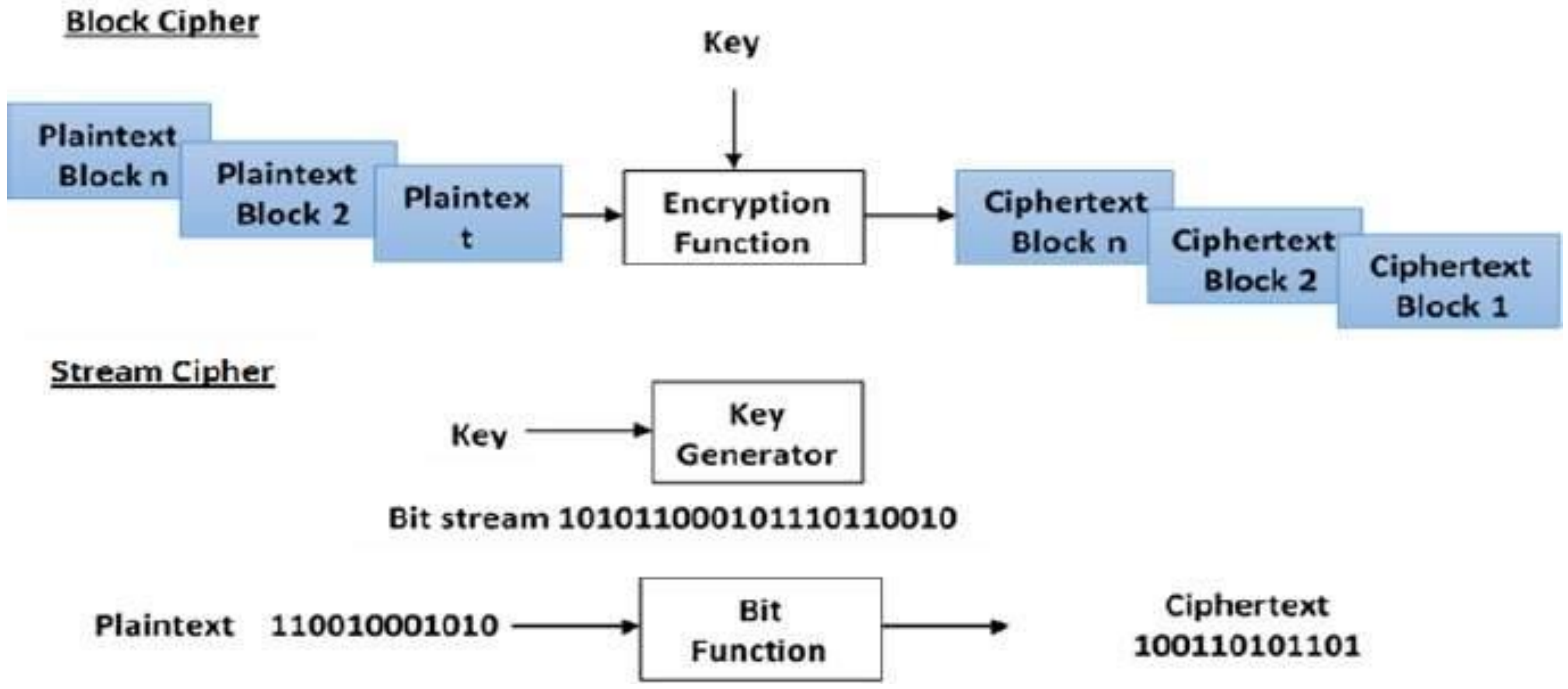
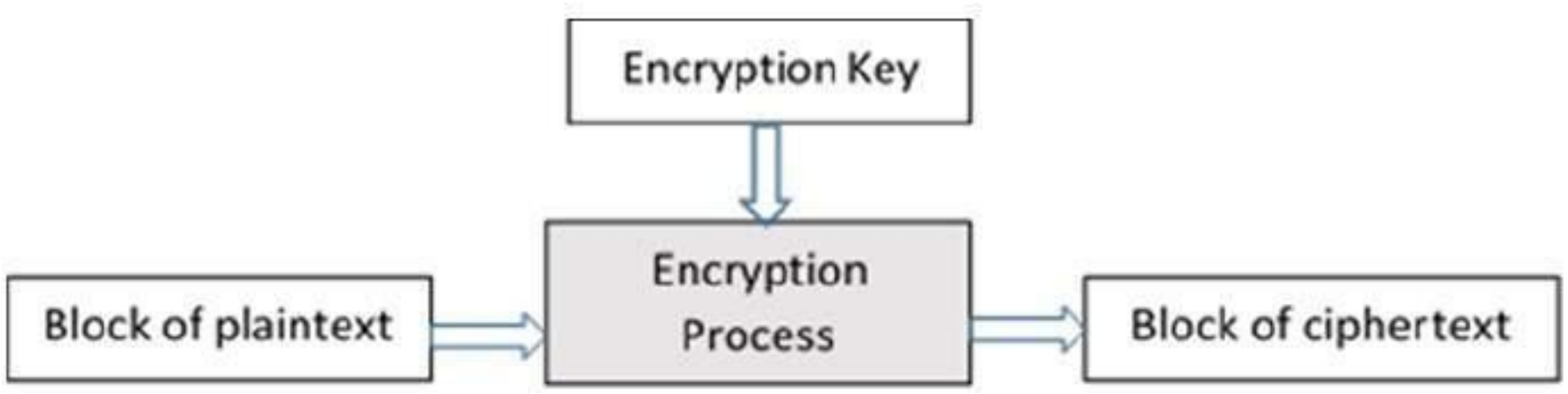
Block Cipher Vs Stream Cipher

- **Stream Cipher:** Encrypts digital data stream one bit or one byte at a time
- **Block Cipher:** A block of plain text is treated as a whole and produces a cipher text block of equal size (Typically 64 or 128 bits are used)

