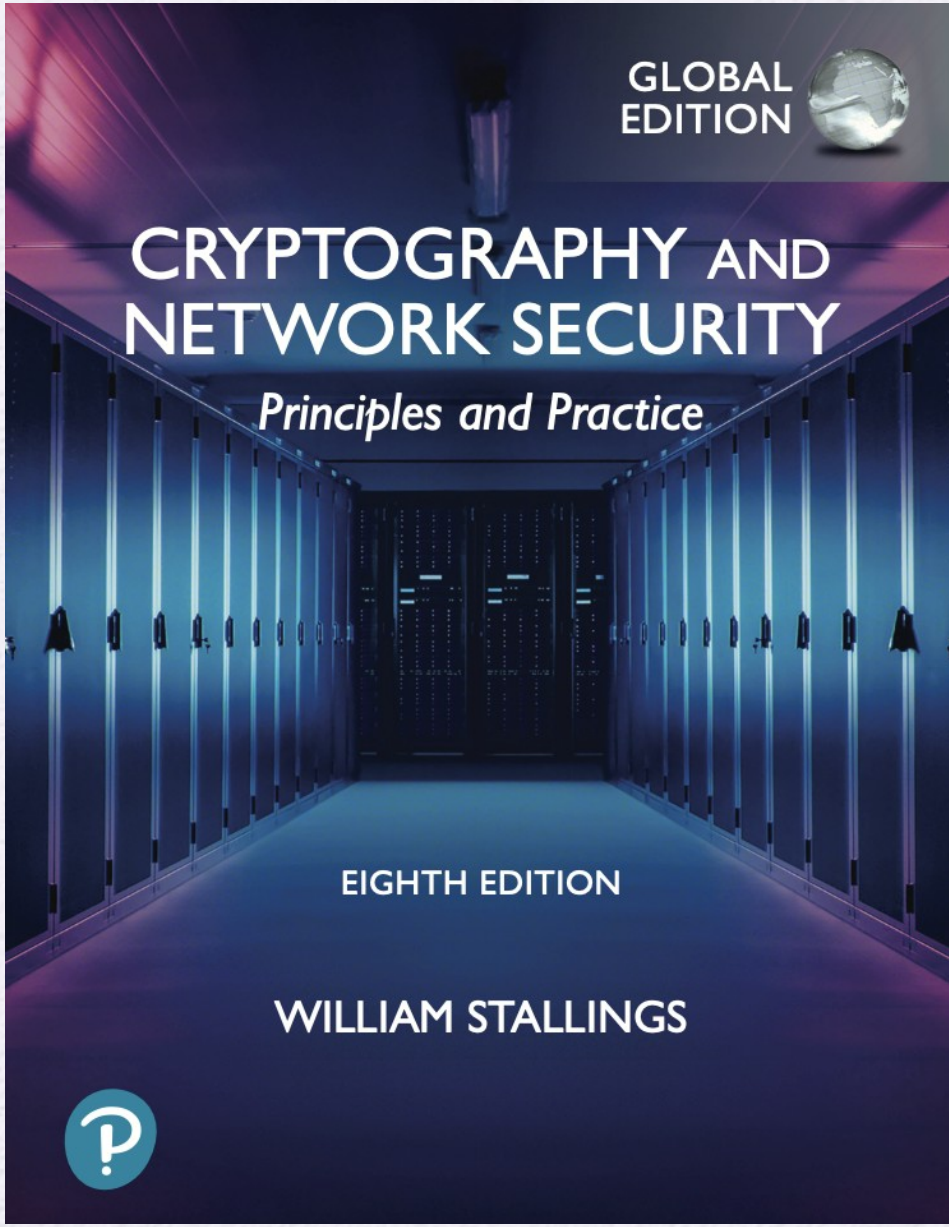
University of Nevada – Reno
Computer Science & Engineering Department

