Mã hóa DES
Data Encryption Standard

Huỳnh Trọng Thưa
htthua@ptithcm.edu.vn

Part 1 - Encryption of DES

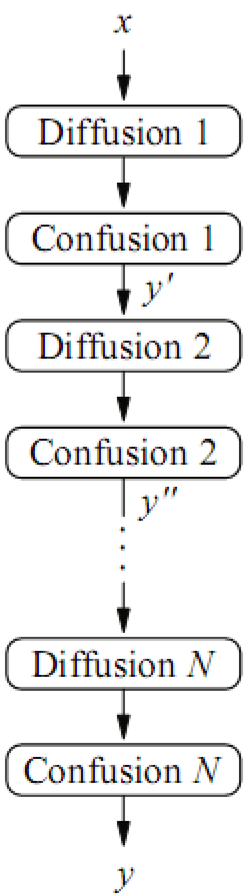
- Feistel structure of DES
- S-boxes
- Key Schedule

Data Encryption Standard (DES) and Alternatives

- Basic design ideas of **block ciphers**, including **confusion (xáo trộn)** and **diffusion (khuếch tán)**, which are important properties of all modern block ciphers
- The internal structure of DES, including Feistel networks, S-boxes and the key schedule.
- Alternatives to DES, including 3DES

Confusion and Diffusion

- Confusion: the relationship between key and ciphertext is obscured.
  – for achieving confusion: substitution, which is found in both DES and AES.
- Diffusion: the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of *hiding statistical properties* of the plaintext.
  – A simple diffusion element is the bit permutation, which is used frequently within DES.



Principle of an *N* round product cipher, where *each round performs a confusion and diffusion operation*

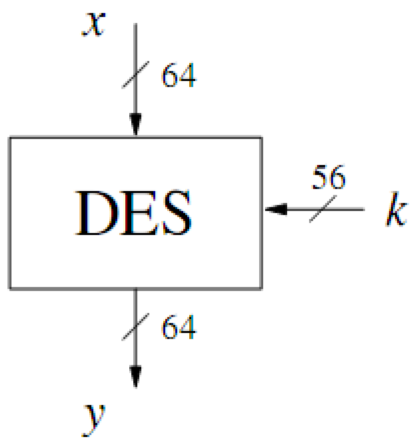
Modern block ciphers

- Changing of **one bit** of plaintext results on average in the change of **half the output bits**, i.e., the second ciphertext looks statistically independent of the first one.

Principle of diffusion of a block cipher

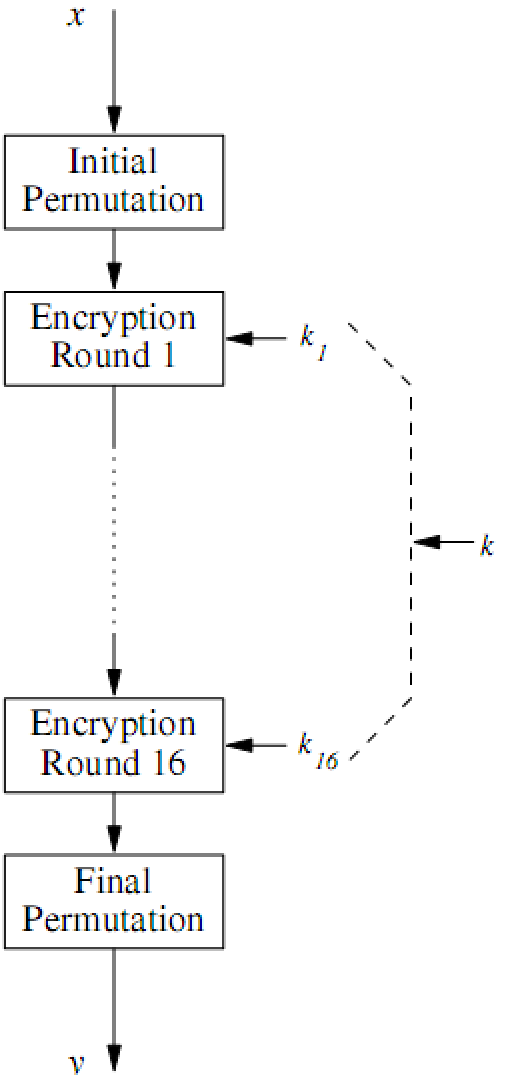
DES block cipher

- DES is a cipher which encrypts blocks of length of 64 bits with a key of size of 56 bits
- DES is a symmetric cipher.
- An iterative algorithm.

