DES

Understanding Cryptography: Chapter 3

Prepared by: Farah Samir Barakat

DES

- “Data Encryption Standard”
- Symmetric-key Cipher
  - Uses same key for encryption and decryption

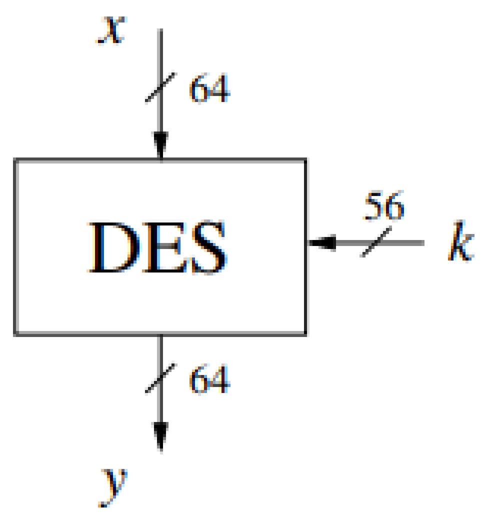
- Block Cipher (most popular)
  - Algorithm is applied to a block of data (rather than bit by bit)

- Not considered secured
  - DES key is too small
  - Can be cracked using Brute Force Attack

DES (cont.)

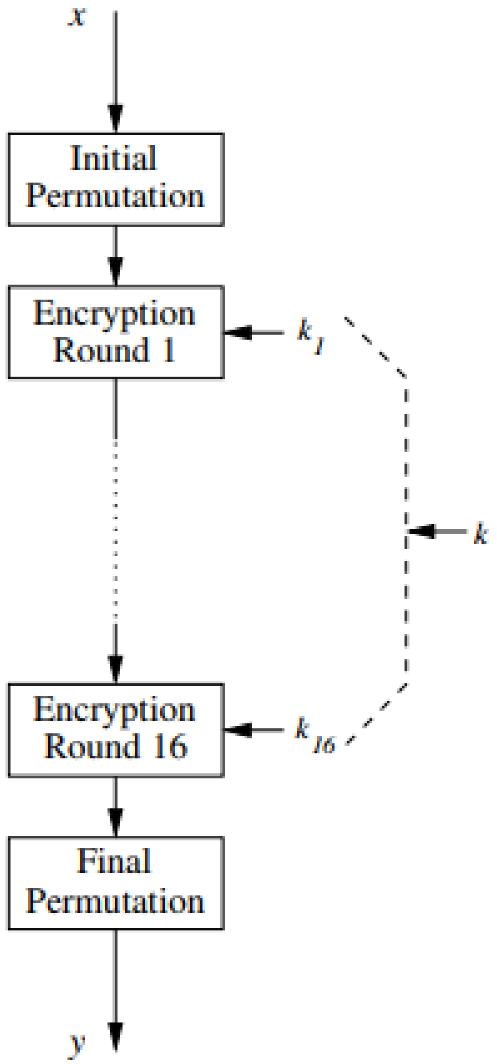
- Block length: 64 bits (16 hexadecimal bits)

- Key size: 56 bits.



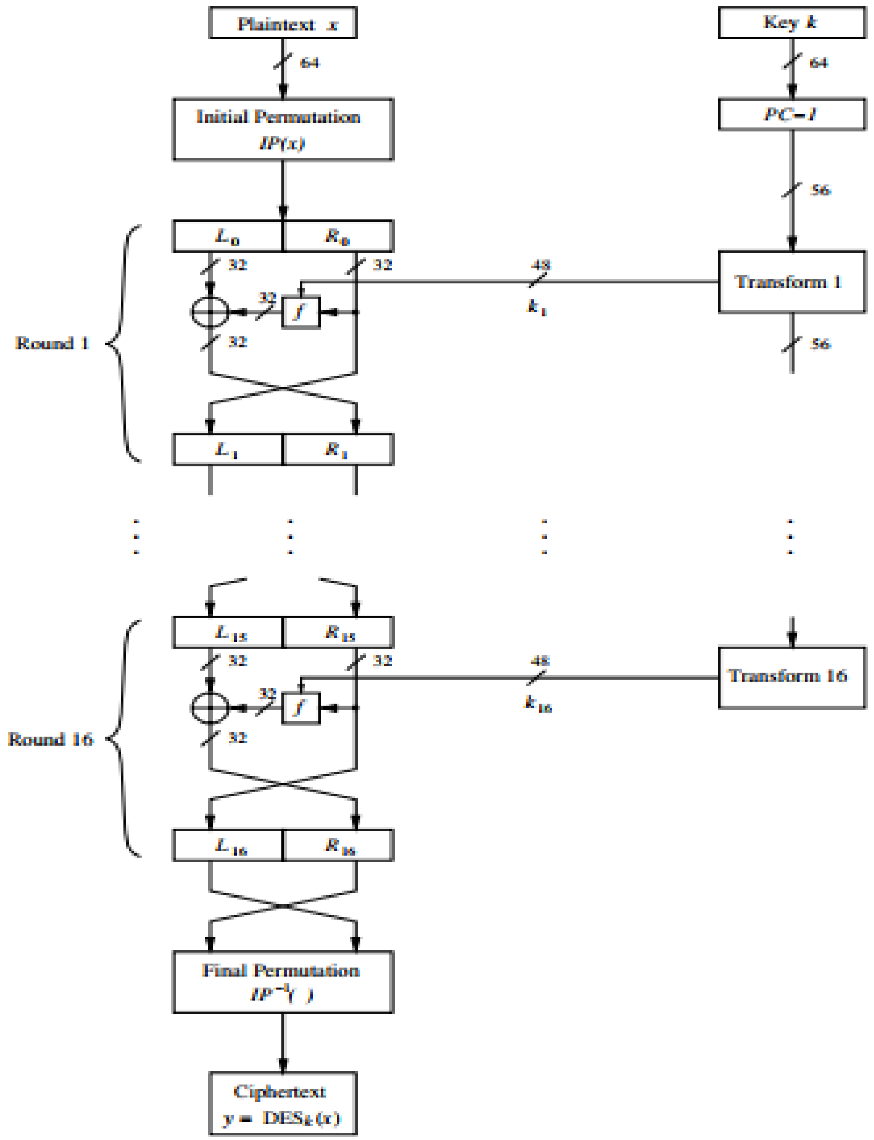
DES (cont.)

