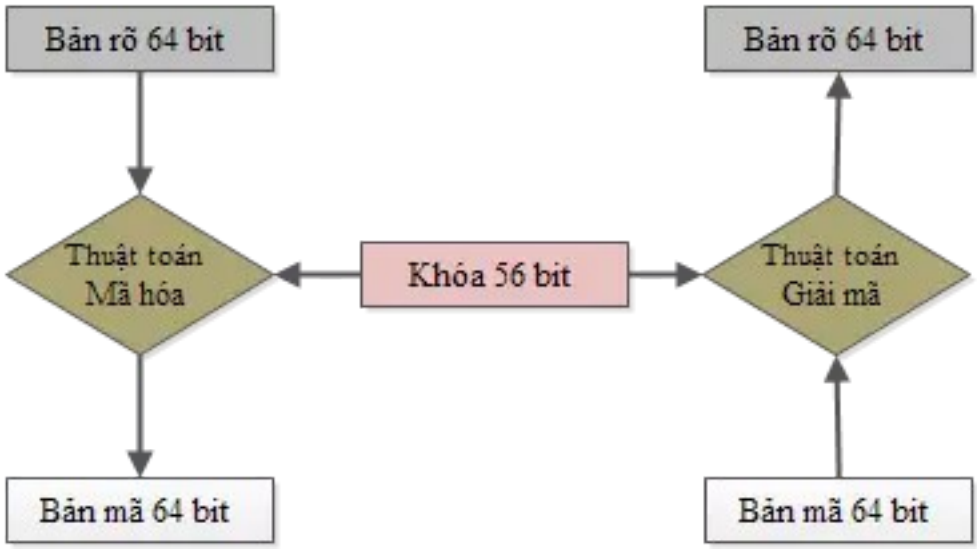
1. Tổng quan về DES

DES (Data Encryption Standard) là chuẩn mã hóa dữ liệu đầu tiên trên thế giới, do Cơ quan an ninh Quốc gia Hoa Kỳ (NSA) đề xuất trên cơ sở cải tiến thuật toán Lucifer do hãng IBM công bố năm 1964. DES đã được sử dụng rộng rãi ở Hoa Kỳ và nhiều quốc gia khác trong các thập kỷ 70, 80, 90 cho đến khi được thay thế bởi Tiêu chuẩn mã hóa dữ liệu tiên tiến AES (Advanced Encryption Standard) vào năm 2002.



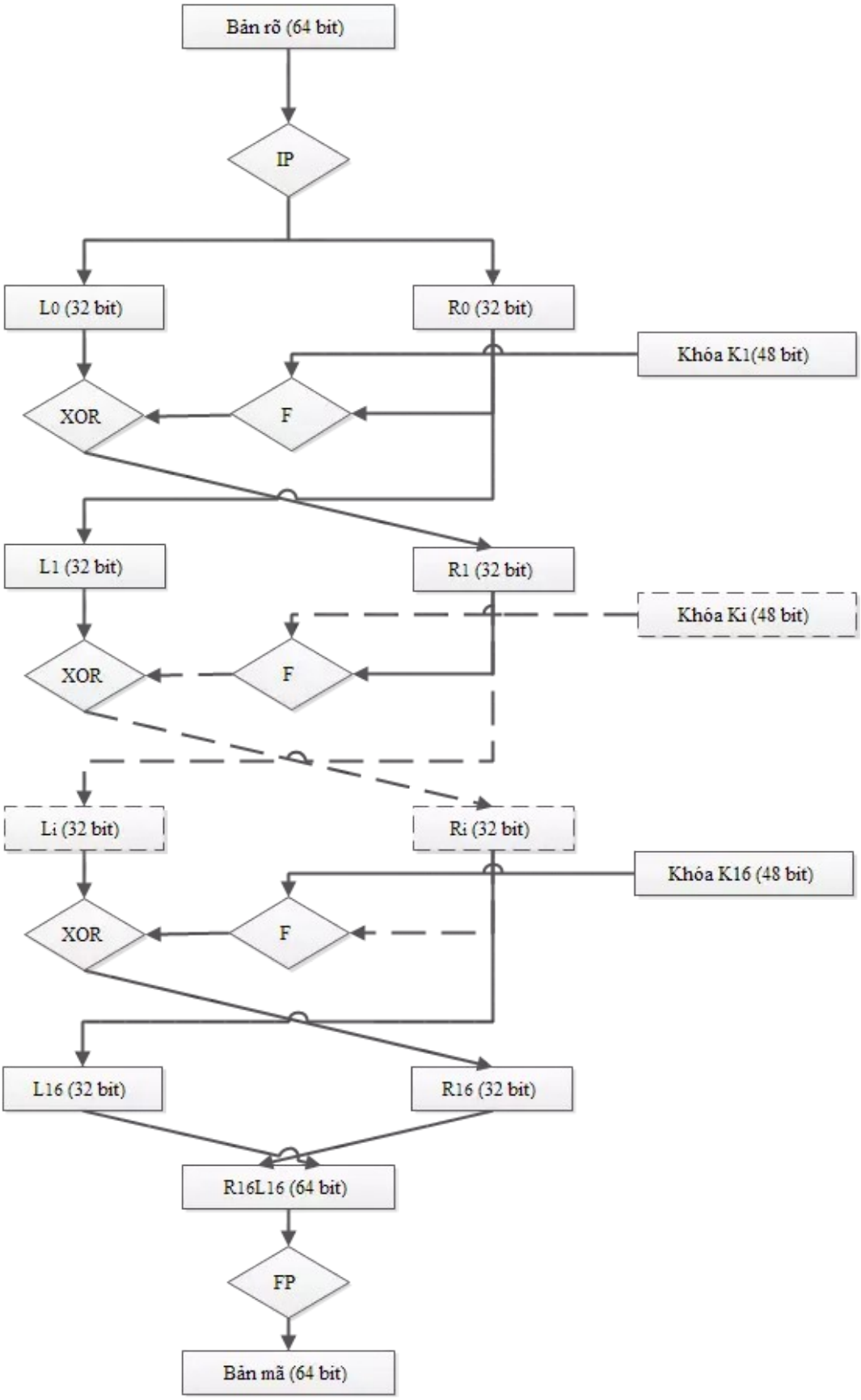
Đầu vào của DES là khối 64 bit, đầu ra cũng là khối 64 bit. Khóa mã hóa có độ dài 56 bit, nhưng thực chất ban đầu là 64 bit, được lấy đi các bit ở vị trí chia hết cho 8 dùng để kiểm tra tính chẵn lẻ.