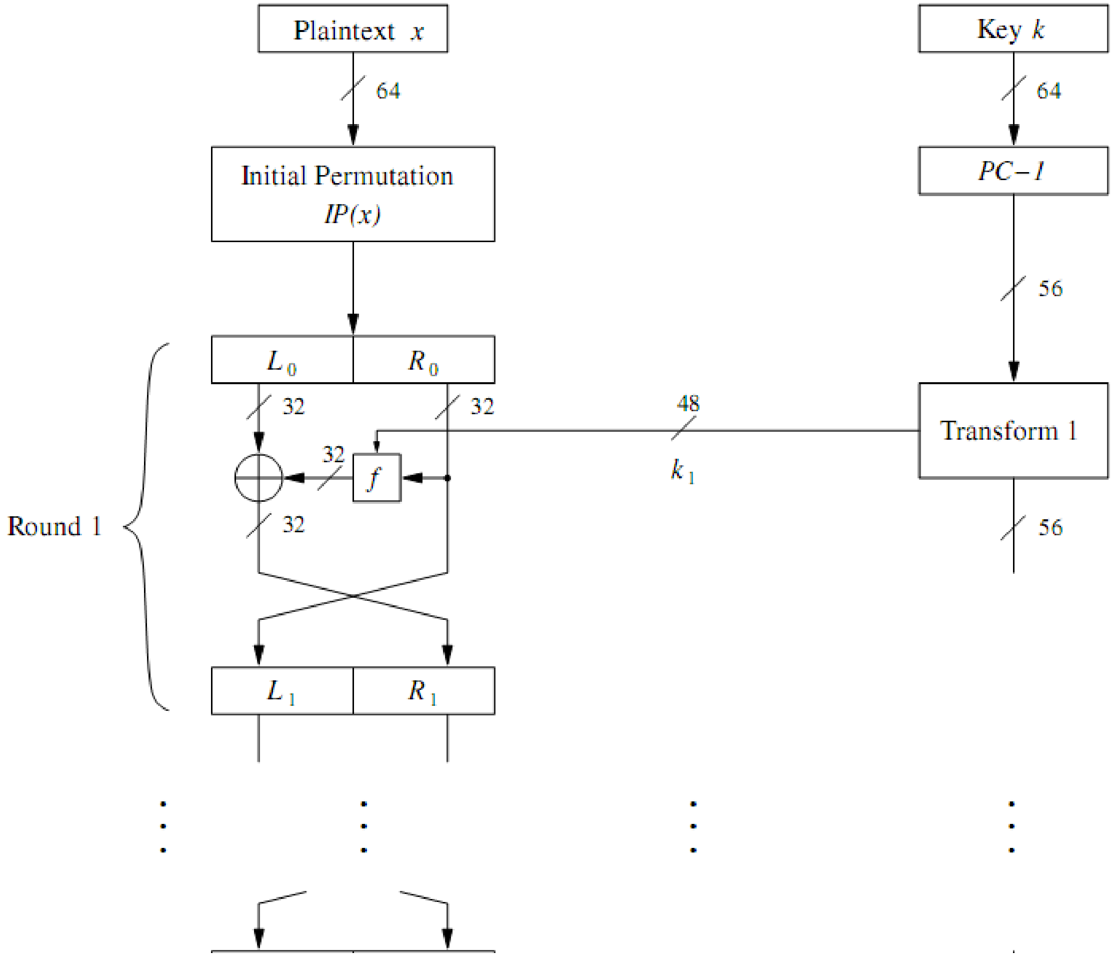


Round structure of DES

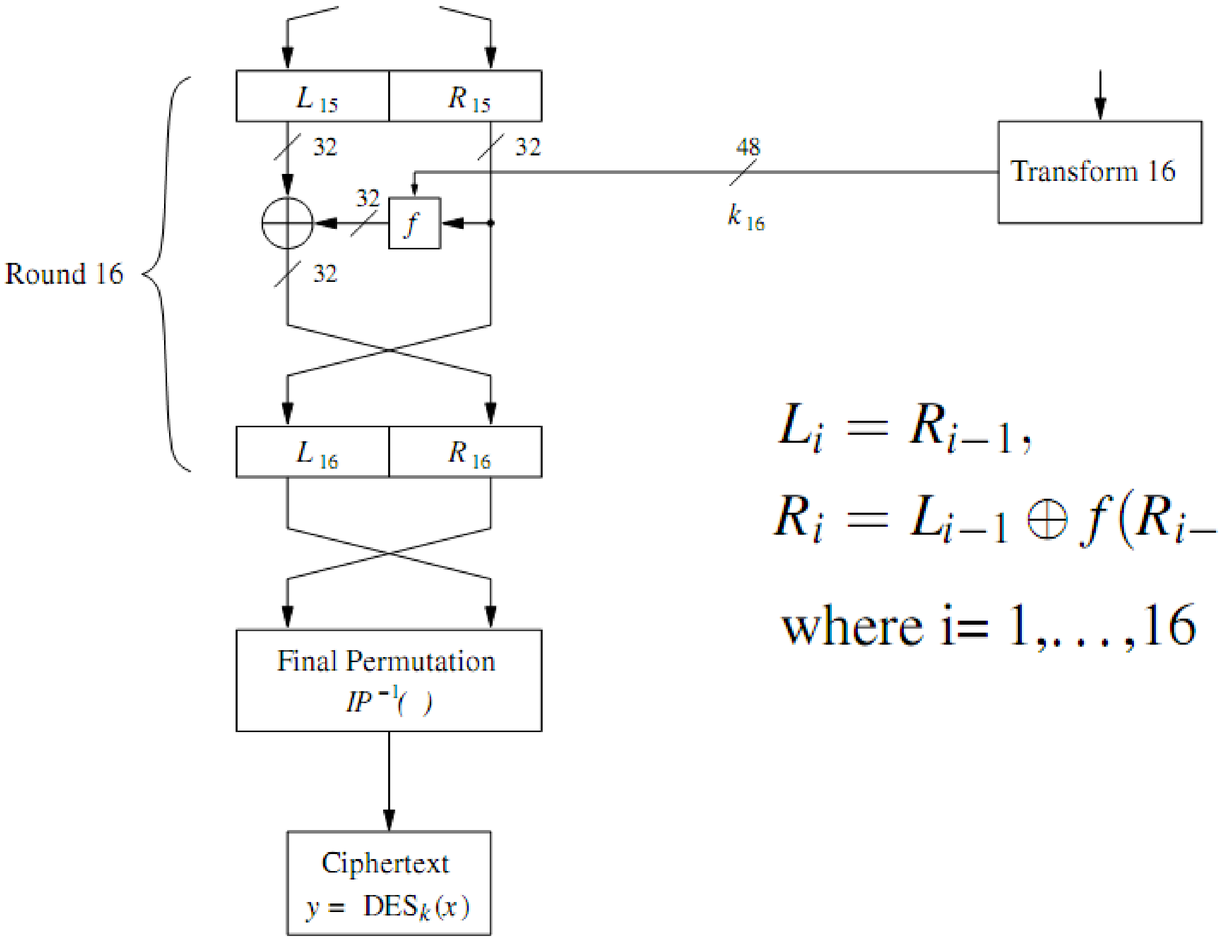
- For each block of plaintext, encryption is handled in 16 rounds which all perform the identical operation.
- In every round a different subkey is used and all subkeys k_i are derived from the main key k.



