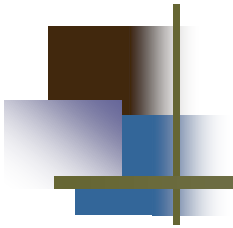


Chap 2-2 Conventional Encryption Message Confidentiality (Symmetric, Block)

DES (Data Encryption Standard)

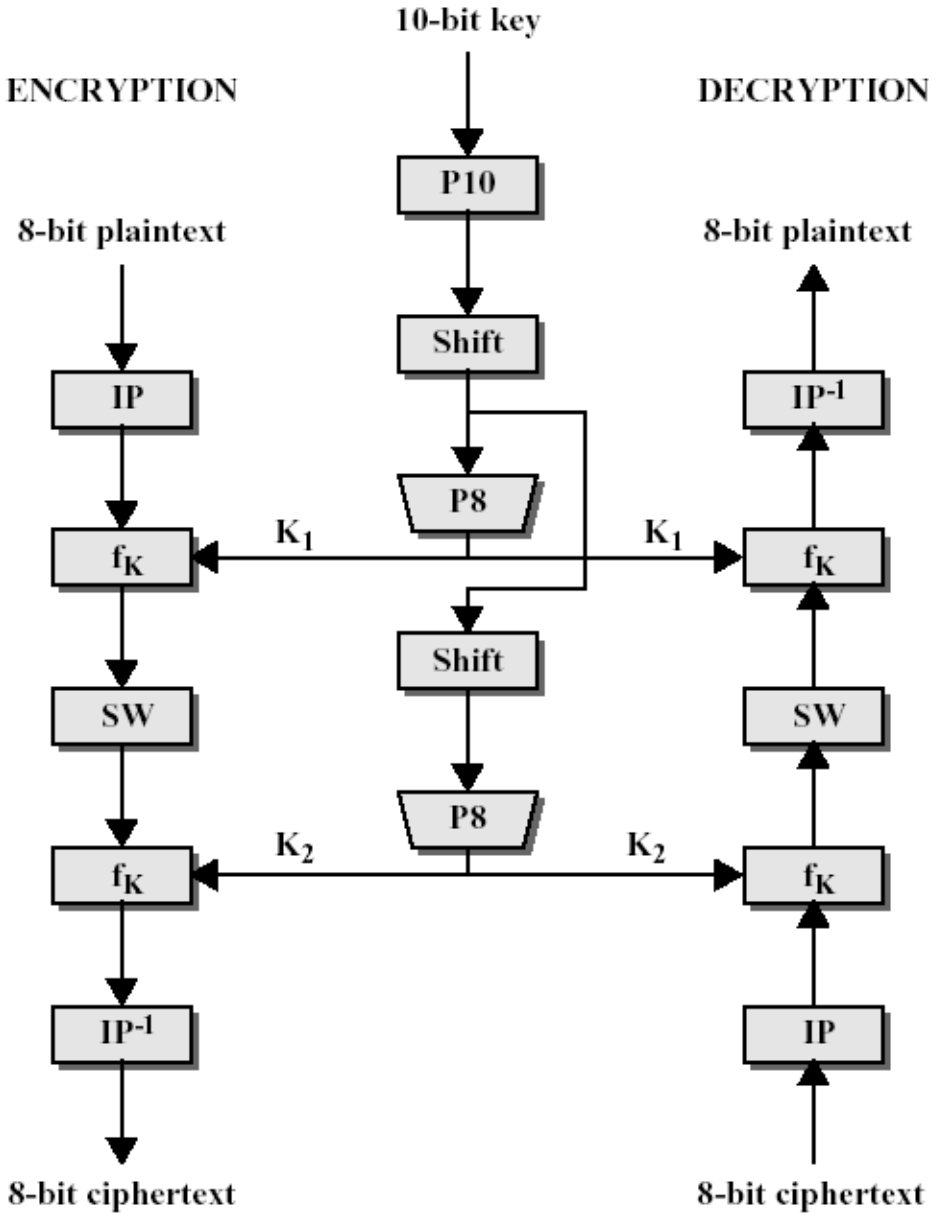
- Requirements

  1. It must provide a high level of security.
  2. It must be completely specified and easy to understand.
  3. The security provided by the algorithm must not be based on the secrecy of the algorithm.
  4. It must be all users and suppliers.

DES (Data Encryption Standard)

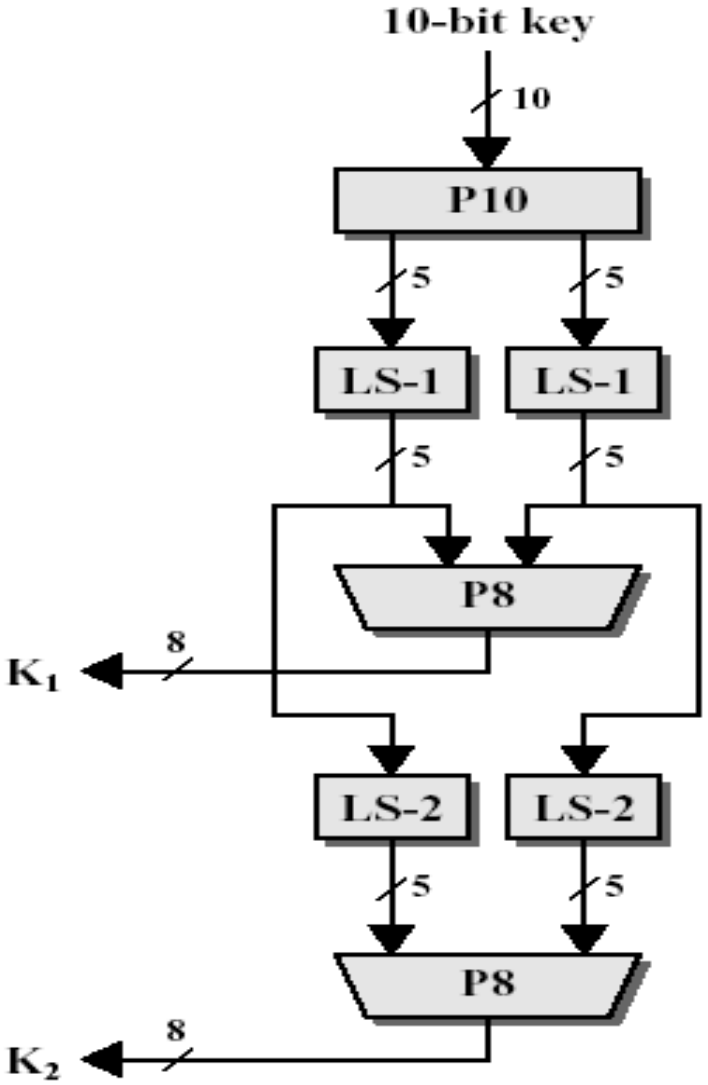
5. It must be adaptable for use in diverse applications.
6. It must be economical to implement in electronic devices and be efficient to use.
7. It must be amenable to validation.
8. It must be exportable.

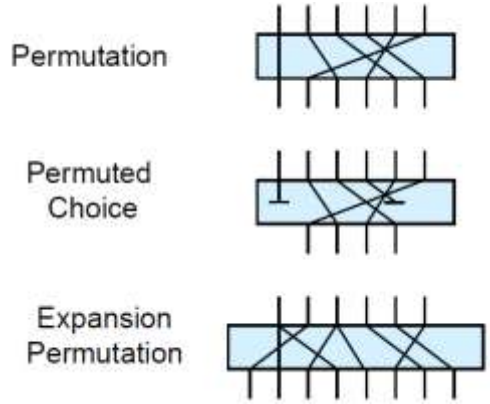
S-DES Structure and Flow Diagram



IP: initial permutation
SW: Switch

Part1. Generation of Key Value
Key generation for simplified DES





1.P10

- 처음 주어진 10비트의 키 의 값은 바꾸지 않고, 각각의 비트들에 Index값 만을 변환 시켜주는 즉 각각의 비트의 위치를 바꿔주는 순열 변환을 하여 준다.

