Data Security and Privacy
DSE 3258

L4 -BLOCK CIPHER

Stream cipher and Block Cipher

- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time
- However, the keystream must be provided to both users in advance via some independent and secure channel. This introduces insurmountable logistical problems if the intended data traffic is very large.
- Accordingly, for practical reasons, the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users. In this approach ,the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong..
- A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key

Stream cipher and Block Cipher

Confusion and Diffusion

Two basic building blocks for any cryptographic system

Confusion:

- Making the relationship between encryption key and the cipher text as complex as possible.
- Relationship between PT and CT is obscured.
- That is given CT no information about PT, Key, encryption algorithm.
- Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by the use of a complex substitution algorithm.
- Ex: substitution

Diffusion:

- Making each plaintext bit affect as many CT bits as possible
- One bit change in PT has significant change in CT. This is equivalent to having each ciphertext digit be affected by many plaintext digits
- An example of diffusion is to encrypt a message M = m1, m2, m3, c of characters with an averaging operation:

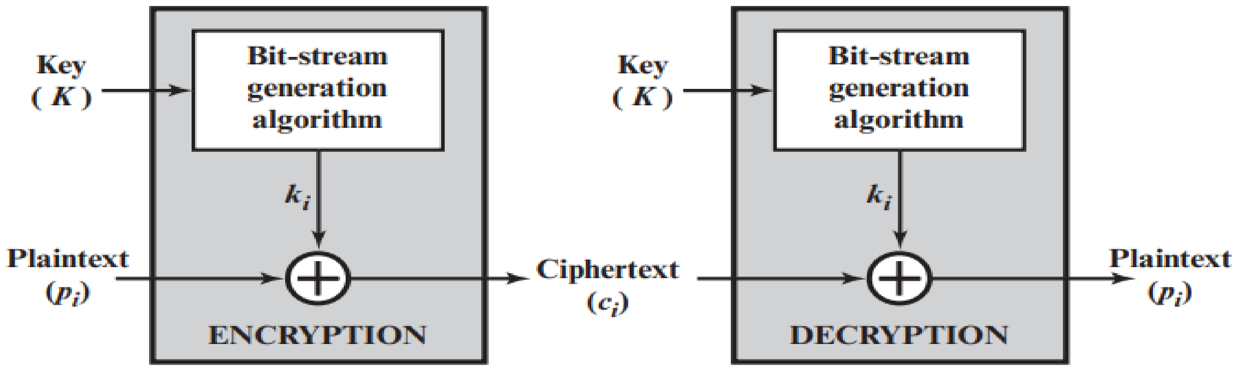
$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

- Ex: permutation or transposition

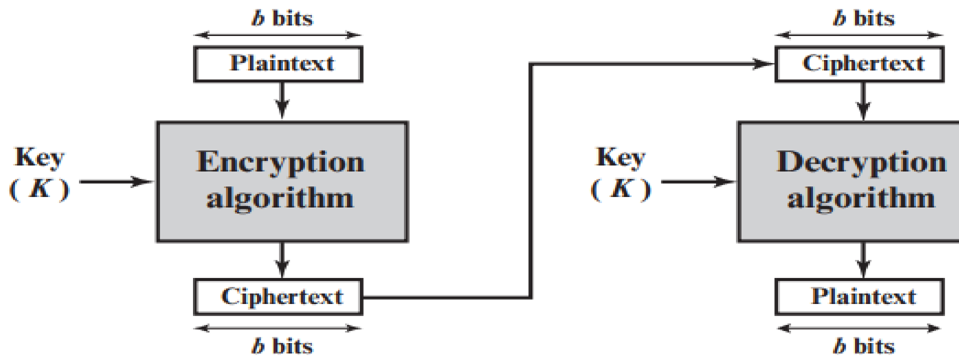
Stream cipher and Block Cipher

Block Cipher	Stream Cipher
Chunk of PT is taking as input and converted to CT	Bit-by-bit PT is converted to CT
PT size is 64 bits or more (multiple of 64)	PT size is 8 bits
Simple operations	Complex operations
Uses confusion and diffusion	Uses only confusion
Decipher is hard	Decipher is easy
Block cipher works on transposition techniques like rail-fence technique, columnar transposition technique, etc.	While stream cipher works on substitution techniques like Caesar cipher, polygram substitution cipher, etc.
Block cipher is slow as compared to a stream cipher.	While stream cipher is fast in comparison to block cipher.

Stream cipher and Block Cipher



(a) Stream cipher using algorithmic bit-stream generator



(b) Block cipher

Figure 4.1 Stream Cipher and Block Cipher

Block Cipher

Plaintext and ciphertext consists of fixed sized blocks
Ciphertext obtained from plaintext by iterating a **round function**
Input to round function consists of key and the output of previous round

Feistel Cipher Structure

- Feistel proposed [FEIS73] block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.
- The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible transformations.
- Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:
- ■ Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- ■ Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

Feistel Cipher Structure

- Plain text is divided into two equal halves and processed through the algorithm.
- **Feistel cipher** refers to a type of block cipher design, not a specific cipher

Encryption:

- Split plaintext block into left and right halves: Plaintext = (L_0, R_0)
- For each round $i = 1, 2, \ldots, n$, compute

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$

  where F is **round function** and K_i is **subkey**
- Ciphertext = (L_n, R_n)

Feistel Cipher

Decryption:

Ciphertext = (L_n,R_n)

- For each round $i=n,n-1,\ldots,1$, compute

  $R_{i-1} = L_i$

  $L_{i-1} = R_i \oplus F(R_{i-1},K_i)$

  where F is round function and K_i is subkey
- Plaintext = (L_0,R_0)
- Formula "works" for any function F
- But only secure for certain functions F