CS454/654 Reliability and Security of Computing Systems - Fall 2024

Lecture 5

Dr. Batyr Charyyev
bcharyyev.com



CHAPTER 4

BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

Stream Cipher

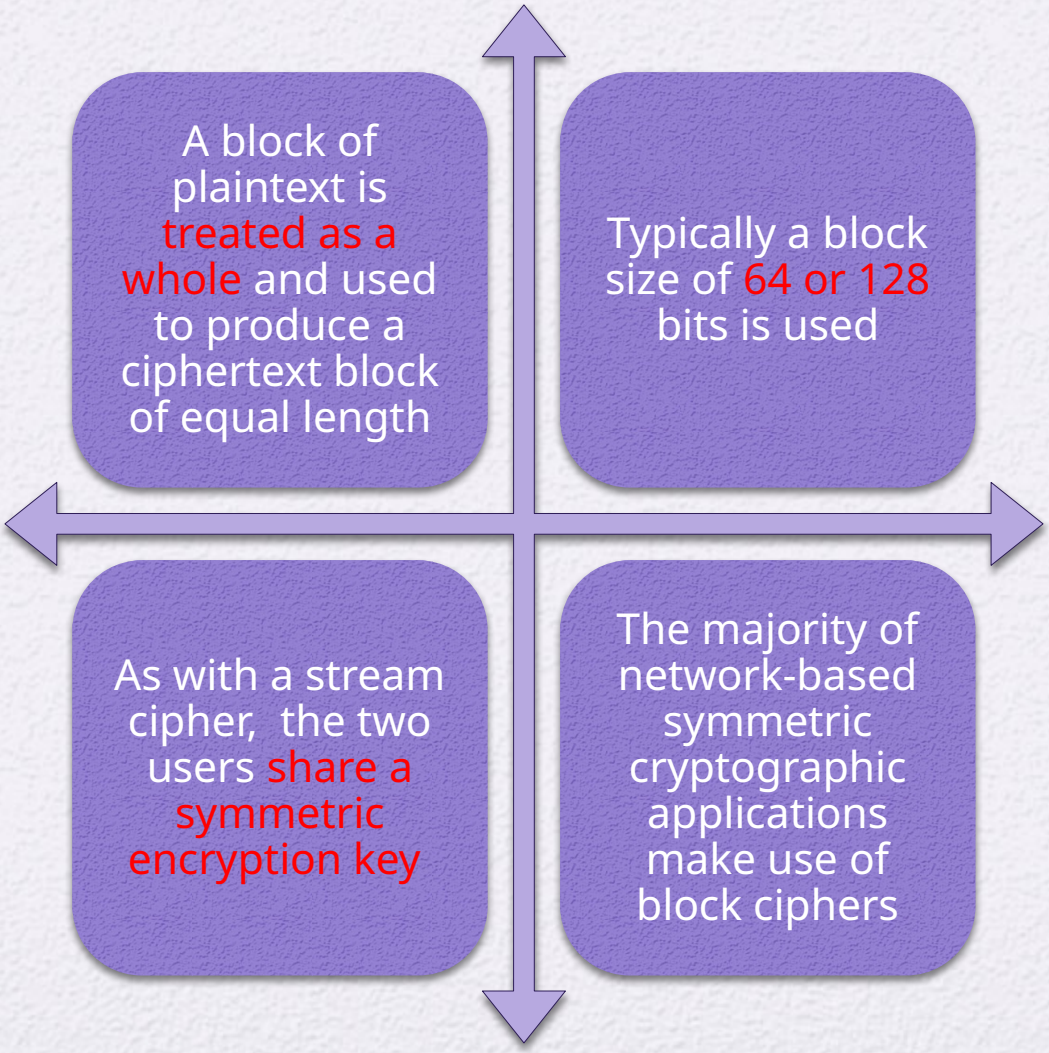
- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. Example of stream cipher is Vernam or Vigenère cipher.
- If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream.
- Sharing the keystream introduces complexities such as extra traffic, transferring keystream over secure channel.
- For practical reasons bit-strea
  kind of function or algorithm)

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation

Block Cipher



- A block cipher operates on a plaintext block of n bits to produce a cipher- text block of n bits.
- **Q: how many possible different blocks?**
- There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block.
- For n = 2

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

- There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block.

- For n = 2

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

- Q: If we limit ourselves to reversible mappings, the number of different transformations is … ?
  - 2^n! Why?
  - For the 1st plaintext, we can choose any of 2^n ciphertext blocks. For the 2nd plaintext, we choose from among 2^n - 1 remaining ciphertext blocks, and so on

- If block size is small, then the system is equivalent to a classical substitution cipher and vulnerable to a statistical analysis of the plaintext. Thus, n should be large to prevent cryptanalysis.
- A reversible substitution for a large block size is not practical. Assume n=4.
- The required key length is (4 bits) * (16 rows) = 64 bits
- For a 64-bit block, which is a desirable length to prevent statistical attacks, the required key length is 64 * 2^{64} = $2^{70} \approx 10^{21}$ bits.