2. Thuật toán

DES là thuật toán mã hóa theo khối, nó xử lý từng khối thông tin của bản rõ có độ dài xác định là 64 bit. Trước khi đi vào 16 chu trình chính, khối dữ liệu cần bảo mật sẽ được tách ra thành từng khối 64 bit, và từng khối 64 bit này sẽ lần lượt được đưa vào 16 vòng mã hóa DES để thực hiện. **Input:** Bản rõ M = m1m2...m64 là một khối 64 bit, khóa 64 bit K = k1k2...k64. **Output:** Bản mã 64 bit C = c1c2... c64

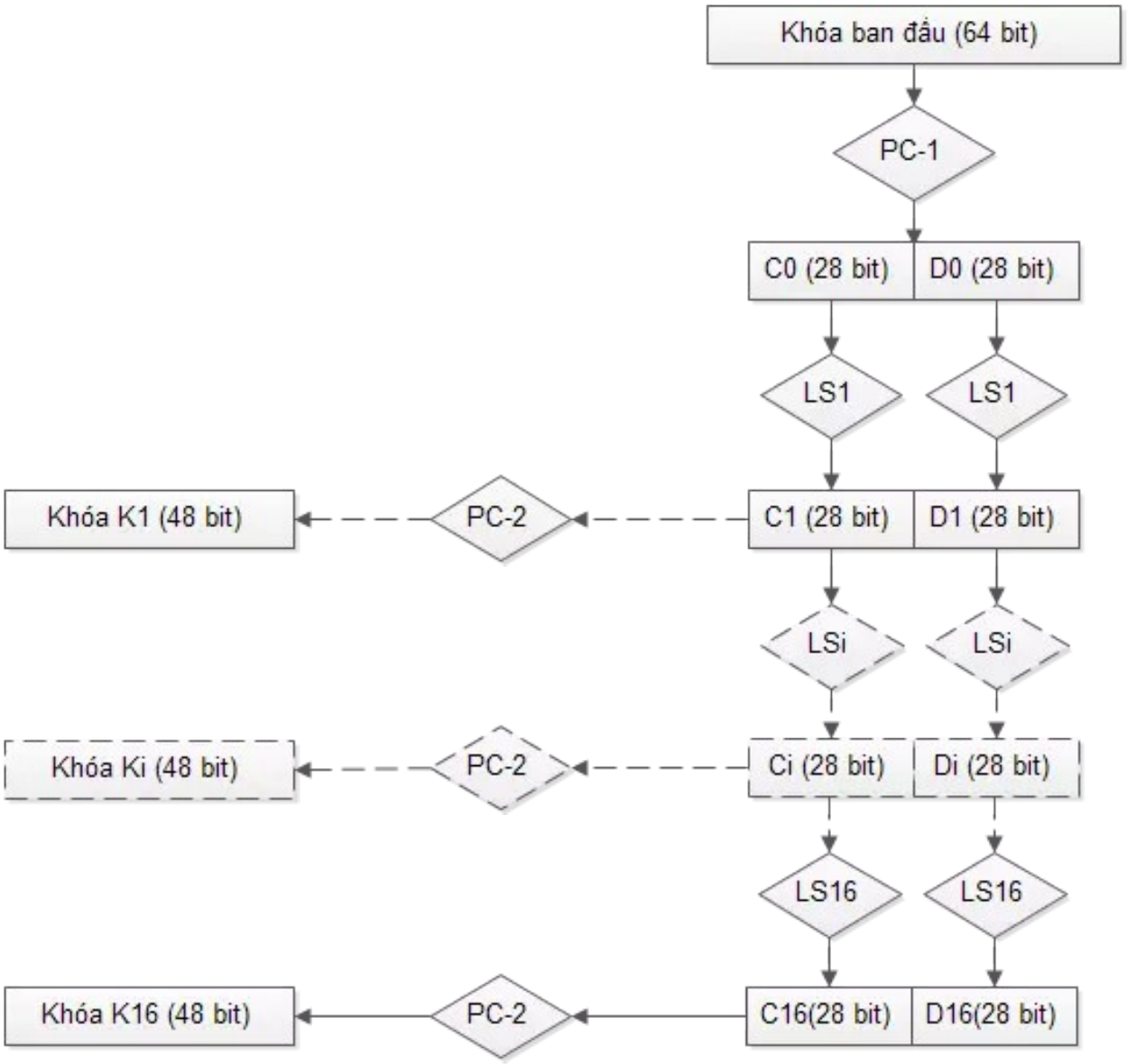
- Bước 1: Sinh khóa con: Sử dụng thuật toán sinh khóa con từ khóa K ta sẽ được 16 khóa con K1, K2, ... K16

