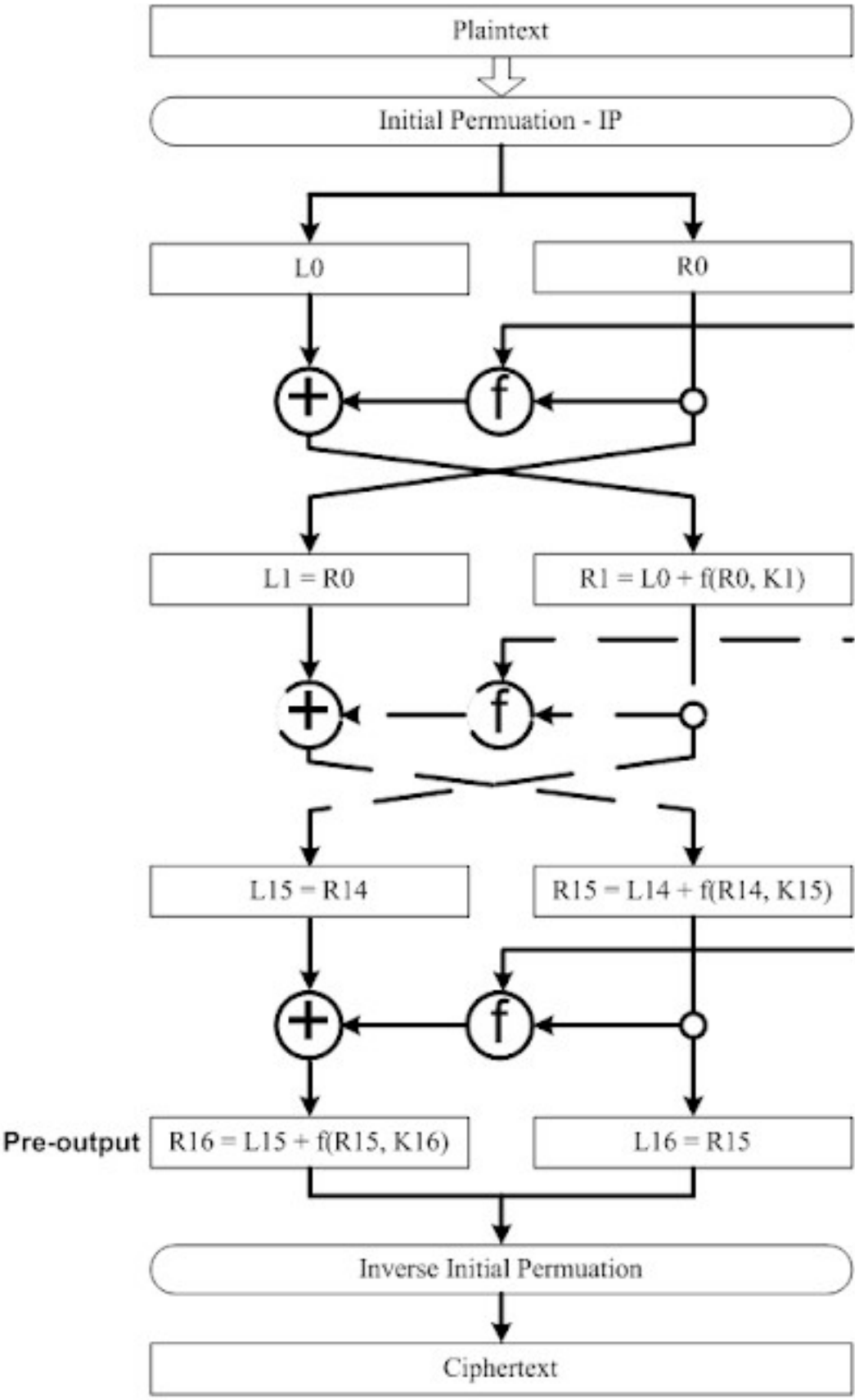
2. Thuật toán mã hóa dữ liệu DES – Encipher

2.1 Lưu đồ thuật toán mã hóa

Thuật toán DES được sử dụng để mã hóa và giải mã các block (khối) dữ liệu 64 bit dựa trên một key (khóa mã) 64 bit. Chú ý, các block được đánh số thứ tự bit từ trái sang phải và bắt đầu từ 1, bit đầu tiên bên trái là bit số 1 và bit cuối cùng bên phải là bit số 64. Quá trình giải mã và mã hóa sử dụng cùng một key nhưng thứ tự phân phối các giá trị các bit key của quá trình giải mã ngược với quá trình mã hóa.