- Product cipher (iterative algorithm)
- Each block is handled in 16 rounds
  - (all perform identical operation)
- Different sub-key in each round.
  - (sub-key is derived from the main key)



DES – Fiestel Structure

- DES in an implementation of Feistel cipher
  - Encryption and decryption are almost the same operation
    - Decryption only requires a reversed key schedule
    - Advantage in software and hardware implementation

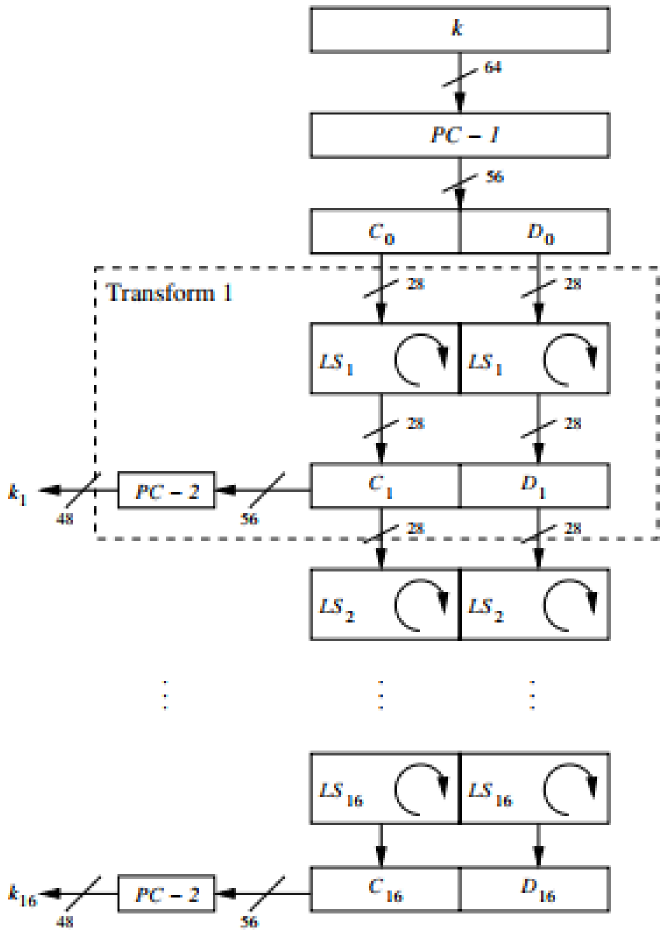


DES Building Blocks

1. Key Schedule
2. Initial and Final Permutation
3. Actual DES Rounds & F-function

Key Schedule

Key Schedule (cont.)



Key Schedule

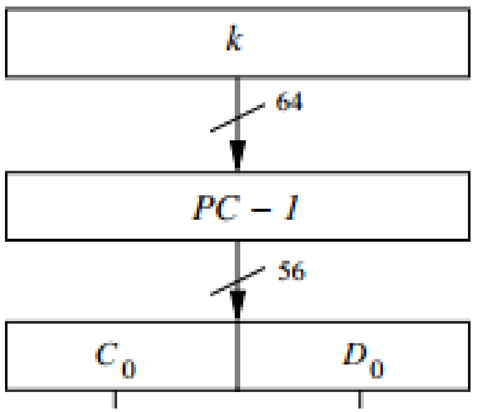
- Keys are originally stored as 64 bits
- They key is reduced to 56 bits by ignoring every 8^{th} bit in the key
  - Bits numbered 8,16,24,32,40,48,56 and 64 are eliminated when the sub-keys are created
- The key schedule derives 16 keys “Ki” (one for each round)
  - Consisting of 48 bits from the original 56-bits
  - Round-Key = Sub-Key
- Key Schedule is very easy to implement in hardware

Key Schedule (create 16 subkeys)

- The 64-bit key is permuted according to the table “PC-1”.
    - Since the first entry in the table is "57“ →this means that the 57th bit of the original key K becomes the first bit of the permuted key K+.
    - The 49th bit of the original key becomes the second bit of the permuted key.
    - The 4th bit of the original key is the last bit of the permuted key.
- Note only 56 bits of the original key appear in the permuted key.
- Next, split this key into left and right halves, C0 and D0, where each half has 28 bits

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 Schedule (create 16 subkeys) (example)