- Bước 2: Sử dụng phép hoán vị khởi đầu IP (Initial Permutation) để hoán vị các bit của M, kết quả nhận được chia thành 2 nửa là L0 = m63m62...m32, R0 = m31m30...m0.
- Bước 3: Với i chạy từ i = 1 đến 16 thực hiện: Tính các Li và Ri theo công thức: Li = Ri-1 Ri = Li-1 XOR f(Ri-1, Ki) trong đó f(Ri-1, Ki) = R(S(E(Ri-1) XOR Ki)). Việc tính f(Ri-1, Ki) sẽ được trình bày chi tiết ở phần sau.
- Bước 4: Đổi vị trí khối L16, R16 ta được khối R16L16 = b1b2...b64.
- Bước 5: Sử dụng phép hoán vị kết thúc FP(Final Permutation – nghịch đảo với hoán vị khởi đầu IP) ta thu được bản mã cần tìm : C = IP-1(b1b2...b64)


Bản rõ (64 bit)
IP
L0 (32 bit)
R0 (32 bit)
Khóa K1(48 bit)
XOR
F
L1 (32 bit)
R1 (32 bit)
Khóa Ki (48 bit)
XOR
F
Li (32 bit)
Ri (32 bit)
Khóa K16 (48 bit)
XOR
F
L16 (32 bit)
R16 (32 bit)
R16L16 (64 bit)
FP
Bản mã (64 bit)

3. Quá trình sinh khóa con

16 vòng lặp của DES chạy cùng thuật toán như nhau nhưng với 16 khóa con khác nhau. Các khóa con đều được sinh ra từ khóa chính của DES bằng thuật toán sinh khóa con.



Khóa ban đầu là 1 xâu có độ dài 64 bit, bit thứ 8 của mỗi byte sẽ được lấy ra để kiểm tra phát hiện lỗi, tạo ra chuỗi 56 bit. Sau khi bỏ các bit kiểm tra ta sẽ hoán vị chuỗi 56 bit này. Hai bước trên được thực hiện thông qua hoán vị ma trận PC-1 (Permuted choice 1).

57	49	41	33	25	17	9	1
58	50	42	34	25	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

Tiếp theo ta kết quả sau khi PC-1 thành 2 phần : C0 : 28 bit đầu. D0 : 28 bit cuối. Mỗi phần sẽ được xử lý 1 cách độc lập. Ci = LSi(Ci-1) Di = LSi(Ci-1) với $1 \leq i \leq 16$. LSi là biểu diễn phép dịch bit vòng (cyclic shift) sang trái 1 hoặc 2 vị trí tùy thuộc vào i.

Vòng lặp	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Số lần dịch trái	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Cuối cùng sử dụng hoán vị cố định PC-2 (Permuted choice 2) để hoán vị chuỗi CiDi 56 bit tạo thành khóa Ki với 48 bit.

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

4. Quá trình mã hóa DES

Chia thành 3 giai đoạn:

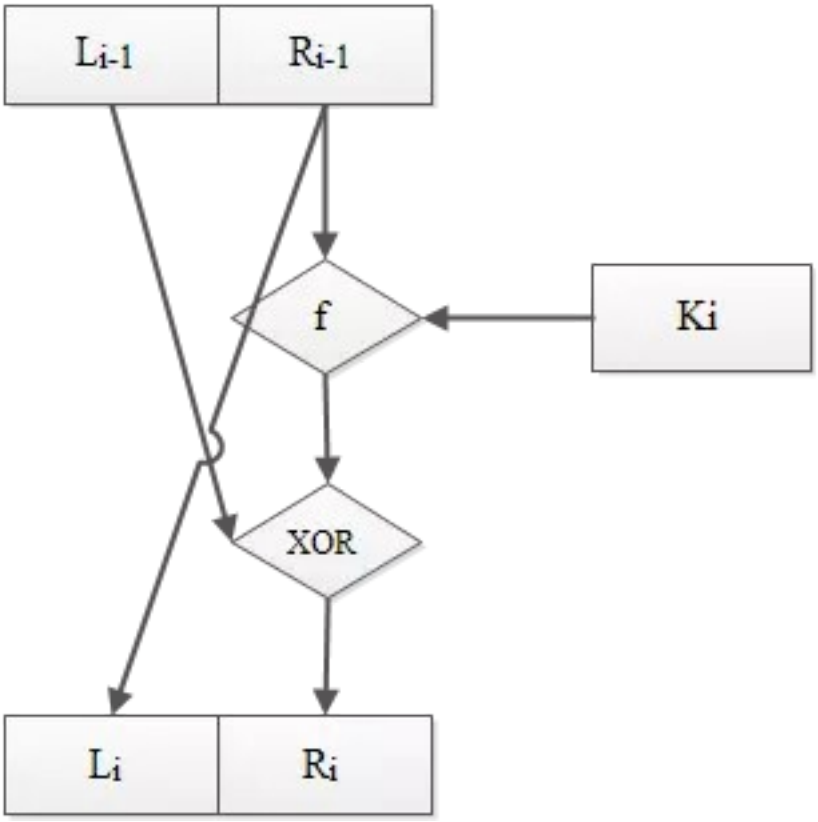
4.1. Giai đoạn 1:

Với bản rõ cho trước x, 1 xâu x’ sẽ được tạo ra bằng cách hoán vị các bit của x theo hoán vị ban đầu 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

Tiếp theo x’ sẽ được chia thành 2 phần L0,R0. x’ = IP(x) = L0R0 Trong đó L0 là 32 bit đầu, R0 là 32 bit cuối.