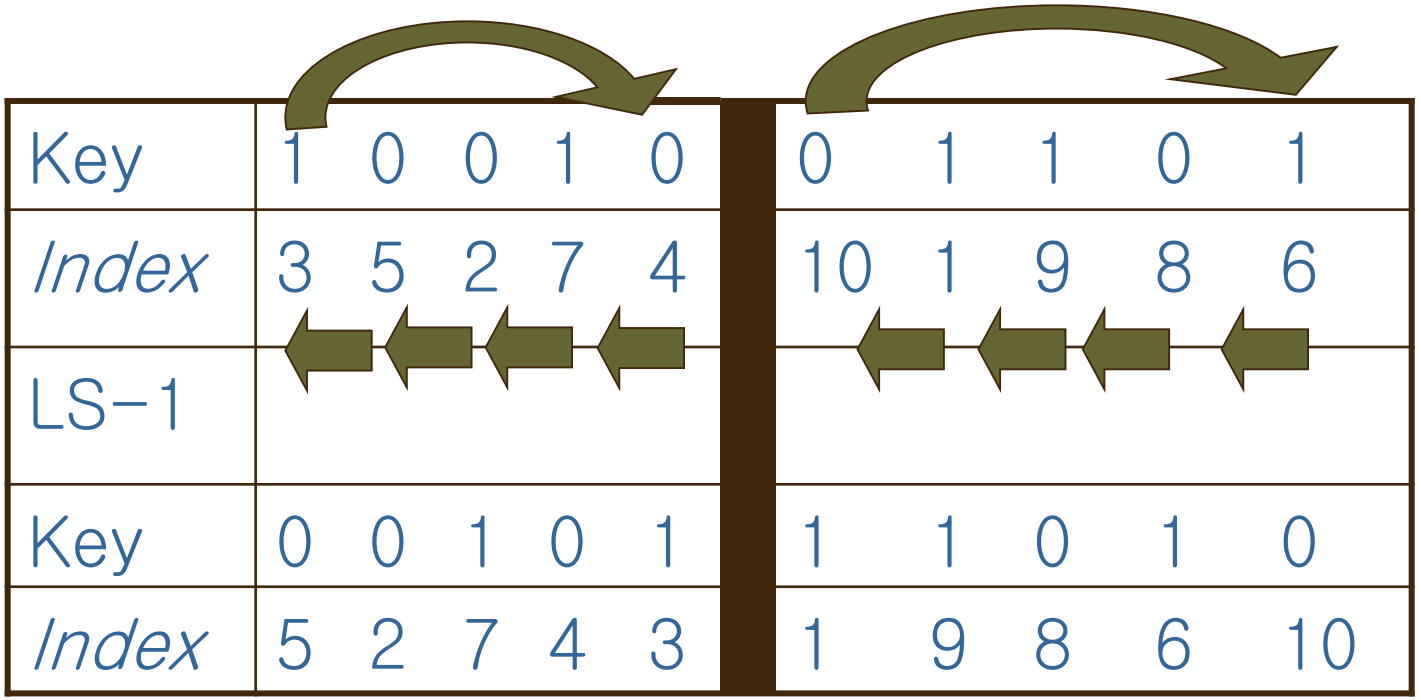
Initial Key Values

P10 실행 후 의 Key 값

Key	1	0	1	0	0	1	1	0	1	0
Index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Key	1	0	0	1	0	0	1	1	0	1
Index	*3*	*5*	*2*	*7*	*4*	*10*	*1*	*9*	*8*	*6*

2.Shift(LS-1)

- 순열 변환이 이루어진 10비트 값을 좌측부터 5비트씩 2부분으로 나누어 각 부분의 5비트를 개별적으로 1비트씩 좌측으로 순환 이동시킨다.



3.P8

- 개별적으로 좌측 1비트씩 순환 이동시킨 2개의 5비트를 합쳐서 다시 10비트로 만들고 Index값 재배치 후 8비트만을 골라서 다시금 순열 변환을 시켜준다.

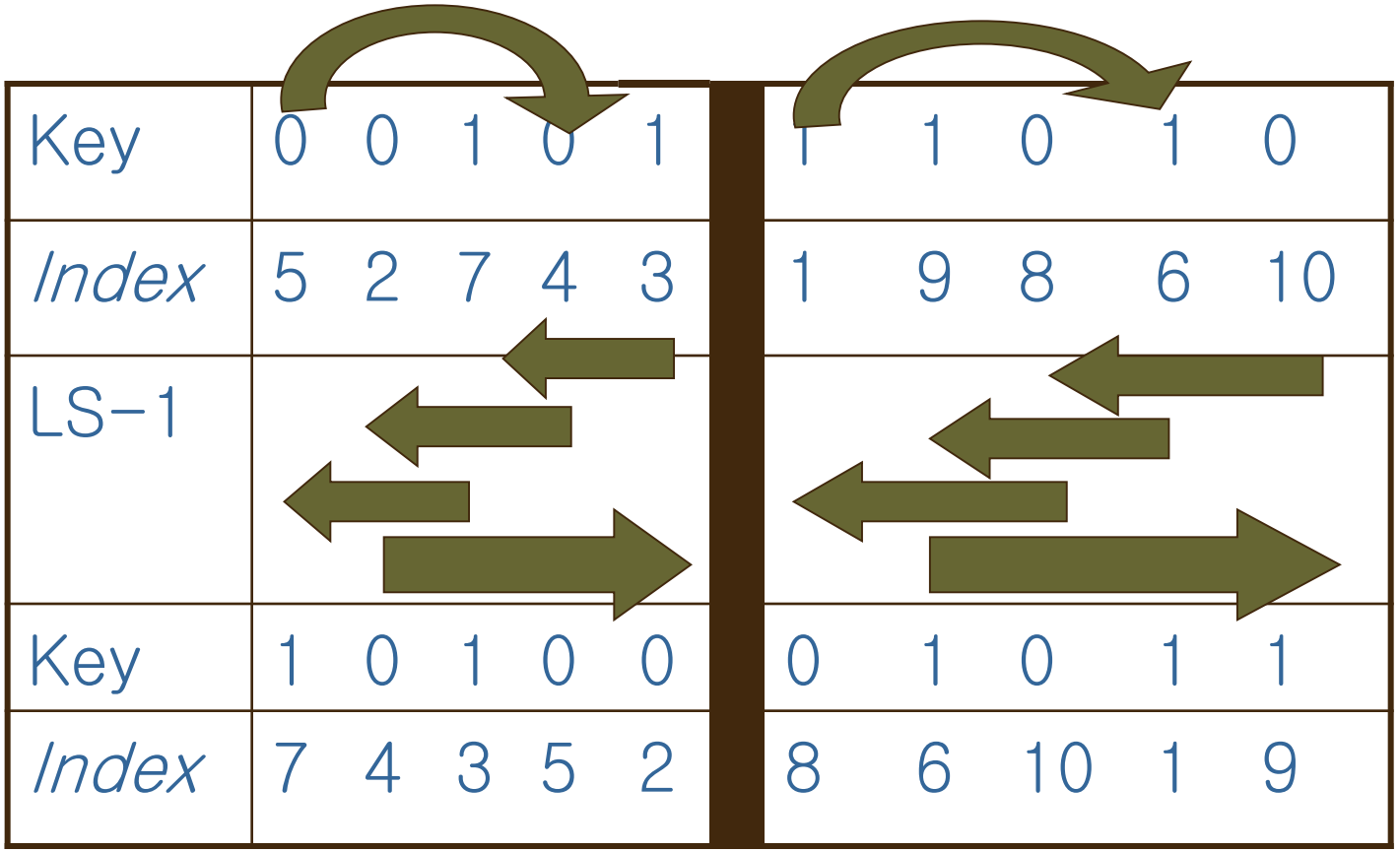
Key값	Key	0	0	1	0	1	1	1	0	1	0
	Index	1	2	3	4	5	6	7	8	9	10
P8 실행 후의 Key 값	Key	1	1	1	0	0	1	0	1		
	Index	*6*	*3*	*7*	*4*	8	5	10	9		

Key1값 생성 : 1 1 1 0 0 1 0 1

4.Shift(LS-2)

- Shift(LS-1) 이루어진 10비트 값을 좌측부터 5비트씩 2부분으로 나누어 각 부분의 5비트를 개별적으로 2비트씩 좌측으로 순환 이동시킨다.

Key값



LS-2 실행 후의 Key 값

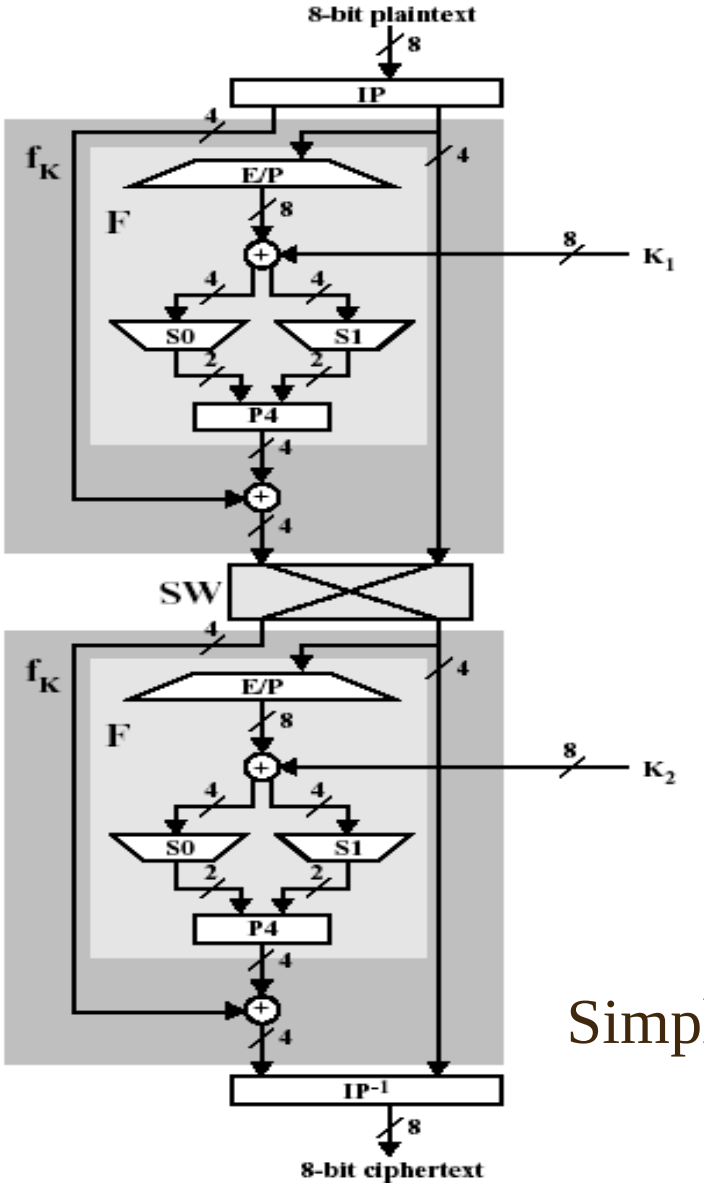
5.P8

- 개별적으로 좌측 2비트씩 순환 이동시킨 2개의 5비트를 합쳐서 다시 10비트로 만들고 Index값 재배치 후 8비트만을 골라서 다시금 순열 변환을 시켜준다.