Feistel Cipher Structure

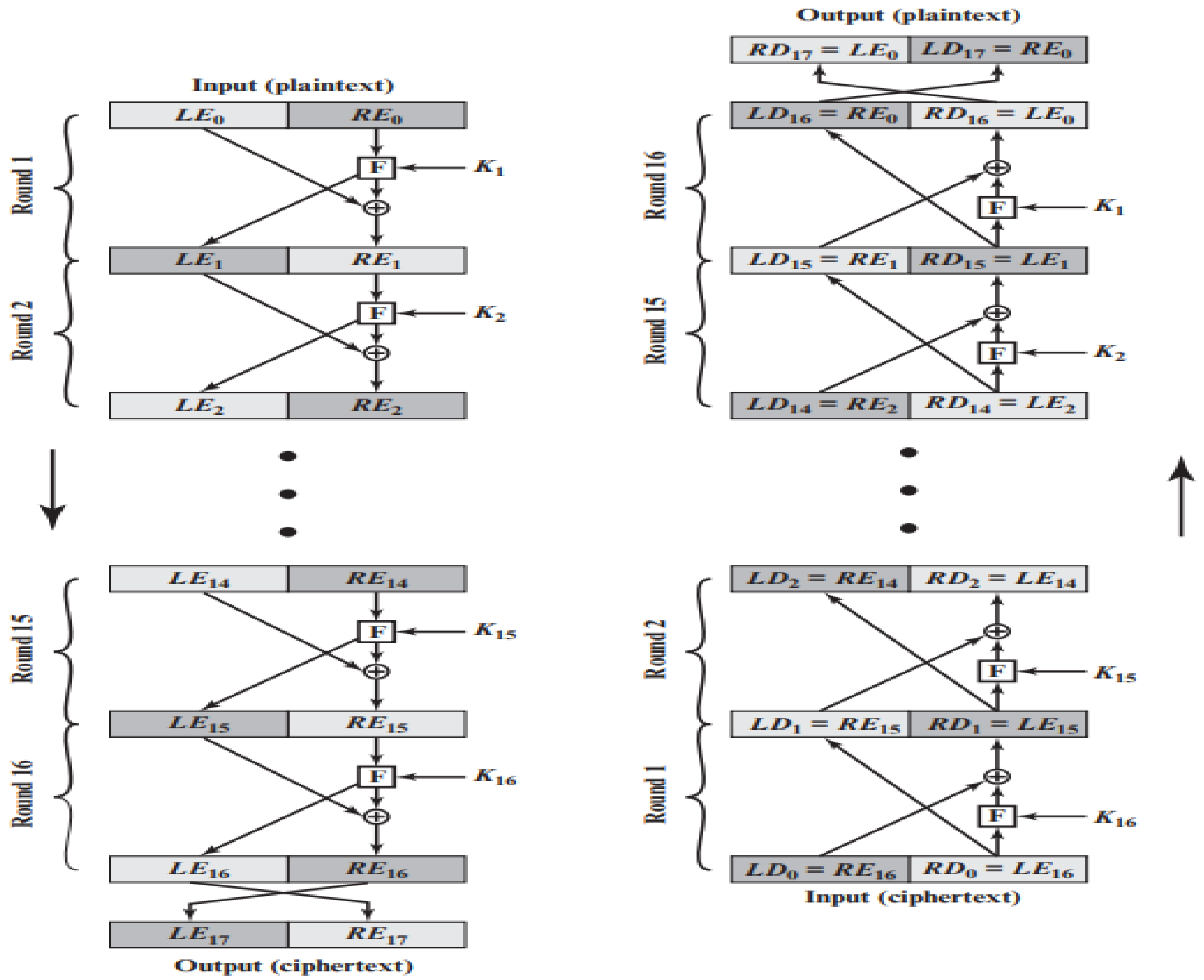
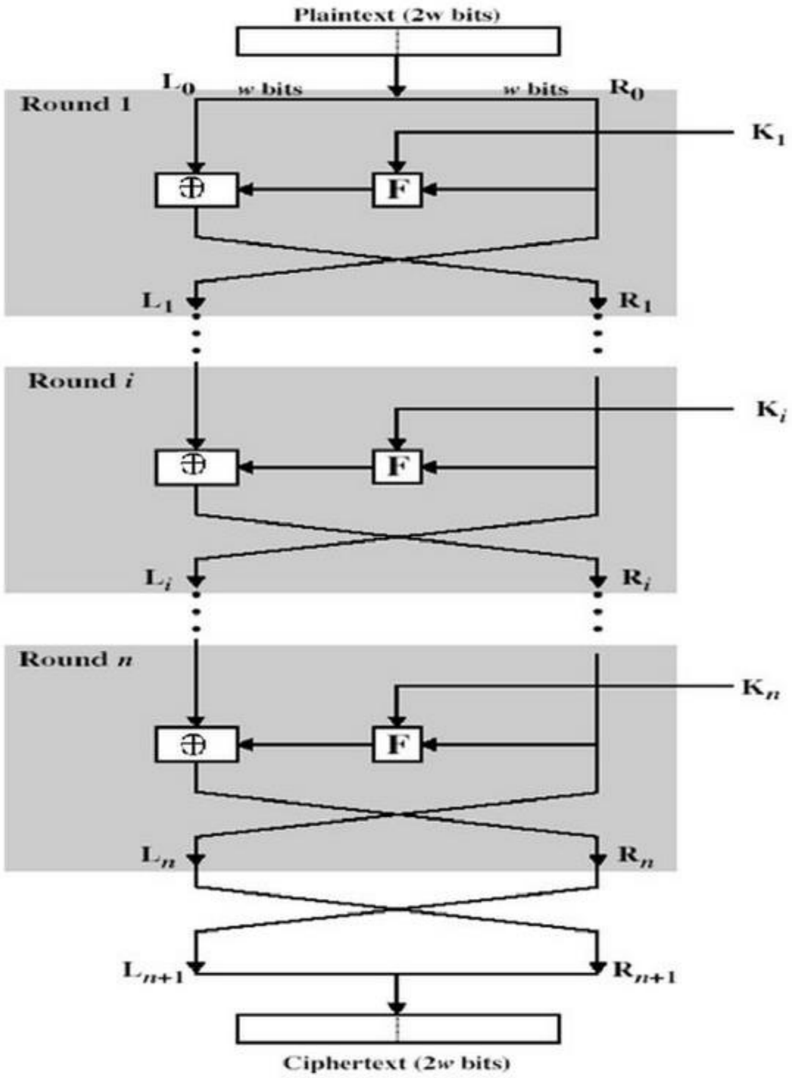


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

Feistel Cipher Structure



Feistel Cipher Structure

A Feistel network depends on the choice of the following parameters and design features:

- **Block size:** Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.
- **Key Size:** Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds:** multiple rounds offer increasing security. Typical is 16 rounds as a single round provide inadequate security.
- **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
- **Round function F** - greater complexity generally means greater resistance to cryptanalysis.

Feistel Cipher Structure

There are two other considerations in the design of a Feistel cipher:

- **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength.

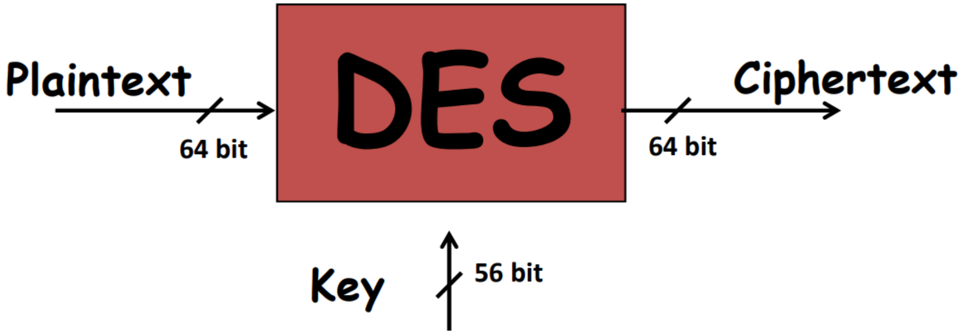
DATA ENCRYPTION STANDARD (DES)

- Symmetric block cipher.
- most widely used block cipher in world
- adopted in 1977 by National Bureau of Standards (NBS), now the National Institute of Standards and Technology (NIST
- encrypts 64-bit data using 56-bit key
- has widespread use
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption

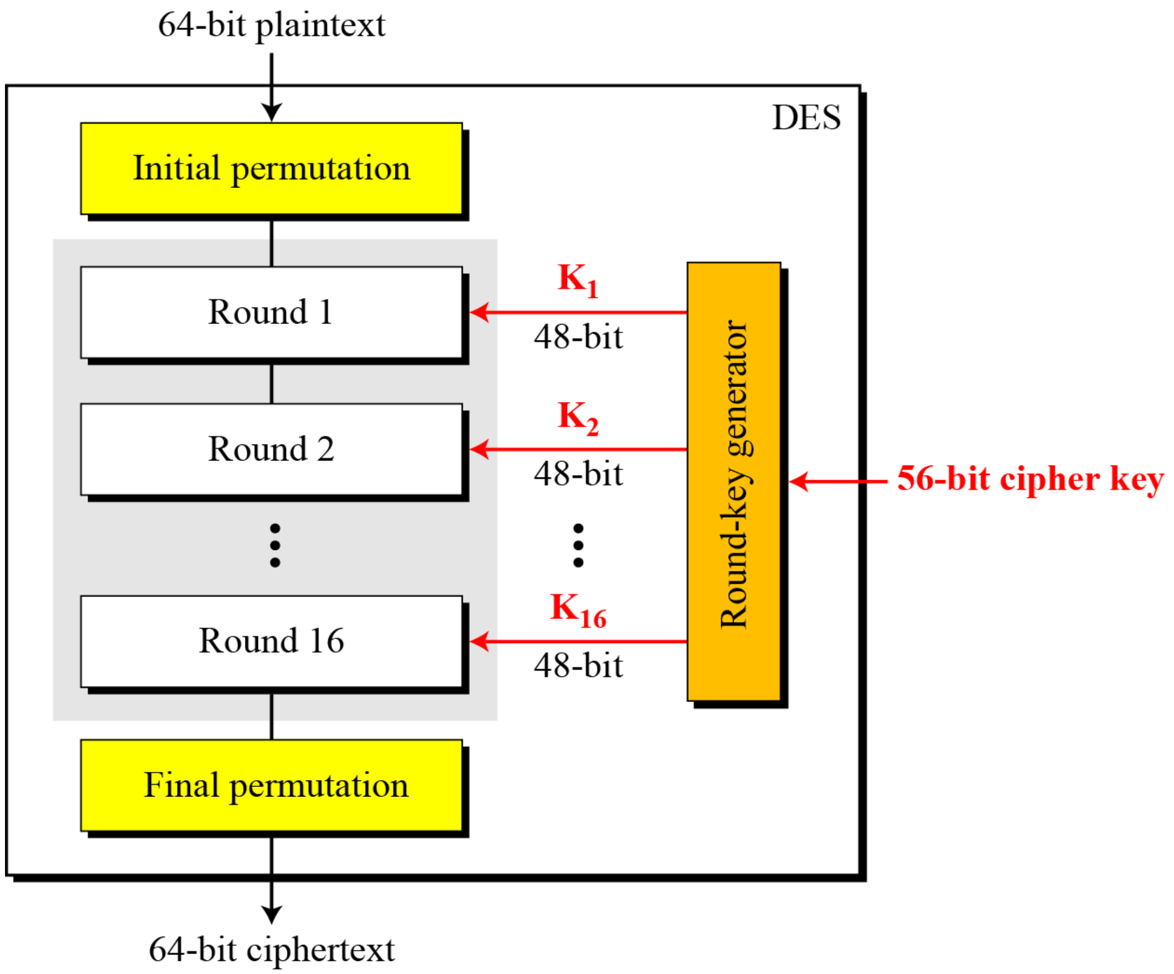
DATA ENCRYPTION STANDARD (DES)

- Follows Feistel structure.
- Block size = 64 bits of plain text
- No of rounds = 16 rounds
- Key size = 56 bits
- No of subkeys = 16 subkeys (16 rounds)
- Sub key size = 48 bits
- Cipher text size = 64 bits

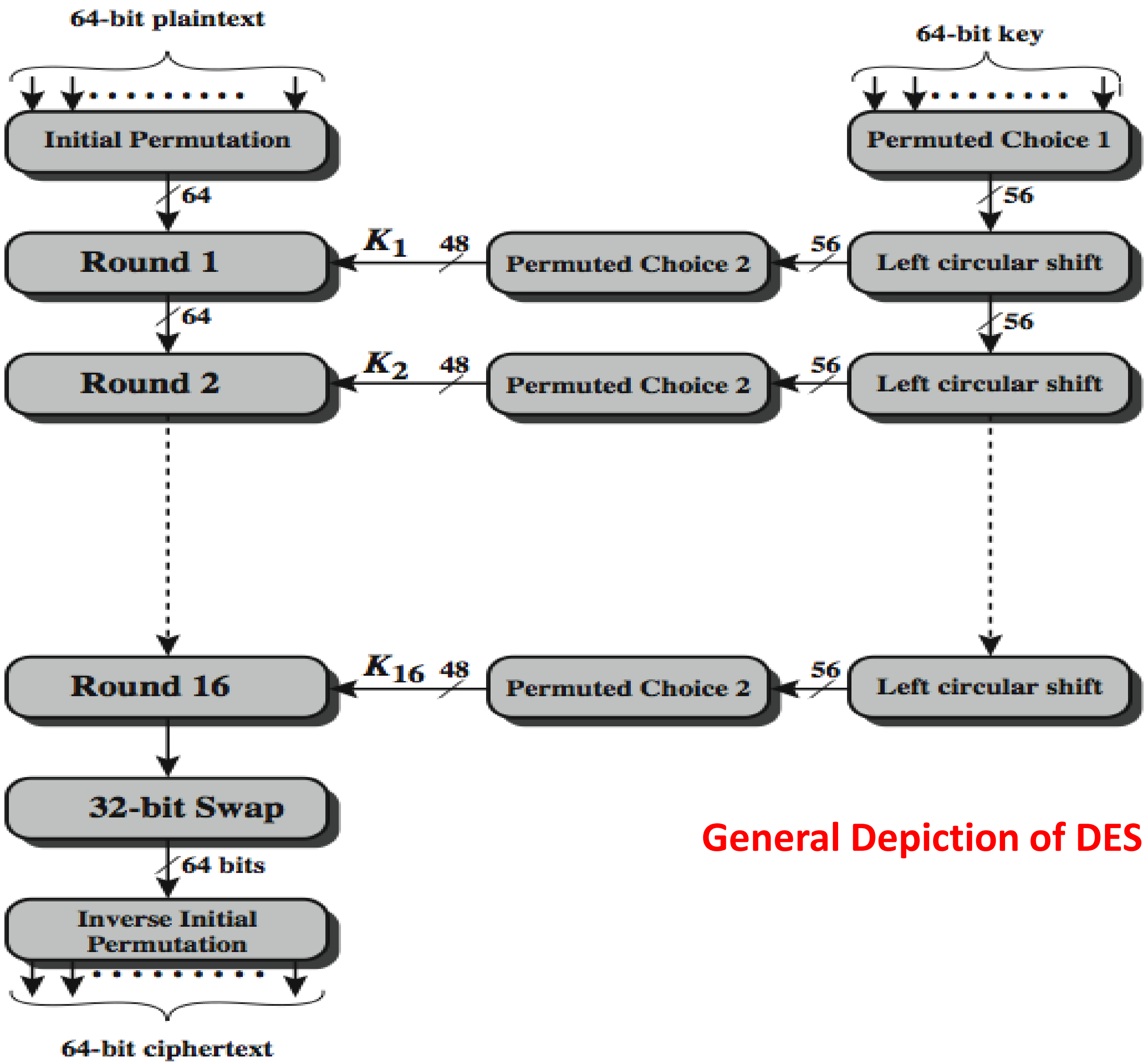
the function expects a 64-bit key as input. However, only 56 of these bits are ever used; the other 8 bits can be used as parity bits or simply set arbitrarily.