Một block dữ liệu sẽ được hoán vị khởi tạo (Initial Permutation) IP trước khi thực hiện tính toán mã hóa với key. Cuối cùng, kết quả tính toán với key sẽ được hoán vị lần nữa để tạo ra , đây là hoán vị đảo của hoán vị khởi tạo gọi là (Inverse Initial Permutation) IP-1. Việc tính toán dựa trên key được định nghĩa đơn giản trong một hàm f, gọi là hàm mã hóa, và một hàm KS, gọi là hàm phân phối key (key schedule). Hàm KS là hàm tạo ra các khóa vòng (round key) cho các lần lặp mã hóa. Có tất cả 16 khóa vòng từ K1 đến K16.


Plaintext
Initial Permuation - IP
L0
R0
f
L1 = R0
R1 = L0 + f(R0, K1)
L15 = R14
R15 = L14 + f(R14, K15)
Pre-output
R16 = L15 + f(R15, K16)
L16 = R15
Inverse Initial Permuation
Ciphertext

Hình 1. Giải thuật mã hóa DES

2.2 Hoán vị khởi tạo - IP

Hoán vị là thay đổi ví trí các bit trong một chuỗi giá trị nhưng không làm thay đổi giá trị của các bit này. Đây là bước đầu tiên trong quy trình mã hóa dữ liệu. 64 bit dữ liệu đầu vào, gọi là plaintext, sẽ được hoán vị theo bảng mô tả sau đây.

***IP*[1]**

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

Hình 2. Sơ đồ hoán vị khởi tạo, ký hiệu IP

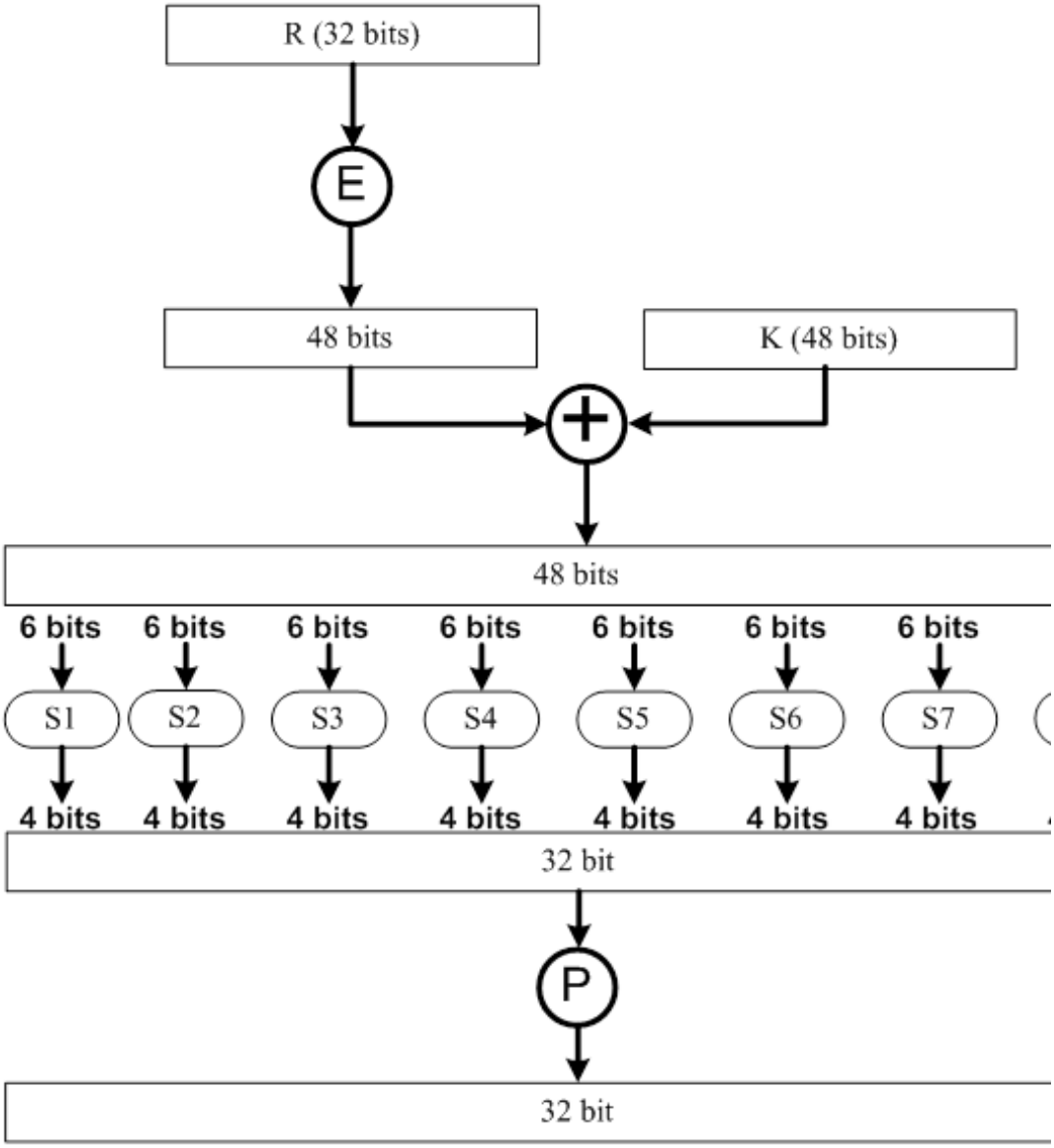
Chuỗi bit đầu vào được đánh số từ 1 đến 64 (tính từ trái qua phải). Sau đó, các bit này được thay đổi vị trí như sơ đồ IP, bit số 58 được đặt vào vị trí đầu tiên, bit số 50 được đặt vào vị trí thứ 2. Cứ như vậy, bit thứ 7 được đặt vào vị trí cuối cùng.