Block and Stream Cipher



Block Cipher



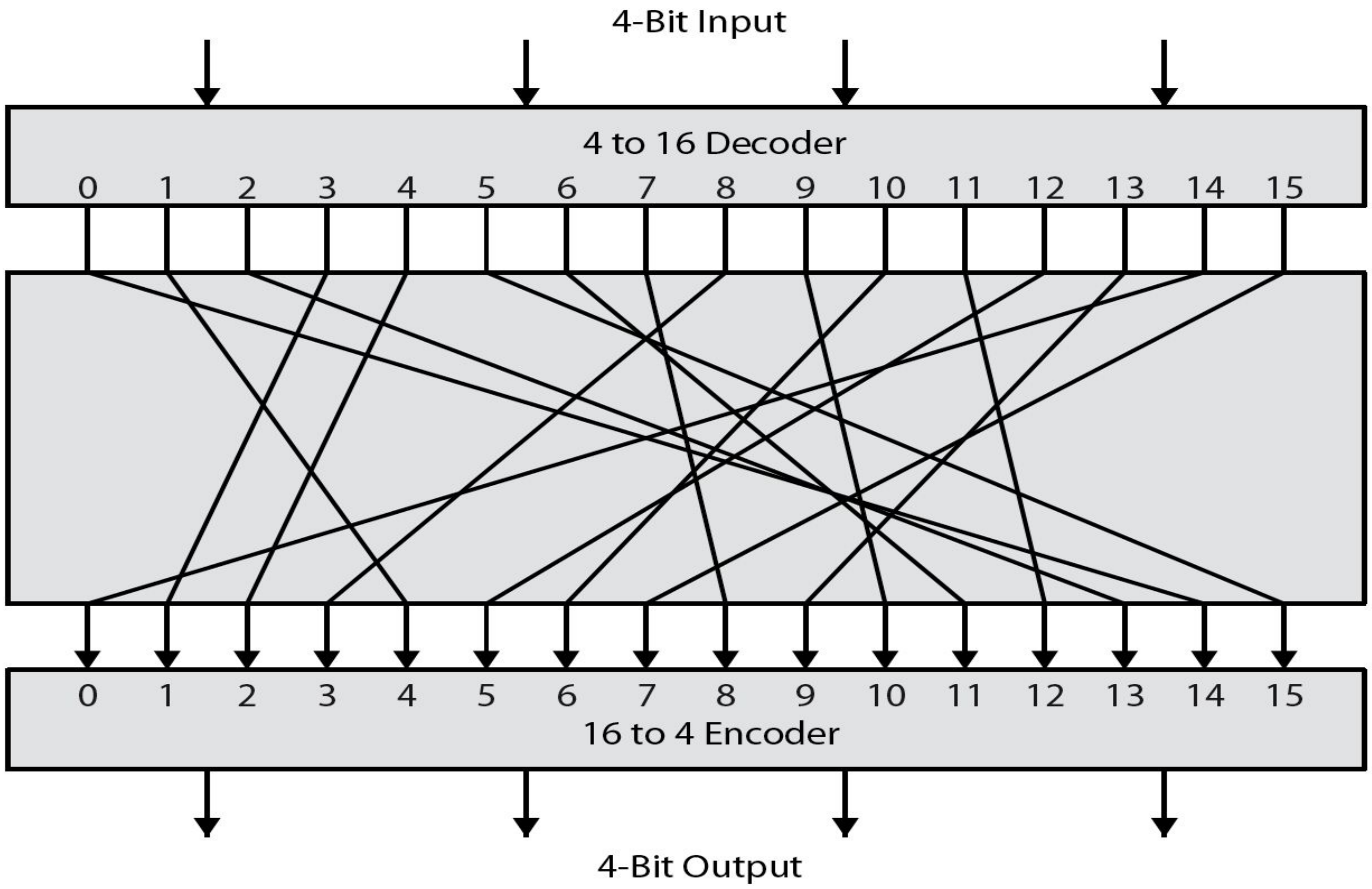
Block Cipher

- Block Size
  - Should not be too big
  - Should not be too small
  - In multiples of 2^m
  - Padding of bits for fixed sized blocks
- Some popular Block cipher Schemes
  - DES
  - AES
  - Idea
  - Blowfish
  - Triple DES   etc

Difference between Block Cipher and Stream Cipher

S.No	**Block Cipher**	**Stream Cipher**
1	Block ciphers encrypt plaintext in chunks	Stream ciphers encrypt plaintext one byte or one bit at a time
2	In blocks of 64,128,168	Stream ciphers operate upon a series of binary digits
3	Block ciphers typically execute at a lower speed	Stream ciphers typically execute at a higher speed
4	Have higher hardware complexity	Have lower hardware complexity
5	Block ciphers are often used in applications where plaintext comes in quantities of known length	Stream ciphers are often used in applications where plaintext comes in quantities of unknowable length