The Feistel structure of DES



The Feistel structure of DES (cont.)



$$L_i = R_{i-1},$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

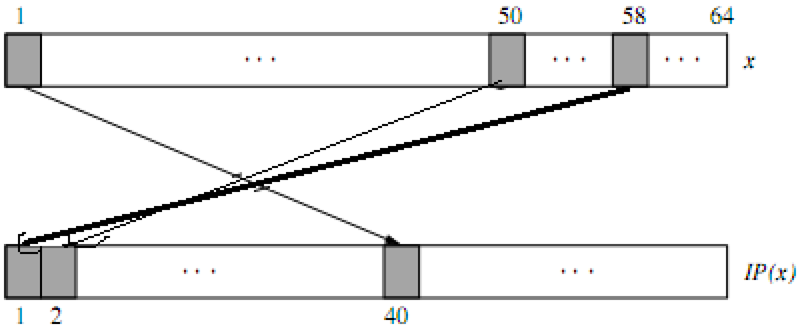
where i= 1,. . . ,16

Internal Structure of DES

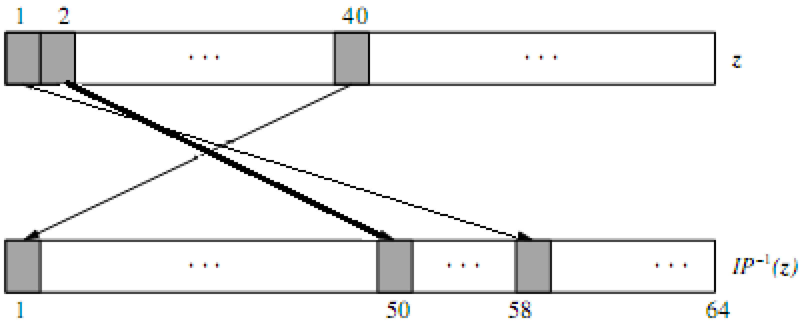
- Initial and Final Permutation
- f – function
- Key Schedule