Sau hoán vị, chuỗi bit mới được phân ra làm hai đoạn, mỗi đoạn 32 bit để bắt đầu vào quy trình tính toán mã hóa với key. Đoạn bên trái ký hiệu là L, đoạn bên phải ký hiệu là R. Đoạn L gồm các bit từ bit số 1 đến bit số 32, đoạn R gồm các bit từ bit số 33 đến bit số 64. Đoạn L của lần tính toán sau sẽ chính là đoạn R của lần tính toán trước. Đoạn R của lần tính toán sau sẽ được tính từ đoạn R trước đó qua hàm mã hóa f(R, K) rồi XOR với đoạn L của lần tính trước đó.

$$L_{n+1} = R_n$$

$$R_{n+1} = L_n \oplus f(R_n, K_{n+1})$$

2.3 Hàm mã hóa f(R,K)



Hình 3. Tính toán hàm mã hóa f(R,K)

Đầu tiên, 32 bit của đoạn R được đánh số từ 1 đến 32 theo thứ tự từ trái qua phải. Giá trị này sẽ được chuyển đổi thông qua bảng tra E để tạo thành một giá trị 48 bit. Bit đầu tiên trong chuỗi giá trị 48 bit là bit số 32 của R, bit thứ 2 là bit số 1 của R, bit thứ 3 là bit số 2 của R và bit cuối cùng là bit số 1 của R.

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

Hình 4. Bảng tra E

Sau khi tra bảng E, giá trị 48 bit được XOR với 48 bit của khóa vòng (cách tạo ra khóa vòng 48 bit sẽ được trình bày sau). Kết quả phép XOR được chia làm 8 block được đánh số từ 1 đến 8 theo thứ tự từ trái qua phải, mỗi block 6 bit. Mỗi block sẽ được biến đổi thông qua các hàm lựa chọn riêng biệt. Tương ứng với 8 block sẽ có 8 hàm chuyển đổi (selection function) riêng biệt là S1, S2, S3, S4, S5, S6, S7 và S8.

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

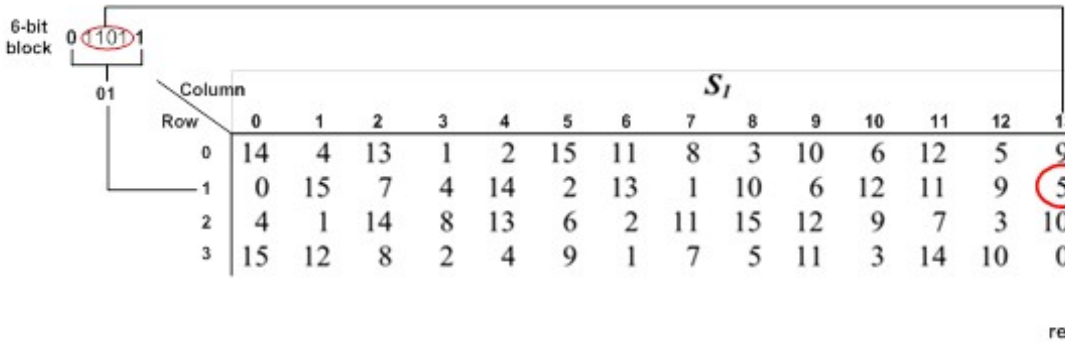
S_8

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

Hình 5. Các hàm S (selection fuction)

Việc chuyển đổi giá trị của các hàm S1, S2, ..., S8 được thực hiện bằng cách tách block 6 bit thành hai phần. Phần thứ nhất là tổ hợp của bit đầu tiên và bit cuối cùng của block để tạo thành 2 bit chọn hàng của bảng S, bảng S có 4 hàng được đánh số từ 0 đến 3 theo thứ tự từ trên xuống. Phần thứ 2 là 4 bit còn lại dùng để chọn cột của bảng S, bảng S có 16 cột được đánh số từ 0 đến 15 theo thứ tự từ trái qua phải. Như vậy, với mỗi block 8 bit ta chọn được 1 giá trị trong bảng S. Giá trị này nằm trong khoảng từ 0 đến 15 sẽ được quy đổi thành chuỗi nhị phân 4 bit tương ứng. Các chuỗi nhị phân có được sau khi chuyển đổi từ S1 đến S8 sẽ được ghép lại theo thứ tự từ trái qua phải để tạo thành một giá trị 32 bit.