- Original 64-bit key **K** = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

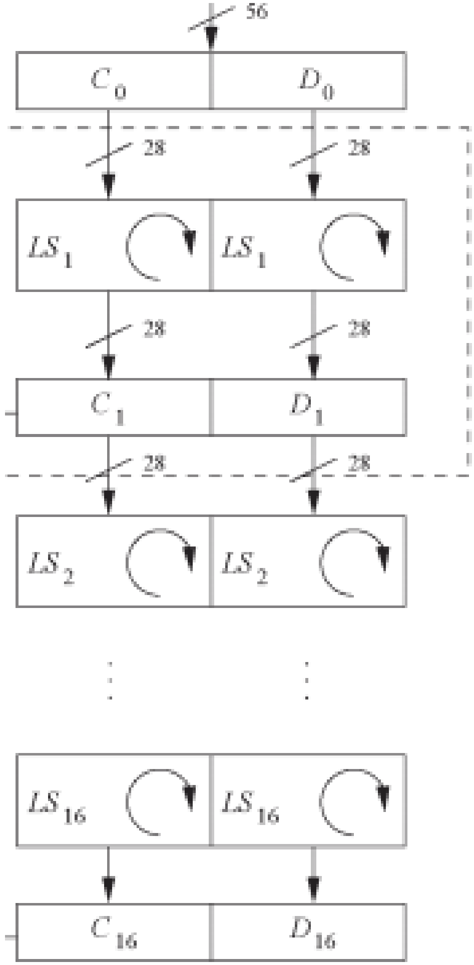
- 56-bit permutation **K+** = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

- C0 = 1111000 0110011 0010101 0101111

- D0 = 0101010 1011001 1001111 0001111

Key Schedule (cont.)

- With C0 and D0 defined, we now have sixteen blocks Cn and Dn, 1<=n<=16.
- Each pair of blocks Cn and Dn is formed from the previous pair Cn-1 and Dn-1, respectively, for n = 1, 2, ..., 16, using left shifting.
  - In rounds 1, 2, 9 ,16, the two halves are each rotated left by one bit.
  - In all other rounds where the two halves are each rotated left by two bits
- To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

Key Schedule (cont.) (example)

- From **C0** = 1111000011001100101010101111 and
  D0 = 0101010101100110011110001111

- By shifting C0 & D0 1 bit to the left we get:
  - C1 = 1110000110011001010101011111
  - D1 = 1010101011001100111100011110
- By shifting C1 & D1 1 bit to the left we get:
  - C2 = 1100001100110010101010111111
  - D2 = 0101010110011001111000111101
- By shifting C2 & D2 2-bits to the left we get:
  - C3 = 0000110011001010101011111111
  - D3 = 0101011001100111100011110101

And so on till we get C16 & D16

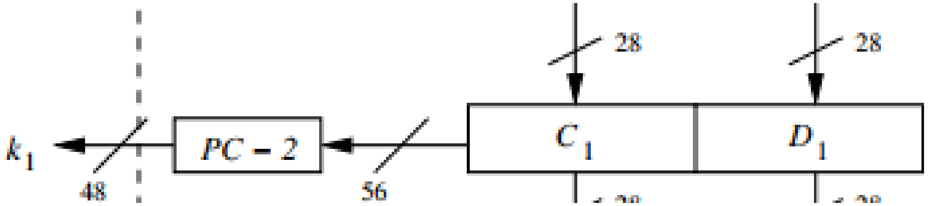
Key Schedule (create 16 subkeys)

- To form the keys Kn, for 1<=n<=16, the following permutation table PC-2 is applied to each pair of CnDn.

- Each pair has 56 bits, but PC-2 only uses 48 of these.
  - The first bit of ***Kn*** is the 14th bit of ***CnDn***
  - The second bit of ***Kn*** is the 17th bit of CnDn
  - The 48th bit of ***Kn*** is the the 32th bit of ***CnDn***.

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



Key Schedule (cont.) (example)

- For the first key we have
  - C1D1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110
- After we apply the permutation PC-2
  - K1 = 000110 110000 001011 101111 111111 000111 000001 110010
  - K2 = 011110 011010 111011 011001 110110 111100 100111 100101
  - K3 = 010101 011111 110010 001010 010000 101100 111110 011001
  - ……………..

And so on till we get K16

Encryption

DES Encryption (Internal Structure)

- The message/text (input) size must be a multiple of 64 bits.
  - Otherwise → padding (adding extra bytes to the end of the message)
    - Once the encrypted message has been decrypted → these extra bytes are thrown away

- Each block of 64-bits is divided into 2 blocks of 32 bits
  - L → left
  - R → right

DES Encryption (Internal Structure)(cont.)

- Message **“M”** in hexadecimal: 0123456789ABCDEF

- **M** in binary: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

- **L** = 0000 0001 0010 0011 0100 0101 0110 0111

- **R** = 1000 1001 1010 1011 1100 1101 1110 1111

Permutation

DES Internal Structure (cont.)

- Initial Permutation (IP) and Final permutation (IP-1)
  - Bitwise permutations
  - Simple cross wiring
  - Easily implemented in Hardware (but not fast in software)
  - Don’t increase the security of DES

Initial Permutation "IP"

- There is an *initial permutation* **IP** of the 64 bits of the message **M**.
- This rearranges the bits according to table "IP".
    - The 58th bit of **M** becomes the first bit of **IP**.
    - The 50th bit of **M** becomes the second bit of **IP**.
    - The 7th bit of **M** is the last bit of **IP**.