Initial and Final Permutation

- are bitwise permutations



bit swaps of the initial permutation



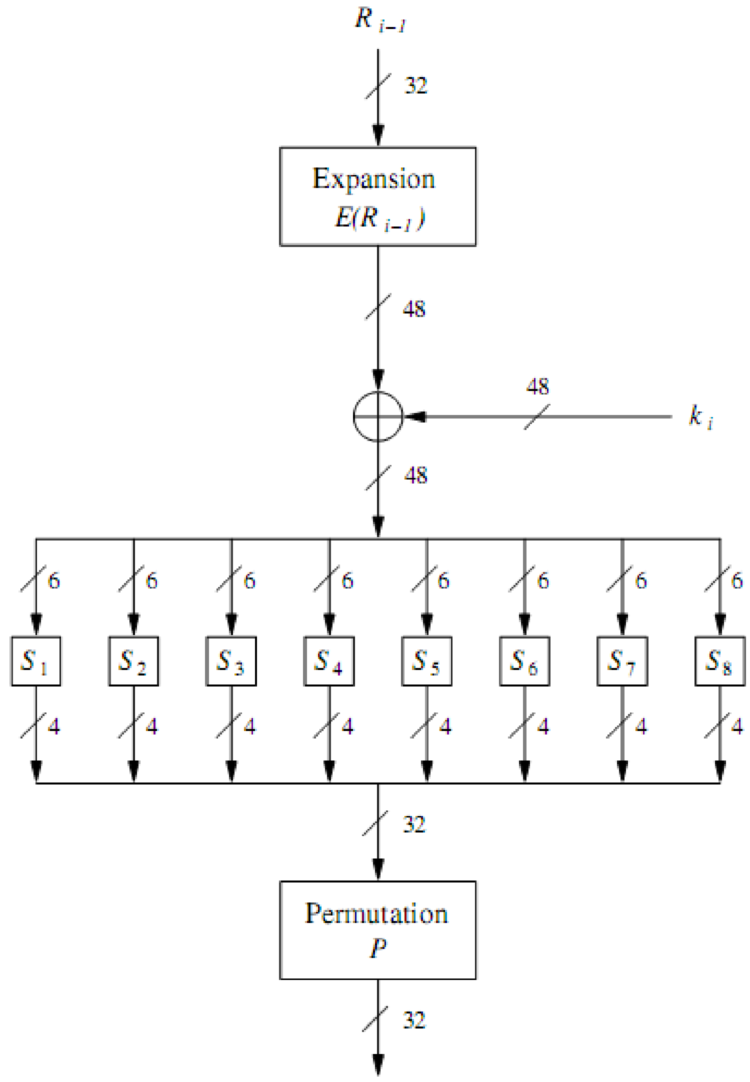
bit swaps of the final permutation

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

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

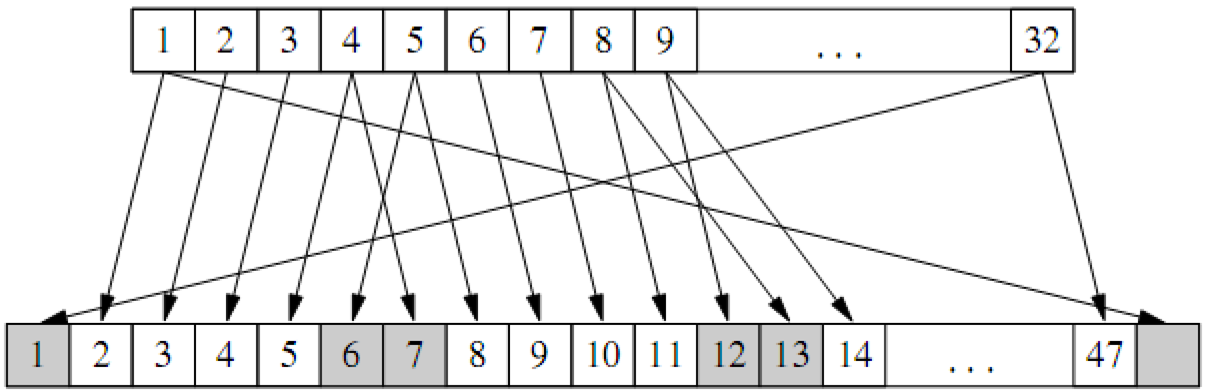
read from left to right, top to bottom

f - function



R i−1
32
Expansion
E(R i−1)
48
48
k i
48
6
6
6
6
6
6
6
6
S 1
S 2
S 3
S 4
S 5
S 6
S 7
S 8
4
4
4
4
4
4
4
4
32
Permutation
P
32

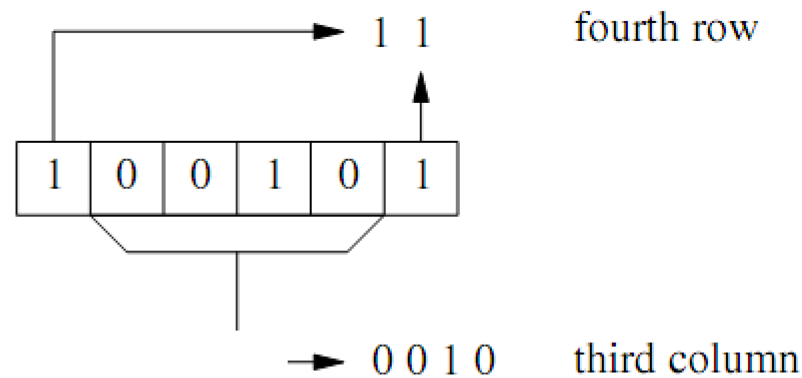
Bit swaps of the expansion function E



E					
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

S-boxes

- Each S-box contains 2^6 =64 entries.
- Each entry is a 4-bit value.



S-box S_1

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Decoding of the input 100101_2 by S-box 1

- **Ex:** The S-box input b =$(100101)_2$ indicates the row 11_2 = 3 (i.e., fourth row, numbering starts with 00_2) and the column 0010_2 = 2 (i.e., the third column). If the input b is fed into S-box 1, the output is S1(37 = 100101_2)= 8 = 1000_2.

S-boxes table for Ref.

S-box S_1

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

S-box S_2

S_2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

S-box S_3

S_3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

S-box S_4

S_4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

S-box S_5

S_5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
1	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
2	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
3	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

S-box S_6

S_6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
1	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
2	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
3	04	03	02	12	09	05	15	10	11	14	01	07	06	00	08	13

S-box S_7

S_7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	04	11	02	14	15	00	08	13	03	12	09	07	05	10	06	01
1	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
2	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
3	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

S-box S_8

S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
3	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

The permutation P within the *f*-function

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

Key Schedule

- PC-1: ignoring every eighth bit (64-bit key -> 56 bits)
- 56-bit key is split into two halves C0 and D0
- The two 28-bit halves are cyclically shifted, i.e., rotated, left by one or two bit positions depending on the round i.