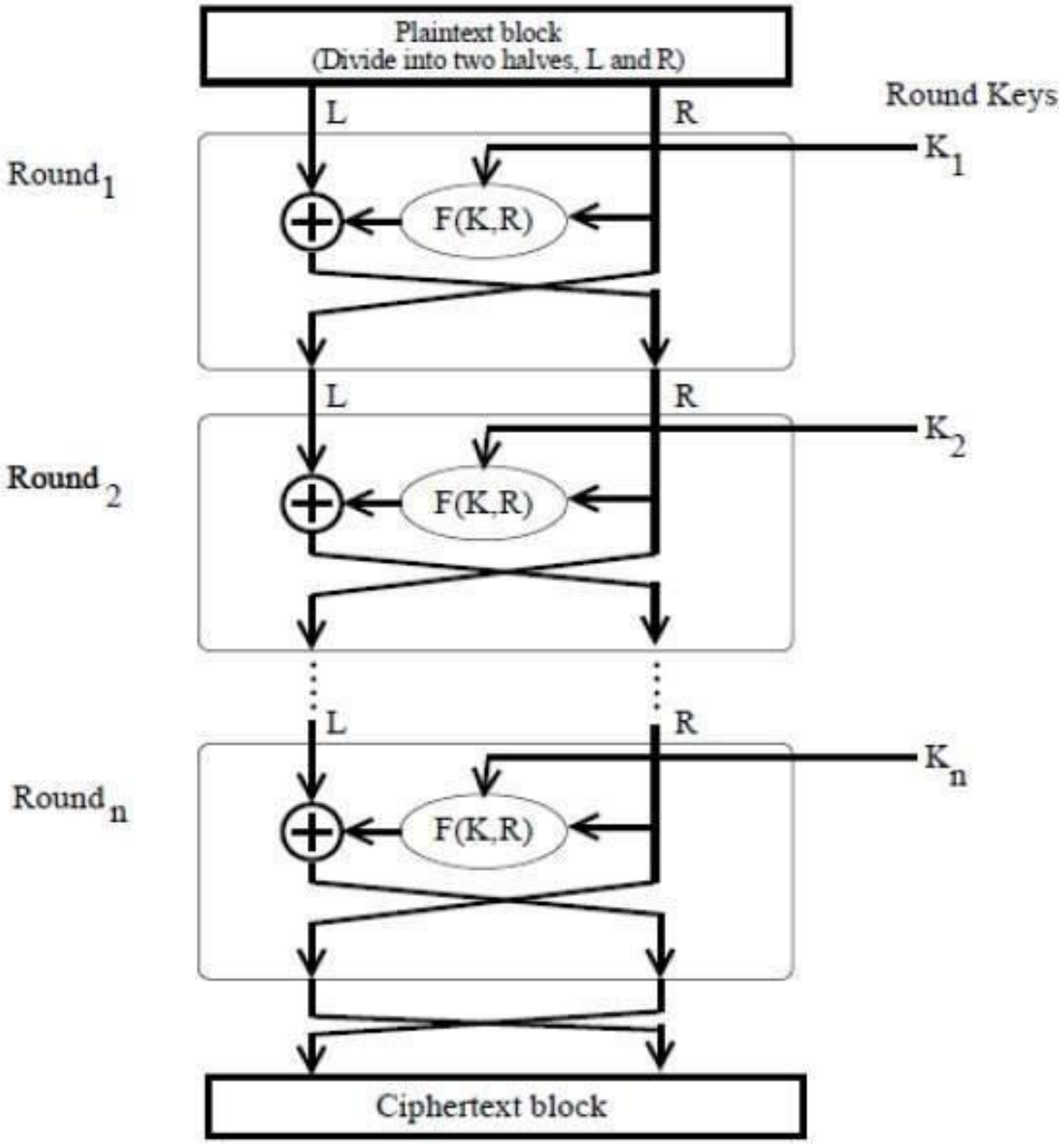
Ideal Block Cipher



Feistel Block Cipher

- It is a design model from which many different block ciphers are derived.
- DES is just one example of a Feistel Cipher.
- A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.

Feistel Structure



The Feistel Cipher

- **Two Principles:**

 Confusion: To make the relationship between cipher text and encryption key as complex as possible

 Diffusion: Each plain text digit effect the value of many cipher text digits or in other words the cipher text is effected by many plaintext digits. (to make the relationship hard between plain text and cipher text)

Feistel Cipher Design Elements

- Block Size
- Key size
- Number of Rounds: 16
- Sub-key Generation Algorithm
- Round Function: F does not have to be invertible
- Fast Software en/decryption
- Ease Analysis

Encryption Step in Feistel Cipher

- The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext
- Each round consisting of a “substitution” step followed by a permutation step.
- The input block to each round is divided into two halves L and R .
- In each round, the right half of the block, R, goes through unchanged.
- The left half, L, goes through an operation that depends on R and the encryption key.
- First, we apply an encrypting function ‘f’ that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a ‘round’.
- The number of rounds are specified by the algorithm design.
- Once the last round is completed then the two sub blocks, ‘R’ and ‘L’ are concatenated in this order to form the ciphertext block.
- More number of rounds gives more secure system
- The difficult part of designing a Feistel Cipher is selection of round function ‘f’.

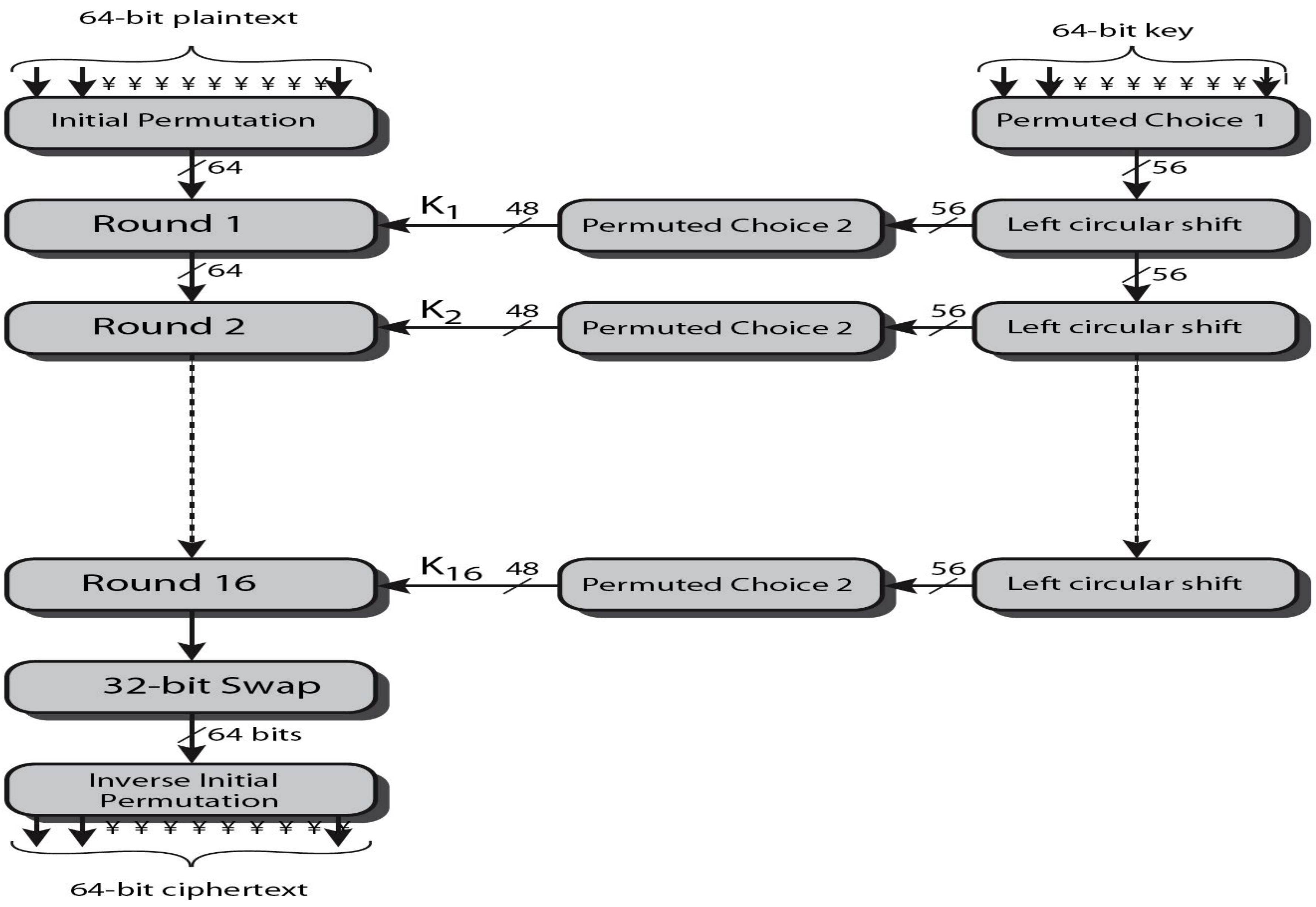
Decryption Step in Feistel Cipher

- The process of decryption in Feistel cipher is almost similar.
- Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.
- The process is said to be almost similar and not exactly same.
- In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.
- The final swapping of ‘L’ and ‘R’ in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.