Key값	Key	1	0	1	0	0	0	1	0	1	1
	Index	1	2	3	4	5	6	7	8	9	10
P8 실행 후의 Key 값	Key	0	1	1	0	0	0	1	1		
	Index	*6*	*3*	*7*	*4*	8	5	10	9		

Key2값 생성 : 0 1 1 0 0 0 1 1

Part2.평문 의 암호화



Simplified DES Scheme Encryption Detail

1. IP

- 주어진 평문 8비트로 순열(치환) 연산을 하여 Index값을 변환 시킨다.

평문	1	0	1	0	0	1	0	1
Index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
값	0	1	1	1	0	1	0	0
Index	*2*	*6*	*3*	*1*	*4*	*8*	*5*	*7*

IP연산 후 얻어진 결과를 좌측부터 4비트씩 두 부분(0111 0100)으로 나눈 후 우측의 4비트 (0100) 를 가지고 F_k 함수 연산을 시작한다.

2.E/P (expansion/permutation)

- 입력된 4비트 값을 8비트로 확장 시키는 연산을 수행한다.

값	0	0	1	0	결과값 00101000
Index	4	1	2	3	
값	**1**	**0**	**0**	**0**	
Index	2	3	4	1	

3.XOR연산

- 확장된 8비트 값(00101000)과 입력받은 Key1의 값을 XOR연산 한다.

값	0	0	1	0	1	0	0	0
Key1	1	1	1	0	0	1	0	1
결과값	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**

- 얻어진 8비트 값을 다시 좌측부터 4 비트씩 2부분(1100 1101)으로 나누어 좌측 4 비트는 S0-box에 다음 4비트는 S1-box의 연산에 쓰여진다.

4.S-box 연산

- 두 부분으로 나누어진 4비트들에 각각 처음과 네번째 입력비트는 S박스의 행 번호 표기를 위한 2비트 숫자로 취급되고 나머지 두번째와 세번째 입력 비트는 S박스의 열번호 표기를 위한 2비트 숫자로 취급된다.

S-0	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

S-1	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

1100 의 입력값 : 01 출력

1101의 입력값 : 00 출력

5.P4

S-BOX 연산에 의해 얻어진값 0100 의 순열을 변환 시키는 연산

값	0	1	0	0
Index	1	2	3	4
값	1	0	0	0
Index	2	4	3	1

6. XOR 연산

- P4에 의해 얻어진 값 1000과 첫 IP연산후 얻어진 값 중 왼쪽 4비트 값을 XOR 연산 시킨다.

값	1	0	0	0
L	0	1	1	1
결과값	1	1	1	1

- 이때 얻어진 값(1111) 과 최초의 IP 연산후에 우측4비트로 나누어진 값(0100)이 스위치 연산에의해 (0100 1111) 의 8비트로 만들어지고 이를 다시 좌측부터 4비트씩 2부분으로 나누어 우측의 (1111)4비트를 가지고 다음의 fk함수연산을 실행한다.

7.E/P

- 입력된 4비트 값을 8비트로 확장 시키는 연산을 수행한다.

Value	1	1	1	1	결과값 11111111
Index	4	1	2	3	
Value	1	1	1	1	
Index	2	3	4	1	

8. XOR연산

- 확장된 8비트 값(11111111)과 입력받은 Key2의 값을 XOR 연산 한다.

값	1	1	1	1	1	1	1	1
Key2	0	1	1	0	0	0	1	1
결과값	1	0	0	1	1	1	0	0

- 얻어진 8비트 값을 다시 좌측부터 4 비트씩 2부분(1001 1100)으로 나누어 좌측 4 비트는 S1-box에 다음 4비트는 S2-box의 연산에 쓰여진다.

9.S-box 연산

- 두 부분으로 나누어진 4비트들에 각각 처음과 네번째 입력비트는 S박스의 행 번호 표기를 위한 2비트 숫자로 취급되고 나머지 두번째와 세번째 입력 비트는 S박스의 열번호 표기를 위한 2비트 숫자로 취급된다.

S-1	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

1001 의 입력값 : 11 출력

S-2	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

1100의 입력값 : 01 출력

10.P4

S-BOX 연산에 의해 얻어진값 1101의 순열을 변환 시키는 연산

값	1	1	0	1
Index	1	2	3	4
값	1	1	0	1
Index	2	4	3	1

11. XOR 연산

- P4에 의해 얻어진 값 1110과 처음 IP연산 후 얻어진 값중 오른쪽 4비트 값을 XOR 연산 시킨다.

값	1	1	0	1
R	0	1	0	0
결과값	1	0	0	1

12. IP^{-1} (Inverse IP)

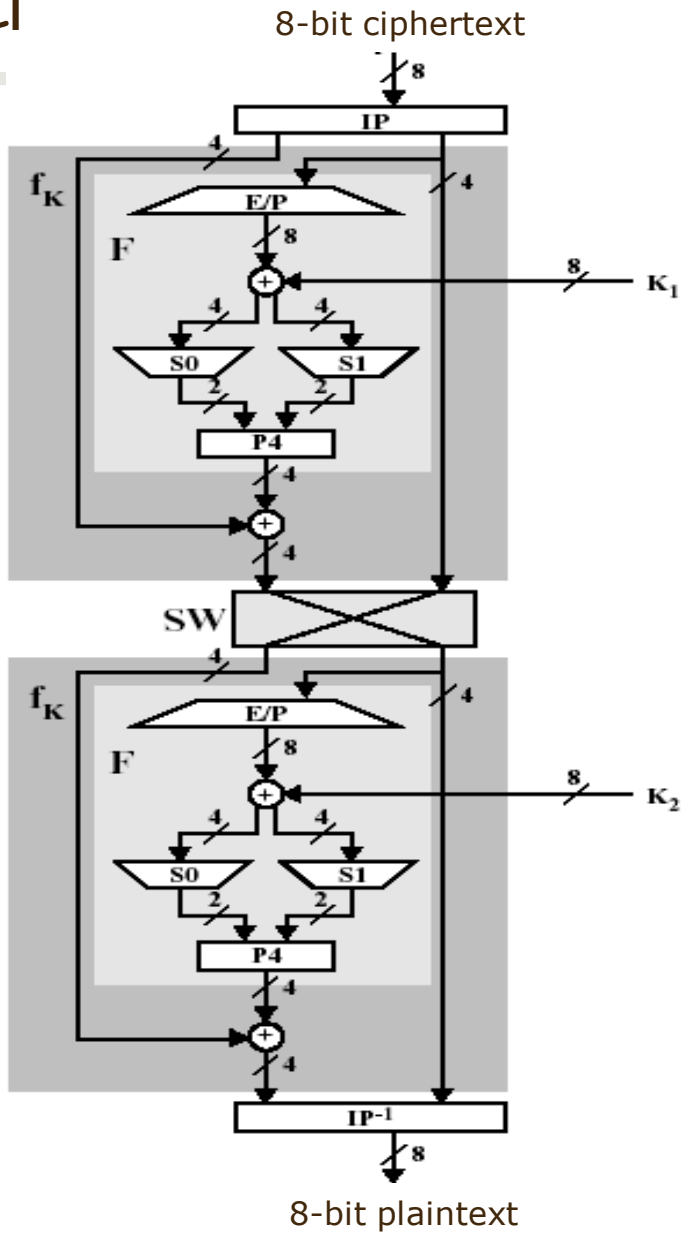
- 전 단계에서 얻어진 4비트 (1001)과 첫번째 fk의 두번째 XOR의 결과 4비트(1111)를 이용하여 역순열 작업을 하여준 후 여기서 얻어진 최종 값이 바로 최종의 S-DES 암호문 값이 된다.

평문	1	0	0	1	1	1	1	1
Index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
IP^{-1}	1	1	0	1	1	0	1	1
Index	*4*	*1*	*3*	*5*	*7*	*2*	*8*	*6*

암호문 : 11011011 (8bit ciphertext)

Part3.복호화

- 최종 으로 얻어진 암호문을 가지고 역순의 연산을 통해 최초로 주어졌던 평문을찾아 내는 방법



1. IP

- 주어진 암호문 8비트로 IP^{-1} 연산의 역인 IP연산 하여 Index값을 변환 시킨다.

암호문	1	1	0	1	1	0	1	1
Index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
값	1	0	0	1	1	1	1	1
Index	*2*	*6*	*3*	*1*	*4*	*8*	*5*	*7*

IP연산 후 얻어진 결과를 좌측부터 4비트씩 두 부분(1001 1111)으로 나눈 후 우측의 4비트 (1111) 를 가지고 f_k 함수 연산을 시작한다.

2.E/P (expansion/permutation)

- 입력된 4비트 값을 8비트로 확장 시키는 연산을 수행한다.

값	1	1	1	1	결과값 11111111
Index	4	1	2	3	
값	1	1	1	1	
Index	2	3	4	1	

3.XOR연산

- 확장된 8비트 값(01010101)과 입력받은 Key2의 값을 XOR연산 한다.