$PC-1$							
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

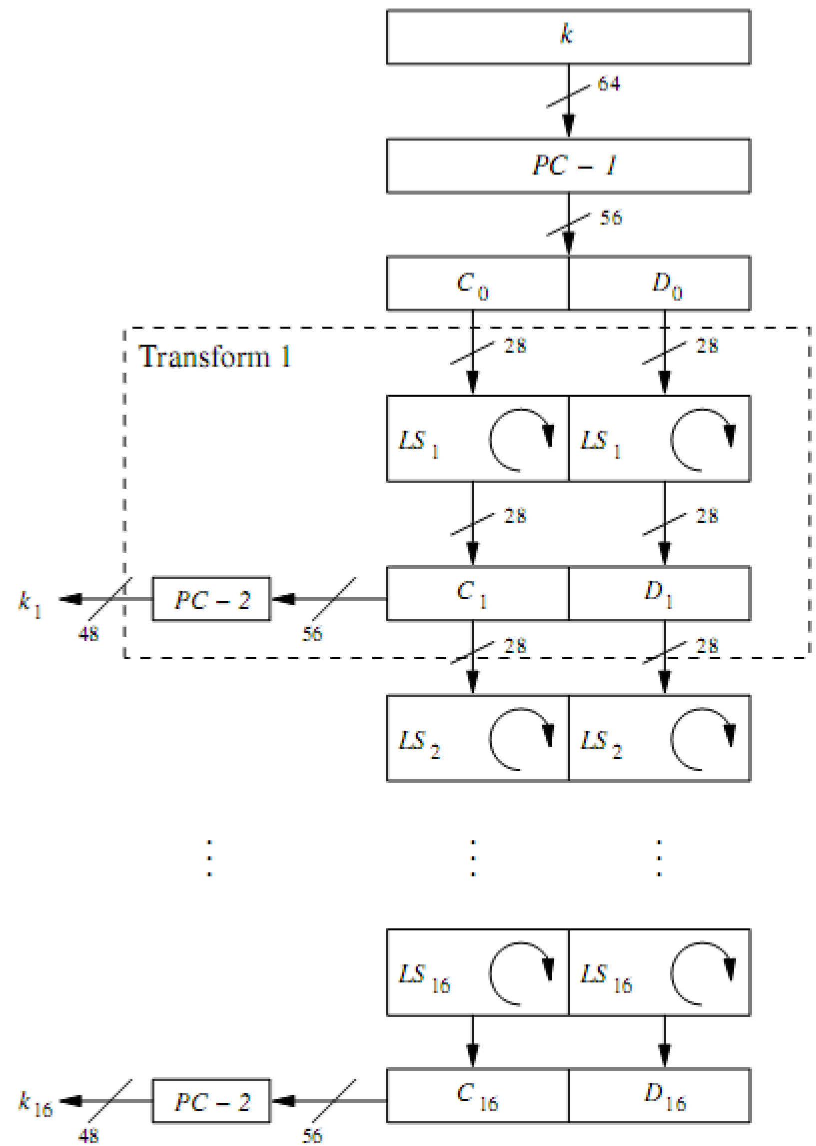
Initial key permutation PC−1

In rounds i = 1,2,9,16, the two halves are rotated left by one bit.
In the other rounds i ≠1,2,9,16, the two halves are rotated left by two bits.