- Next the permuted block IP is divided into
    - a left half L0 of 32 bits
    - a right half R0 of 32 bits

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

Initial Permutation (Example)

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

- After applying the initial permutation to M, we get:
  - IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010
  - The 58th bit of **M** is "1“ → becomes he first bit of **IP**.
  - The 50th bit of **M** is "1“→ becomes the second bit of **IP**.
  - The 7th bit of **M** is "0“ → becomes the last bit of **IP**

- **L0** = 1100 1100 0000 0000 1100 1100 1111 1111

- **R0** = 1111 0000 1010 1010 1111 0000 1010 1010

Actual DES Rounds

and

F - function

F function

- Plays a crucial role for the security of DES
- Confusion and Diffusion are realized within f
  - Confusion: relationship between key and ciphertext is unknown
  - Diffusion: influence of 1 plaintext symbol is spread over many ciphertext symbols (to hide statistical properties of the plaintext)
- Input:
  - Data block of 32 bits
  - Key Kn of 48 bits
- Output
  - Block of 32 bits

F function (cont.)

- 1<=***n***<=16 (16 iterations)
- In round n
  - the left 32 bits of the current step are the right 32 bits of the previous result “Rn-1”
  - The right 32 bits of the current step are the left 32 bits of the previous step “Ln-1” XORed with the calculation f .
  - Let + → XOR addition, (bit-by-bit addition modulo 2)

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

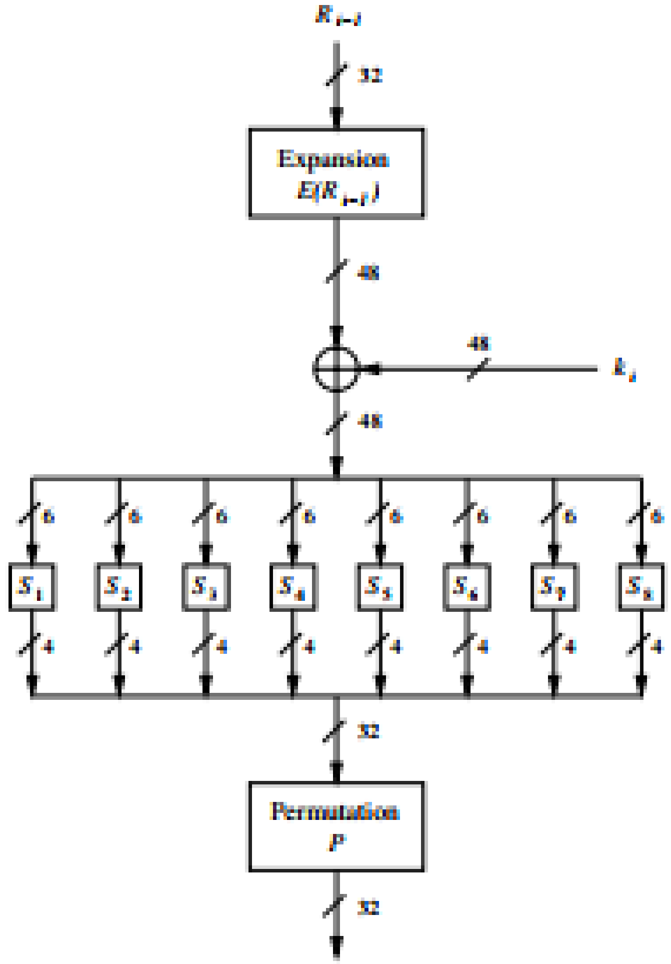
F function (cont.) (example)

- **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
  - **L0** = 1100 1100 0000 0000 1100 1100 1111 1111
  - **R0** = 1111 0000 1010 1010 1111 0000 1010 1010
- **K1** = 000110 110000 001011 101111 111111 000111 000001 110010

- For n = 1
  - L1 = R0 = 1111 0000 1010 1010 1111 0000 1010 1010
  - R1 = L0 + f(R0,K1)

F function (cont.)

- 4 Steps:
  - Expansion E
  - XOR with round key
  - S-box substitution
  - Permutation