DATA ENCRYPTION STANDARD (DES)



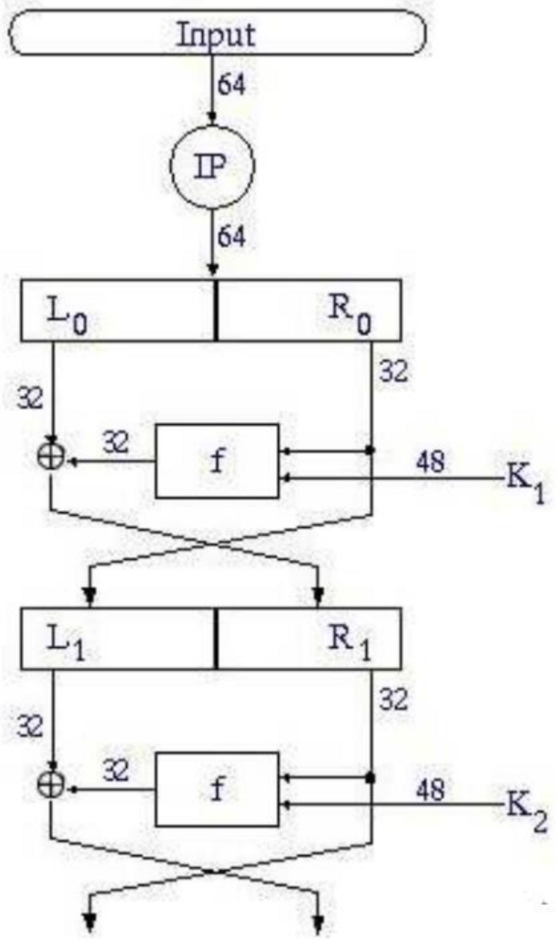
DATA ENCRYPTION STANDARD (DES)



General Depiction of DES Encryption Algorithm

DATA ENCRYPTION STANDARD (DES)

- $IP(x) = L_0R_0$
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- $y = IP^{-1}(R_{16}L_{16})$

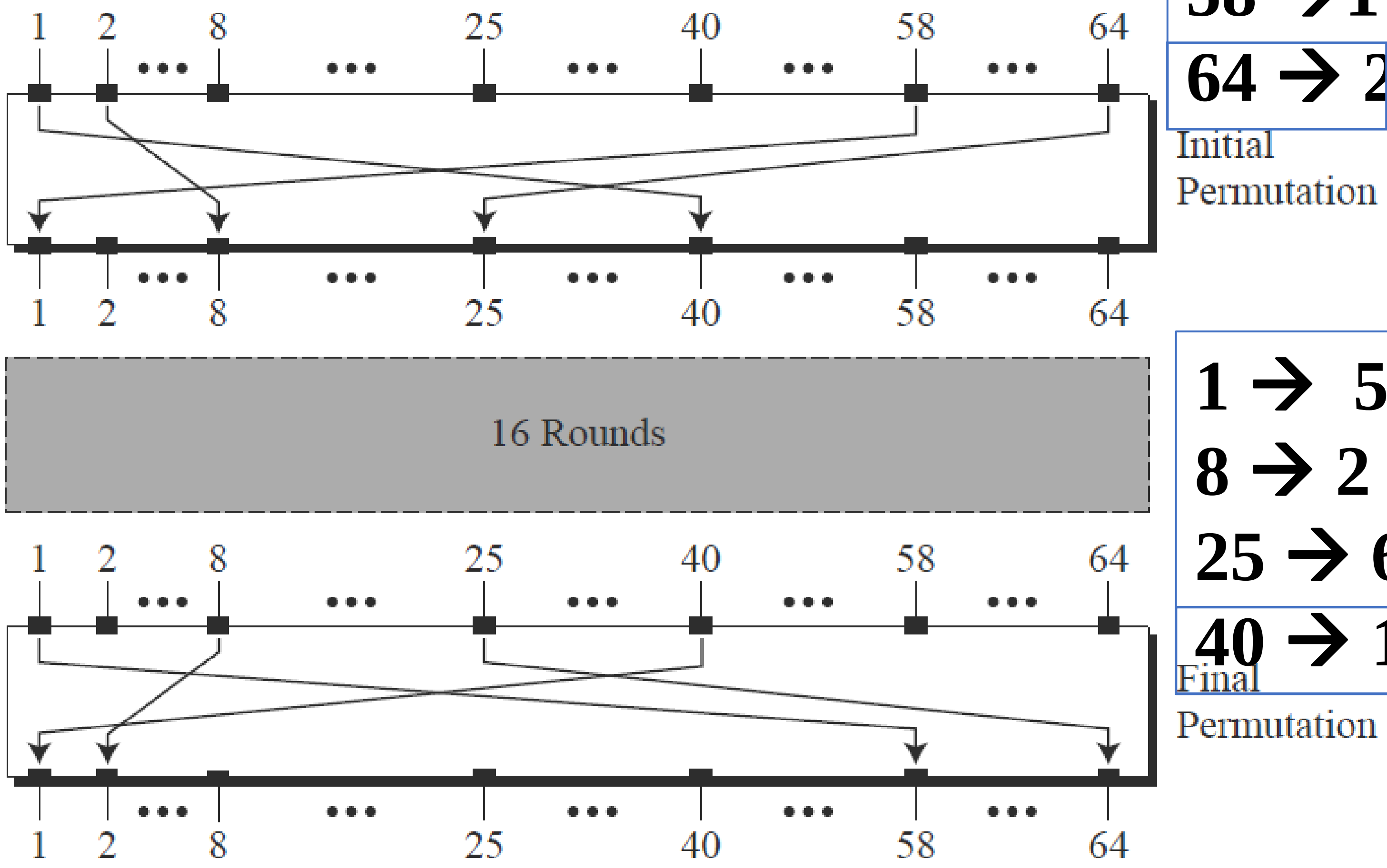


DATA ENCRYPTION STANDARD (DES)

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
- This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions.
- The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key.
- The left and right halves of the output are swapped to produce the preoutput.
- Finally, the preoutput is passed through a permutation [IP-1] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.
- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

DATA ENCRYPTION STANDARD (DES) (contd..)

1 → 40
2 → 8
58 → 1
64 → 25

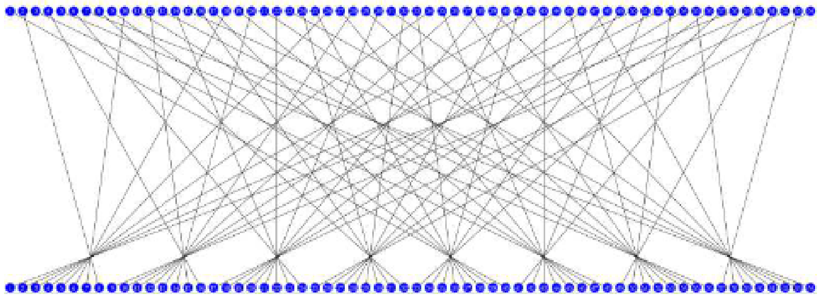


1 → 58
8 → 2
25 → 64
40 → 1

DATA ENCRYPTION STANDARD (DES) (contd..)

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



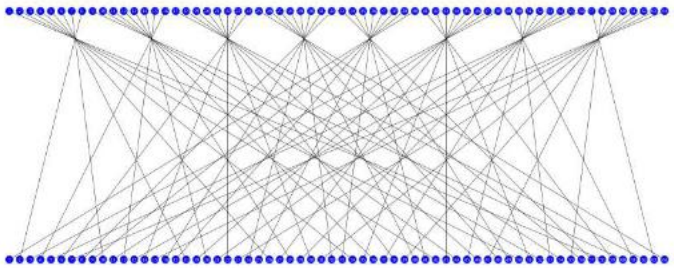
This table specifies the input permutation on a 64-bit block.

- The meaning is as follows:
- the first bit of the output is taken from the 58th bit of the input; the second bit from the 50th bit, and so on, with the last bit of the output taken from the 7th bit of the input.
- This information is presented as a table for ease of presentation:
- it is a vector, not a matrix.

DATA ENCRYPTION STANDARD (DES) (contd..)