Table 4.1 Encryption and Decryption Tables for Substitution Cipher of Figure 4.2

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher

- Is a practical application of a proposal (1945) by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions.
- Claude Shannon, often regarded as the father of information theory, introduced the concepts of confusion and diffusion in the context of cryptography to describe essential properties of a secure cipher.
- The structure of Fiestel Cipher is used by many significant symmetric block ciphers currently in use.
  - Camellia block cipher is a Feistel structure; it is used as one of the symmetric ciphers in TLS.

Claude Shannon confusion and diffusion

- Shannon's concern was to prevent cryptanalysis based on statistical analysis.
- The goal of **confusion** is to obscure the connection between the key and the ciphertext, making it difficult for an attacker to determine the key even if they have several pairs of plaintext and corresponding ciphertext. Achieved Through: Substitution.
- The mechanism of **diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to prevent attempts to deduce the key.

Claude Shannon confusion and diffusion

- In diffusion, each plaintext digit affect the value of many ciphertext digits; generally, this is equivalent to having each ciphertext digit be affected by many plaintext digits.

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

- An example of diffusion is to encrypt a message M = m1, m2, m3, … of characters with an averaging operation adding k successive letters to get a ciphertext letter y_n.

Claude Shannon confusion and diffusion

- With diffusion the distribution of how often each letter appears in the ciphertext will be more uniform compared to their distribution in the plaintext. Helps to prevent statistical crypto analysis with frequency distribution.
- Example: plain text = HELLO WORLD
  - Distribution is H: 1, E: 1, L: 3, O: 2, W: 1, R: 1, D: 1
- Encrypted HJMQSO
  - Distribution: H:1, J:1, M:1, Q:1, S:1, O:1
  - More uniform

- H (8), E (5), L (12) → Average: (8+5+12)/3 = 25/3 ≈ 8 → H
- E (5), L (12), L (12) → Average: (5+12+12)/3 = 29/3 ≈ 10 → J
- L (12), L (12), O (15) → Average: (12+12+15)/3 = 39/3 = 13 → M
- L (12), O (15), W (23) → Average: (12+15+23)/3 = 50/3 ≈ 17 → Q
- O (15), W (23), R (18) → Average: (15+23+18)/3 = 56/3 ≈ 19 → S
- W (23), R (18), D (4) → Average: (23+18+4)/3 = 45/3 = 15 → O

Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions (Confusion) and permutations (Diffusion).