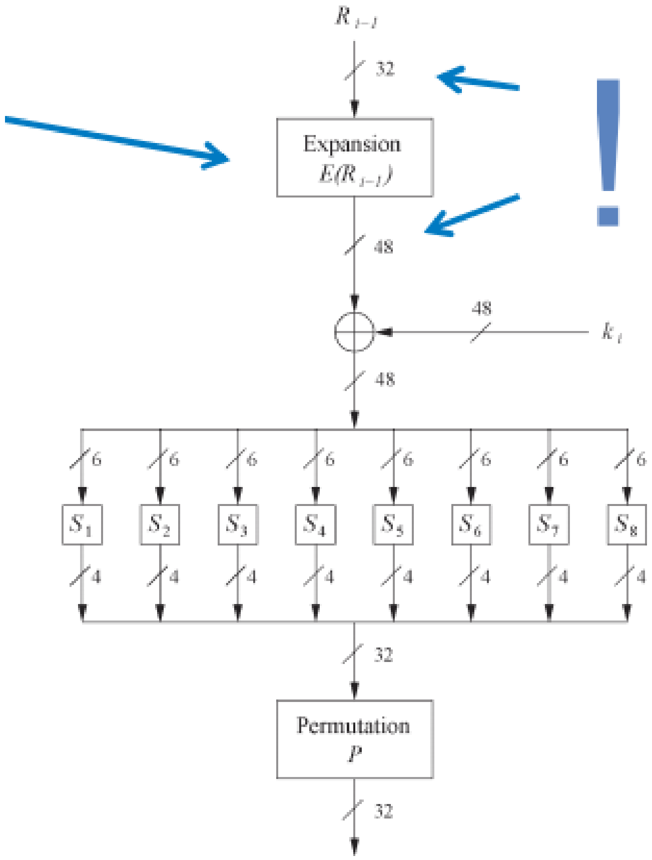
F calculation - Expansion

- Expand each block Rn-1 from 32 bits to 48 bits done by
  - Partitioning the input into eight 4-bit blocks
  - Expanding each block to 6 bits
  - Using a selection E box that repeats some of the bits in Rn-1 → function E
  - E-box is a special type of permutation

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

- E(Rn-1) has
  - 32 bit input block
  - 48 bit output block.

- Main purpose of expansion
  - Increase diffusion of DES
    - Since certain input bits influence 2 different output locations



(Note that each block of 4 original bits has been expanded to a block of 6 output bits.)

F calculation – Expansion (Example)

- R0 = 1111 0000 1010 1010 1111 0000 1010 1010

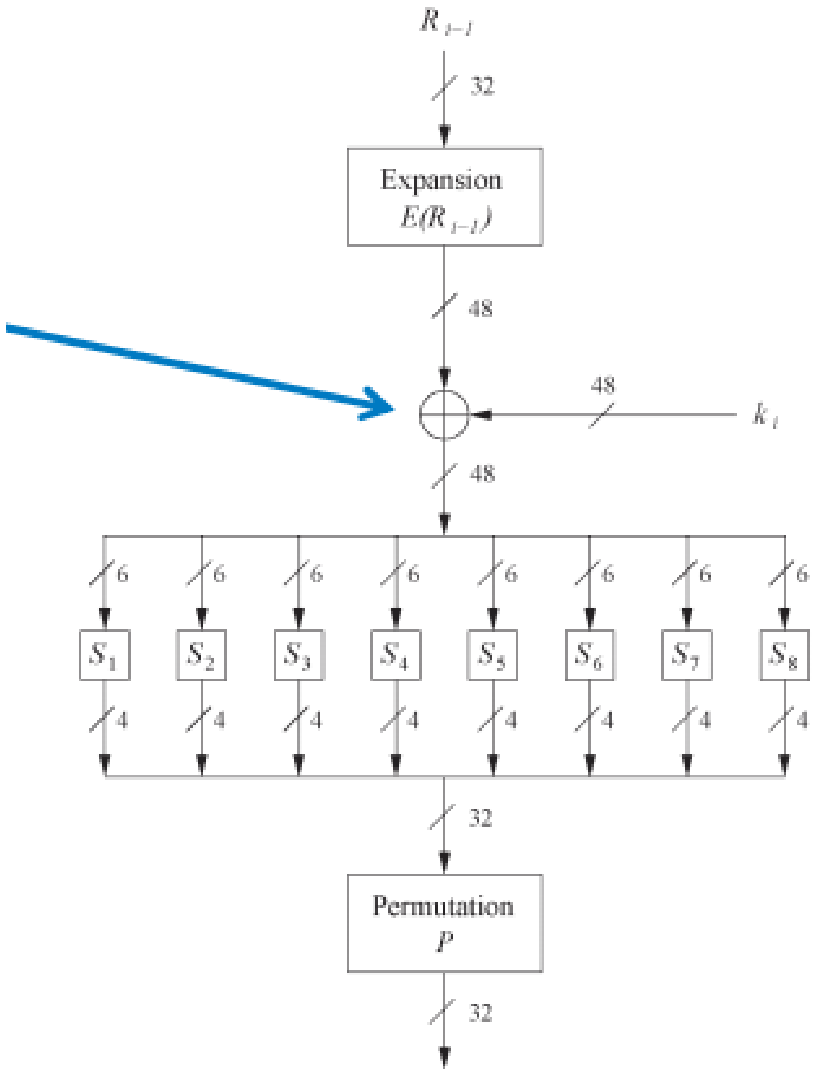
- Steps:
  - The first three bits of E(Rn-1) are the bits in positions 32, 1 and 2 of Rn-1
  - ……………………
  - The last 2 bits of E(Rn-1) are the bits in positions 32 and 1