Ví dụ về việc thực hiện chuyển đổi hàm S, giả sử, giá trị block 6 bit đầu tiên là 011011 = 1 + 2+8+16=27. Ta tách chuỗi này ra làm hai tổ hợp giá trị là 01 (bit đầu tiên và bit cuối cùng) và 1101 (4 bit ở giữa). Hai tổ hợp này được dùng để chọn hàng và cột tương ứng như hình minh họa sau:



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

Hình 6. Minh họa việc sử dụng hàm S1

Tổ hợp 01 sẽ chọn hàng 1, tổ hợp 1101 sẽ chọn cột 13 và kết quả trả về là 5 có giá trị nhị phân 4 bit là 0101.

Qua bước chuyển đổi với các hàm lựa chọn S, kết quả thu được là một giá trị 32 bit. Giá trị này được đưa qua một hàm hoán vị P để tạo ra giá trị hàm f.

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

Hình 7. Hoán vị P trong thuật toán tính hàm mã hóa f(R,K)

Giá trị 32 bit thu được từ các chuyển đổi với hàm lựa chọn S sẽ được đánh số từ 1 đến 32 theo thứ tự từ trái qua phải.

Theo bảng hoán vị P, bit đầu tiên sau hoán vị sẽ là bit số 16, bit thứ 2 sẽ là bit số 7 và bit cuối cùng sẽ là bit số 25. Hàm tính toán mã hóa f(R, K) được định nghĩa như sau:

$$f(R,K) = P(S_1(B_1)S_2(B_2) \ldots S_8(B_8))$$

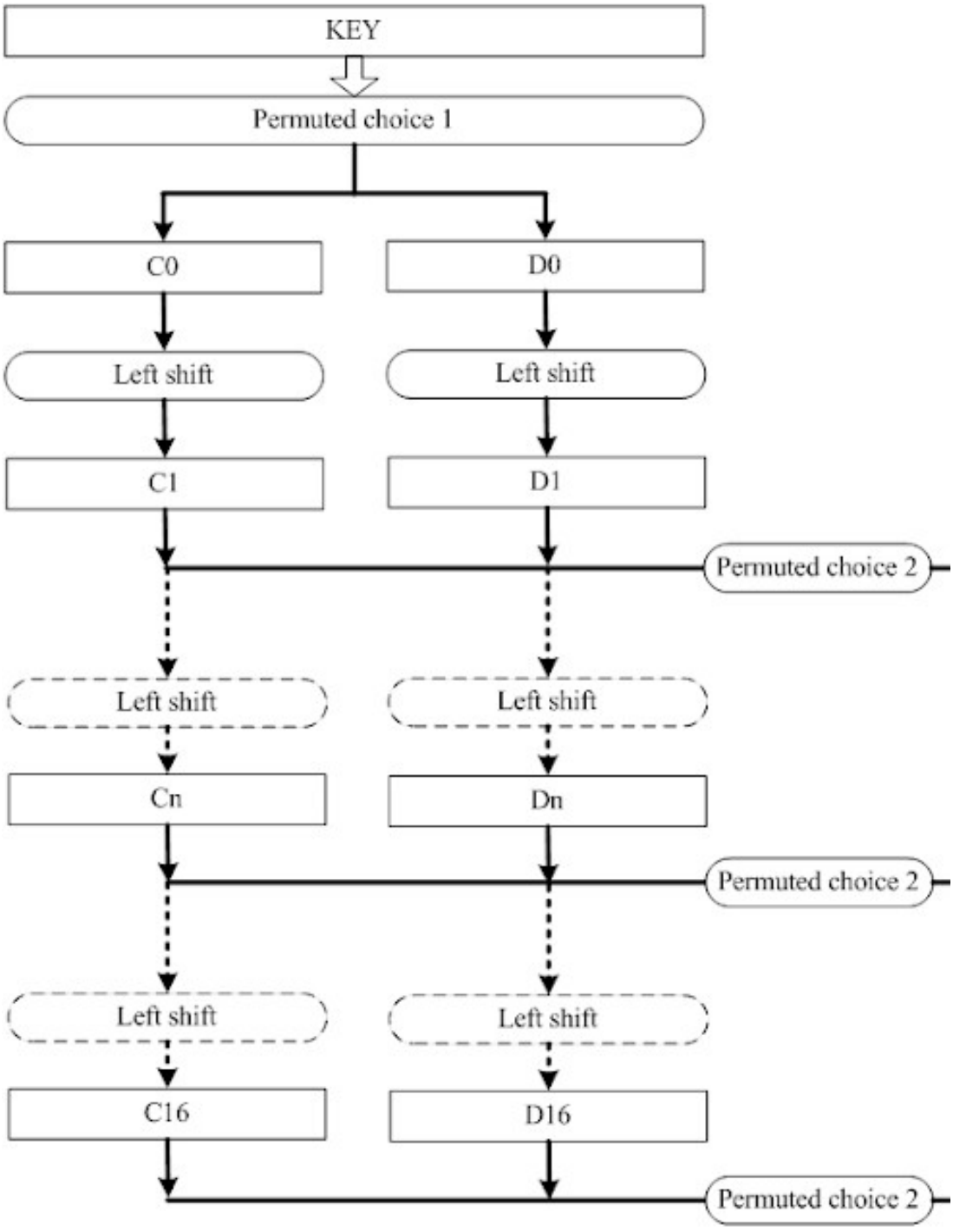
$$B_1B_2 \ldots B_8 = K \oplus E(R)$$

Trong đó:

- P(): là phép hoán vị P
- Sn: là phép chuyển đổi block n (n chạy từ 1 đến 8) với hàm lựa chọn S
- Bn: là block 6 bit thứ n (n chạy từ 1 đến 8). Block này lấy từ phép toán XOR giữa khóa vòng K và giá trị hàm E(R)
- E(R): là hàm chuyển đổi giá trị R 32 bit thành giá trị 48 bit

2.4 Tính khóa vòng - KS

Một key có 64 bit nhưng chỉ có 56 bit được sử dụng để thực hiện tính toán giá trị khóa vòng. Key được chia làm 8 byte. Các bit ở vị trí 8, 16, 32, 40, 48, 56 và 64 là các bit parity được sử dụng để kiểm tra độ chính xác của key theo từng byte vì khi key được phân phối trên đường truyền đến bộ mã hóa giải mã thì có thể xảy ra lỗi. Parity được sử dụng là parity lẻ (odd parity).



Hình 8. Thuật toán tính khóa vòng

Key gốc sẽ được thực hiện hoán vị lựa chọn PC-1. Key được đánh số từ 1 đến 64 theo thứ tự từ trái qua phải.

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

Hình 9. Bảng hoán vị lựa chọn PC-1

Bảng hoán vị lựa chọn PC-1 có hai phần. Phần đầu dùng để xác định giá trị C0 và phần sau dùng để xác định giá trị D0. Theo bảng trên thì C0 là chuỗi bit có thứ tự là 57, 49, 41, ..., 36 lấy từ key gốc, D0 là chuỗi bit có thứ tự là 63, 55, 47, ..., 4 lấy từ key gốc.

Sau khi xác định được giá trị ban đầu để tính key là C0 và D0 thì các khóa vòng Kn (với n từ 1 đến 16) sẽ được tính theo nguyên tắc giá trị của khóa vòng thứ n sẽ được tính từ giá trị khóa vòng thứ n-1.

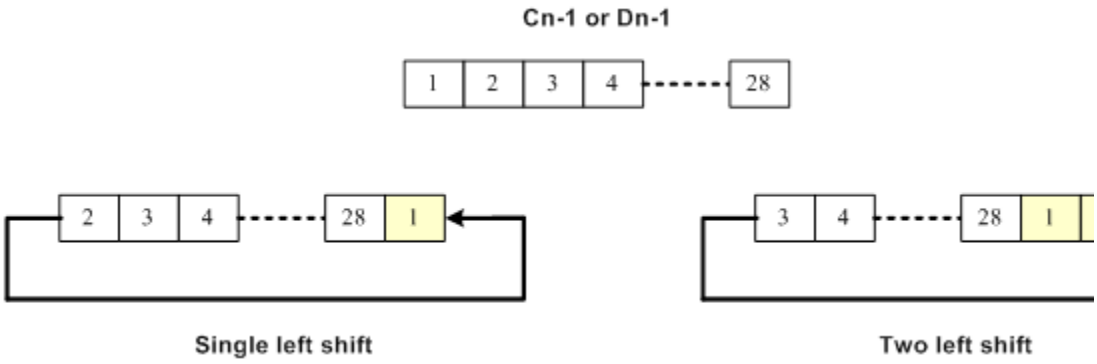
$$K_n = PC\_2(C_n D_n)$$

Trong đó Cn và Dn được tạo từ Cn-1 và Dn-1 bằng cách dịch trái các giá trị này với số bit được quy định trong bảng sau đây:

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Hình 10. Bảng quy định số bit dịch trái khi tính khóa vòng

Ví dụ, theo bảng trên, C3 và D3 có được từ C2 và D2 bằng cách dịch trái 2 bit. Hay C16 và D16 có được từ C15 và D15 bằng cách dịch trái 1 bit. Dịch trái ở đây được hiểu là quay trái như minh họa sau đây:



Hình 11. Minh họa phép dịch trái khi tính khóa vòng

Sau khi tính được Cn và Dn thì chuỗi CnDn sẽ được đánh số từ 1 đến 56 theo thứ tự từ trái sang phải và được hoán vị lựa chọn lần 2 theo bảng hoán vị PC-2.

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

Hình 12. Bảng hoán vị lựa chọn PC-2

Như vậy bit đầu tiên của khóa vòng Kn là bit số 14 của chuỗi CnDn, bit thứ 2 là bit số 17 của chuỗi CnDn và bit cuối cùng là bit số 32 của chuỗi CnDn.

2.5 Hoán vị khởi tạo đảo IP-1

Đây là bước cuối cùng để tạo ra giá trị mã hóa. Giá trị của lần lặp mã hóa cuối cùng sẽ được hoán vị khởi tạo đảo IP-1 và tạo ra giá trị mã hóa plaintext.

IP^{-1}

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

Hình 13. Bảng hoán vị khởi tạo đảo IP-1

2.6 Ví dụ về mã hóa DES

Giả sử ta có dữ liệu cần mã hóa và key là:

- M = 00123456789abcde (Hex) = 0000 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1111
- K = 0133457799bbcdff (Hex) = 0000 0001 0011 0011 0100 0101 0111 0111 1001 10001 1011 1011 1100 1100 1111 1111

Từ hai đầu vào này, giá trị của từng bước tính toán mã hóa DES sẽ được minh họa chi tiết sau đây.

2.6.1 Hoán vị khởi tạo – IP

Thông điệp M được đánh số vị trí bit từ trái qua phải như sau:

M	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	.....	1	1	1	1
STT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	.....	61	62	63	64

Sắp xếp lại thứ tự các bit của M theo bảng hoán vị IP, kết quả có được sau bước này là:

IP(M) = 98fecc00e054f0aa (Hex) = 1001 1000 1111 1110 1100 1100 0000 0000 1110 0000 0101 0100 1111 0000 1010 1010

2.6.2 Tính toán giá trị các khóa vòng – KS

Để tính toán hàm mã hóa f, chúng ta cần có giá trị khóa cho từng lần lặp mã hóa. Key ban đầu được đánh số thứ tự bit từ trái qua phải như sau:

M	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	.....	1	1	1	1
STT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	.....	61	62	63	64

Sau khi sắp xếp các bit theo bảng hoán vị PC-1 ở Hình 1-8, kết quả thu được là hai giá trị đầu như sau:

- C0 = f0ccaab (Hex) = 1111 0000 1100 1100 1010 1010 1011
- D0 = aaccf0a (Hex) = 1010 1010 1100 1100 1111 0000 1010

Để tính khóa vòng đầu tiên K1 thì C0 và D0 sẽ được dịch (quay) trái 1 bit. Giá trị thu được sau khi dịch trái là:

- C1 = e199557 (Hex) = 1110 0001 1001 1001 0101 0101 0111
- D1 = 5599e15 (Hex) = 0101 0101 1001 1001 1110 0001 0101

Hai chuỗi C1 và D1 được ghép lài thành một chuỗi 56 bit là e1995575599e15. Chuỗi này được hoán vị bằng bảng PC-2 ở Hình 1-11 để được giá trị khóa vòng 48 bit thứ nhất K1.

- K1 = 1b02efdb49a5 (Hex)

Để tính khóa vòng K2 thì lấy C1 và D1 dịch trái với số lượng bit theo bảng ở Hình 1-9, rồi lấy kết quả hoán vị theo bảng PC-2. Quá trình cứ tiếp tục cho đến khóa vòng cuối cùng là K16. Kết quả các khóa vòng 48 bit thu được là:

- K2 = 69aed925ae66 (Hex)
- K3 = 55fc8ab4acd2
- K4 = 72add2ad8657
- K5 = 7cec071fe6c2
- K6 = 63a51e3cc545
- K7 = 6c84b78ae4c6
- K8 = f7883aece781
- K9 = c0dbeb27b839
- K10 = b1f347631d76
- K11 = 215fc30d89be
- K12 = 7171f5455cd5
- K13 = 95c5d14b80fd
- K14 = 5743b783đ8d
- K15 = bf91850a17b5
- K16 = cb3d0bbc7072

2.6.3 Tính hàm mã hóa f(R,K)

Sau bước hoán vị khởi tạo IP, giá trị IP(M) sẽ được tách làm hai phần là:

- R0 = e054f0aa (Hex) = 1110 0000 0101 0100 1111 0000 1010 1010
- L0 = 98fecc00 (Hex) = 1001 1000 1111 1110 1100 1100 0000 0000

Giá trị 32 bit của R0 được tra qua bảng E để tạo ra một giá trị 48 bit.