값	1	1	1	1	1	1	1	1
Key2	0	1	1	0	0	0	1	1
결과값	1	0	0	1	1	1	0	0

- 얻어진 8비트 값을 다시 좌측부터 4 비트씩 2부분(1001 1100)으로 나누어 좌측 4 비트는 S0-box에 다음 4비트는 S1-box의 연산에 쓰여진다.

4.S-box 연산

- 두 부분으로 나누어진 4비트들에 각각 처음과 네번째 입력비트는 S박스의 행 번호 표기를 위한 2비트 숫자로 취급되고 나머지 두번째와 세번째 입력 비트는 S박스의 열번호 표기를 위한 2비트 숫자로 취급된다.

S-0	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	**3**	1	3	2

1001 의 입력값 : 11 출력

S-1	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	**1**	0
3	2	1	0	3

1100의 입력값 : 01 출력

5.P4

S-BOX 연산에 의해 얻어진값 1101 의 순열을 변환 시키는 연산

값	1	1	0	1
Index	1	2	3	4
값	1	1	0	1
Index	2	4	3	1

6. XOR 연산

- P4에 의해 얻어진 값 1101과 암호화 과정에서 IP연산후에 얻어진 값 중 왼쪽 4비트 값을 XOR 연산 시킨다.

값	1	1	0	1
L	1	0	0	1
결과값	0	1	0	0

7.E/P

- 입력된 4비트 값을 8비트로 확장 시키는 연산을 수행한다.

Value	0	0	1	0	결과값 00101000
Index	4	1	2	3	
Value	1	0	0	0	
Index	2	3	4	1	

8. XOR연산

- 확장된 8비트 값(00101000)과 입력받은 Key1의 값을 XOR 연산 한다.

값	0	0	1	0	1	0	0	0
Key1	1	1	1	0	0	1	0	1
결과값	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**

- 얻어진 8비트 값을 다시 좌측부터 4 비트씩 2부분(1100 1101)으로 나누어 좌측 4 비트는 S1-box에 다음 4비트는 S2-box의 연산에 쓰여진다.

9.S-box 연산

- 두 부분으로 나누어진 4비트들에 각각 처음과 네번째 입력비트는 S박스의 행 번호 표기를 위한 2비트 숫자로 취급되고 나머지 두번째와 세번째 입력 비트는 S박스의 열번호 표기를 위한 2비트 숫자로 취급된다.

S-1	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

1100 의 입력값 : 01 출력

S-2	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

1101의 입력값 : 00 출력

10.P4

S-BOX 연산에 의해 얻어진값 0001의 순열을 변환 시키는 연산

값	0	1	0	0
Index	1	2	3	4
값	1	0	0	0
Index	2	4	3	1

11. XOR 연산

- P4에 의해 얻어진 값 0001과 첫 IP연산 후 얻어진 값 중 오른쪽 4비트 값을 XOR 연산 시킨다.

값	1	0	0	0
R	1	1	1	1
결과값	0	1	1	1

12. IP^{-1} (Inverse IP)

- 전 단계에서 얻어진 4비트 (0111)과 첫번째 fk의 두번째 XOR의 결과 4비트(0100)를 이용하여 역순열 작업을 하여준 후 여기서 얻어진 최종 값이 바로 최종의 S-DES 평문값이 된다.

Input	0	1	1	1	0	1	0	0
Index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
IP^{-1}	1	0	1	0	0	1	0	1
Index	*4*	*1*	*3*	*5*	*7*	*2*	*8*	*6*

평문 : 10100101 (8bit plaintext)

Feistel Cipher Structure

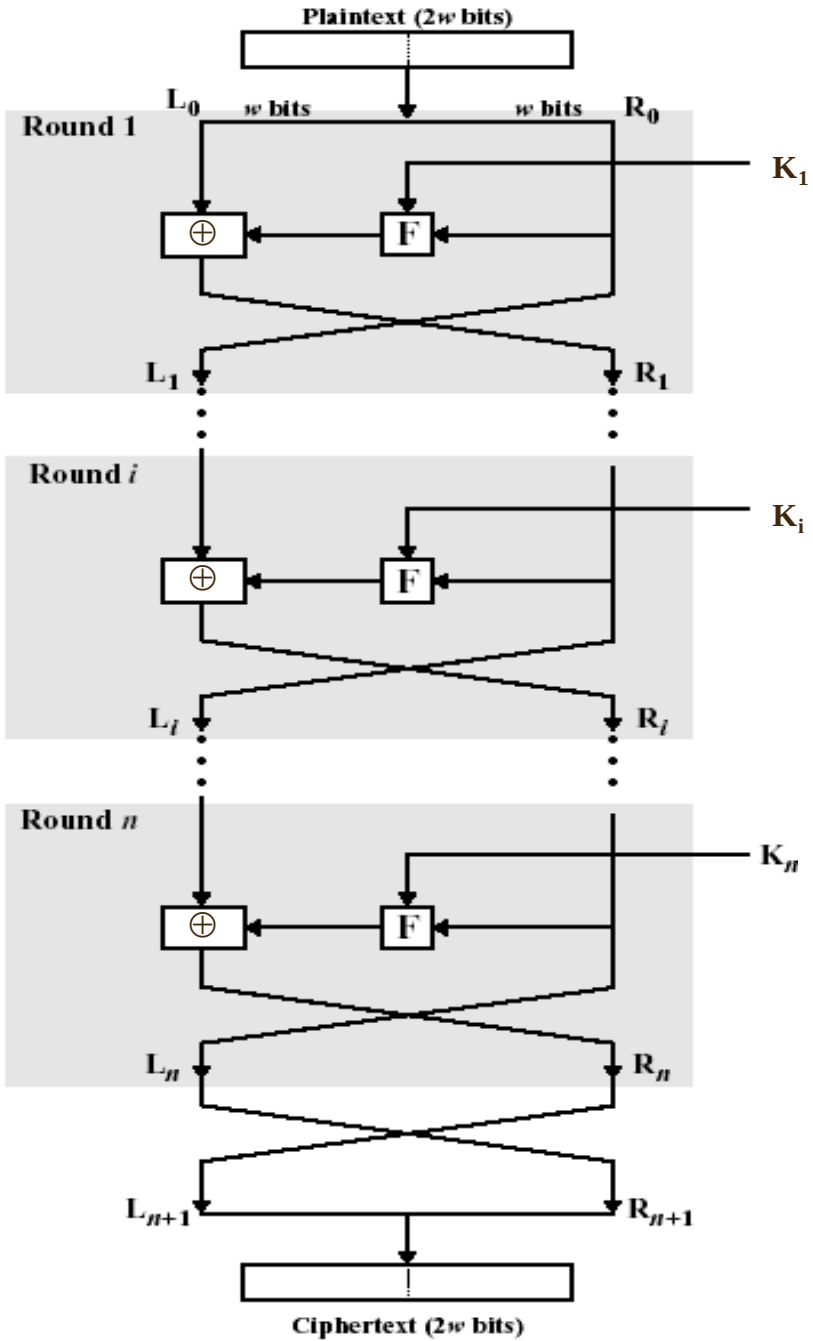
- Virtually all conventional block encryption algorithms, including DES have a structure first described by Horst Feistel of IBM in 1973

- The realization of a Fesitel Network depends on the choice of the following parameters and design features (see next slide):