Final Permutation (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



The final permutation is the inverse of the initial permutation; the table is interpreted similarly.

• That is, the output of the Final Permutation has bit 40 of the preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output

DATA ENCRYPTION STANDARD (DES) (contd..)

Example 1

- Example:
  - Find the output of the initial permutation box when the input is given in hexadecimal as:

0x0002 0000 0000 0001

- The input has only two 1s (bit 15 and bit 64); the output must also have only two 1s (the nature of straight permutation).
- Using Table, we can find the output related to these two bits.
- Bit 15 in the input becomes bit 63 in the output.
- Bit 64 in the input becomes bit 25 in the output.
- So the output has only two 1s, bit 25 and bit 63. The result in hexadecimal is

0x0000 0080 0000 0002

```
0: 0000 (bits 1-4)
0: 0000 (bits 5-8)
0: 0000 (bits 9-12)
2: 0010 <--bit 15 is 1 (bits 13-16)
```

```
0: 0000 (bits 49-52)
0: 0000 (bits 53-56)
0: 0000 (bits 57-60)
1: 0001 <--bit 64 is 1 (bits 61-64)
```

DATA ENCRYPTION STANDARD (DES) (contd..)

Example 2

Find the output of the initial permutation box when the input is given in hexadecimal as:

0x0000 0080 0000 0002

Solution

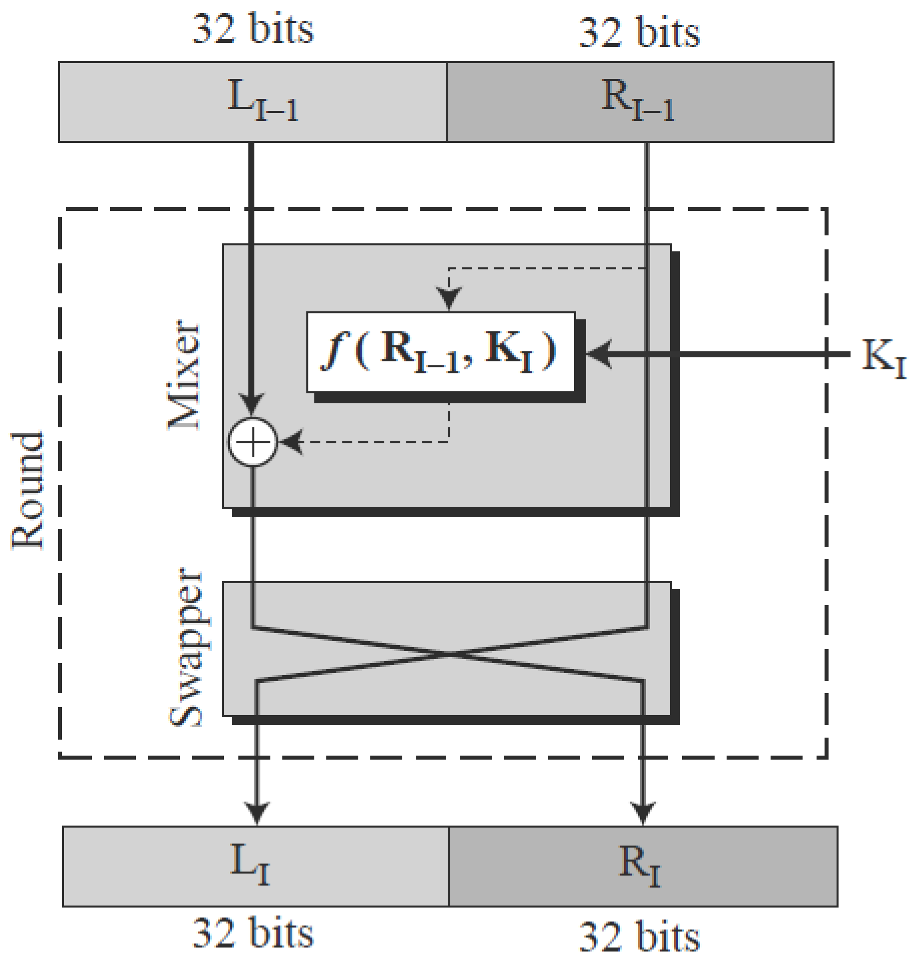
Only bit 25 and bit 64 are 1s; the other bits are 0s. In the final permutation, bit 25 becomes bit 64 and bit 63 becomes bit 15. The result is

0x0002 0000 0000 0001

DESRounds (contd..)

DES Rounds:

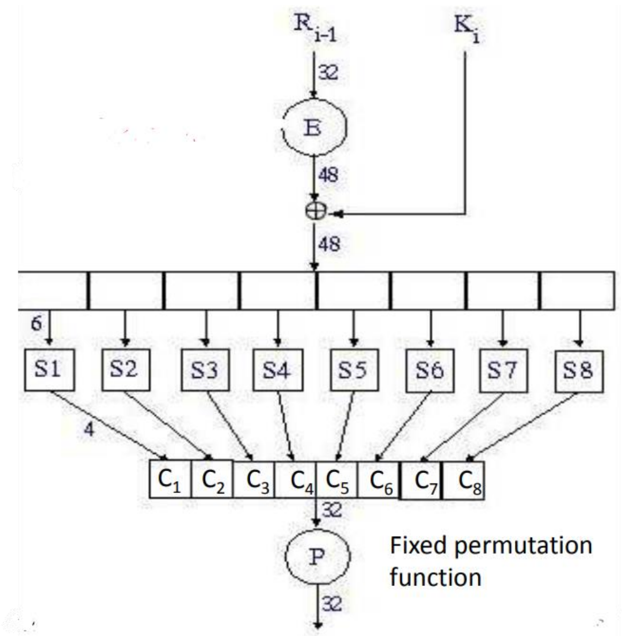
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