4..2. Giai đoạn 2:



Tính toán 16 lần bằng 1 hàm xác định. Ta sẽ tính Li, Ri ($1 \leq i \leq 16$) theo quy tắc: Li = Ri-1. Ri = Li-1 XOR f(Ri-1, Ki). Với Ki là khóa được sinh ra ở quá trình tạo khóa, f là một hàm sẽ được trình bày ở phần sau.

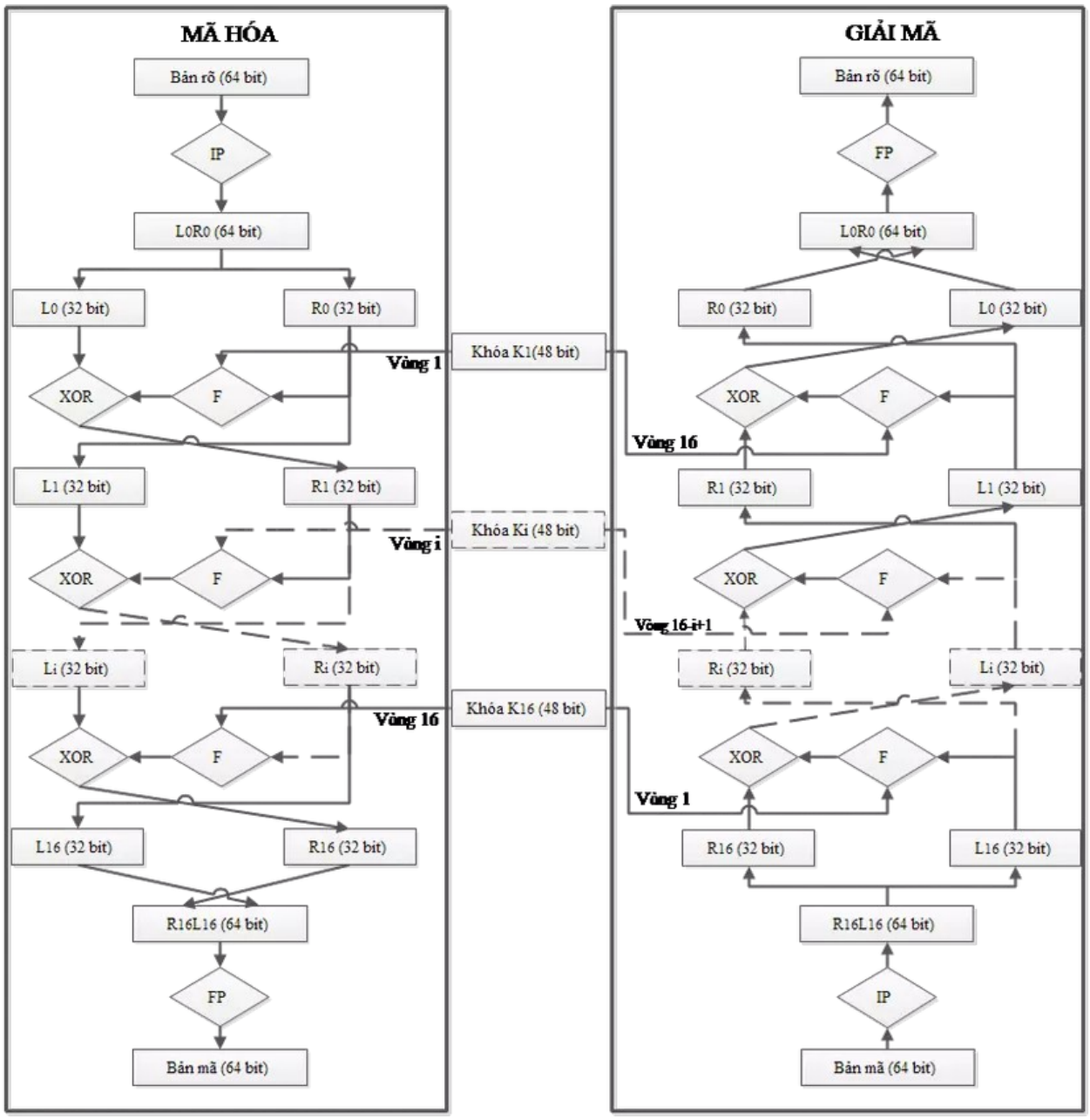
4.3. Giai đoạn 3:

Áp dụng hoán vị kết thúc FP cho xâu bit R16L16 ta thu được bản mã y: y = FP(R16L16).

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	59	17	57	25

5. Giải mã DES

Quá trình giải mã của DES cũng tương tự quá trình mã hóa. Chỉ khác nhau ở: Li = Ri-1. Ri = Li-1 XOR f(Ri-1, K16-i+1). Như vậy khóa K của hàm F sẽ đi từ khóa K16 đến khóa K1.