Key schedule for DES encryption

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

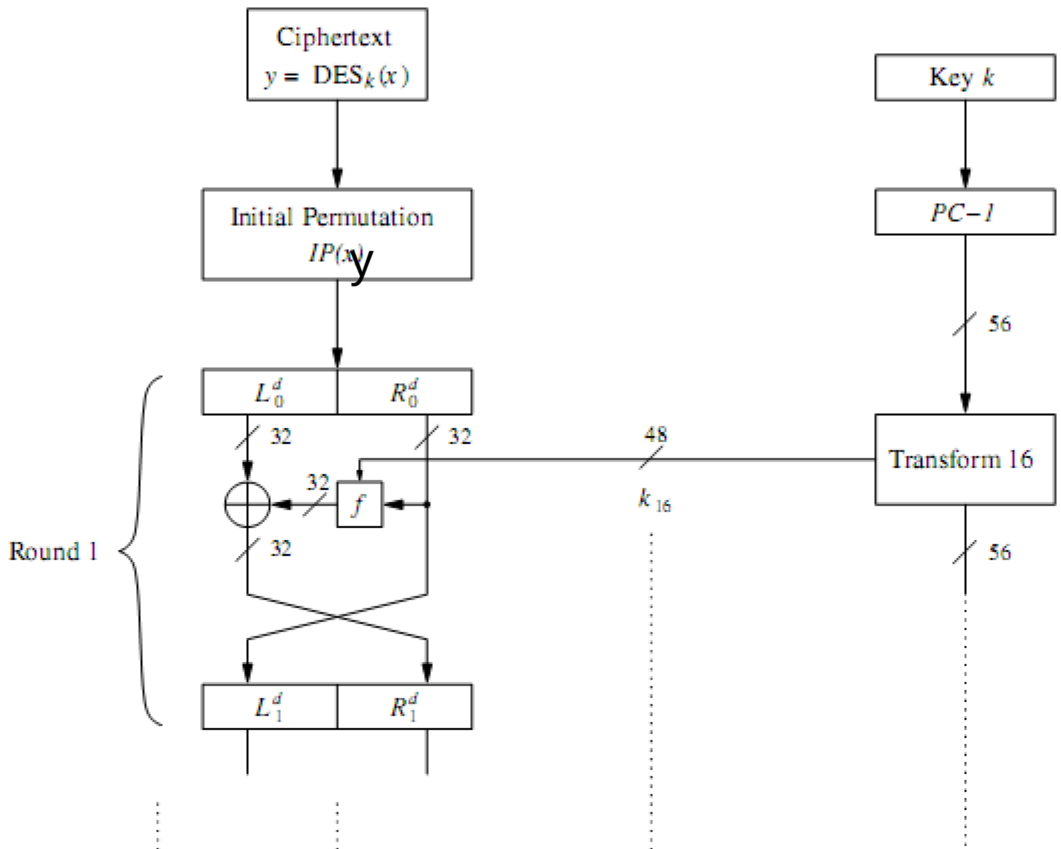
Round key permutation PC−2



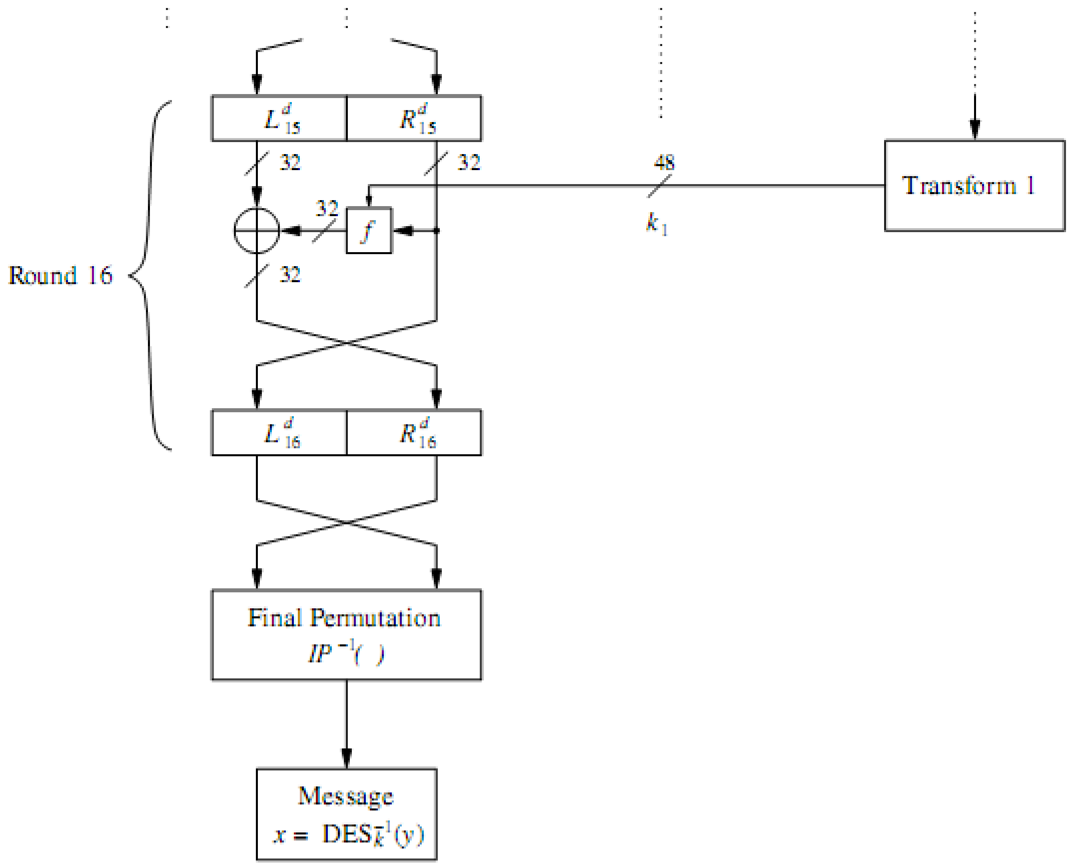
Part 2 - Descryption of DES

- Descryption of DES
- Security of DES
- DES Alternatives

Block diagram for DES decryption



Block diagram for DES decryption (cont.)