- E(R0) = 011110 100001 010101 010101 011110 100001 010101 010101

F calculation - XoR

- Next in the ***f*** calculation
  - The 48-bit result of the expansion **E**(***Rn-1***) is XORed with the key ***Kn***

$$\boldsymbol{K_n} + \mathbf{E}(\boldsymbol{R_{n\text{-}1}}).$$



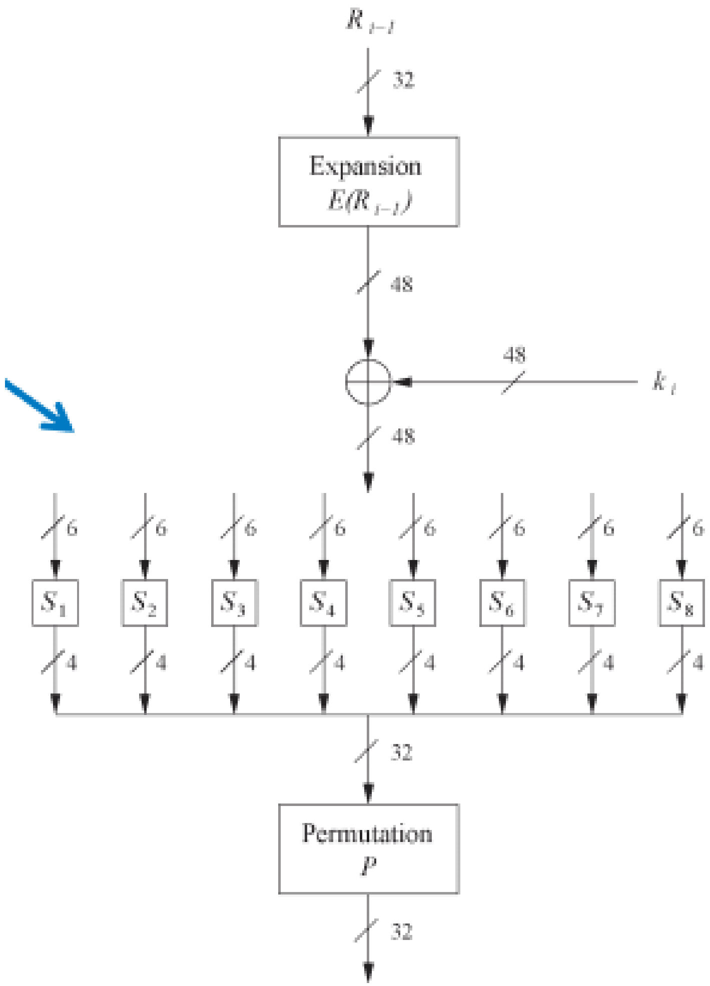
F calculation (example)

- **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.

F calculation – S-box substitution

- After expanding Rn-1 from 32 bits to 48 bits, using the selection table, and XORed the result with the key Kn → we now have 48 bits (eight groups of six bits).
- The eight 6-bit blocks are fed into 8 different substitution boxes “S-boxes”
  - (each group of six bits is used as addresses in"S-boxes“ tables.)
- Each S-box is a lookup table → maps a 6-bit input into a 4-bit output
  - Each group of six bits will give us an address in a different S-box.
  - Located at that address will be a 4 bit number.
  - This 4 bit number will replace the original 6 bits.
- The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the S-boxes)



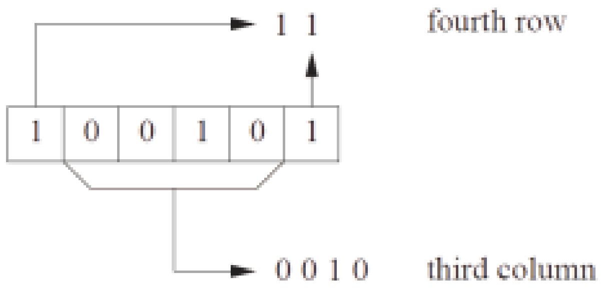
S-box

- Each S-box contains 64 entries.
- Each S-box is represented by a table with 16 columns and 4 rows
- The most significant bit “MSB” and the lease significant bit “LSB” of each 6-input → row of the table.
- The four inner bits → column of the table

S-box (example)

- Decoding of the input $(100101)_2$ by S-box 1
  - $(11)_2$ → 3 so the 4th row
  - $(0010)_2$ → 2 so the 3rd column

  - Note: The counting begins from 0 that's why 3 represents 4th row [0 1 2 3]



S-box (cont.)

- The S-boxes are fixed, but each different from the other.
- They were carefully designed to thwart advanced mathematical attacks ex: differential cryptanalysis.
- The S-boxes are the core of DES in terms of cryptographic strength.
  - They are the only nonlinear element in the algorithm
    - Otherwise an attacker could express the DES input and output with a system of linear equations.
  - They provide confusion

S-box (cont.)

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_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_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_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 (cont.)

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_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_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

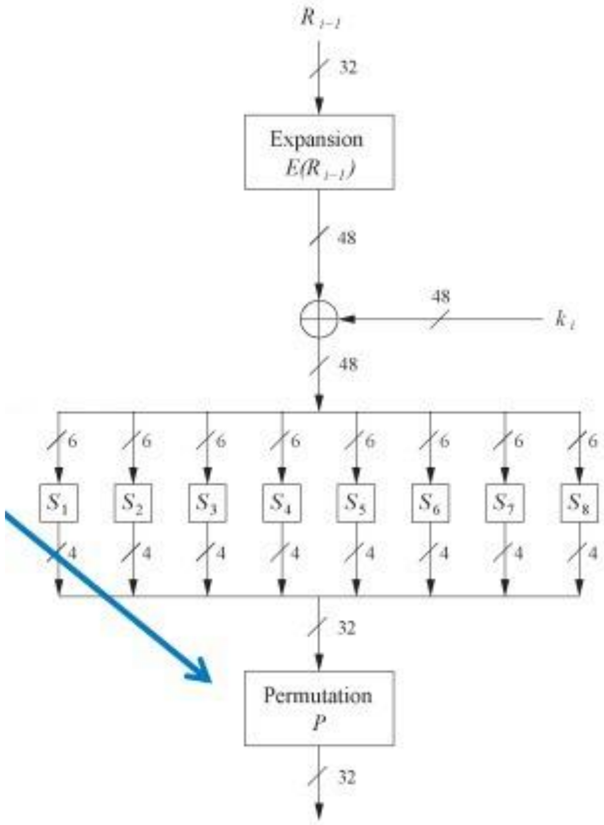
S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
3	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

F calculation - Permutation “P”

- The final stage in the calculation of f is to do a permutation P of the S-box output to obtain the final value of f.