- Permutation: No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

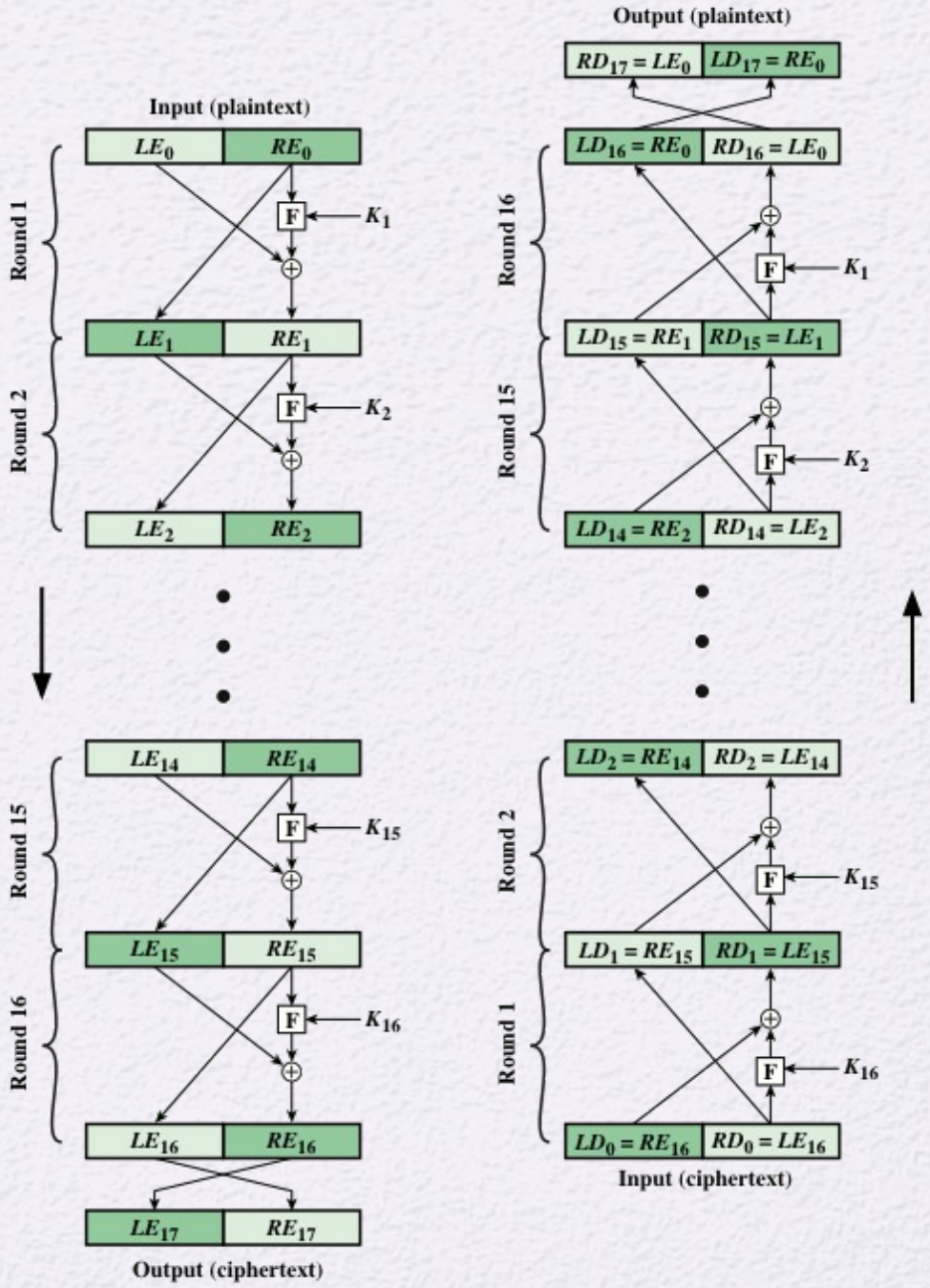


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

1. **Plaintext Block**:
   - Assume a 8-bit block of plaintext: 01010101
   - Split into two 4-bit halves: $L_0 = 0101$, $R_0 = 0101$
2. **Round 1**:
   - Subkey K_1 = 0011 (example subkey)
   - Apply the round function F:
     - $F(R_0, K_1) = R_0 \oplus K_1 = 0101 \oplus 0011 = 0110$
   - Compute the new halves:
     - $L_1 = R_0 = 0101$
     - $R_1 = L_0 \oplus F(R_0, K_1) = 0101 \oplus 0110 = 0011$
3. **Ciphertext Block**:
   - Combine L_1 and R_1 to get the ciphertext: 01010011