Conventional Encryption Data Encryption Standard(DES)

- DES is an implementation of a Feistel Cipher.
- It uses 16 round Feistel structure.
- The block size is 64-bit.
- Key length is 64-bit, DES has an effective key length of 56 bits,8 of the 64 bits of the key are parity bits

DES Encryption Overview



DES Encrypting Details

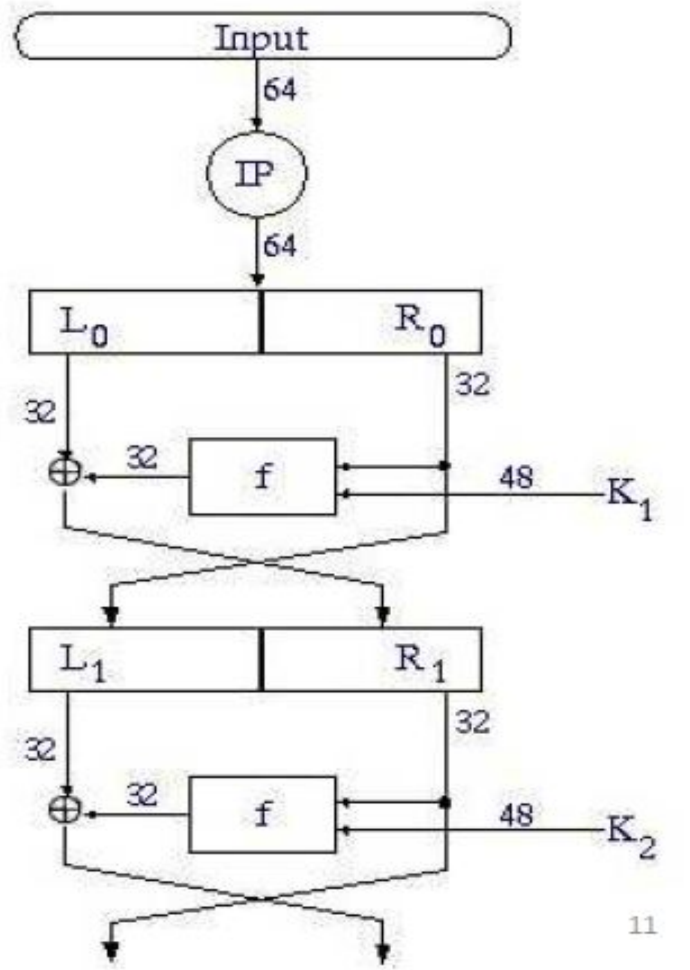
Processing takes place in 3 phases

1) Initial Permutation
2) Phase consisting of 16 rounds of the same function which involves both permutation and substitution
3) The o/p of the last round consists of 64 bits that are a function of input plaintext and key.
4) Left and right halves output are swapped to produce the preoutput
5) Finally the pre output is passed through inverse permutation

DES Encrypting Details

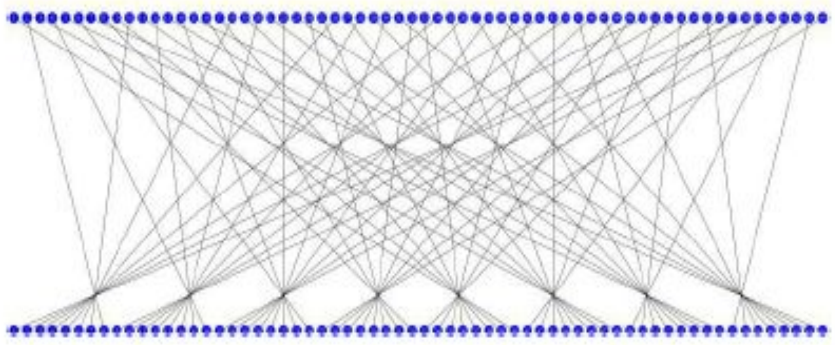
- $IP(x) = L_0R_0$
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- $y = IP^{-1}(R_{16}L_{16})$

Note: IP means Initial Permutation



DES Initial Permutation

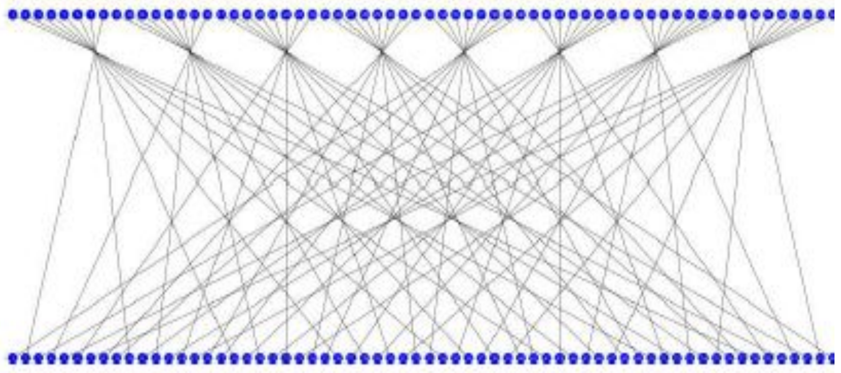
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7



- 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.

DES Final Permutation

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.