- P yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

- Introduces diffusion (unlike IP and IP-1)
    - The 4 output bits of each S-box are permuted in such a way that they affect several different S-boxes in the following round.



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

Avalanche Effect

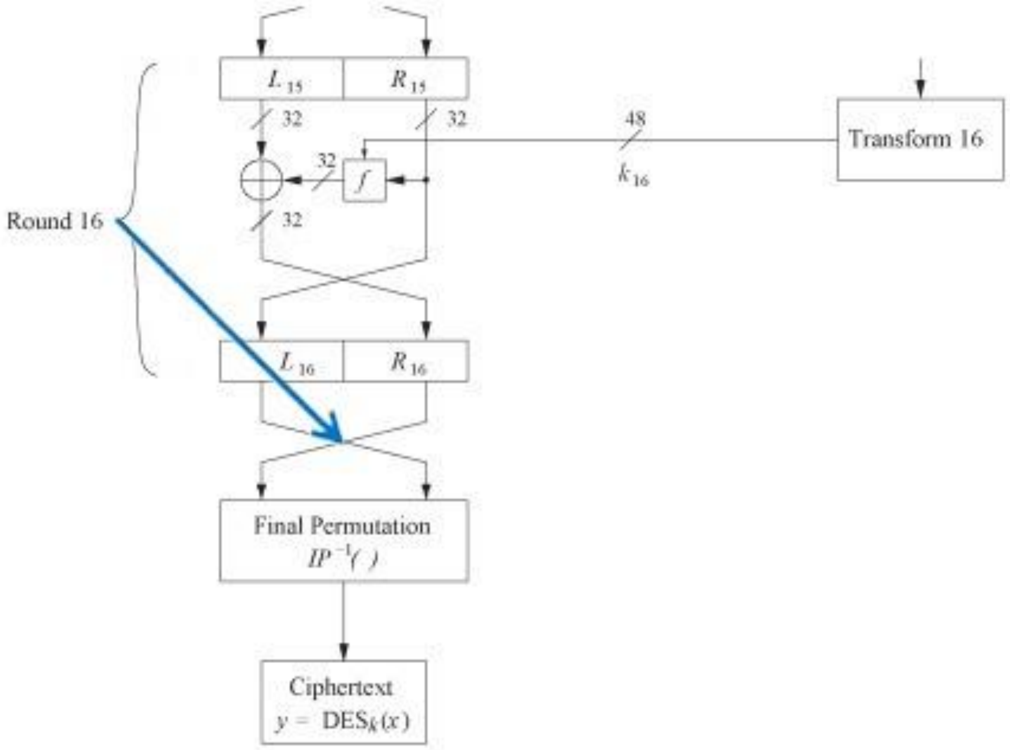
- The diffusion caused by the expansion, S-boxes and the permutation P guarantees that every bit at the end of the fifth round is a function of every plaintext bit and every key bit.

- This behavior is known as the "*Avalanche Effect*."
  - "A small change in plaintext results in the very great change in the ciphertext."

Final Permutation (IP-1)

- At the end of the sixteenth round we have the blocks L16 & R16
- We then reverse the order of the two blocks into the 64-bit block R16L16