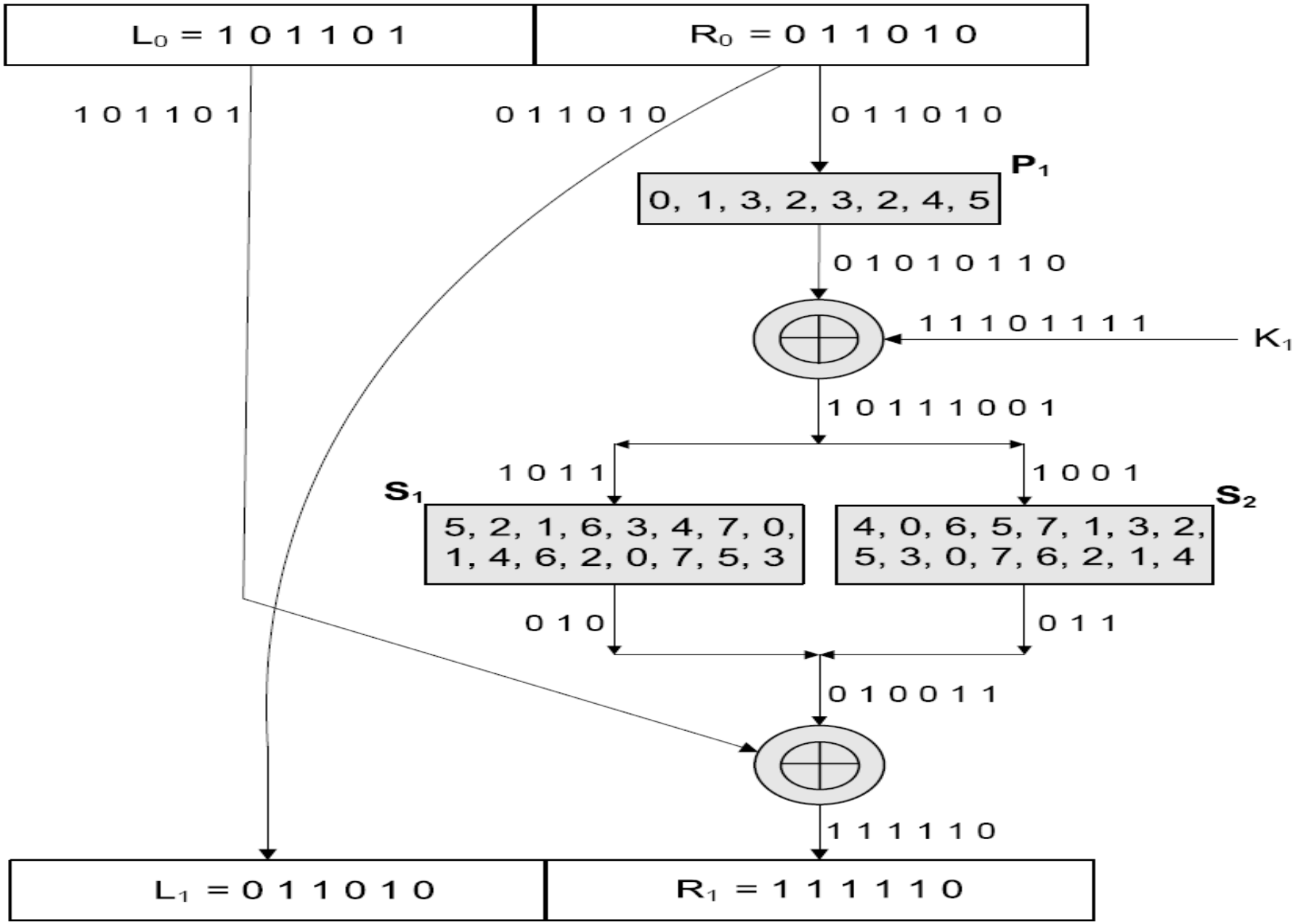
Feistel Cipher Structure

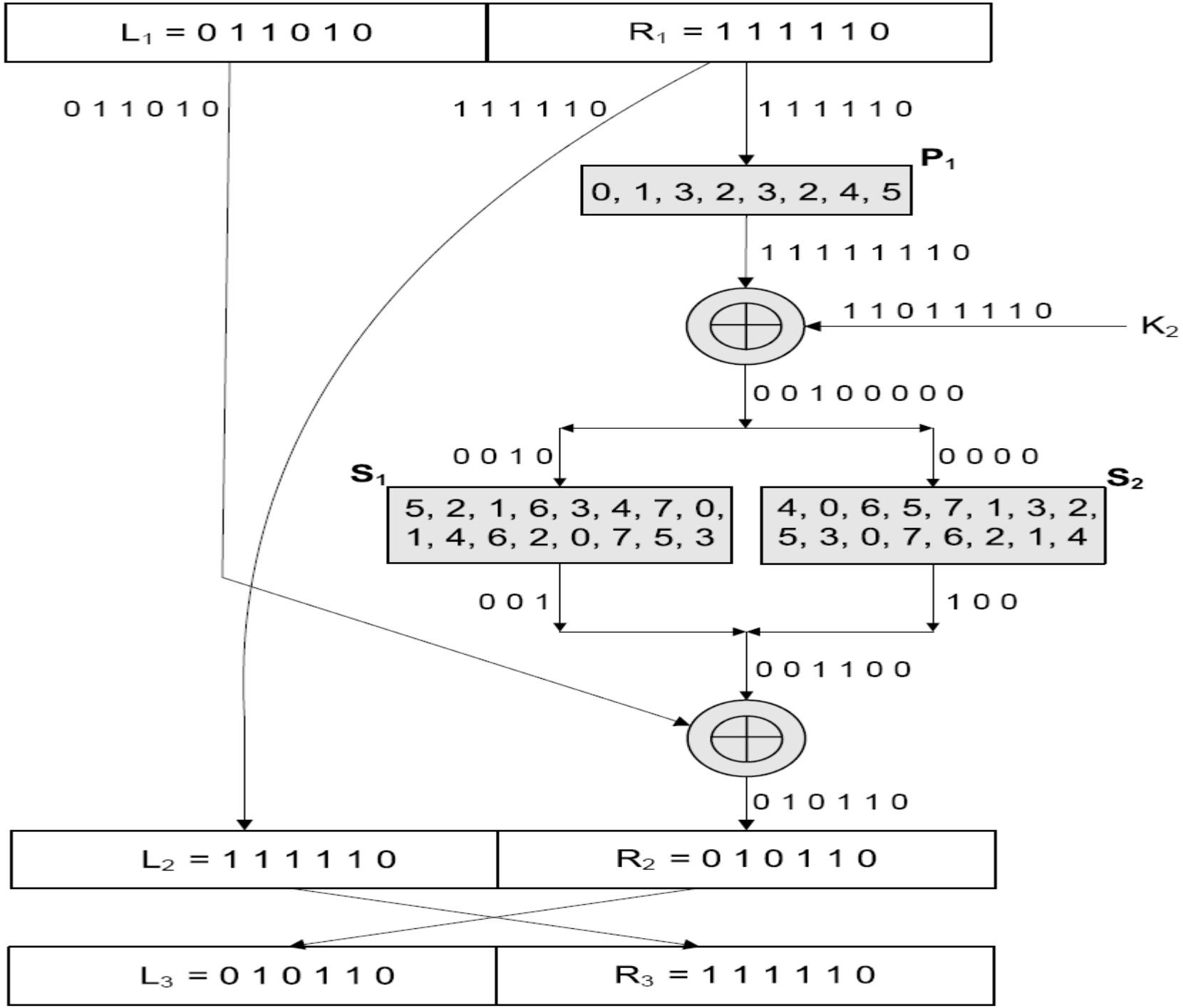
- **Block size:** larger block sizes mean greater security
- **Key Size:** larger key size means greater security
- **Number of rounds:** multiple rounds offer increasing security
- **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
- **Fast software encryption/decryption:** the speed of execution of the algorithm becomes a concern

Classical Feistel Network



An Example of a Feistel Cipher
(Block Size 12, Two Rounds, Key Size 9, Subkey Size 8)

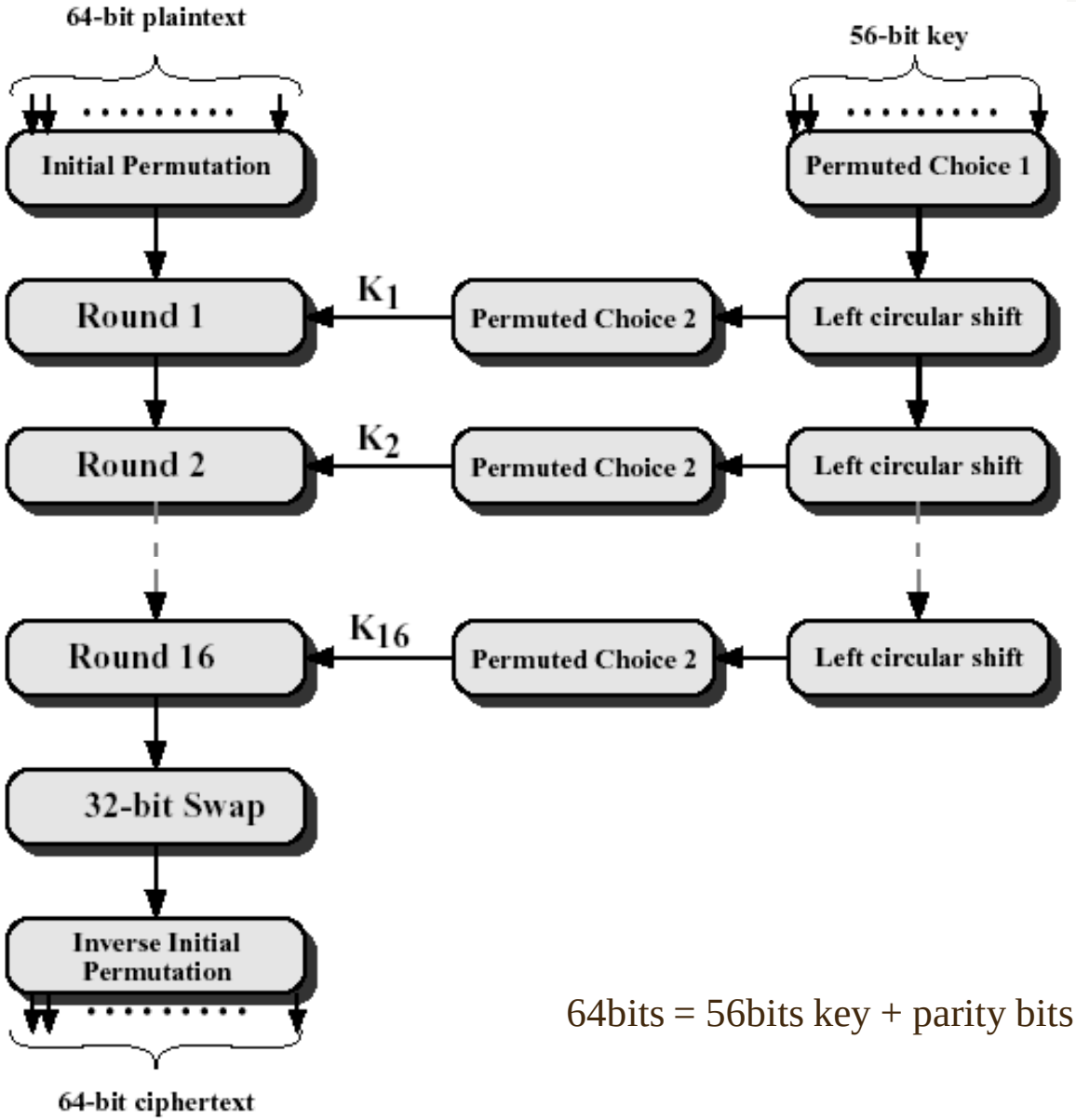



L1 = 0 1 1 0 1 0
R1 = 1 1 1 1 1 0
0 1 1 0 1 0
1 1 1 1 1 0
1 1 1 1 1 0
P1
0, 1, 3, 2, 3, 2, 4, 5
1 1 1 1 1 1 1 0
1 1 0 1 1 1 1 0
K2
0 0 1 0 0 0 0 0
0 0 1 0
0 0 0 0
S1
S2
5, 2, 1, 6, 3, 4, 7, 0,
1, 4, 6, 2, 0, 7, 5, 3
4, 0, 6, 5, 7, 1, 3, 2,
5, 3, 0, 7, 6, 2, 1, 4
0 0 1
1 0 0
0 0 1 1 0 0
0 1 0 1 1 0
L2 = 1 1 1 1 1 0
R2 = 0 1 0 1 1 0
L3 = 0 1 0 1 1 0
R3 = 1 1 1 1 1 0

