đó được cộng modulo 2 (XOR) với khóa khởi tạo tương ứng K_i 48 bit để tạo thành dãy 48 bit mới chính là đầu vào của khối S-Boxes. Khối S-Boxes này có nhiệm vụ chuyển 48 bit đầu vào thành 32 bit đầu ra. Nhìn vào sơ đồ có thể thấy khối S-Boxes được chia thành 8 phần, mỗi phần 6 bit. Mỗi khối 6 bit tương ứng với một bảng S-box để tạo thành khối mới 4 bit.

		Column															
Box	Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1																	
	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Với mỗi khối 6 bit: Lấy bit đầu và bit cuối tạo thành chỉ số hàng, 4 bit còn lại tạo thành chỉ số cột và dựa vào bảng S-box như trên để tìm ra giá trị tương ứng với 4 bit.

Kết quả đầu ra của khối S-Boxes được đưa vào khối Straight P-box để hoán đổi các vị trí tạo thành 32 bit mới thông qua bảng sau:

P

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

(Quay trở lại ví dụ)

Từ IP ⇒

$\boldsymbol{L_o}$ = 1100 1100 0000 0000 1100 1100 1111 1111

$\boldsymbol{R_0}$ = 1111 0000 1010 1010 1111 0000 1010 1010

Xét $f(R_0, K_1)$ với

$\boldsymbol{K_1}$ = 000110 110000 001011 101111 111111 000111 000001 110010

Từ R_0 (expansion P-box) ⇒

011110 100001 010101 010101 011110 100001 010101 010101

(XOR K_1) ⇒

011000 010001 011110 111010 100001 100110 010100 100111

(S-Boxes) ⇒

0101 1100 1000 0010 1011 0101 1001 0111

(Straight P-box) ⇒

$f(R_0, K_1)$ = 0010 0011 0100 1010 1010 1001 1011 1011

(XOR L_0) ⇒

Mixer = 1110 1111 0100 1010 0110 0101 0100 0100

(Swapper) ⇒

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

$R_1 = \text{Mixer} = 1110\ 1111\ 0100\ 1010\ 0110\ 0101\ 0100\ 0100$

Ở round 16 ta bỏ phần Swapper và được kết quả:

$\boldsymbol{L_{16}} = 0000\ 1010\ 0100\ 1100\ 1101\ 1001\ 1001\ 0101$

$\boldsymbol{R_{16}} = 0100\ 0011\ 0100\ 0010\ 0011\ 0010\ 0011\ 0100$

Bước 4: Final Permutation (Hoán vị kết cuối)

L_{16} và R_{16} tạo thành dãy 64 bit sau đó được hoán đổi vị trí thông qua bảng hoán vị kết cuối sau:

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Kết quả sau hoán vị kết cuối chính là Ciphertext cần tìm.

(Quay trở lại ví dụ)

$L_{16}R_{16}$

$= 0000\ 1010\ 0100\ 1100\ 1101\ 1001\ 1001\ 0101\ 0100\ 0011\ 0100\ 0010\ 0011\ 0010\ 0011\ 01$

(Final Permutation) ⇒ Ciphertext

1000 0101 1110 1000 0001 0011 0101 0100 0000 1111 0000 1010 1011 0100 0000 0101

⇒ 85E813540F0AB405

RC4

Ví dụ: Giả sử ta có bản tin gốc Plaintext $P = [9\ 6\ 1\ 8]$. Sử dụng bộ khoá 4×3 bit $K = [2\ 0\ 0\ 9]$. Tìm Ciphertext với bộ khởi tạo $S = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]$

-Giải_

$S = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]$

$T = [2\ 0\ 0\ 9\ 2\ 0\ 0\ 9]$

Bước 1: Tìm hoán vị khởi tạo của S với quy luật sau:

j = 0;

for i = 0 to 7 do

j = (j + S[i] + T[i]) mod 8

Swap(S[i],S[j]);

end

T array		2	0	0	9	2	0	0	9
i	j	S0	S1	S2	S3	S4	S5	S6	S7
	0	0	1	2	3	4	5	6	7
0	2	2	1	0	3	4	5	6	7
1	3	2	3	0	1	4	5	6	7
2	3	2	3	1	0	4	5	6	7
3	4	2	3	1	4	0	5	6	7
4	6	2	3	1	4	6	5	0	7
5	3	2	3	1	5	6	4	0	7
6	3	2	3	1	0	6	4	5	7
7	3	2	3	1	7	6	4	5	0

Ta tìm được hoán vị khởi tạo $S = [2\ 3\ 1\ 7\ 6\ 4\ 5\ 0]$

Bước 2: Tạo Keystream 4×3 bit sau đó ta XOR từng bit của KeyStream với từng bit của Plaintext để tạo Ciphertext

```
i, j = 0;
while (true) {
        i = (i + 1) mod 8;
        j = (j + S[i]) mod 8;
        Swap (S[i], S[j]);
        t = (S[i] + S[j]) mod 8;
       KeyStream = S[t];
}
```

Plaintext		9		6		1		8	
Keystream		1		0		0		0	
Ciphertext		8		6		1		8	
i	j	S0	S1	S2	S3	S4	S5	S6	S7
0	0	2	3	1	7	6	4	5	0
1	3	2	7	1	3	6	4	5	0
2	4	2	7	6	3	1	4	5	0
3	7	2	7	6	0	1	4	5	3
4	0	1	7	6	0	2	4	5	3

RSA (Rivest-Shamir-Adleman)

- Là hệ mật mã hóa khóa công khai
- Thuật toán:

Bước 1: Chọn p, q là hai số nguyên tố rất lớn.

Bước 2: Tính $\phi(n) = (q-1)(p-1)$ với $n = p.q$

Bước 3: Chọn một số ngẫu nhiên e thỏa mãn: $1 < e < \phi(n)$ và $\gcd(\phi(n), e) = 1$

Bước 4: Tìm d sao cho $d \times e = 1 (mod\ \phi(n))$

- Mật mã hóa: $C = P^e mod\ n$
- Giải mật mã hóa: $P = C^d mod\ n$

Note: Sử dụng giải thuật Square & Multiply để mật mã hóa và giải mật mã hóa.