Single Round of DES

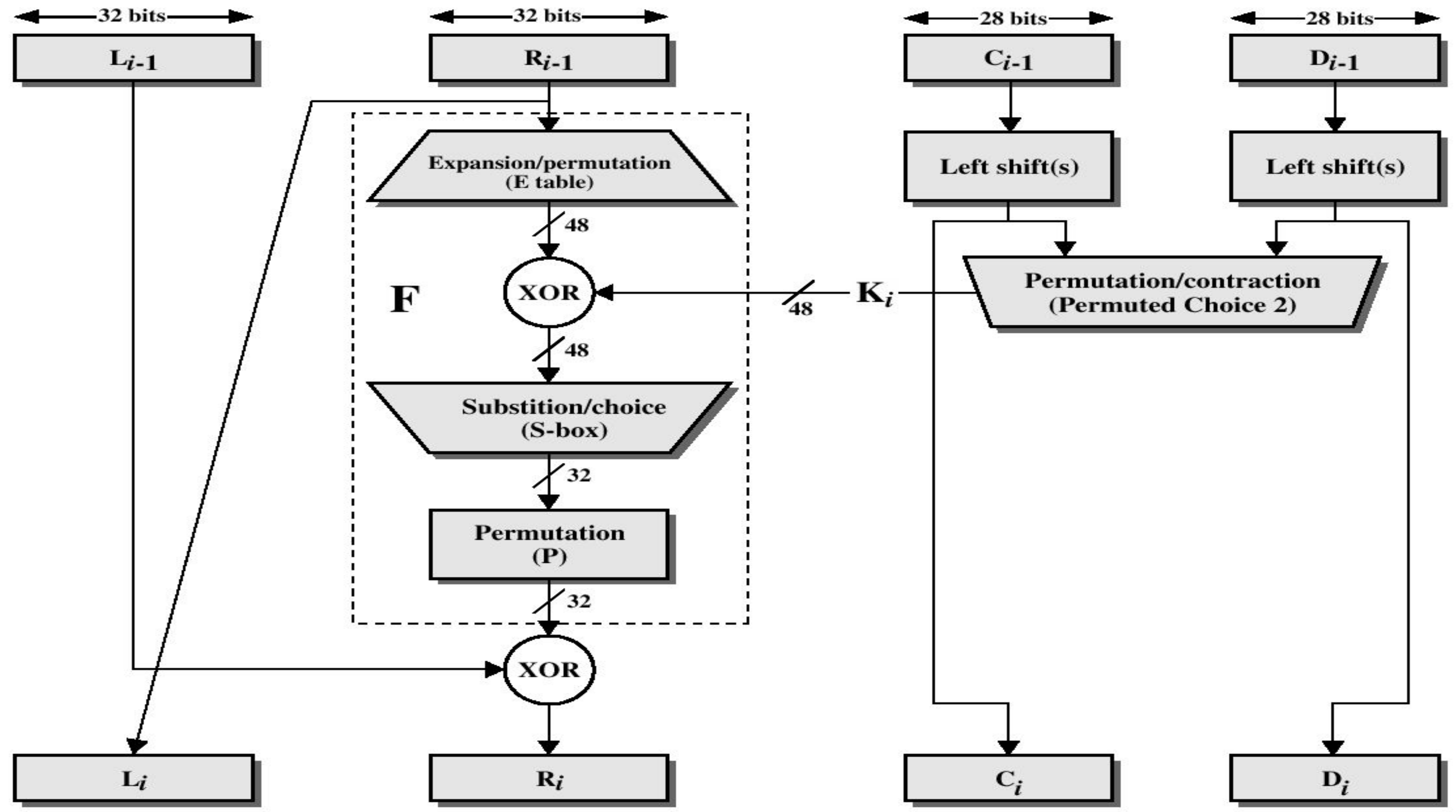


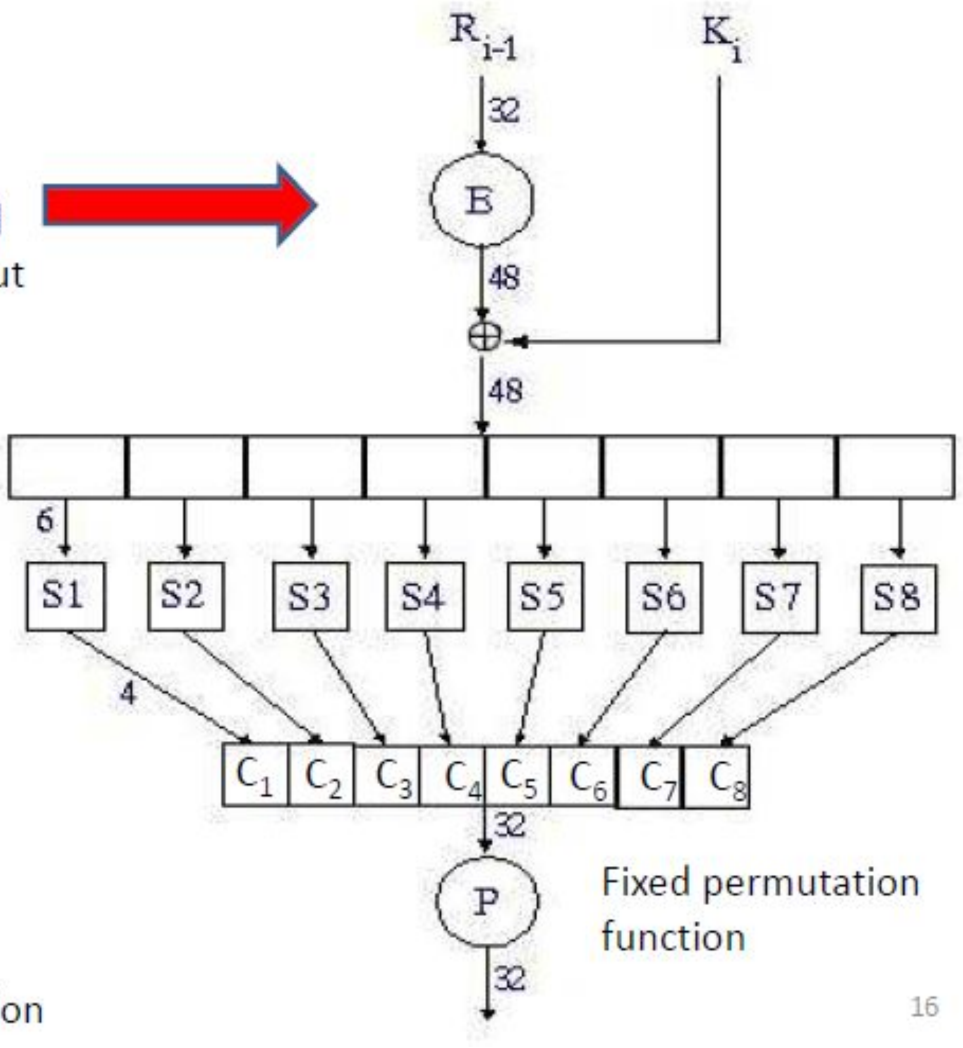
Figure 2.4 Single Round of DES Algorithm