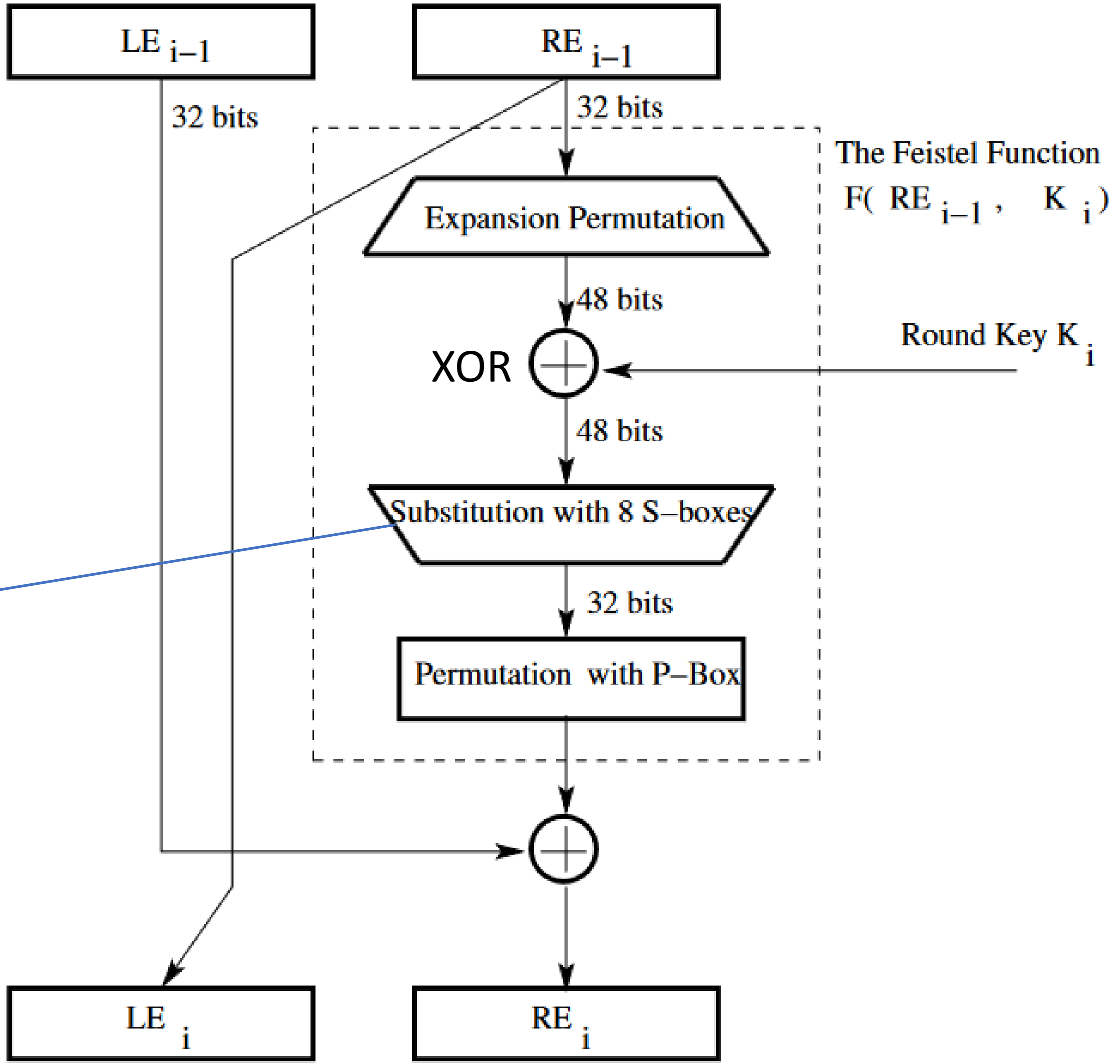


A round in DES (encryption site)

DES Rounds (contd..)

DES "f(•)" Function:





DESRounds (contd..)

DES “f(•)” Function:

Expansion Function (E):

E is an expansion function which takes a block of 32 bits as input and produces a block of 48 bits as output

E-step entails the following:

– first divide the 32-bit block into eight 4-bit words

– attach an additional bit on the left to each 4-bit word that is the last bit of the previous 4-bit word

– attach an additional bit to the right of each 4-bit word that is the beginning bit of the next 4-bit word.

Note that what gets prefixed to the first 4-bit block is the last bit of the last 4-bit block. By the same token, what gets appended to the last 4-bit block is the first bit of the first 4-bit block

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

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

DES Rounds (contd..)

DES “f(•)” Function:

The S-Box for the Substitution Step in Each Round

S-box is to introduce diffusion in the generation of the output from the input.

The 48-bit input word is divided into eight 6-bit words and each 6-bit word fed into a separate S-box. Each S-box produces a 4-bit output. Therefore, the 8 S-boxes together generate a 32-bit output.

8 S-Box

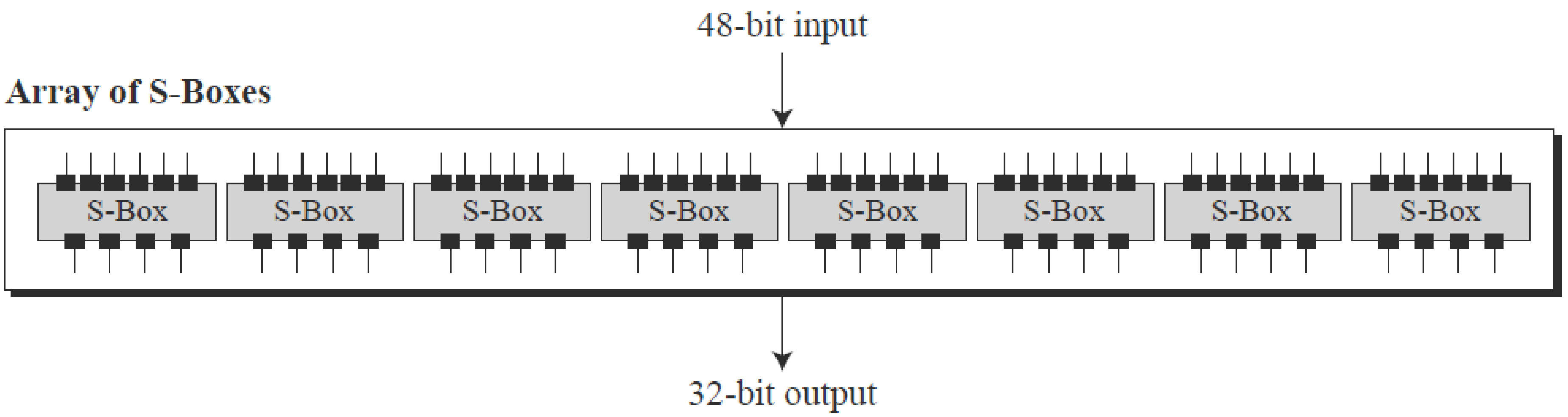
Each of the eight S-boxes consists of a 4 × 16 table lookup for an output 4-bit word.

- S = matrix 4x16, values from 0 to 15
- B (6 bit long) = $b_1b_2b_3b_4b_5b_6$
  - b_1b_6 ➔ r = row of the matrix (2 bits: 0,1,2,3)
  - $b_2b_3b_4b_5$ ➔ c = column of the matrix (4 bits:0,1,...15)
- C (4 bit long) = Binary representation of S(r, c)

DES Rounds (contd..)

DES "f(•)" Function:

There are 8 S-boxes



Each row in all eights tables is a random permutation of the 16 integers, 0 through 15, and no two permutations are the same in all of the eight tables taken together.

DES Rounds (contd..)

DES "f(•)" Function(Feistel Function):

Example S-boxes

Column #

Row #	S_1	1	2	3	…			7								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

S(i, j) <16, can be represented with 4 bits

Assignment: B=011011, C=?

Example: B =101111

b_1b_6 = 11 = row 3

$b_2b_3b_4b_5$= 0111 = column 7

C=7=<u>0111</u>

The 4×16 substitution table for S-box 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-box S_2

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

S-box 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-box S_4

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

S-box 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-box S_6

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

S-box 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-box S_8

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

Example

The input to S-box 1 is <u>1</u>0001<u>1</u>. What is the output?

Solution If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

The input to S-box 8 is <u>0</u>0000<u>0</u>. What is the output?

Solution If we write the first and the sixth bits together, we get 00 in binary, which is 0 in decimal. The remaining bits are 0000 in binary, which is 0 in decimal. We look for the value in row 0, column 0, in Table 6.10 (S-box 8). The result is 13 in decimal, which is 1101 in binary. So the input 000000 yields the output 1101.

DATA ENCRYPTION STANDARD (DES) (contd..)

DES Rounds:

The P-Box Permutation in the Feistel Function

- The last operation in the Feistel function is a permutation with a “Permutation with P-Box”--- 32-bit input and a 32-bit output.

- The input/output relationship for this operation is shown in Table and follows the same general rule as previous tables.

P-Box Permutation							
15	6	19	20	28	11	27	16
0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8
18	12	29	5	21	10	3	24

This permutation ‘table’ says that the 0th output bit will be the 15th bit of the input, the 1st output bit the 6th bit of the input, and so on, for all of the 32 bits of the output that are obtained from the 32 bits of the input.