- E(R0) = 7002a97a1555 (Hex)

Giá trị này được XOR với khóa vòng thứ nhất K1 và được kết quả

- XOR(E(R0), K1) = 6b0046a15cf0 (Hex)

Giá trị trên được chia thành 8 nhóm theo thứ tự từ trái qua phải, mỗi nhóm 6 bit để đưa đến các bảng S.

Bảng S	Giá trị đầu vào	Giá trị đầu ra	
		Số thập phân	Số b
S1	011010	9	1(
S2	110000	5	01
S3	000001	13	11
S4	000110	3	0(
S5	101000	10	1(
S6	010101	13	11
S7	110011	5	01
S8	110000	0	0(

Hình 14. Ví dụ về tra bảng S

Các giá trị đầu ra sau bước tra các bảng S sẽ được ghép lại theo thứ tự từ S1 đến S8 để được 1 giá trị 32 bit.

- S1()S2()S3()S4()S5()S6()S7()S8() = 10010101110100111010110101010000 = 95d3ad50 (Hex)

Giá trị này được hoán vị bằng bảng P để cho ra giá trị của hàm f.

- f(R0,K1) = 97d1619a (Hex) = 1001 0111 1101 0001 0110 0001 1001 1010

Tương tự, ta có giá trị hàm f tại các vòng lặp mã hóa còn lại như sau:

- f(R1,K2) = 88488d0b (Hex)
- f(R2,K3) = da3b2692
- f(R3,K4) = f44950b2
- f(R4,K5) = d83237fd
- f(R5,K6) = afc43b25
- f(R6,K7) = 4e5123a2
- f(R7,K8) = 6cfdecb8
- f(R8,K9) = fb0600b1
- f(R9,K10) = d51508e4
- f(R10,K11) = fcf67146
- f(R11,K12) = 704fa3a5
- f(R12,K13) = 7bfe2806
- f(R13,K14) = 65fc7a48
- f(R14,K15) = 513f1d11
- f(R15,K16) = cbf5252d

2.6.4 Giá trị tại mỗi vòng lặp mã hóa

Giá trị của hàm f(R,K) được sử dụng để tính giá trị Rn và Ln tại mỗi vòng lặp mã hóa theo công thức:

$$L_n = R_{n-1}$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

Thay vào công thức trên ta có các kết quả như sau:

Vòng lặp thứ n	f(R,K)	Ln	Rn
0		98fecc00	e054f0aa
1	97d1619a	e054f0aa	0f2fad9a
2	88488d0b	0f2fad9a	681c7da1
3	da3b2692	681c7da1	d5148b08
4	f44950b2	d5148b08	9c552d13
5	d83237fd	9c552d13	0d26bcf5
6	afc43b25	0d26bcf5	33911636
7	4e5123a2	33911636	43779f57
8	6cfdecb8	43779f57	5f6cfa8e
9	fb0600b1	5f6cfa8e	b8719fe6
10	d51508e4	b8719fe6	8a79f26a
11	fcf67146	8a79f26a	4487eea0
12	704fa3a5	4487eea0	fa3651cf
13	7bfe2806	fa3651cf	3f79c6a6
14	65fc7a48	3f79c6a6	9fca2b87
15	513f1d11	9fca2b87	6e46dbb7
16	cbf5252d	6e46dbb7	543f0eaa

2.6.5 Hoán vị khởi tạo đảo IP-1

Giá trị R16 và L16 của vòng lặp mã hóa cuối cùng sẽ được ghép lại thành một chuỗi 64 bit để thực hiện hoán vị theo bảng IP-1.