DES Function

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

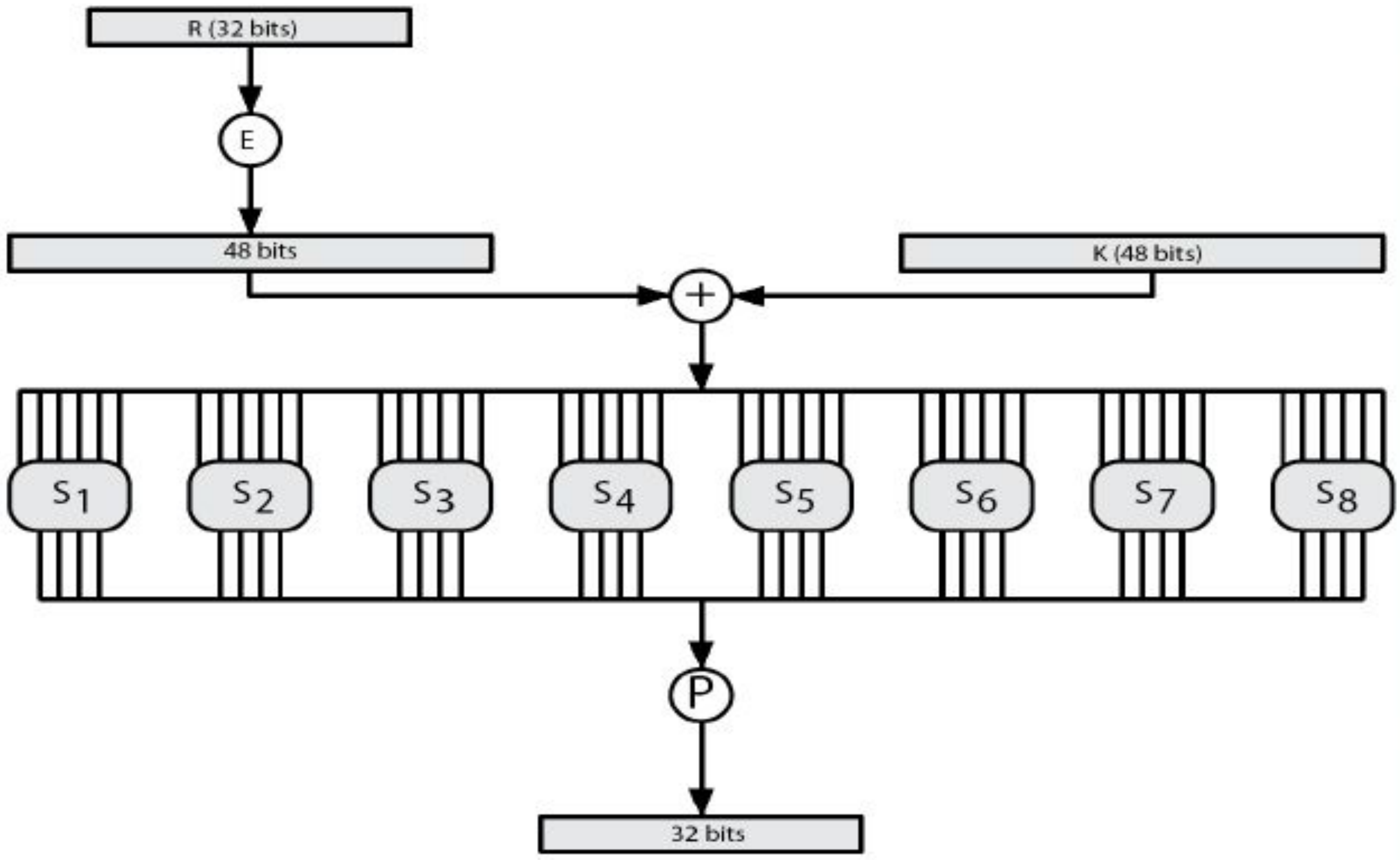
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

16 bits appear twice, in the expansion



Fixed permutation function

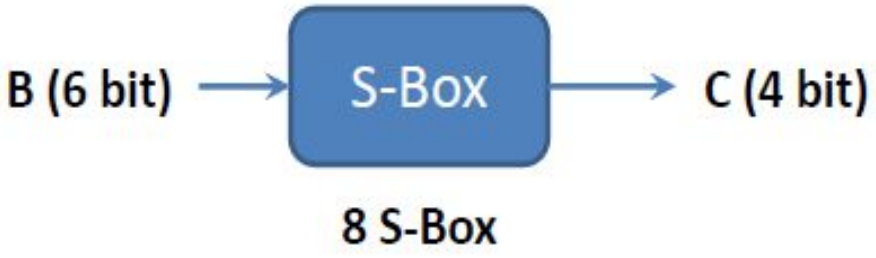
DES S-Boxes



DES S-Boxes

- S-boxes are the only *non-linear* elements in DES desigr

Each of the unique selection functions **$S_1,S_2,\ldots,S_8$**, takes a 6-bit block as input and yields a 4-bit block as output



- 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 S-Boxes

Example (S1)

Row #	S_1	1	2	3	…			7								15	Column #
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

Example: B =101111

b_1b_6 = 11 = row 3

$b_2b_3b_4b_5$= 0111 = column 7

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

Another example: B=011011, C=?