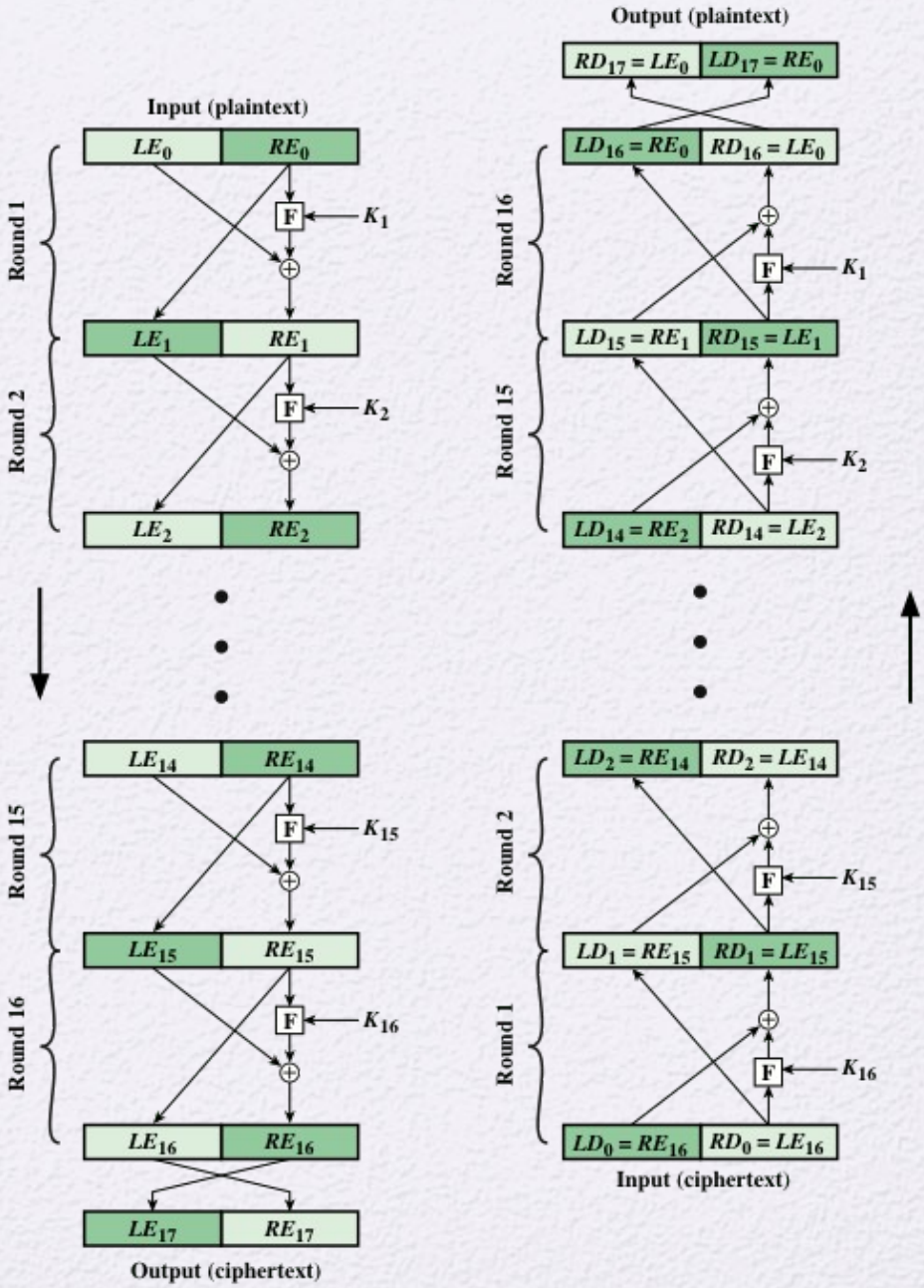


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

Decryption

1. **Ciphertext Block**:
   - Split into two 4-bit halves: $L_1 = 0101$, $R_1 = 0011$
2. **Round 1** (using subkey K_1):
   - Apply the round function F:
     - $F(R_1, K_1) = R_1 \oplus K_1 = 0011 \oplus 0011 = 0000$
   - Compute the new halves:
     - $R_0 = L_1 = 0101$
     - $L_0 = R_1 \oplus F(R_1, K_1) = 0011 \oplus 0000 = 0011$
3. **Plaintext Block**:
   - Combine L_0 and R_0 to get the original plaintext: 01010101

Note: K_i in reverse order. That is, use K_n in the first round, K_{n-1} in the second round, and so on, until K_1 is used in the last round.