Private key: (n, d) dùng để giải mật mã hóa.

Public key: (n, e) dùng để mật mã hóa.

Ví dụ:

$p = 23;\ q = 67$

Tính được: $n = p.q = 1541;\ \phi(1541) = 66 * 22 = 1452$

Chọn $e = 5$ do $gcd(5,1452) = 1$

Tính được $d = 581$ (sử dụng thuật toán Euclidean mở rộng)

Giải mật mã hóa: $P = 401^{581}\ (mod\ 1541)$

Ta có $d = 581 = 1001000101$. Khởi tạo $P1 = 1$

bit	P_1	$P_1 = P_1^2\ mod\ 1541$	$P_1 = P_1\ if\ bit$ "0" $P_1 = P_1 * 401\ (mod\ 1541)\ if\ bit$ "1"
1	1	1	401
0	401	537	537
0	537	202	202
1	202	738	66

0	66	1274	1274
0	1274	403	403
0	403	604	604
1	604	1140	1004
0	1004	202	202
1	202	738	66 = P

Knap Sack

Private key: là một chuỗi siêu tăng

Permutation table (dùng để hoán đổi vị trí các giá trị của Public Key)

Chọn n với điều kiện n lớn hơn tổng các giá trị trong chuỗi siêu tăng; sau đó chọn r với $gcd(r, n) = 1$

$$Public\ key\ =\ Private\ key * r\ (mod\ n)$$

Ví dụ:

$$Private\ key = [7, 11, 19, 39, 79, 157, 313].$$

Chọn $n\ =\ 900;\ r\ =\ 37$

$$Permutation\ table = [4\ 3\ 5\ 2\ 1\ 7\ 6]$$

$$Public\ key\ =\ Private\ key\ x\ 37\ (mod\ 900)\ =\ [259\ 407\ 703\ 543\ 223\ 409\ 781]$$

Sau khi qua permutation table ta tính được $Public\ key\ =\ [543\ 407\ 223\ 703\ 259\ 781\ 409]$

Plaintext: "g" $\Rightarrow$ $(ASCII)$ 1100111

- Encrypt: $C\ =\ 1\ \times\ 543\ +\ 1\ \times\ 407\ +\ 0\ \times\ 223 + \cdots +\ 1 \times\ 409\ =\ 2399$
- Decrypt: Tìm phần tử nghịch đảo nhân của r trong tập Z_n

Cụ thể: Nghịch đảo nhân của 37 trong tập Z_{900} là 73

$2399\ x\ 73\ (mod\ 900)\ =\ 527\ =\ 7\ +\ 11\ +\ 39\ +\ 157\ +\ 313$ (Tách thành tổng của các phần tử trong Private key)

$$\Rightarrow\ 1101011\ \Rightarrow\ (Permutation\ table)\ \Rightarrow\ 1100111 \Rightarrow\ "g"$$

Diffie-Hellman key change
(Trao đổi khóa DH)

Concepts

- Phương thức trao đổi khóa DH là một phương thức trao đổi thông tin công khai qua một môi trường mạng không an toàn nhưng phục vụ mục đích để các bên tham gia cùng nắm được khóa bí mật.
- Trao đổi khóa DH không phải là thuật toán để mật mã hóa dữ liệu.

Phương thức thực hiện

- Chọn một số nguyên tố rất lớn p (thường là 1024 bits)
- Chọn một phần tử nguyên tố g

$$g^{ab}(mod\ p) = g^{b^a}(mod\ p) = g^{ab}(mod\ p)$$

Bước 1:

- Alice tạo ra một giá trị ngẫu nhiên bí mật a sau đó tính $g^a(mod\ p)$ để gửi cho Bob.
- Bob tạo ra một giá trị ngẫu nhiên bí mật b sau đó tính $g^b(mod\ p)$ để gửi cho Alice.

Bước 2: $g^{ab}(mod\ p) = g^{b^a}(mod\ p) = g^{ab}(mod\ p)$ là shared session key.

Ví dụ:

Chọn $p = 353;\ g = 3$

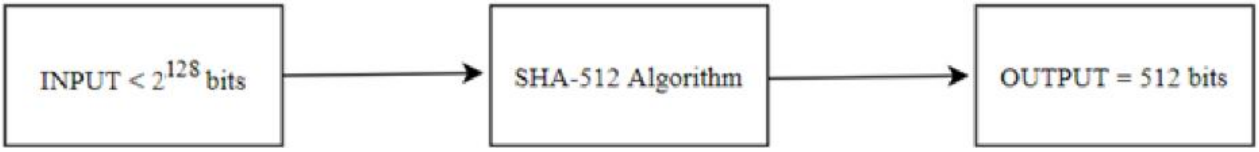
Alice chọn $a = 97 \Longrightarrow$ gửi $3^{97}(mod\ 353) = 40$ cho Bob

Bob chọn $b = 233 \Longrightarrow$ gửi $3^{233}(mod\ 353) = 248$ cho Alice

$\Longrightarrow$ Shared session key = $40^{233}(mod\ 353) = 248^{97}\ (mod\ 353) = 160$

SHA-512

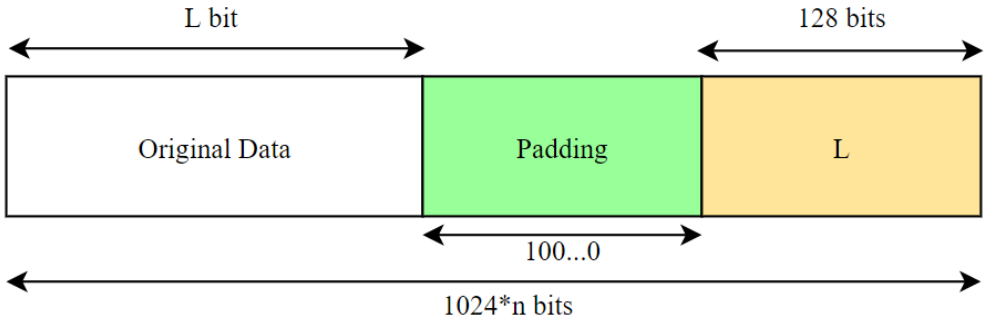
Cấu trúc hàm băm SHA-512



Hình 5.1. Sơ đồ mật mã hóa dữ liệu sử dụng hàm băm SHA- 512.