DES S-Boxes

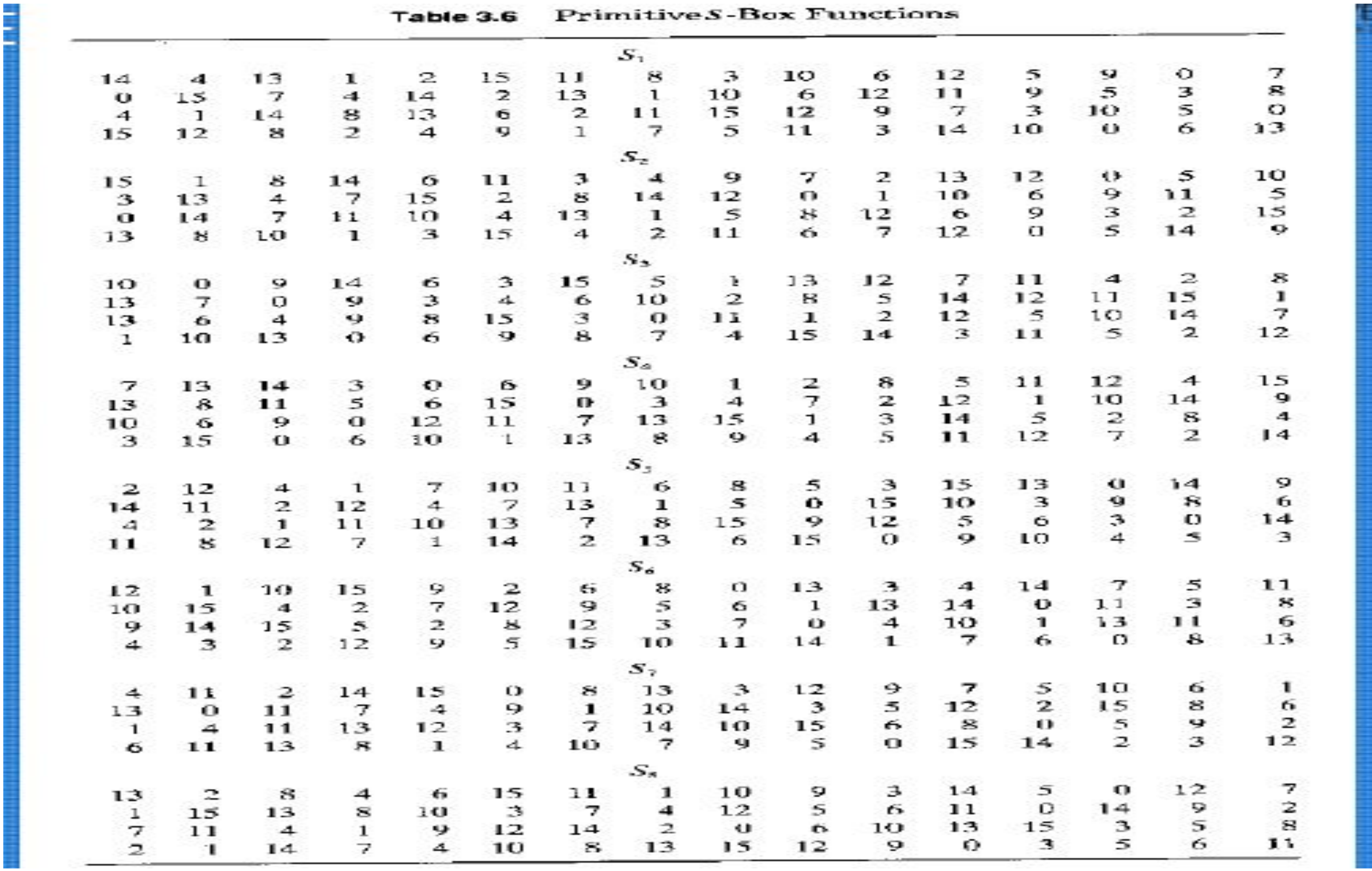
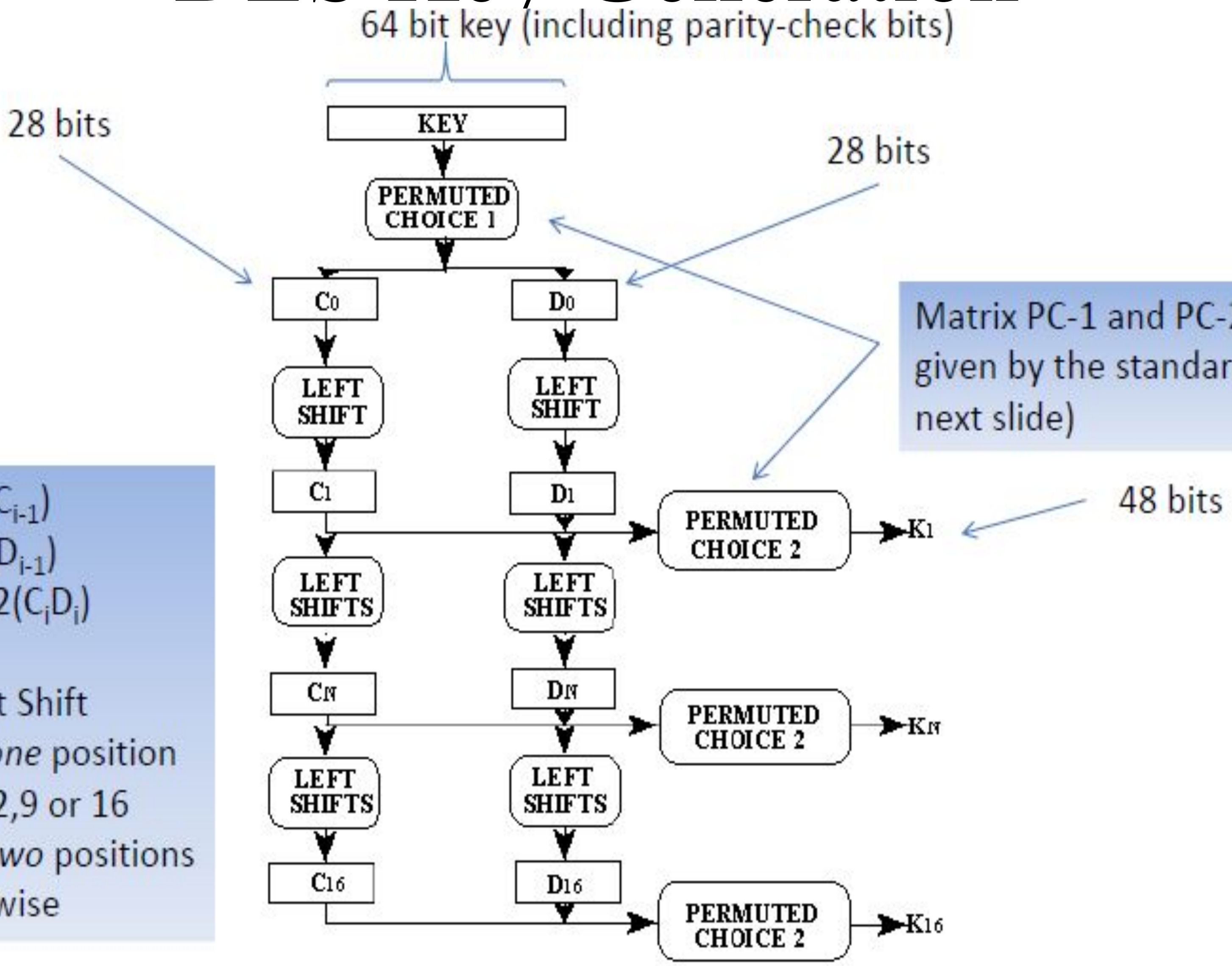


Table 3.6 Primitive *S*-Box Functions

							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_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_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_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_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_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_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_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