6. Hàm F

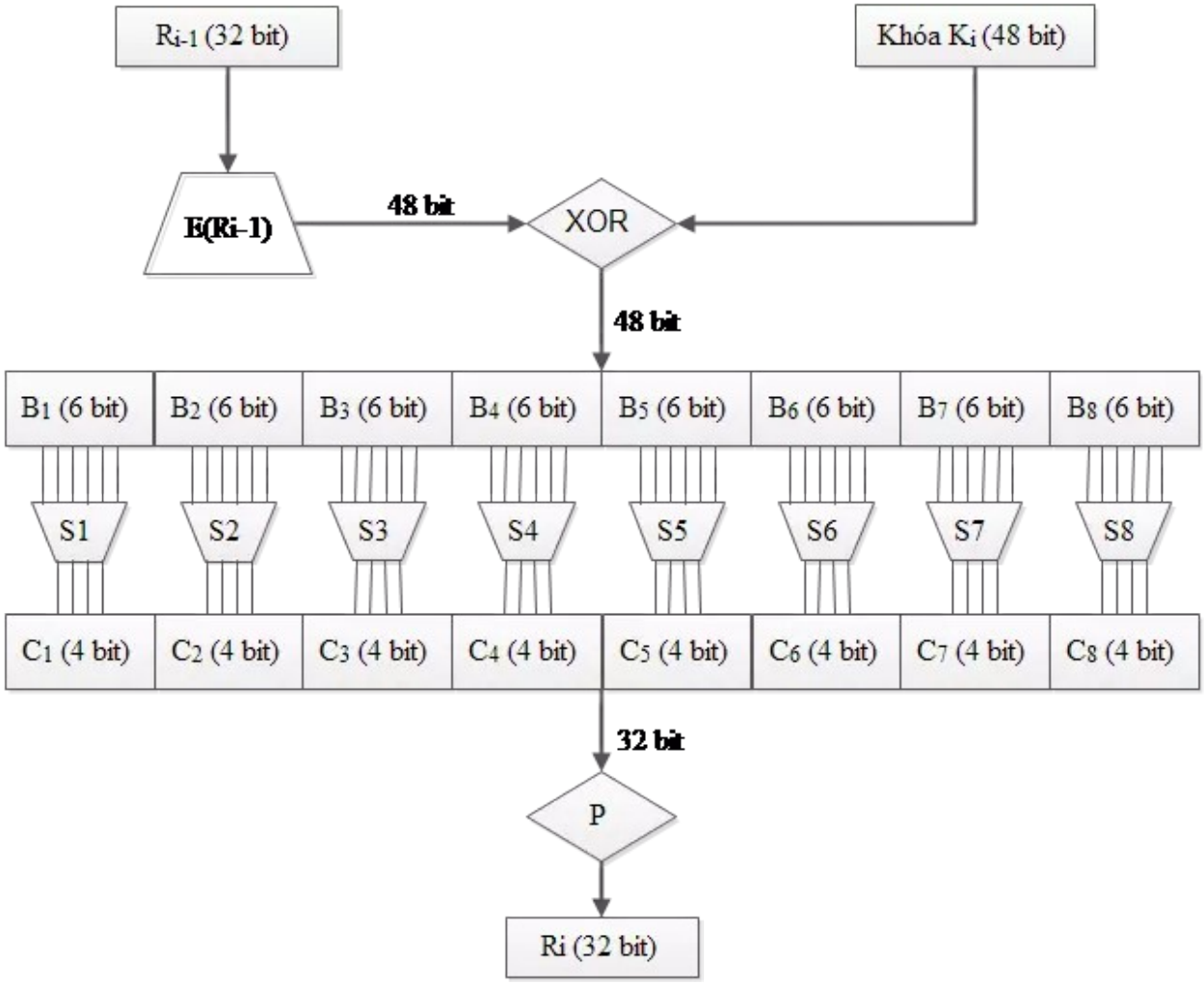
Đầu vào hàm f có 2 biến:

- Biến thứ nhất: Ri-1 là xâu bit có độ dài 32 bit.
- Biến thứ hai: Ki là xâu bit có độ dài 48 bit. Đầu ra của hàm f là xâu có độ dài 32 bit.

Quy trình hoạt động của hàm f như sau:

- Biến thứ nhất Ri-1 được mở rộng thành một xâu có độ dài 48 bit theo một hàm mở rộng hoán vị E (Expansion permutation). Thực chất hàm mở rộng E(Ri-1) là một hoán vị có lặp trong đó lặp lại 16 bit của Ri-1.
- Tính E(Ri-1) XOR Ki.
- Tách kết quả của phép tính trên thành 8 xâu 6 bit B1, B2, ..., B8.
- Đưa các khối 8 bit Bi vào 8 bảng S1, S2, ..., S8 (được gọi là các hộp S-box). Mỗi hộp S-Box là một bảng 4*16 cố định có các cột từ 0 đến 15 và các hàng từ 0 đến 3. Với mỗi xâu 6 bit Bi = b1b2b3b4b5b6 ta tính được SiBi như sau: hai bit b1b6 xác định hàng r trong hộp Si, bốn bit b2b3b4b5 xác định cột c trong hộp Si. Khi đó, Si(Bi) sẽ xác định phần tử Ci = Si(r,c), phần tử này viết dưới dạng nhị phân 4 bit. Như vậy, 8 khối 6 bit Bi ($1 \leq i \leq 8$) sẽ cho ra 8 khối 4 bit Ci với ($1 \leq i \leq 8$).
- Xâu bit C = C1C2C3C4C5C6C7C8 có độ dài 32 bit được hoán vị theo phép toán hoán vị P (hộp P-Box). Kết quả P(C) sẽ là kết quả của hàm

f(Ri-1,Ki).