Reversed Key Schedule

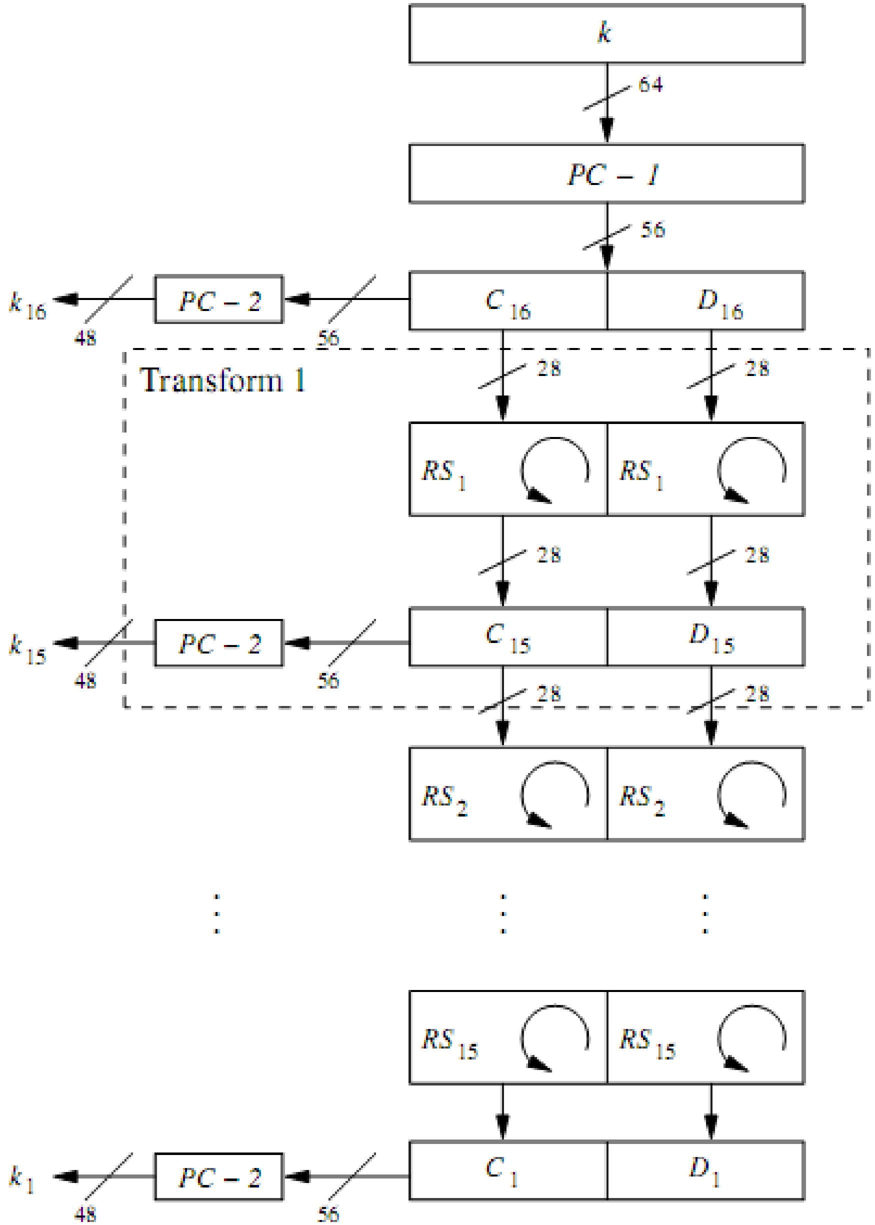
- k_{16} can be directly derived after PC−1.

$$
\begin{aligned}
k_{16} &= PC-2(C_{16}, D_{16}) \\
&= PC-2(C_0, D_0) \\
&= PC-2(PC-1(k))
\end{aligned}
$$

$$
\begin{aligned}
k_{15} &= PC-2(C_{15}, D_{15}) \\
&= PC-2(RS_2(C_{16}), RS_2(D_{16})) \\
&= PC-2(RS_2(C_0), RS_2(D_0))
\end{aligned}
$$

- Round 1, the key is not rotated.
- Rounds 2, 9, and 16 the two halves are rotated right by one bit.
- Other rounds 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14 and 15 the two halves are rotated right by two bits.

Reversed key schedule for decryption of DES



Why is the decryption function essentially the same as the encryption function?

$$(L_0^d, R_0^d) = IP(Y) = IP(IP^{-1}(R_{16}, L_{16})) = (R_{16}, L_{16})$$

$$L_0^d = R_{16}$$
$$R_0^d = L_{16} = R_{15}$$

$$L_1^d = R_0^d = L_{16} = R_{15}$$

$$R_1^d = L_0^d \oplus f(R_0^d, k_{16}) = R_{16} \oplus f(L_{16}, k_{16})$$
$$R_1^d = [L_{15} \oplus f(R_{15}, k_{16})] \oplus f(R_{15}, k_{16})$$
$$R_1^d = L_{15} \oplus [f(R_{15}, k_{16}) \oplus f(R_{15}, k_{16})] = L_{15}$$

Why is the decryption function essentially the same as the encryption function? (cont.)

$$L_i^d = R_{16-i},$$
$$R_i^d = L_{16-i}$$

where i = 0,1,...,16. In particular, after the last decryption round:

$$L_{16}^d = R_{16-16} = R_0$$
$$R_{16}^d = L_0$$

Finally, at the end of the decryption process, we have to reverse the initial permutation:

$$IP^{-1}(R_{16}^d, L_{16}^d) = IP^{-1}(L_0, R_0) = IP^{-1}(IP(x)) = x$$

Security of DES

- The key space is too small, i.e., the algorithm is vulnerable against brute-force attacks.
- The design criteria of the S-boxes was kept secret and there might have existed an analytical attack that exploits mathematical properties of the S-boxes, but which is only known to the DES designers.