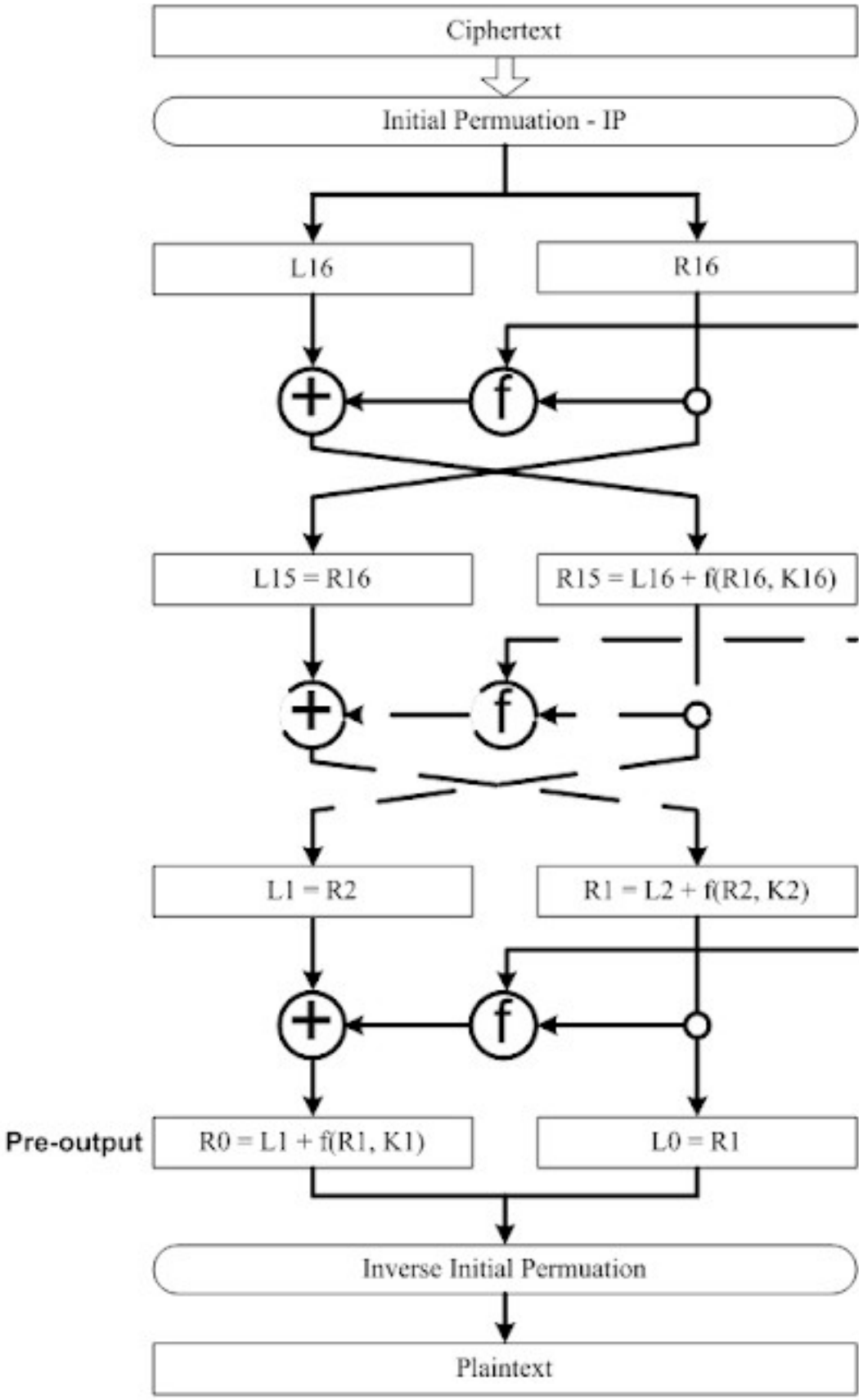
Kết quả của phép hoán vị này chính là giá trị mã hóa (ciphertext) cần tính.

IP-1(R16, L16) = 1abff69d5a93e80b (Hex) = 0001 1010 1011 1111 1111 0110 1001 1101 0101 1010 1001 0011 1110 1000 0000 1011

3. Thuật toán giải mã dữ liệu DES

Các bước của quá trình giải mã dữ liệu được thực hiện tương tự như quá trình mã hóa dữ liệu. Trong quá trình giải mã có một số thay đổi như sau:

- Đầu vào lúc này là dữ liệu cần giải mã (ciphertext) và đầu ra là kết quả giải mã được (plaintext).
- Khóa vòng sử dụng trong các vòng lặp giải mã có thứ tự ngược với quá trình mã hóa. Nghĩa là, tại vòng lặp giải mã đầu tiên, khóa vòng được sử dụng là K16. Tại vòng lặp giải mã thứ 2, khóa vòng được sử dụng là K15, và tại vòng lặp giải mã cuối cùng thì khóa vòng được sử dụng là K1.


Ciphertext
Initial Permuation - IP
L16
R16
f
L15 = R16
R15 = L16 + f(R16, K16)
L1 = R2
R1 = L2 + f(R2, K2)
Pre-output
R0 = L1 + f(R1, K1)
L0 = R1
Inverse Initial Permuation
Plaintext

Hình 16. Quá trình giải mã dữ liệu DES