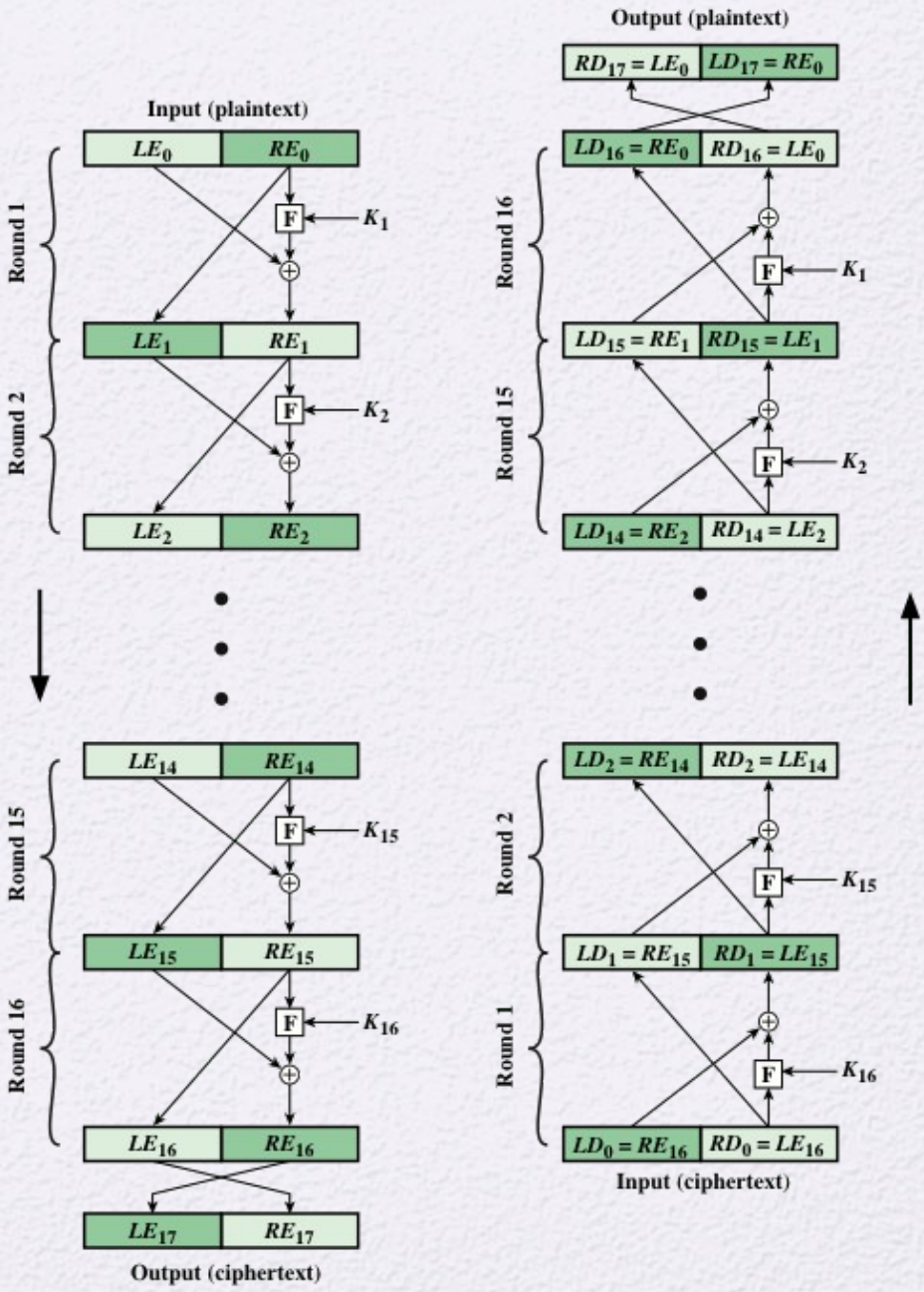


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

Feistel Cipher Design Features

- **Block/key size:** Larger block/key sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function F:** Greater complexity generally means greater resistance to cryptanalysis.

CHAPTER 4

BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

Data Encryption Standard (DES)

- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard.
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001.
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA).
  - Data are encrypted in 64-bit blocks using a 56-bit key.
  - 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)

- https://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

APPENDIX C

DATA ENCRYPTION STANDARD

The overall scheme for DES encryption is illustrated in Figure C.1, which repeats Figure 4.5. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.[1]