DES Alternatives

- Advanced Encryption Standard (AES) and the AES Finalist Ciphers
- Triple DES (3DES) and DESX
- Lightweight Cipher PRESENT

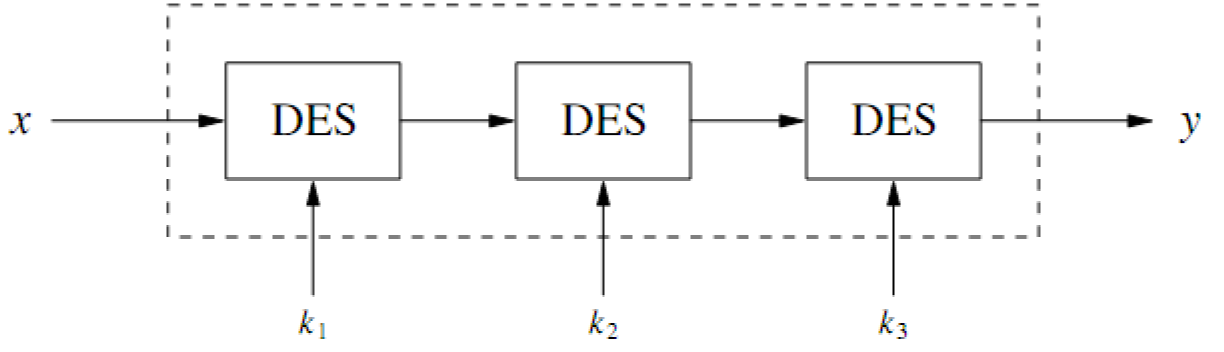
Advanced Encryption Standard (AES) and the AES Finalist Ciphers

- AES is with its three key lengths of 128, 192 and 256 bit secure
- Against brute-force attacks for several decades
- There are no analytical attacks with any reasonable chance of success known.

Triple DES (3DES) and DESX

- 3DES consists of three subsequent DES encryptions with different keys

$$y = DES_{k_3}(DES_{k_2}(DES_{k_1}(x)))$$



Another version of 3DES is

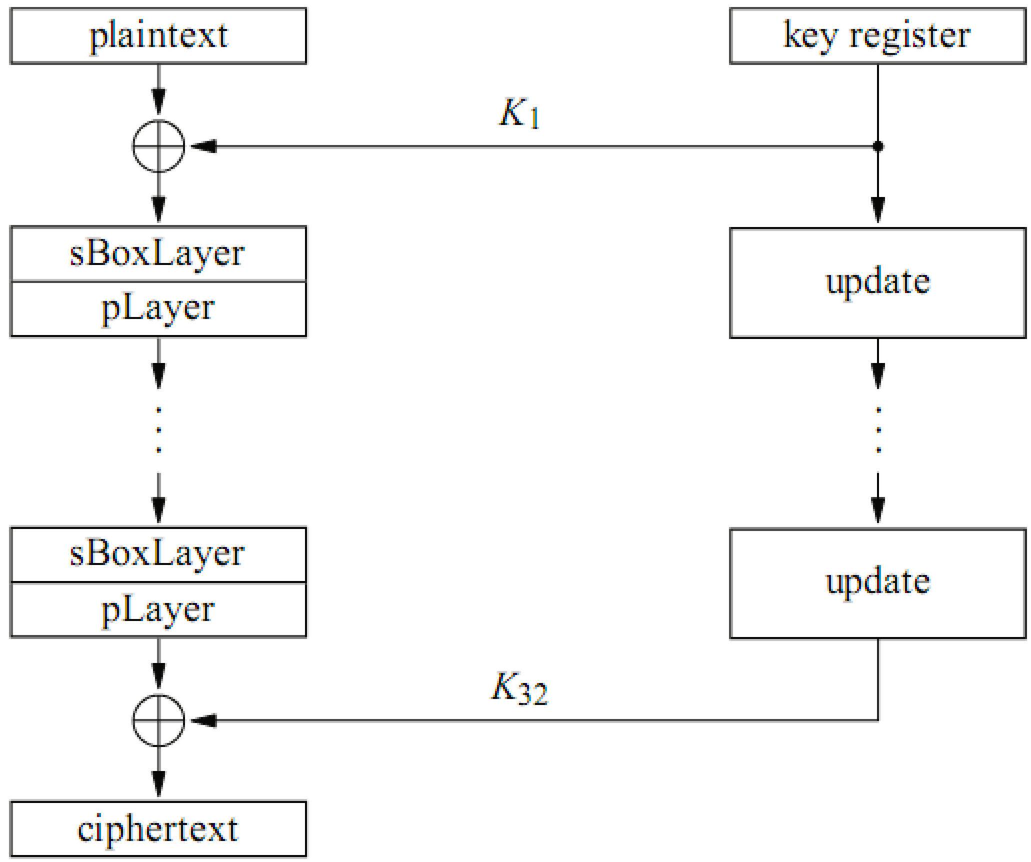
$$y = DES_{k_3}(DES_{k_2}^{-1}(DES_{k_1}(x)))$$

A different approach for strengthening DES is to use key whitening

$$y = DES_{k,k_1,k_2}(x) = DES_k(x \oplus k_1) \oplus k_2$$

Lightweight Cipher PRESENT

generateRoundKeys()
for $i = 1$ to 31 **do**
  addRoundKey(STATE,K_i)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
addRoundKey(STATE,K_{32})



Next class

- Advanced Encryption Standard (AES)