6.1. Hàm mở rộng E

Hàm ở rộng E sẽ tăng độ dài Ri-1 từ 32 bit lên 48 bit bằng cách thay đổi thứ tự các bit cũng như lặp lại các bit. Việc thực hiện này nhằm hai mục đích:

- Làm độ dài của Ri-1 cùng cỡ với khóa K để thực hiện việc cộng modulo XOR.
- Cho kết quả dài hơn để có thể được nén trong suốt quá trình thay thế. Tuy nhiên, cả hai mục đích này nhằm một mục tiêu chính là bảo mật dữ liệu. Bằng cách cho phép 1 bit có thể chèn vào hai vị trí thay thế, sự phụ thuộc của các bit đầu ra với các bit đầu vào sẽ trải rộng

ra.

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

6.2. Các hộp S-box

Sau khi thực hiện phép XOR giữa E(Ri-1) và Ki, kết quả thu được chuỗi 48 bit chia làm 8 khối đưa vào 8 hộp S-box. Mỗi hộp S-Box sẽ có 6 bit đầu vào và 4 bit đầu ra. Kết quả thu được là một chuỗi 32 bit tiếp tục vào hộp P-Box.

- Mỗi hàng trong mỗi hộp S là hoán vị của các số nguyên từ 0 đến 15
- Các hộp S-box phi tuyến tính. nói cách khác, đầu ra không phải là biến đối tuyến tính của đầu vào.
- Sự thay đổi của một bit, hai bit hoặc nhiều hơn sẽ dẫn đến sự biến đổi ở đầu ra.
- Nếu hai đầu vào của một S-box bất kì chỉ khác nhau 2 bit ở giữa (bit 3 và 4) thì đầu ra sẽ khác nhau ít nhất 2 bit. Nói cách khác, S(x) và S(x XOR 001100) phải khác nhau ít nhất 2 bit.

S1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	0	14	2	13	6	15	0	9	10	4	5	3

S6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

6.3. Hộp P-box

Mỗi 4 bit đầu ra của các hộp S-box sẽ được ghép lại, theo thứ tự các hộp và được đem vào hộp P-box. Hộp P-Box đơn giản chỉ là hoán vị các bit với nhau.

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

Ví dụ:

Ta có 1 bản tin M, M = 0123456789ABCDEF và một khóa K = 13345799BBCDDFF1 với M, K được định dạng dưới dạng hệ thập lục phân, ta tiến hành mã hóa và giải mã theo giải thuật DES theo các bước sau. Ta viết lại M, K dưới dạng nhị phân , chúng ta nhận được khối 64bit của của văn bản M và khóa K M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 chia 64 bit trên thành 2 nửa trái và phải, lấy từ trái qua phải mỗi phần 32 bit

L = 0000 0001 0010 0011 0100 0101 0110 0111

R = 1000 1001 1010 1011 1100 1101 1110 1111

DES hoạt động trên các khối 64-bit sử dụng kích thước chính của 56- bit. Các khóa thực sự được lưu trữ dài 64 bit, nhưng mỗi bit thứ 8 trong khóa không được sử dụng (tức là bit số 8, 16, 24, 32, 40, 48, 56, và 64). Tuy nhiên, chúng ta sẽ vẫn đánh số các bit 1-64, đi từ trái sang phải, trong các tính toán sau đây. Tuy nhiên, ta sẽ thấy, tám bit đề cập đến được loại trừ khi chúng ta tạo ra khóa. Đưa K về dạng nhị phân:

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Thuật toán DES sử dụng các bước sau:

Bước 1: Tạo 16 khóa con, mỗi khóa có chiều dài 48 bit Khóa 64 bit được hoán vị theo bảng dưới đây, PC-1. Chỉ có 56 bit của khóa ban đầu xuất hiện trong khóa hoán vị 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

57	49	41	33	25	17	9
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Từ khóa 64 bit ban đầu: K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001 Từ bảng hoán vị trên ta thấy: bit thứ 57 của K là bit '1', thứ 49 của K là bit '1', bít thứ 41, 33, 25, 17, 9 lần lượt là 11000.... Chúng ta nhận được hoán vị 56 bit:

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Tiếp theo, chia khóa này thành 2 nửa trái và phải , C0 và D0, mỗi nửa 28 bit, ta được: C0 = 1111000 0110011 0010101 0101111 D0 = 0101010 1011001 1001111 0001111 Với C0 và D0 đã được xác định, bây giờ chúng ta tạo ra 16 khối CnDn 1<=n<=16. Mỗi cặp CnDn được hình thành từ các cặp trước nó Cn-1Dn-1 bằng cách sử dụng quy luật sau đối với khối trước nó

Vòng lặp	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Số lần dịch trái	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Điều này có nghĩa là , ví dụ C3, D3 thu được nhờ sự dịch trái 2 lần C2, D2, hay C2, D2 thu được nhờ sự dịch trái 1 lần của C1, D1 Ta đã có
C0 = 1111000 0110011 0010101 0101111
D0 = 0101010 1011001 1001111 0001111