Conventional Encryption Algorithms

- Data Encryption Standard (DES)
  - The most widely used encryption scheme
  - The algorithm is reffered to the Data Encryption Algorithm (DEA)
  - DES is a block cipher
  - The plaintext is processed in 64-bit blocks
  - The key is 56-bits in length

General Depiction of DES Encryption Algorithm



64bits = 56bits key + parity bits or simply set arbitrary

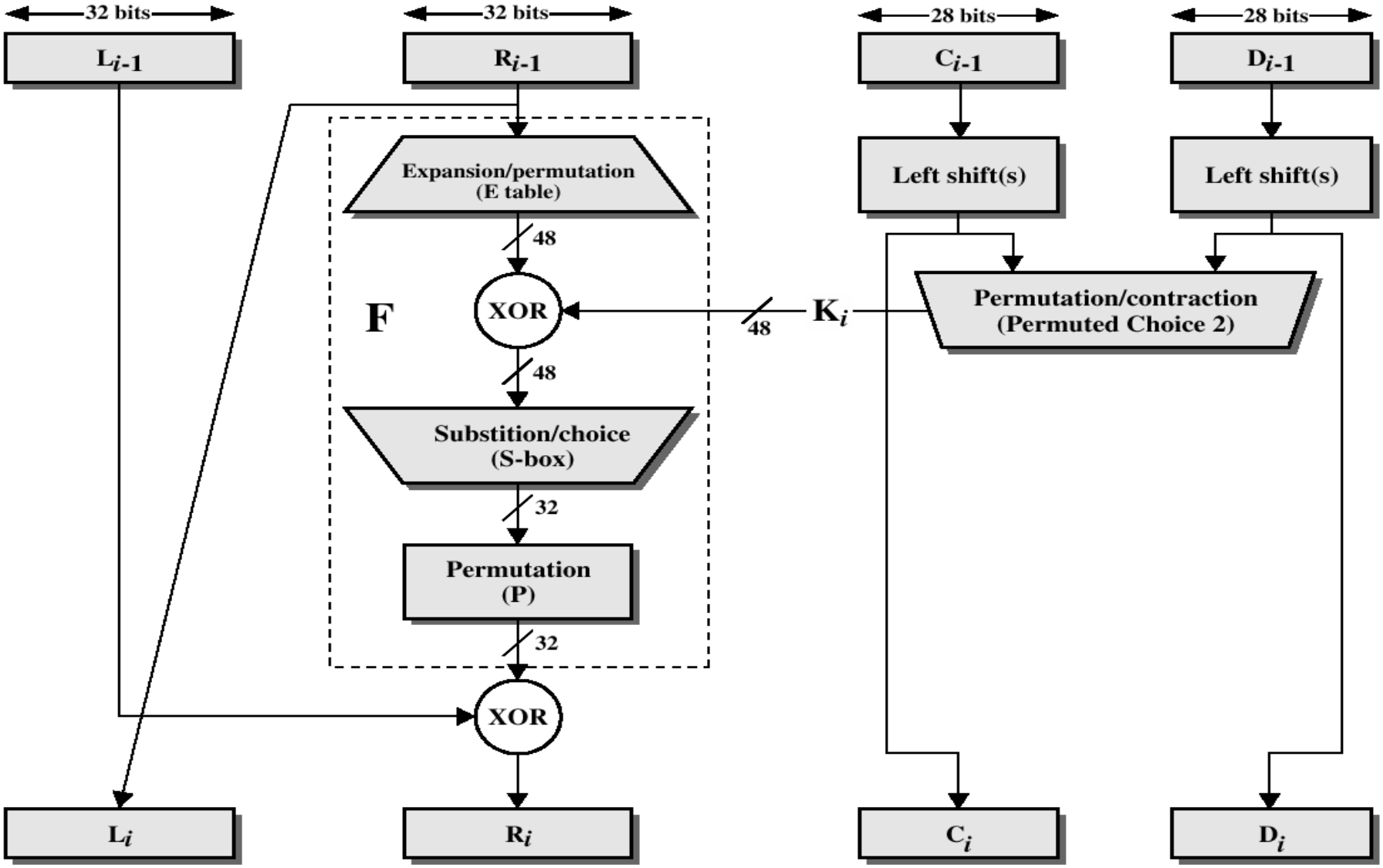
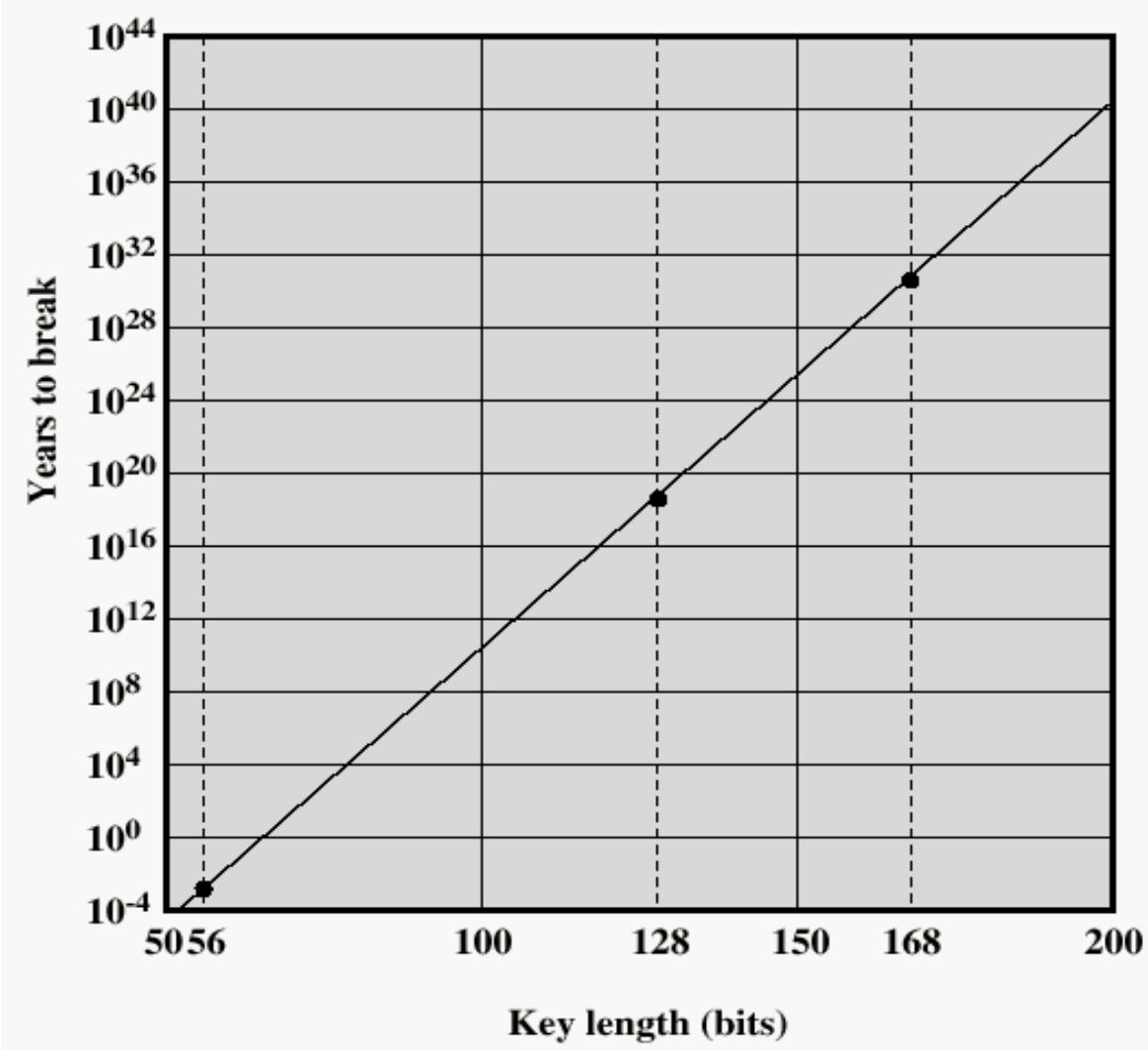