Nhìn vào sơ đồ ta có thể dễ dàng thấy được: thuật toán nhận dữ liệu đầu vào có độ dài bất kỳ nhỏ hơn 2^{128} bits và trả về kết quả đầu ra là 512 bits.

Dữ liệu đầu vào hàm băm



Hình 5.2. Dữ liệu đầu vào hàm băm

Dữ liệu đầu vào hàm băm là một số nguyên lần 1024 bits, gồm 3 phần được thể hiện như trên Hình 5.2.

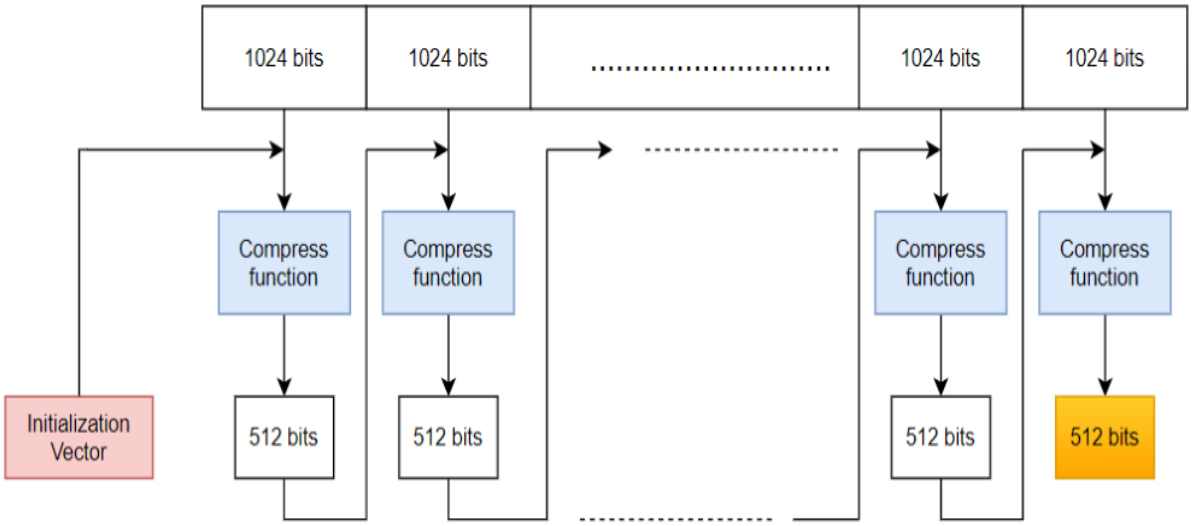
- Phần thứ nhất: là bản tin gốc cần mật mã hóa (Original Data).
- Phần thứ ba: gồm 128 bits, thể hiện độ dài của bản tin gốc.
- Phần thứ hai: phần được thêm vào để độ dài của dữ liệu đầu vào hàm băm thỏa mãn là một số nguyên lần 1024 bits. Phần này được bắt đầu bởi bit 1 và bit 0 phía sau.

Ví dụ: Bản tin gốc là “HiRose”.

Ta nhận thấy bản tin gốc có độ dài là 48 bits $=> L = 48 = 00\ldots00110000$

Padding length $= (-L - 128) mod 1024 = (-48 - 128) mod 1024 = 848$ bits

Sơ đồ thuật toán SHA-512



Hình 5.3. Sơ đồ thuật toán SHA-512

Thuật toán SHA-512 tuân theo lưu đồ Merkle-Damgard như thể hiện trên Hình 5.3.

Nhìn vào sơ đồ hình vẽ ta có cái nhìn tổng quát về thuật toán SHA-512: Dữ liệu đầu vào được chia thành n khối, mỗi khối 1024 bits. Khối 1024 bits đầu tiên kết hợp với vector khởi tạo ban đầu 512 bits là đầu vào của hàm nén. Hàm này có nhiệm vụ chuyển những dữ liệu đầu vào thành 512 bits đầu ra. Công việc này được lặp lại đến hết khối 1024 bits thứ n. Giá trị 512 bits sau hàm nén cuối cùng chính là dữ liệu đầu ra của quá trình mật mã hóa sử dụng hàm băm SHA-512.

Vector khởi tạo ban đầu là chuỗi 512 bits, được chia thành 8 words như sau:

a	b	c	d	e	f	g	h

Với:

a = 0x6A09E667F3BCC908

b = 0xBB67AE8584CAA73B

c = 0x3C6EF372FE94F82B

d = 0xA54FF53A5F1D36F1

e = 0x510E527FADE682D1

f = 0x9B05688C2B3E6C1F

g = 0x1F83D9ABFB41BD6B

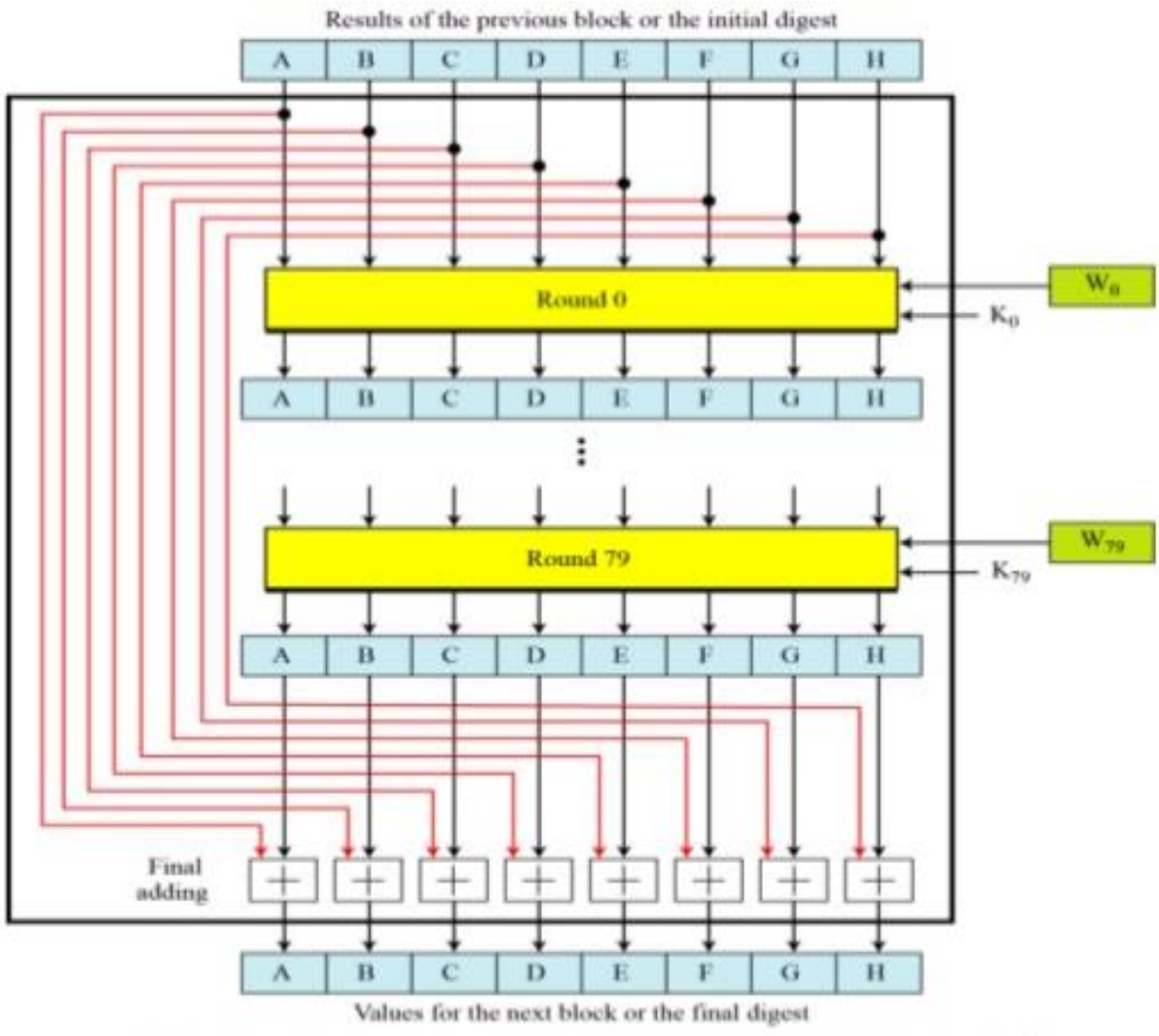
h = 0x5BE0CD19137E2179

Sơ đồ hàm nén (Compress Function)

Như đã trình bày ở những phần trên, số hàm nén tương ướng với số khối 1024 bits. Đầu vào mỗi hàm nén gồm:

- 512 bits được khởi tạo ban đầu hoặc kết quả đầu ra của hàm nén trước đó
- Khối 1024 bits thứ i tương ứng với hàm nén thứ i

Một hàm nén gồm 80 rounds. Mỗi round có nhiệm vụ chuyển khối 512 bits hay 8 words đầu vào thành chuỗi 512 bits hay 8 words đầu ra khác thông qua các giá trị Wi và Ki tương ứng như thể hiện ở Hình 5.4.



Hình 5.4. Sơ đồ hàm nén

Kết quả đầu ra của round thứ 80 được cộng modulo 2 với giá trị đầu vào round thứ nhất để thu được kết quả 8 words sau cùng là Hash Value thứ i.

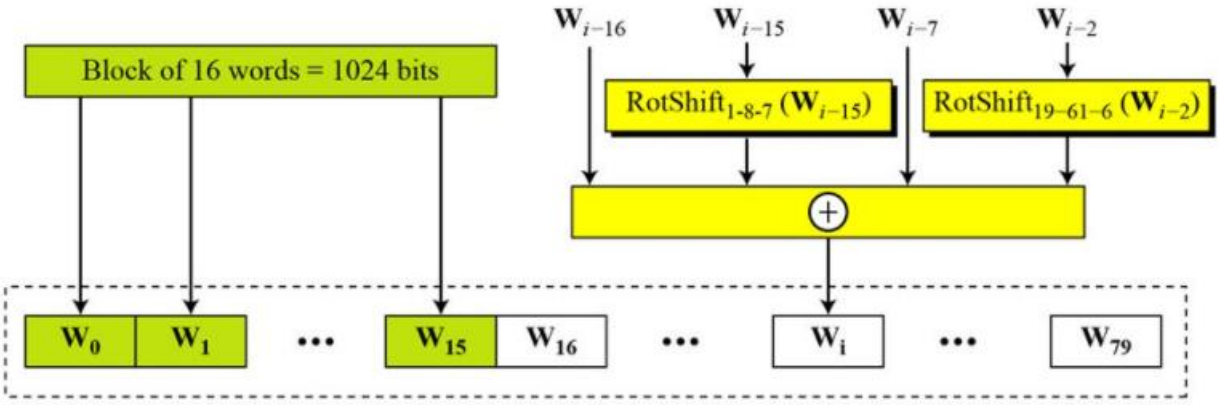
Các giá trị K_i là các hằng số được định nghĩa trước. Cụ thể được thể hiện trên Bảng 5.1 như sau:

Bảng 5.1. Bảng giá trị khởi tạo hằng số vòng K

K0	428a2f98d728ae22	K27	bf597fc7beef0ee4	K54	5b9cca4f7763e373
K1	7137449123ef65cd	K28	c6e00bf33da88fc2	K55	682e6ff3d6b2b8a3
K2	b5c0fbcfec4d3b2f	K29	d5a79147930aa725	K56	748f82ee5defb2fc
K3	e9b5dba58189dbbc	K30	06ca6351e003826f	K57	78a5636f43172f60
K4	3956c25bf348b538	K31	142929670a0e6e70	K58	84c87814a1f0ab72
K5	59f111f1b605d019	K32	27b70a8546d22ffc	K59	8cc702081a6439ec
K6	923f82a4af194f9b	K33	2e1b21385c26c926	K60	90befffa23631e28
K7	ab1c5ed5da6d8118	K34	4d2c6dfc5ac42aed	K61	a4506cebde82bde9
K8	d807aa98a3030242	K35	53380d139d95b3df	K62	bef9a3f7b2c67915
K9	12835b0145706fbe	K36	650a73548baf63de	K63	c67178f2e372532b
K10	243185be4ee4b28c	K37	766a0abb3c77b2a8	K64	ca273eceea26619c
K11	550c7dc3d5ffb4e2	K38	81c2c92e47edaee6	K65	d186b8c721c0c207
K12	72be5d74f27b896f	K39	92722c851482353b	K66	eada7dd6cde0eb1e
K13	80deb1fe3b1696b1	K40	a2bfe8a14cf10364	K67	f57d4f7fee6ed178
K14	9bdc06a725c71235	K41	a81a664bbc423001	K68	06f067aa72176fba
K15	c19bf174cf692694	K42	c24b8b70d0f89791	K69	0a637dc5a2c898a6
K16	e49b69c19ef14ad2	K43	c76c51a30654be30	K70	113f9804bef90dae
K17	efbe4786384f25e3	K44	d192e819d6ef5218	K71	1b710b35131c471b
K18	0fc19dc68b8cd5b5	K45	d69906245565a910	K72	28db77f523047d84
K19	240ca1cc77ac9c65	K46	f40e35855771202a	K73	32caab7b40c72493
K20	2de92c6f592b0275	K47	106aa07032bbd1b8	K74	3c9ebe0a15c9bebc
K21	4a7484aa6ea6e483	K48	19a4c116b8d2d0c8	K75	431d67c49c100d4c

K22	5cb0a9dcbd41fbd4	K49	1e376c085141ab53	K76	4cc5d4becb3e42b6
K23	76f988da831153b5	K50	2748774cdf8eeb99	K77	597f299cfc657e2a
K24	983e5152ee66dfab	K51	34b0bcb5e19b48a8	K78	5fcb6fab3ad6faec
K25	a831c66d2db43210	K52	391c0cb3c5c95a63	K79	6c44198c4a475817
K26	b00327c898fb213f	K53	4ed8aa4ae3418acb		

Các giá trị Wi được tạo từ khối 1024 bits hay 16 words theo sơ đồ Hình 5.5



Hình 5.5. Sơ đồ khối tạo giá trị Wi

Theo sơ đồ hình vẽ, các giá trị W_0 đến W_{15} tương ứng với từng word trong khối 1024 bits, các giá trị từ W_{16} đến W_{79} được tính theo công thức:

$W_i = W_{i-16}\ XOR\ \text{RotShift}_{1-8-7}(W_{i-15})\ XOR\ W_{i-7}\ XOR\ W_{i-2}$ với $16 \leq i \leq 79$

Trong đó:

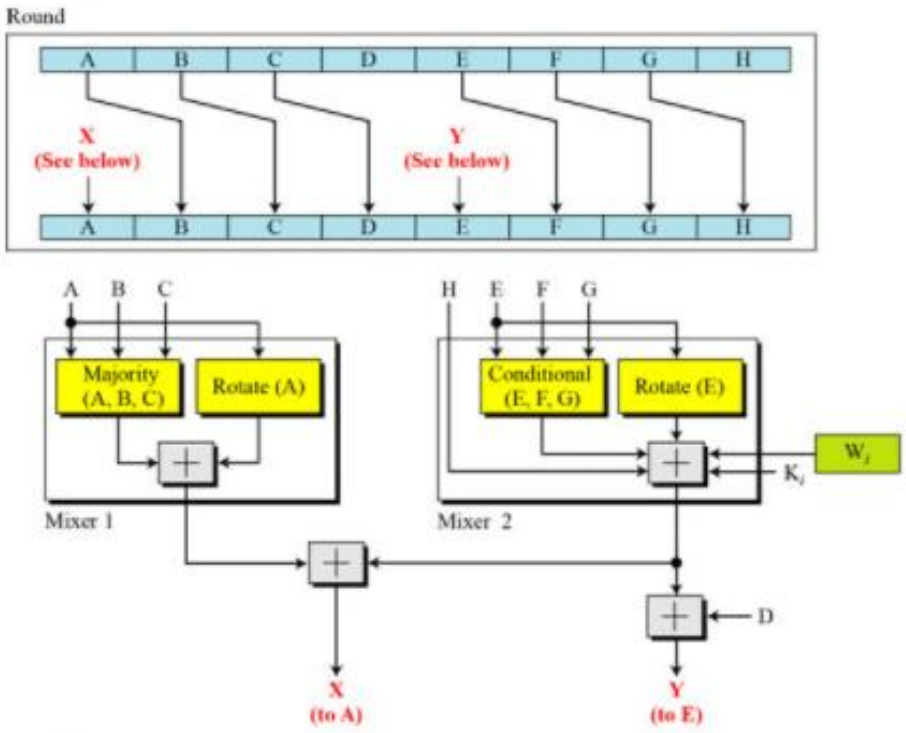
$$\text{RotShift}_{1-m-n}(x)\text{: RotR}_1(x) \oplus \text{RotR}_m \oplus \text{ShR}_n(x)$$