Từ bảng quy luật trên ta thấy, C1D1 được sinh ra khi thực hiện dịch trái 1 lần đối với C0D0 ta thu được :
C1 = 111000 0110011 0010101 01011111
D1 = 101010 1011001 1001111 00011110

Tương tự ứng với bảng quy luật trên ta thu được các khối CnDn (1<=n<=16) như sau:
C2 = 11000011001100101010101111111
D2 = 01010101100110011111000111101
C3 = 00001100110010101010111111111
D3 = 01010110011001111100011110101
C4 = 00110011001010101011111111100
D4 = 01011001100111110001111010101
C5 = 11001100101010101111111110000
D5 = 01100110011111000111101010101
C6 = 00110010101010111111111000011
D6 = 10011001111000111110101010101
C7 = 11001010101011111111100001100
D7 = 01100111100011111010101010110
C8 = 00101010101111111110000110011
D8 = 10011110001111010101011001
C9 = 01010101011111111100001100110
D9 = 00111100011110101010110011
C10 = 01010101111111110000110011001
D10 = 11110001111010101011001100
C11 = 01010111111111000011001100101
D11 = 11000111101010101011001100111
C12 = 01011111111000011001100101010
D12 = 00011110101010101100110011111
C13 = 01111111110000110011001010101
D13 = 01111010101010110011001111100
C14 = 11111110000110011001001010101
D14 = 11101010101011001100111110001
C15 = 11111000011001100101010101111
D15 = 10101010101100110011111000111
C16 = 11110000110011001010101011111
D16 = 01010101011001100111100011111
Bây giờ chúng ta xây dựng các khóa Kn (1<=k<=16) bằng cách áp dụng bảng hoán vị sau cho mỗi cặp CnDn. Mỗi cặp có 56 bit, nhưng PC-2 chỉ sử dụng 48 trong số này. 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

Từ C1D1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110 Từ bảng trên ta thấy: bit thứ 14 của C1D1 là bít đầu tiên của K1 tức là bit ‘0’, bit thứ 17 là bít thứ 2 của K1, tức bit ‘0’...bít cuối cùng của K1 là bít thứ 32 của C1D1 tức là bit ‘0’

K1 = 000110 110000 001011 101111 111111 000111 000001 110010

Tương tự cho các khóa khác, ta có:
K2 = 011110 011010 111011 011001 110110 111100 100111 100101
K3 = 010101 011111 110010 001010 010000 101100 111110 011001
K4 = 011100 101010 110111 010110 110110 110011 010100 011101
K5 = 011111 001110 110000 000111 111010 110101 001110 101000
K6 = 011000 111010 010100 111110 010100 000111 101100 101111
K7 = 111011 001000 010010 110111 111101 100001 100010 111100
K8 = 111101 111000 101000 111010 110000 010011 101111 111011
K9 = 111000 001101 101111 101011 111011 011110 011110 000001
K10 = 101100 011111 001101 000111 101110 100100 011001 001111
K11 = 001000 010101 111111 010011 110111 101101 001110 000110
K12 = 011101 010111 000111 110101 100101 000110 011111 101001
K13 = 100101 111100 010111 010001 111110 101011 101001 000001

K14 = 010111 110100 001110 110111 111100 101110 011100 111010
K15 = 101111 111001 000110 001101 001111 010011 111100 001010
K16 = 110010 110011 110110 001011 000011 100001 011111 110101

Bước 2: Mã hóa 64 bit khối dữ liệu Có một hoán vị IP ban đầu của 64 bit của dữ liệu M, sắp xếp lại các bit này theo quy luật bảng dưới đây 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

Áp dụng hoán vị cho khối văn bản ban đầu M, ta nhận được:
M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Chia khối IP hoán vị thành 1 nửa trái L0 32 bit và một nửa phải R0 32bit, từ IP ta nhận được:
L0 = 1100 1100 0000 0000 1100 1100 1111 1111
R0 = 1111 0000 1010 1010 1111 0000 1010 1010

Bây giờ chúng ta tiến hành thông qua 16 vòng lặp, sử dụng một hàm f mà hoạt động trên 2 khối – một khối dữ liệu 32bit và 1 khóa Kn 48bit để sinh ra một khối 32bit Ln = Rn-1, Rn = Ln-1 f(Rn-1, Kn). For n = 1, ta có K1 =

000110 110000 001011 101111 111111 000111 000001 110010. L1 = R0= 1111 0000 1010 1010 1111 0000 1010 1010. R1 = L0 f(R0,K1) Để tính hàm f, chúng ta mở rộng mỗi khối Rn-1 từ 32bit lên 48bit bằng cách sử dụng một bảng một bảng lựa chọn mà lặp đi lặp lại một số bit trong Rn-1. Ta gọi việc sử dụng mỗi lựa chọn bảng này là hàm E. Vì vậy, E(Rn-1) có khối đầu vào 32bit và một khối đầu ra 48bit 48bit đầu ra được viết như 8 khối 6bit thu được bằng cách chọn các các bit trong đầu vào của nó theo theo 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