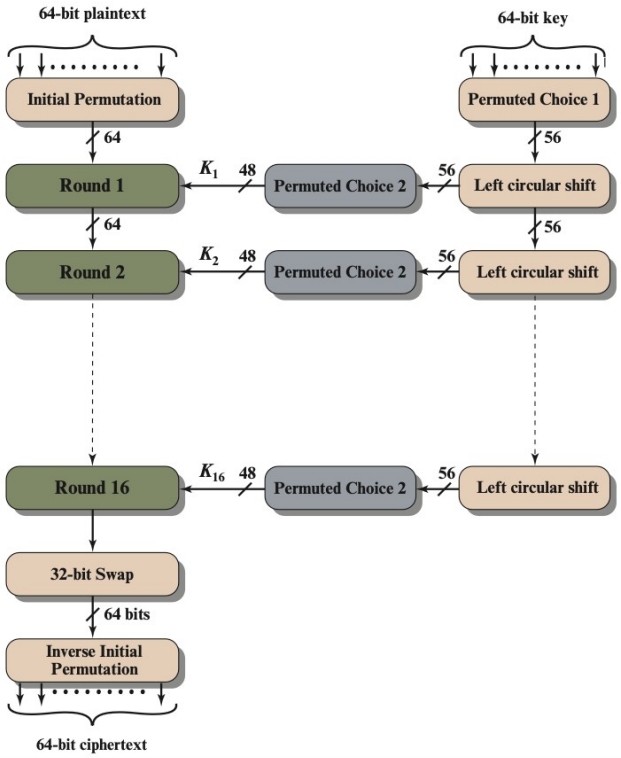


Figure C.1 General Depiction of DES Encryption Algorithm

Let M be the plain text message M = 0123456789ABCDEF

M = 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

Let K be the hexadecimal key K = 133457799BBCDFF1

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Key sizes is 56- bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64).

Step 1: Create 16 subkeys, each of which is 48-bits long.

The 64-bit key is permuted according to the following 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.

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

Note only 56 bits of the original key appear in the permuted key.

From the original 64-bit key

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111

Step 1: Create 16 subkeys, each of which is 48-bits long.

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key (of size 56 bits) into left and right halves, C_0 and D_0, where each half has 28 bits.

C_0 = 1111000 0110011 0010101 0101111 D_0 = 01

With C_0 and D_0 defined, we now create 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 the following schedule of "left shifts" of the previous block.

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

Step 1: Create 16 subkeys, each of which is 48-bits long.

C_0 = 1111000 0110011 0010101 0101111 D_0 = 0101010 1011001 100111 0001111

For example, C3 and D3 are obtained from C2 and D2, respectively, by two left shifts, and C16 and D16 are obtained from C15 and D15, respectively, by one left shift.

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

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

Step 1: Create 16 subkeys, each of which is 48-bit

We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs $C_n D_n$. Each pair has 56 bits, but PC-2 only uses 48 of these.

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

1st bit of K_n is the 14th bit of $C_n D_n$, the 2nd bit the 17th,

C_0 = 11110000110011001010101011111
D_0 = 01010101011001100111100011111

C_1 = 11100001100110010101010111111
D_1 = 10101010110011001111100011110

C_2 = 11000011001100101010101111111
D_2 = 01010101100110011110001111101

C_3 = 00001100110010101010111111111
D_3 = 01010110011001111000111101011

C_4 = 00110011001010101011111111100
D_4 = 01011001100111100011110101011

C_5 = 11001100101010101111111100000
D_5 = 01100110011110001111010101011

C_6 = 00110010101010111111111000011
D_6 = 10011001111000111101010101011

C_7 = 11001010101011111111000011100
D_7 = 01100111100011110101010101100

C_8 = 00101010101111111110000110011
D_8 = 10011110001111010101011001100

C_9 = 01010101011111111100001100110
D_9 = 00111100011110101010110011001

C_{10} = 01010101111111100001100110011
D_{10} = 11110001111010101011001100110

C_{11} = 01010111111111000011001100101
D_{11} = 11000111101010101100110011011

C_{12} = 01011111111000011001100101011
D_{12} = 00011110101010101100110011111

C_{13} = 01111111110000110011001010101
D_{13} = 01111010101010110011001111100

C_{14} = 11111110000110011001010101011
D_{14} = 11101010101011001100111100011

C_{15} = 11111000011001100101010101111
D_{15} = 10101010101100110011111000111

C_{16} = 11110000110011001010101011111
D_{16} = 01010101011001100111100011111

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

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

Step 1: Create 16 subkeys, each of which is 48-bits long.

K (64bit)= 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

K+ (56bit) = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

C_0 = 1111000 0110011 0010101 0101111

D_0 = 0101010 1011001 1001111

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

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

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_{16} = 1111000011001100101010101111
D_{16} = 0101010101100110011110001111

Step 1: Create 16 subkeys, each of which is 48-bits long.

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

56-bit

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

48-bit

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

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

Step 2: Encode each 64-bit block of data.

Take an initial permutation (IP) of the 64 bits of the message data M.

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

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

Next divide the permuted block IP into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111 R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

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

Perform 16 iterations, for 1<=n<=16, using a [illegible] ata block of 32 bits and a key K_n of 48 bits and produce [illegible] (similar to Feistel Ciphers)

Step 2: Encode each 64-bit block of data.

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

Example: For $n = 1$, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0 + f(R_0, K_1)$

How F works?

First expand each block R_{n-1} from 32 bits to 48 bits, using a selection table that repeats some of the bits in R_{n-1} .