DES Rounds (contd..)

The DES Key Schedule: Generating the Round Keys

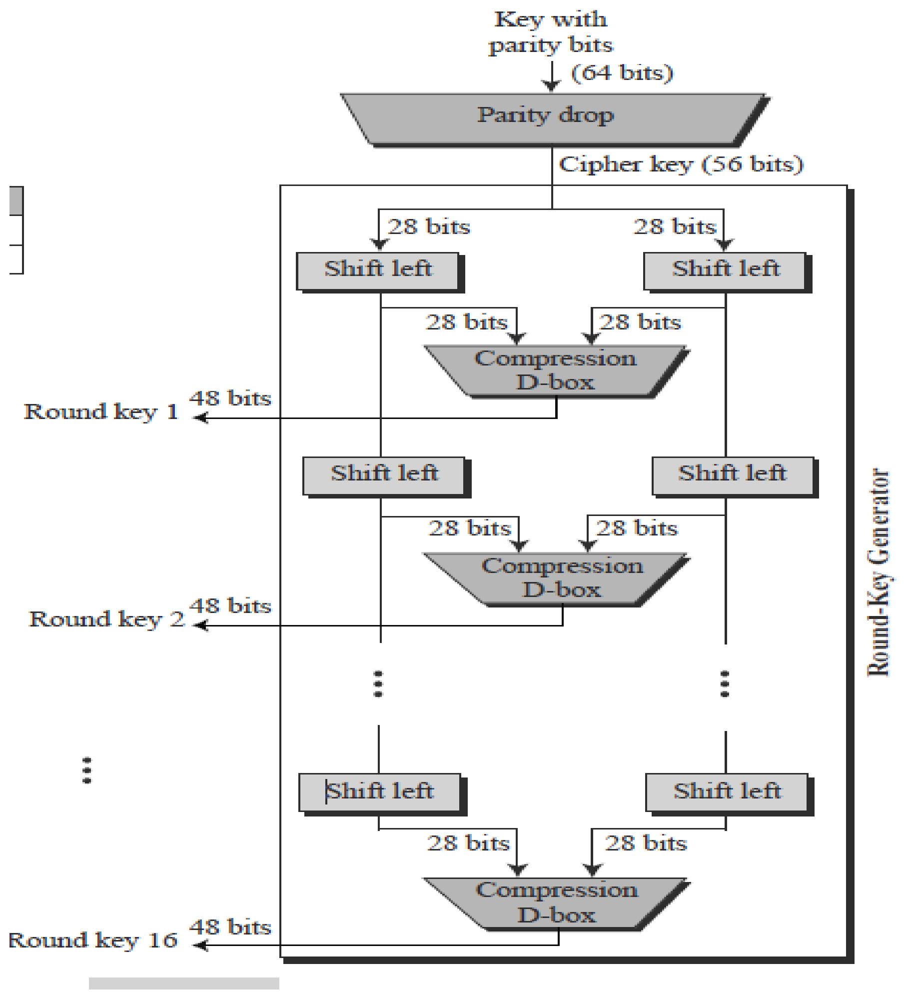
The **round-key generator** creates sixteen 48-bit keys out of a 56-bit cipher key.

However, the cipher key is normally given as a 64-bit key in which 8 extra bits are the parity bits, which are dropped before the actual key-generation process.

At the beginning of each round, we divide the 56 relevant key bits into two 28 bit halves and circularly shift to the left each half by one or two bits, depending on the round, as shown in the table

Shifting

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



DES Rounds (contd..)

The DES Key Schedule: Generating the Round Keys

- The 56-bit encryption key is represented by 8 bytes, with the last bit (the least significant bit) of each byte used as a parity bit.
- The relevant 56 bits are subject to a permutation at the beginning before any round keys are generated. This is referred to as **Key Permutation 1 (PC1)**
- At the beginning of each round, we divide the 56 relevant key bits into two 28 bit halves and circularly shift to the left each half by one or two bits, depending on the round,
- For generating the round key, we join together the two halves and apply a 56 bit to 48 bit contracting permutation this is referred to as **Key Permutation 2 (PC2)**, The resulting 48 bits constitute round key.
- The contraction permutation in **Key Permutation 2**, along with the one-bit or two-bit rotation of the two key halves prior to each round, is meant to ensure that each bit of the original encryption key is used in roughly 14 of the 16 rounds.

DES Rounds (contd..)

The DES Key Schedule: Generating the Round Keys

Parity-check bits (namely, bits 8,16, 4,32,40,48,56,64) are not chosen, they do not appear in **PC-1**

Left						
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
Right						
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

The permutation order for the bits is given by reading the entries shown from the upper left corner to the lower right corner.

This permutation tells us that the 0th bit of the output will be the 57 th bit of the input (in a 64 bit representation of the 56-bit encryption key), the 1st bit of the output the 49th bit of the input, and so on

PC-2 selects the 48-bit subkey for each round from the 56-bit key-schedule state

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

THE STRENGTH OF DES

1. The Use of 56-Bit Keys

- With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 * 1016 keys. Thus, on the face of it, a brute-force attack appears impractical.
- Assuming that, on average, half the key space has to be searched, a single machine

2. Timing attack

- The authors conclude that DES appears to be fairly resistant to a successful timing attack but suggest some avenues to explore.
- A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts. A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.

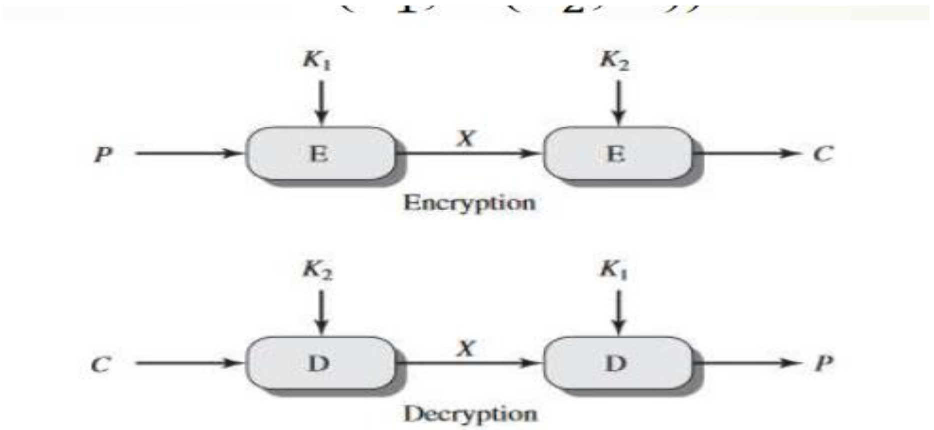
THE STRENGTH OF DES

3. The Nature of the DES Algorithm

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent.