Ta tính E(R0) từ R0 như sau: R0 = 1111 0000 1010 1010 1111 0000 1010 1010 E(R0) = 011110 100001 010101 010101 011110 100001 010101 010101 Ta thấy mỗi khối 4bit ban đầu đã được mở rộng đến một khối 6 bit đầu ra. Tiếp theo để tính f , ta XOR đầu ra E(Rn-1) với khóa Kn: Kn E(Rn-1) Ví dụ: Cho K1, E(R0) ta có
K1= = 000110 110000 001011 101111 111111 000111 000001 110010
E(R0) = 011110 100001 010101 010101 011110 100001 010101 010101
K1 E(R0) = 011000 010001 011110 111010 100001 100110 010100 100111

Tiếp theo ta sẽ sử dụng mỗi nhóm 6bit như các địa chỉ trong bảng gọi lại “S boxes”. Mỗi nhóm 6bit sẽ cung cấp 1 địa chỉ trong một S box khác nhau. Nằm tại địa chỉ đó sẽ có 1 số 4bit. Số 4bit này sẽ thay thế 6bit ban đầu .

Kết quả cuối cùng là 8 nhóm 6bit được chuyển thành 8 nhóm 4bit, tổng số sẽ là 32 bit. Viết kết quả trước đó với 48bit dưới dạng sau:

Kn E(Rn-1) = B1B2B3B4B5B6B7B8

Trong đó mỗi Bi là một nhóm 6 bit như ta đã xã định với trường hợp K1 E(R0) ở trên. Bây giờ ta phải tính

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)

Với Si(Bi) tương ứng là đầu ra của hộp thứ i S box Mỗi hàm S1, S2,..., S8 có một khối 6bit là đầu vào và một khối 4bit là đầu ra. Bảng xác định S1: S1

S1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S1 là hàm được xác định trong bảng này và B là một khối 6 bit đầu vào, S1(B) được xác định như sau: xác định bit đầu tiên và cuối cùng của B: nó là một số nhị phân có dạng 00,01, 10 hoặc 11, tức là có giá trị từ 0-3 dạng thập phân. Đặt giá trị này là i. còn lại 4 bit giữa của B sẽ có giá trị trong khoảng từ 0 đến 15 dạng thập phân (vì dạng nhị phân của B nằm trong khoảng 0000-1111). Đặt giá trị này là j. Tra trong bảng số S1 ở trên hàng thứ i và cột thứ j. Đó là một số trong khoảng từ 0 đến 15 và biểu diễn nó bởi một khối 4 bit. Khối đó là đầu ra S1(B) của S1 cho đầu vào B. Ví dụ, khối đầu vào B1 = 011000, bit đầu tiên là "0" và bit cuối cùng "0" 00= 0 => i=0. Đây là hàng 0. bốn bit giữa là "1100", 1101 = 12=> j=12. vì vậy cột là cột số 12. Tìm tới hàng 0 cột 12 (0,12) = 5 , 5 nhị phân là 0101. Do đó S1(B1) = 0101. Ta có các bảng xác định S2, S3, ...S8 như sau: S2

S2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S3

S3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S4

S4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S5

S5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	0	14	2	13	6	15	0	9	10	4	5	3

S6

S6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S7

S7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

S8

S8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

Đối với vòng đầu tiên, đã đã tính toán và có:

K1 E(R0) = 011000 010001 011110 111010 100001 100110 010100 100111

Từ các bảng trên với B1, B2, ..., B8 đã xác định ta tính được: S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111

Cuối cùng trong việc tính toán f là làm một hàm hoán vị P của các đầu ra S-box để có được giá trị cuối cùng của f: f(R0,K1) = P(S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8))

Hoán vị P được xác định trong bảng dưới đây: 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

16	7	20	21
19	13	30	6
22	11	4	25

Từ đầu ra của 8 boxes: S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111 Ta có:
f(R0,K1) = 0010 0011 0100 1010 1010 1001 1011 1011
R1 = L0
f(R0,K1) = 1100 1100 0000 0000 1100 1100 1111 1111 0010 0011 0100 1010 1010 1001 1011 1011 = 1110 1111 0100 1010 0110 0101 0100 0100

Vòng lặp tiếp theo , sẽ có L2 = R1, R2 = L1 f(R1, K2) cứ làm tương tự trên cho 16 vòng lặp.

Vòng cuối cùng ta có các khối L16R16. Ta đảo ngược thứ tự của 2 khối và áp dụng một hoán vị IP-1 được xác định bởi bảng sau : IP-1

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

Xử lý tất cả 16 khối sử dụng phương pháp xác định như trên, tới vòng lặp thứ 16 ta có được :
L16 = 0100 0011 0100 0010 0011 0010 0011 0100
R16 = 0000 1010 0100 1100 1101 1001 1001 0101

Chúng ta đảo ngược thứ tự của 2 khối này và áp dụng hoán vị cuối cùng
R16L16 = 0000 1010 0100 1100 1101 1001 1001 01010100 0011 0100 0010 0011 0010 0011 0100

IP-1 = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

Chuyển đổi IP-1 về dạng thập lục phân ta được : 85E813540F0AB405

Kết luận : Vậy dạng mã hóa của M = 0123456789ABCDEF là C = 85E813540F0AB405

Giải mã : Quá trình giải mã chỉ đơn giản là nghịch đảo của mã hóa, các bước thực hiện tương tự như trên nhưng đảo ngược thứ tự các khóa được áp dụng, tức là áp dụng khóa K16 tới K1, làm hoàn toàn giống các bước và trình tự như mã hóa.