Figure 2.4 Single Round of DES Algorithm

DES

- **The overall processing at each iteration:**
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \otimes F(R_{i-1}, K_i)$
- Concerns about:
  - The algorithm and the key length (56-bits)

Time to break a code (10^6 decryptions/µs)



Triple DEA

- Use three keys and three executions of the DES algorithm (encrypt-decrypt-encrypt)

$$C = E_{K3}[D_{K2}[E_{K1}[P]]]$$

  - C = ciphertext
  - P = Plaintext
  - $E_K[X]$ = encryption of X using key K
  - $D_K[Y]$ = decryption of Y using key K
- Effective key length of 168 bits

Triple DEA

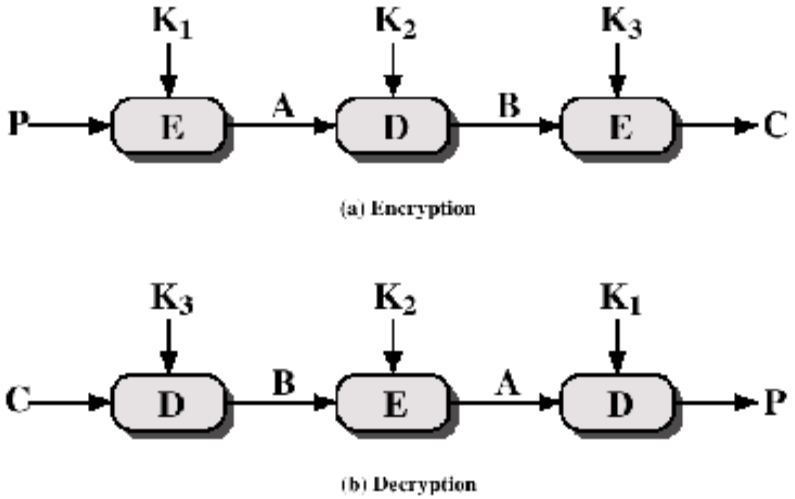


Figure 2.6 Triple DEA

Other Symmetric Block Ciphers

- **International Data Encryption Algorithm (IDEA)**
  - 128-bit key
  - Used in PGP(Pretty Good in Privacy)
- **Blowfish**
  - Easy to implement
  - High execution speed
  - Run in less than 5K of memory

Other Symmetric Block Ciphers

- **RC5**
  - Suitable for hardware and software
  - Fast, simple
  - Adaptable to processors of different word lengths
  - Variable number of rounds
  - Variable-length key
  - Low memory requirement
  - High security
  - Data-dependent rotations
- **Cast-128**
  - Key size from 40 to 128 bits
  - The round function differs from round to round

Cipher Block Modes of Operation

- Cipher Block Chaining Mode (CBC)
  - The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block.
  - Repeating pattern of 64-bits are not exposed
  - For first block, need “initialization vector”, IV
  - IV must be known to sender and receiver (often all 0’s)

$$C_i = E_k[C_{i-1} \oplus P_i]$$

$$D_K[C_i] = D_K[E_K(C_{i-1} \oplus P_i)]$$

$$D_K[C_i] = (C_{i-1} \oplus P_i)$$

$$C_{i-1} \oplus D_K[C_i] = C_{i-1} \oplus C_{i-1} \oplus P_i = P_i$$

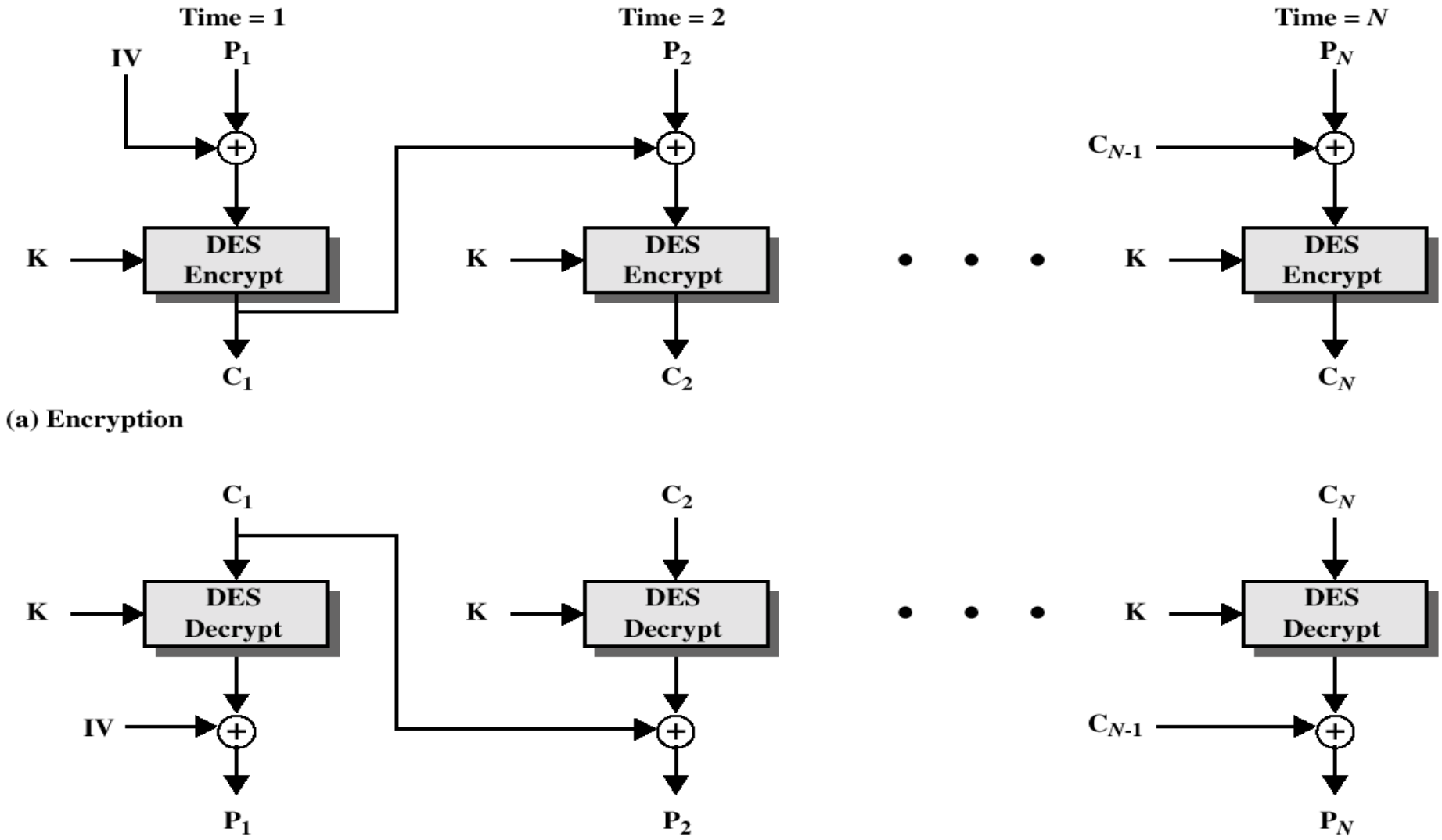
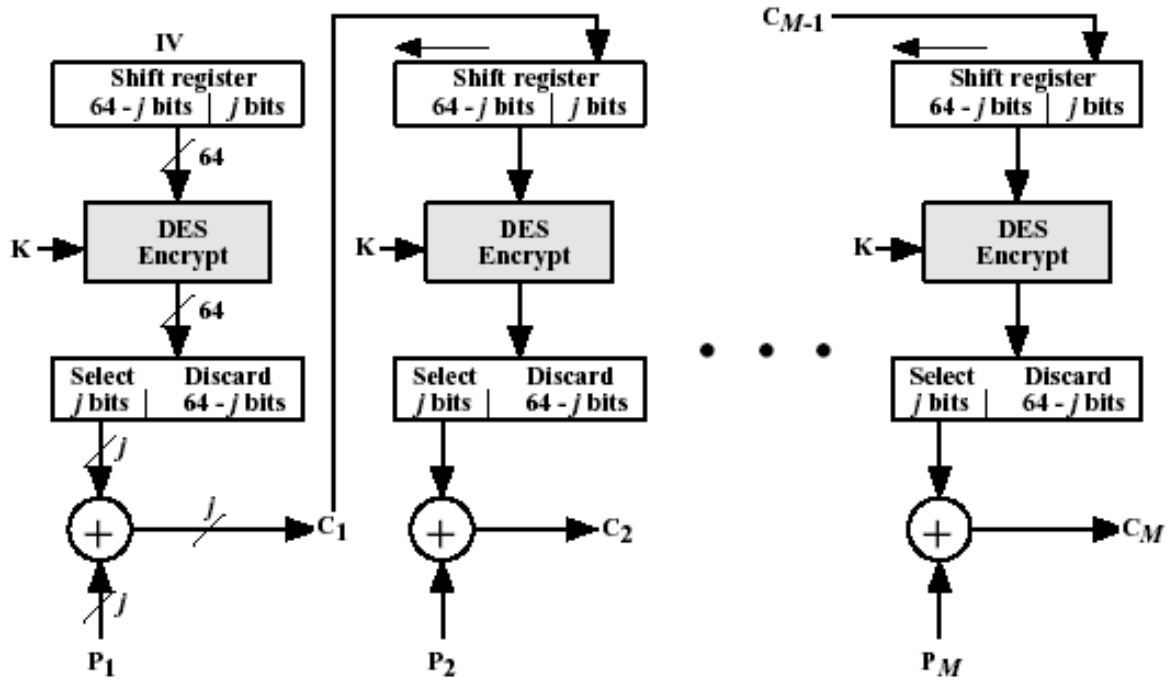