Double DES

- DES uses a 56-bit key, this raised concerns about brute force attacks.
- One proposed solution: double DES.
- Apply DES twice using two keys, K1 and K2.
- – Encryption: $C = E_{K2} [E_{K1} [P]]$
- – Decryption: $P = D_{K2} [D_{K1} [C]]$
- This leads to a 2x56=112 bit key, so it is more secure than DES



(a) Double encryption

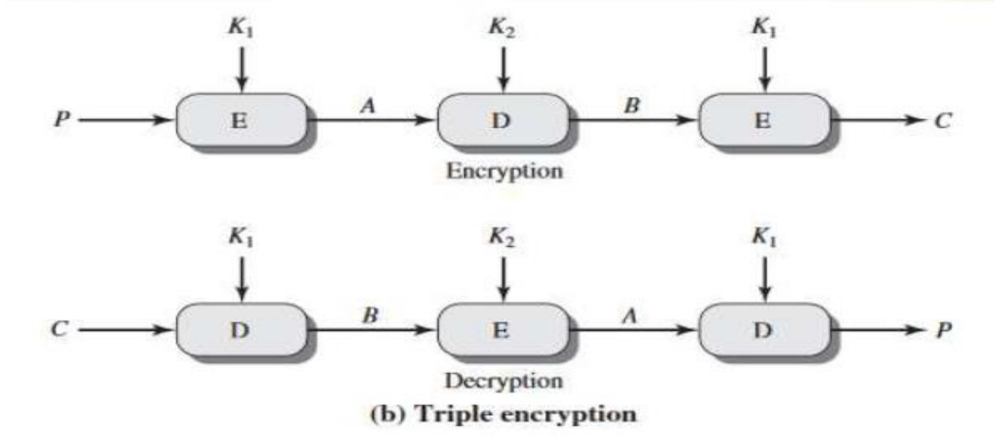
Triple DES

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

The encryption-decryption process is as follows −

- Encrypt the plaintext blocks using single DES with key K_1.
- Now decrypt the output of step 1 using single DES with key K_2.
- Finally, encrypt the output of step 2 using single DES with key K_3.
- The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. User first decrypt using $K_{3,}$ then encrypt with $K_{2,}$ and finally decrypt with K_1.

Triple DES

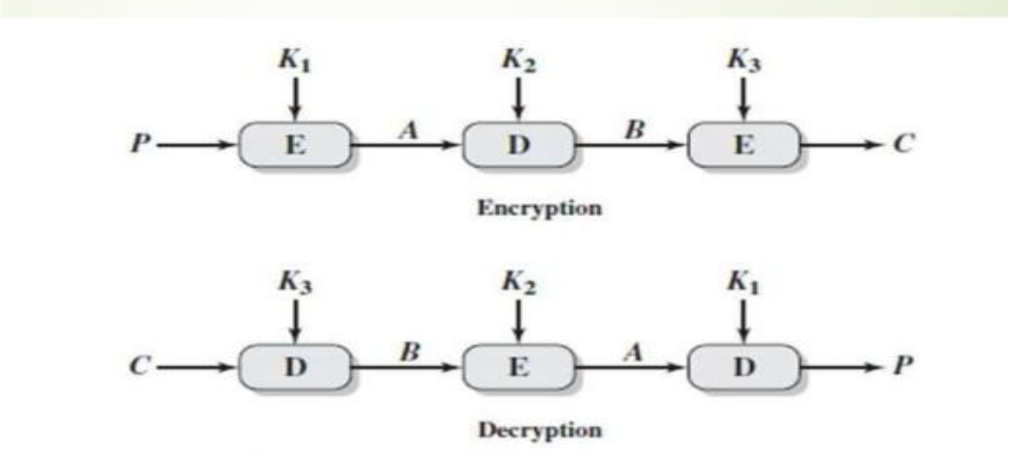


Triple DES with 2-key

The function follows an encrypt-decrypt-encrypt (EDE) sequence.

C = E(K1, D(K2, E(K1, P)))

P = D(K1, E(K2, D(K1, C)))



Triple DES with 3-key

3-key 3DES has an effective key length of 168 bits and is defined as

C = E(K3, D(K2, E(K1, P)))

P = D(K1, E(K2, D(K3, C))

Example (DES Round key generation)

- Generate round-1 key using the key in Hex K=0x133457799BBCDFF1
- Given data: PC1 and PC2=

Left						
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
Right						
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

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

- **Solution:**
- Step1: Convert Hex number to 64 bit binary format
- 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Example:

- Step2: Remove parity bits and apply PC1

Hex digits	Bit position	bits	bits	bits	bits	bi s	bits	bits	bits
13	1-2-3-4-5-6-7-8	0	0	0	1	0	0	1	1
34	9-10-11-12-13-14-15-16	0	0	1	1	0	1	0	0
57	17-18-19-20-21-22-23-24	0	1	0	1	0	1	1	1
79	25-26-27-28-29-30-31-32	0	1	1	1	1	0	0	1
9B	33-34-35-36-37-38-39-40	1	0	0	1	1	0	1	1
BC	41-42-43-44-45-46-47-48	1	0	1	1	1	0	0	0
DF	49-50-51-52-53-54-55-56	1	1	0	1	1	1	1	1
F1	57-58-59-60-61-62-63-64	1	1	1	1	0	0	0	1

Left						
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
Right						
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

1	1	1	1	0	0	0
0	1	1	0	0	1	1
0	0	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	1	0	1	0
1	0	1	0	0	0	1
1	0	0	1	1	1	1
0	0	0	1	1	1	1

- Parity bits
- Apply PC1: we get 56 bit key permutation
- 1111000 0110011 0010101 0101111 0101010 1010001 1001111 0001111

Example

- Step3: Split this key into left and right halves, $\boldsymbol{C_0}$ and $\boldsymbol{D_0}$, where each half has 28 bits.

- $\boldsymbol{C_0}$ = 1111000 0110011 0010101 0101111
  $\boldsymbol{D_0}$ = 0101010 1011001 1001111 0001111

- **Step4:** Left shift → Round 1 i-shift

- $\boldsymbol{C_0}$ = 111000 0110011 0010101 01011111
  $\boldsymbol{D_0}$ = 101010 1011001 1001111 00011110

Step5: Apply PC2

1	1	1	0	0	0	0
1	1	0	0	1	1	0
0	1	0	1	0	1	0
1	0	1	1	1	1	1
1	0	1	0	1	0	1
0	1	0	0	0	1	1
0	0	1	1	1	1	0
0	0	1	1	1	1	0

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

1-7
8-14
15-21
22-28
29-35
36-42
43-49
50-56

1	1	1	0	0	0	0
1	1	0	0	1	1	0
0	1	0	1	0	1	0
1	0	1	1	1	1	1
1	0	1	0	1	0	1
0	1	0	0	0	1	1
0	0	1	1	1	1	0
0	0	1	1	1	1	0

0	0	0	1	1	0	1	1
0	0	0	0	0	0	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	1	0

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010-----→48 bit key

C_0 = 1111000011001100101010101111
D_0 = 0101010101100110011110001111

C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110

C_2 = 1100001100110010101010111111
D_2 = 0101010110011001111000111101

C_3 = 0000110011001010101011111111
D_3 = 0101011001100111100011110101

C_4 = 0011001100101010101111111100
D_4 = 0101100110011110001111010101

C_5 = 1100110010101010111111110000
D_5 = 0110011001111000111101010101

C_6 = 0011001010101011111111000011
D_6 = 1001100111100011110101010101

C_7 = 1100101010101111111100001100
D_7 = 0110011110001111010101010110

C_8 = 0010101010111111110000110011
D_8 = 1001111000111101010101011001

C_9 = 0101010101111111100001100110
D_9 = 0011110001111010101010110011

C_{10} = 0101010111111110000110011001
D_{10} = 1111000111101010101011001100

C_{11} = 0101011111111000011001100101
D_{11} = 1100011110101010101100110011

C_{12} = 0101111111100001100110010101
D_{12} = 0001111010101010110011001111

C_{13} = 0111111110000110011001010101
D_{13} = 0111101010101011001100111100

C_{14} = 1111111000011001100101010101
D_{14} = 1110101010101100110011110001

C_{15} = 1111100001100110010101010111
D_{15} = 1010101010110011001111000111

C_{16} = 1111000011001100101010101111
D_{16} = 0101010101100110011110001111

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101