First three bits of $E(R_{n-1})$ are the bits in positions 32, 1 and 2 of R_{n-1} while the last 2 bits of $E(R_{n-1})$ are the bits in positions 32 and 1.

Example: We calculate $\mathbf{E}(R_0)$ from R_0 as follows:

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

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

Next in the F calculation, we XOR the output $E(R_{n-1})$ with the key $K_n + \mathbf{E}(R_{n-1})$.

Example: For K_1 , $\mathbf{E}(R_0)$, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101
$K_1 + \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

Step 2: Encode each 64-bit block of data.

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

How F works?

To this point we have expanded R_{n-1} from 32 bits to 48 bits, using the selection table, and XORed the result with the key K_n

$$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$$
$$\mathbf{E}(R_0) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$$
$$K_1 + \mathbf{E}(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$$

$$K_n + \mathbf{E}(R_{n-1}) = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8,$$

Group of six bits

where each B_i is a group of 6 bits.

$$S_1(B_1) S_2(B_2) S_3(B_3) S_4(B_4) S_5(B_5) S_6(B_6) S_7(B_7) S_8(B_8)$$

S1

Row No.	Column Number 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

S8

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

Step 2: Encode each 64-bit block of data.

$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$

$E(R_0) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$

$K_1 + E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$

$$K_n + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8,$$

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

S1

Row No.	Column Number															
	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

B = 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1.

The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13.

In row 1, column 13 appears 5 and binary value is 0101

Thus B1=011011 converted to S1(B1) = 0101

Step 2: Encode each 64-bit block of data.

$K_1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010$

$E(R_0) = 011110\ 100001\ 010101\ 010101\ 011110\ 100001\ 010101\ 010101$

$K_1 + E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$

$K_n + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8,$

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

S1

Row No.	Column Number 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

S8

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

$K_1 + E(R_0) = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8) = 0101\ 1100\ 1000\ 0010\ 1011\ 0101\ 1001\ 0111$

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

$f = 0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011$

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

Step 2: Encode each 64-bit block of data.

Next iteration L2 = R1, which is the block we just calculated, and then we must calculate R2 =L1 + f(R1, K2) after iterating 16 times we should get $R_{16}L_{16}$

$$R_1 = L_0 + f(R_0, K_1)$$

$$= 1100\ 1100\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111$$
$$+ 0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011$$
$$= 1110\ 1111\ 0100\ 1010\ 0110\ 0101\ 0100\ 0100$$

and apply a final permutation $\mathbf{IP^{-1}}$ as defined by the following table:

$\mathbf{IP^{-1}}$

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

Encryption of M = 0123456789ABCD

is C = 85E813540F0AB405.

DES Decryption: uses the same algorithm as encryption, except that the application of the subkeys is reversed. Additionally, the initial and final permutations are reversed.

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101

We reverse the order of these two blocks and apply the final permutation to

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

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

which in hexadecimal format is

85E813540F0AB405.

Avalanche effect: a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

We change single bit in plaintext and inp
becomes 12468aceeca86420

The 2nd column of the table shows the intermediate 64-bit values at the end of each round for the two plaintexts.

The 3rd column shows the number of bits that differ between the two intermediate values.

Table 4.3 Avalanche Effect in DES: Change in Plaintext

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Avalanche effect: a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

We change single bit in key and key becomes 1f1571c947d9e859

Table 4.4 Avalanche Effect in DES: Change in Key

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	27
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

CHAPTER 4

BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

THE STRENGTH OF DES

Two concerns: key size and the nature of the algorithm.

Table 4.5 Average Time Required for Exhaustive Key Search

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

A single PC can break DES in about a year; if multiple PCs work in parallel, the time is drastically shortened.

Today's supercomputers should be able to find a key in about an hour.

Fortunately, there are a number of alternatives to DES, the most important of which are AES and triple DES, discussed in Chapters 6 and 7,

THE STRENGTH OF DES

Two concerns: key size and the nature of the algorithm.

Nature of the algorithm

- The design criteria for the S-boxes were not made public when DES was first introduced. This lack of transparency has led to suspicion and concern within the cryptographic community.
- Over the years, researchers have discovered certain patterns and behaviors in the S-boxes that were not initially expected.
- Despite this, no one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.

The tables defining the functions $S_1,\ldots,S_8$ are the following:

S1

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

S2

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

S3

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

S4

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