Figure 2.7 Cipher Block Chaining (CBC) Mode

Cipher Feedback (CFB) Mode

- Allows use of DES as a *stream cipher* (appropriate when data inherently arrives in bits/bytes)
- Start with IV
- Encrypt
- XOR (MSB) *j* bits of output with *j* bit plaintext
  - Result is ciphertext
- Shift IV by *j* bits, insert ciphertext

J-bit CFM Mode (Encryption)



Location of Encryption Device

- **Link encryption:**
  - A lot of encryption devices
  - High level of security
  - Decrypt each packet at every switch
- **End-to-end encryption**
  - The source encrypt and the receiver decrypts
  - Payload encrypted
  - Header in the clear
- **High Security:** Both link and end-to-end encryption are needed (see Figure 2.9)

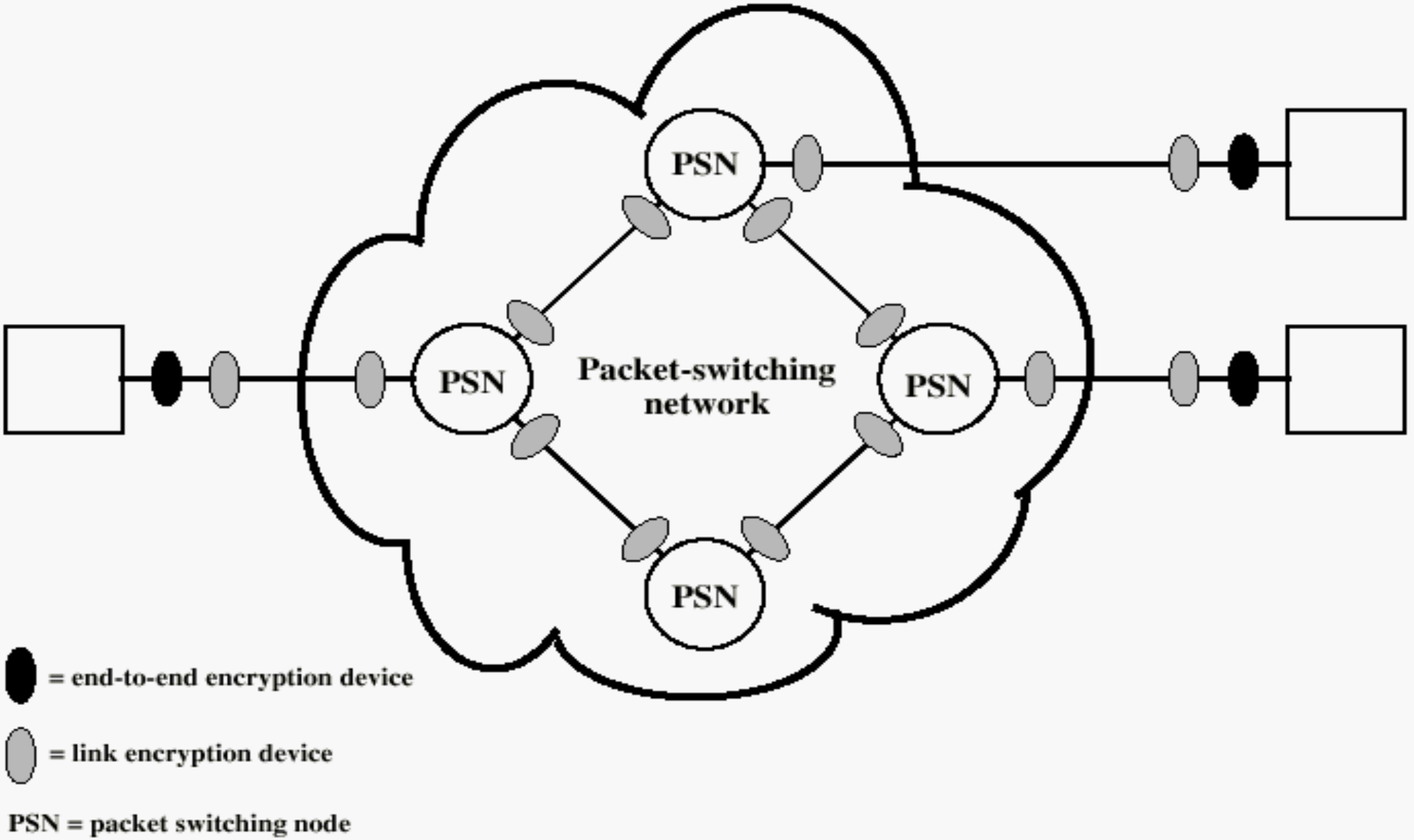


Figure 2.9 Encryption Across a Packet-Switching Network

Key Distribution

1. A key could be selected by A and physically delivered to B.
2. A third party could select the key and physically deliver it to A and B.
3. If A and B have previously used a key, one party could transmit the new key to the other, encrypted using the old key.
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B.

Key Distribution (See Figure 2.10)

- **Session key:**
  - Data encrypted with a one-time session key. At the conclusion of the session the key is destroyed
- **Permanent key:**
  - Used between entities for the purpose of distributing session keys

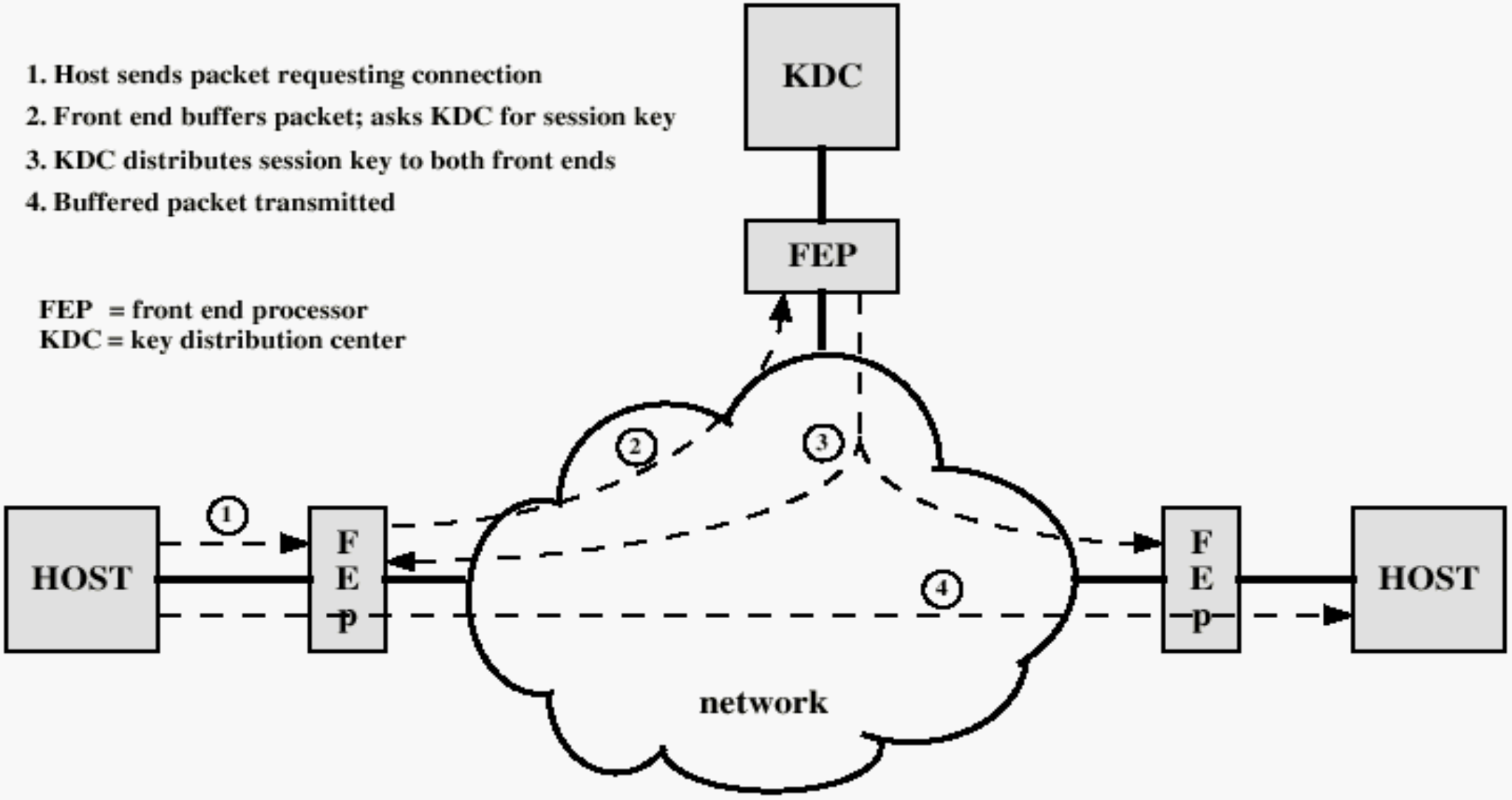


Figure 2.10 Automatic Key Distribution for Connection-Oriented Protocol

Summary

- Conventional Description Algorithms
- Requirements of DES
- Key generation for simplified DES
  - Simplified DES Scheme Encryption Detail
- Location of Encryption Device
- Key distribution by KDC