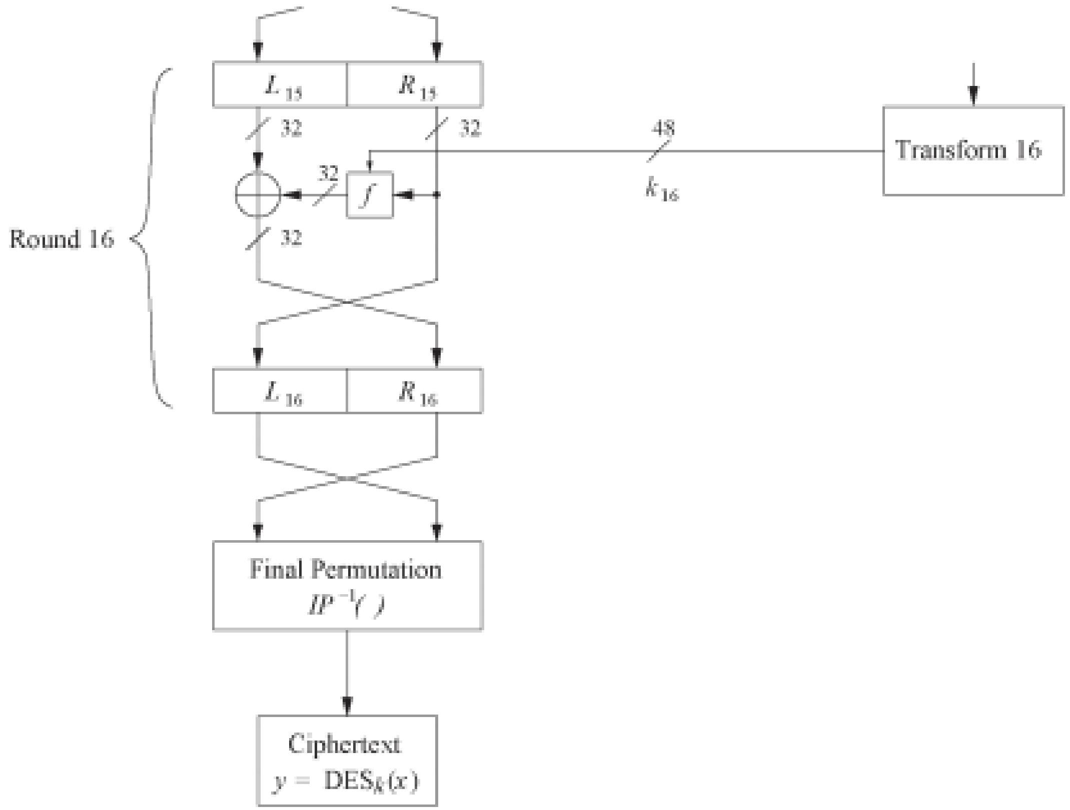
- Apply a final permutation **IP-1** as defined by table IP-1
- The output of the algorithm has
  - bit 40 of the pre-output block as its first bit
  - bit 8 as its second bit, and so on..
  - until bit 25 of the pre-output block is the last bit of the output



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

Final Permutation (IP-1) (cont.)



Example

- L16 = 0100 0011 0100 0010 0011 0010 0011 0100
- R16 = 0000 1010 0100 1100 1101 1001 1001 0101

- We reverse the order of these two blocks and apply the final permutation
  - R16L16 = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

- IP-1 = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101
- which in hexadecimal format is 85E813540F0AB405 → Ciphertext

Decryption

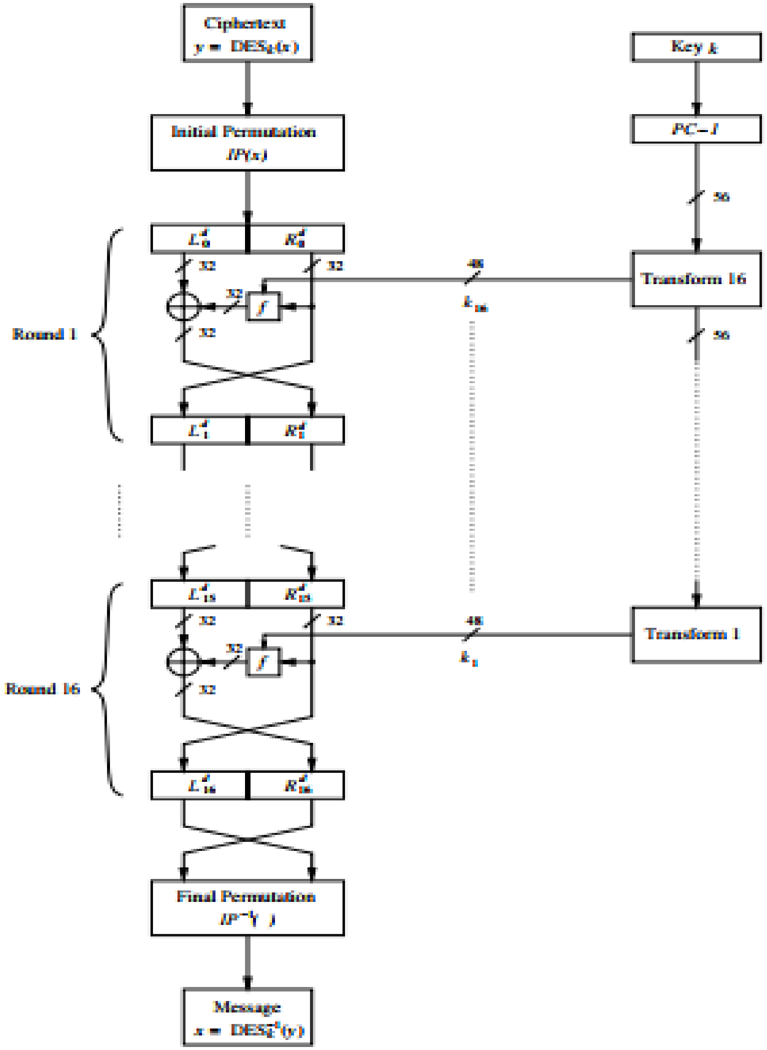
DES - Decryption

- Decryption is simply the inverse of encryption

- The same steps, but reversing the order in which the sub-keys are applied.
  - In round 1 → sub-key 16 is needed
  - In round 2 → sub-key 15 is needed
  - ……
  - Round-keys: K16, K15,K14,…….K1.

- To compute kn we need Cn and Dn which can be derived from Cn+1 and n+1 through cyclic right shifts (RS)
  - In decryption round 1, the key is not rotated.
  - In decryption rounds 2, 9, and 16 the two halves are rotated right by one bit.
  - In the other rounds 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14 and 15 the two halves are rotated right by two bits

DES – Decryption (cont.)



DES – Decryption Key Schedule