$\text{RotR}_i(x)$: Xoay phải phần tử x bởi i bit

$\text{ShR}_n(x)$: Dịch phải tham số x theo i bit

Cấu trúc mỗi round trong hàm nén

Như đã trình bày ở phần trước, mỗi hàm nén gồm 80 rounds, mỗi round có cấu trúc như Hình 5.6



Hình 5.6. Cấu trúc mỗi round trong hàm nén

Trong đó:

Major Function:

$$(A_j\ AND\ B_j) \oplus (B_j\ AND\ C_j) \oplus (C_j\ AND\ A_j)$$

Conditional Function:

$$(E_j\ AND\ F_j) \oplus (NOT\ E_j\ AND\ G_j)$$

Rotate Functions:

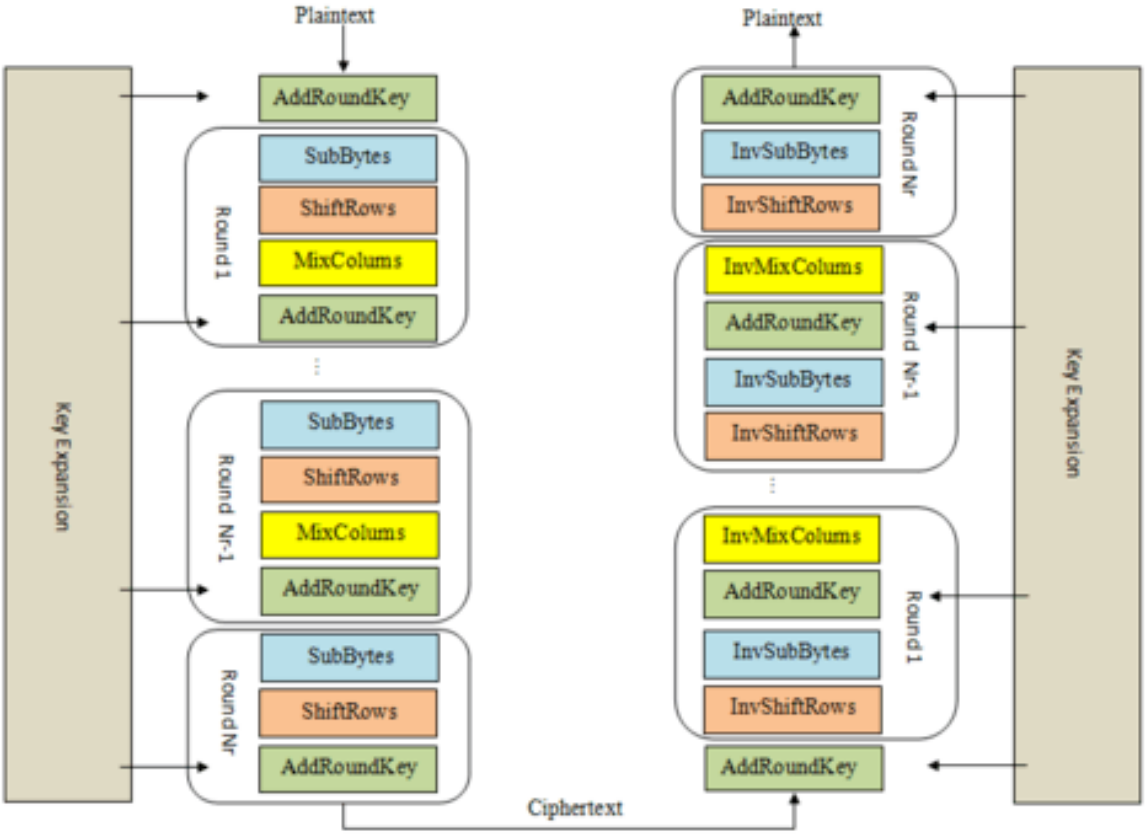
$$\text{Rotate}(A): \text{RotR}_{28}(A) \oplus \text{RotR}_{34}(A) \oplus \text{RotR}_{29}(A)$$

$$\text{Rotate}(E): \text{RotR}_{28}(E) \oplus \text{RotR}_{34}(E) \oplus \text{RotR}_{29}(E)$$

AES

- Là bộ mật mã hoá khối có khoá đối xứng.
- Block cipher = 128 bit
- Key length= 128bit / 192bit / 256bit
- #Round = 10 / 12 / 14.

Cấu trúc bộ mật AES



Plaintext đầu vào là 128 bit hay 16 byte được chuyển đổi thành khối trạng thái (ma trận 4x4).

9 round đầu gồm 4 loại chuyển đổi, round cuối cùng có 3 loại chuyển đổi.

SubBytes/ InvSubBytes: Có 2 cách (tra bảng hoặc tính toán trong GF(2^8))

Cách 1: Tra bảng

Substitution

- **SubBytes**

Example

14 → Row 1, Column 4 → FA

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

SubBytes Table

Substitution

- **Inverse SubBytes**

Example

FA → Row F, Column A → 14

	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

InvSubBytes Table

Cách 2: Tính toán trong trường $GF(2^8)$

Ví dụ 1: SubBytes của $0C$

- Bước 1: Tìm nghịch đảo nhân của 0C trong trường $GF(2^8)$ sử dụng thuật toán Euclidean mở rộng. $0C = (00001100)_2 = x^3 + x^2$

q	r_1	r_2	r	t_1	t_2	t
$x^5+x^4+x^3+x^2+1$	$x^8+x^4+x^3+x+1$	x^3+x^2	x^2+x+1	0	1	$x^5+x^4+x^3+x^2+1$
x	x^3+x^2	x^2+x+1	x	1	$x^5+x^4+x^3+x^2+1$	$x^6+x^5+x^4+x^3+x+1$
x	x^2+x+1	x	$x+1$	$x^5+x^4+x^3+x^2+1$	$x^6+x^5+x^4+x^3+x+1$	$x^7+x^6+x^3+x+1$
1	x	$x+1$	1	$x^6+x^5+x^4+x^3+x+1$	$x^7+x^6+x^3+x+1$	$x^7+x^5+x^4$
$x+1$	$x+1$	1	0	$x^7+x^6+x^3+x+1$	$x^7+x^5+x^4$	$x^8+x^4+x^3+x+1$
	1	0		$x^7+x^5+x^4$	$x^8+x^4+x^3+x+1$	

Multiplicative inverse of 0C ~ B0 (1011 0000)

Kết quả ta tìm được nghịch đảo nhân của $0C$ trong trường $GF(2^8)$ là $b = 1011\ 0000$

- Bước 2: Nhân ma trận b (ở dạng cột) tìm được ở Bước 1 với một ma trận cố định theo công thức:

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

$b = 1011\ 0000 = [b7\ b6\ b5\ b4\ b3\ b2\ b1\ b0]$

Sau Bước 2 ta tính được $s = [s7\ s6\ s5\ s4\ s3\ s2\ s1\ s0] = [1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ ]$

- Bước 3: XOR s tìm được ở trên với giá trị cố định $y = [0\ 1\ 1\ 0\ 0\ 0\ 1\ 1]$ ta thu được giá trị SubBytes của 0C

$$[1\ 0\ 0\ 1\ 1\ 1\ 0\ 1]\ XOR\ [0\ 1\ 1\ 0\ 0\ 0\ 1\ 1] = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 0] = FE$$

Ví dụ 2: Tìm đầu ra của phép biến đổi Inv Subbyte với đầu vào là 10110101 bằng biến đổi đại số trong trường $GF(2^8)$.

Ngược lại với SubByte:

- Bước 1: XOR giá trị đầu vào với giá trị cố định y = [0 1 1 0 0 0 1 1] thu được giá trị s

S = [1 0 1 1 0 1 0 1] XOR [0 1 1 0 0 0 1 1] = [1 1 0 1 0 1 1 0] = [s7 …. s0]

- Bước 2: Nhân ma trận s (ở dạng cột) tìm được ở Bước 1 với ma trận cố định theo công thức:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

$$B = [b7 \ldots b0] = [1\ 0\ 1\ 0\ 1\ 1\ 1\ 0] = x^7 + x^5 + x^3 + x^2 + x = AE$$

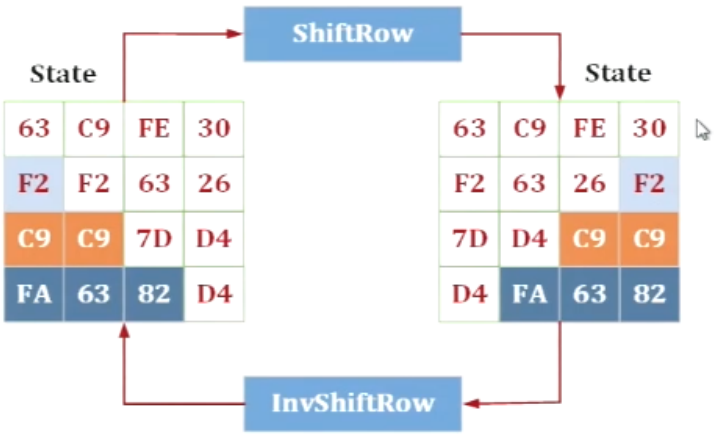
- Bước 3: Tìm nghịch đảo nhân của AE trong trường $GF(2^8)$

q	r1	r2	r	t1	t2	t
x	$x^8 + x^4 + x^3 + x + 1$	$x^7 + x^5 + x^3 + x^2 + x$	$x^6 + x^2 + x + 1$	0	1	x
x	$x^7 + x^5 + x^3 + x^2 + x$	$x^6 + x^2 + x + 1$	x^5	1	x	$1 + x^2$
x	$x^6 + x^2 + x + 1$	x^5	$x^2 + x + 1$	x	$1 + x^2$	x^3
$x^3 + x^2 + 1$	x^5	$x^2 + x + 1$	$x + 1$	$1 + x^2$	x^3	$x^6 + x^5 + x^3 + x^2 + 1$
x	$x^2 + x + 1$	$x + 1$	1	x^3	$x^6 + x^5 + x^3 + x^2 + 1$	$x^7 + x^6 + x^4 + x$
$x + 1$	$x + 1$	1	0	$x^6 + x^5 + x^3 + x^2 + 1$	$x^7 + x^6 + x^4 + x$	

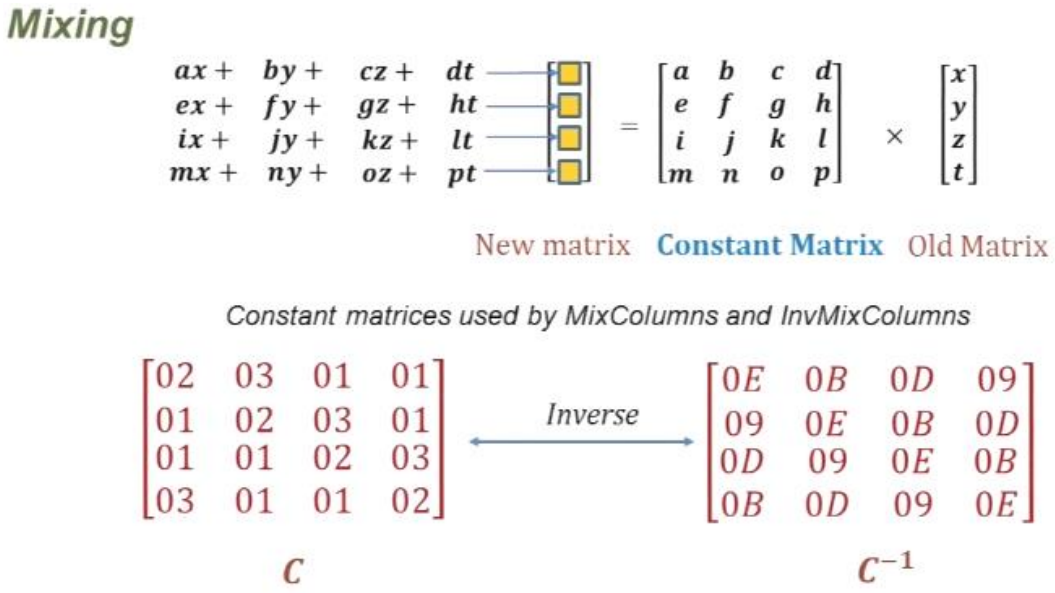
Ta tìm được nghịch đảo nhân của AE trong trường GF(2^8) là:

$$x^7 + x^6 + x^4 + x = 11010010 = D2$$

ShiftRows/InvShiftRows:

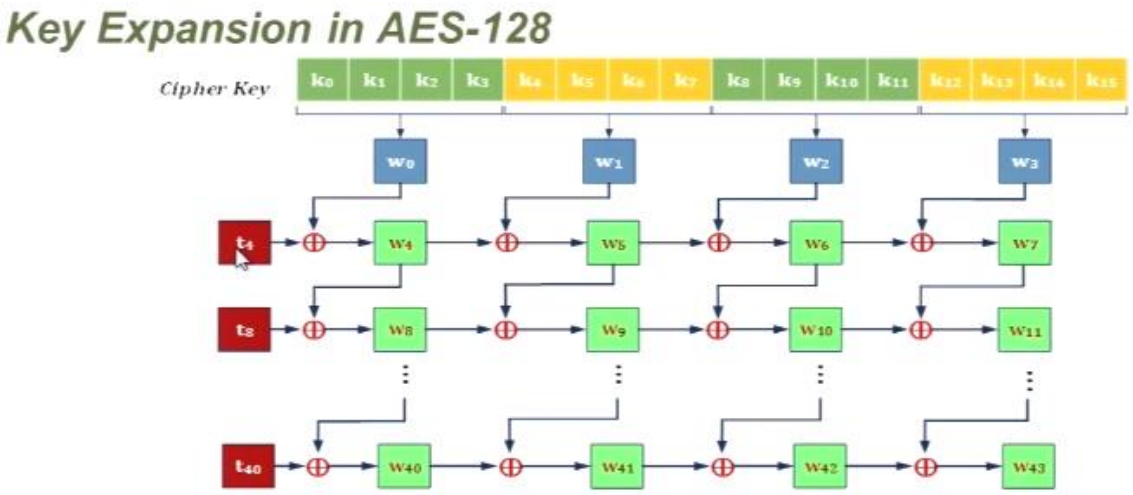


MixColumns/InvMixColumns:

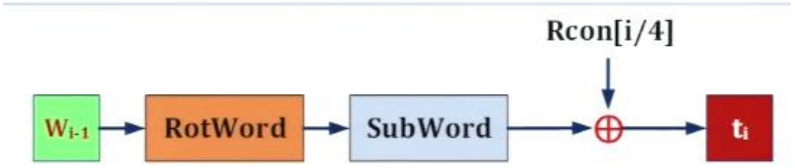


Mở rộng khoá:

Tương tự như Plaintext, cipher key ban đầu là 128 bit hay 16 byte được chuyển đổi thành khối trạng thái. Việc mở rộng khoá được thực hiện như sơ đồ sau:



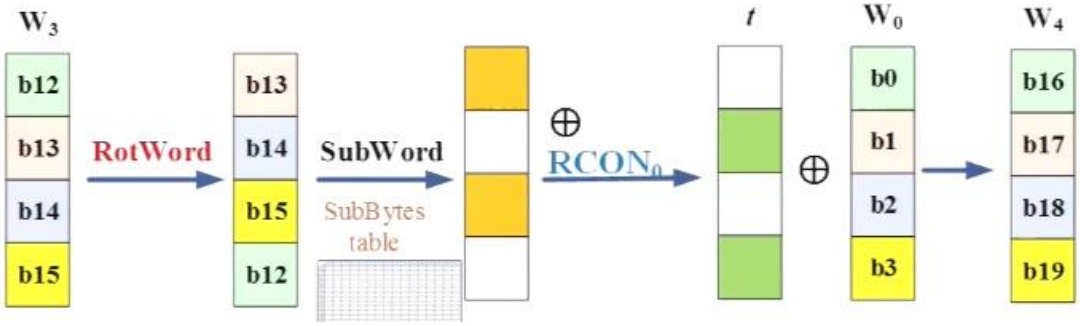
Các giá trị t_4 đến t_{40} được tính theo công thức:



RCON Constant (Hexa)

Round 1 $RCON_0$	*Round 2* $RCON_1$	*Round 3* $RCON_2$	*Round 4* $RCON_3$	*Round 5* $RCON_4$	*Round 6* $RCON_5$	*Round 7* $RCON_6$	*Round 8* $RCON_7$	*Round 9* $RCON_8$	*Round 10* $RCON_9$
01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Ví dụ:



Plaintext: ONELOVEONEFUTURE. Sử dụng phương pháp AES, hãy thực hiện:

- Chuyển đổi thành khối trạng thái
- Phép thay thế (SubBytes) với khối trạng thái tại bước 1.
- Phép hoán vị (ShiftRows) với khối trạng thái tại bước 2.
- Phép nhân: $(03)_{16} \times (83)_{16}$ với đa thức tối giản: $x^8 + x^4 + x^3 + x + 1$.

BÀI GIẢI

Sử dụng bảng mã ASCII để chuyển Plaintext thành khối trạng thái:

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Ch
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Plaintext

4F	4F	4E	54
4E	56	45	55
45	45	46	52
4C	4F	55	45

SubBytes

84	84	2F	20
2F	B1	6E	FC
6E	6E	5A	00
29	84	FC	6E

ShiftRows

84	84	2F	20
B1	6E	FC	2F
5A	00	6E	6E
6E	29	84	FC