DES Key Generation

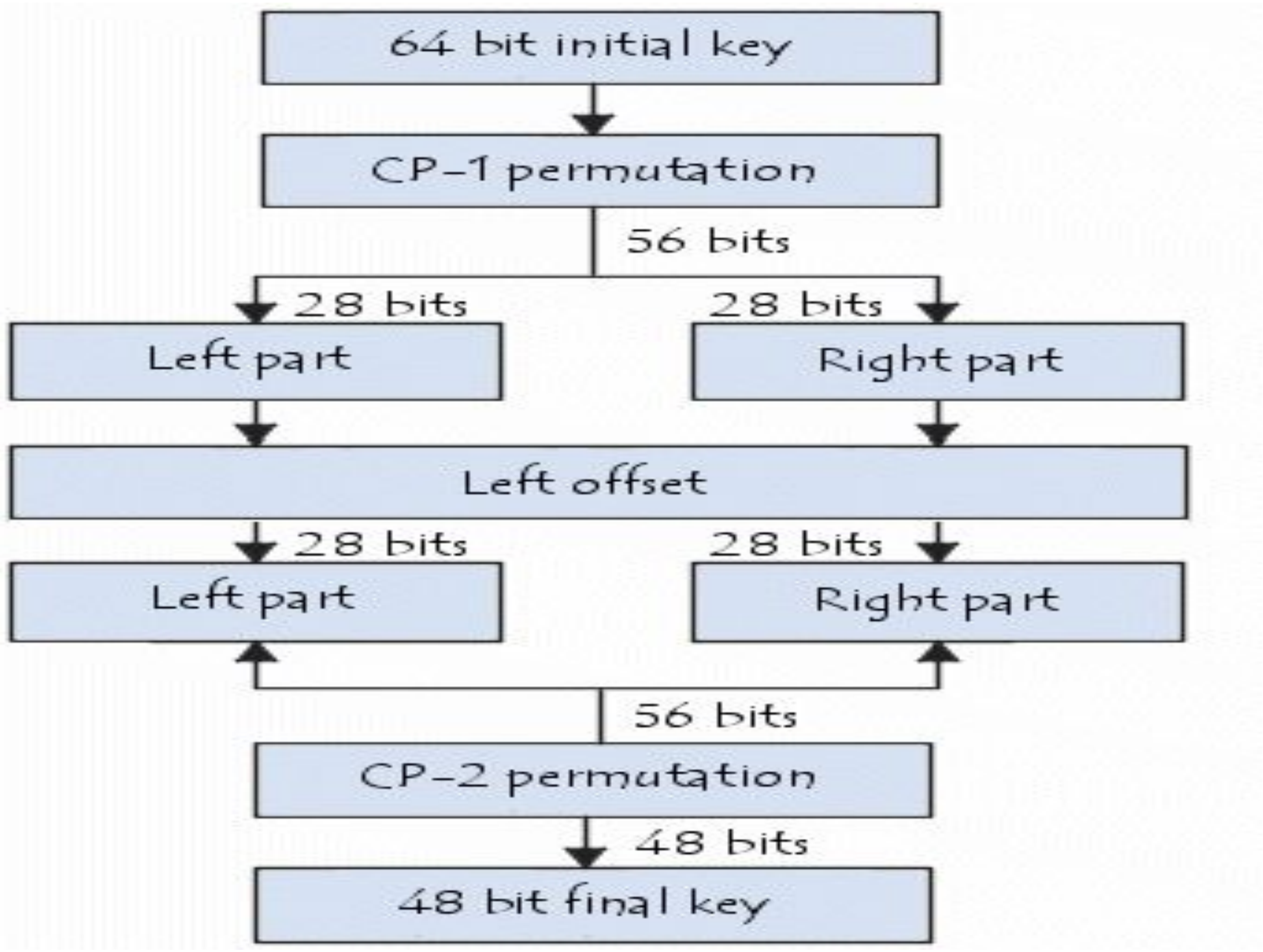


Matrix PC-1 and PC-2 are given by the standard (see next slide)

$C_i=LS_i(C_{i-1})$
$D_i=LS_i(D_{i-1})$
$K_i=PC\text{-}2(C_iD_i)$

LS=Left Shift
-shift *one* position
if i=1,2,9 or 16
-shift *two* positions
otherwise

DES Key Generation



DES Permuted Choice 1 and 2
PC1 and PC2

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

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

Avalanche Effect

- The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly
- A change in one bit of the plain text should produce a change in many bits of the cipher text

Strength of DES

- **Use of 56-bit keys**
  - 2^{56} (possible keys) = 7.2×10^{16} keys
  - Brute Force attack impossible
  - May take years to break the code
- **The Nature of DES Algorithm**
  - related to S boxes
  - Design of S box is not made public
  - Till now nobody is able to discover the weakness of S boxes

Double DES

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

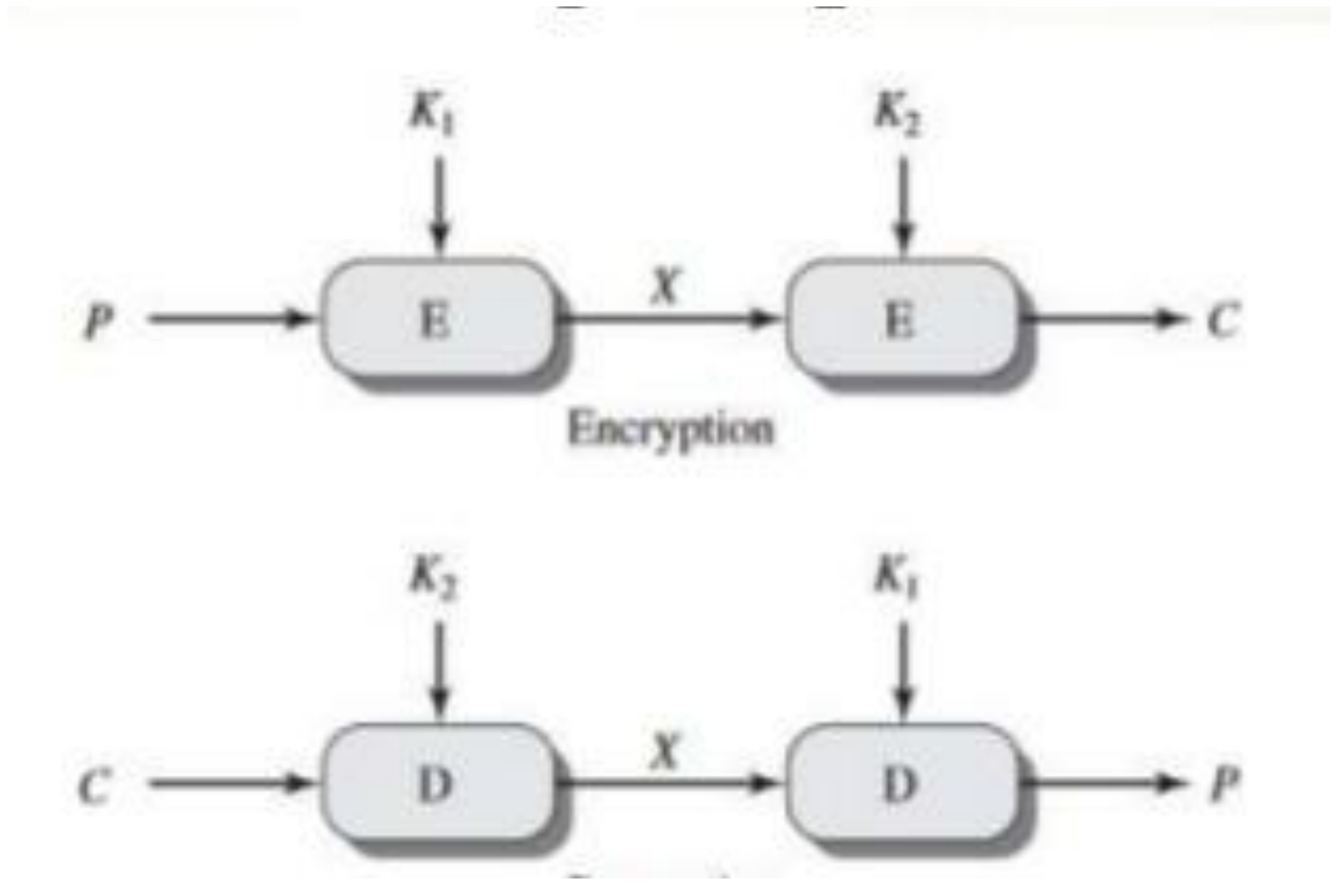
$$\boxed{C = E[K2[E_{K1},P]]}$$

  - C = ciphertext
  - P = Plaintext

- Effective key length of 128 bits

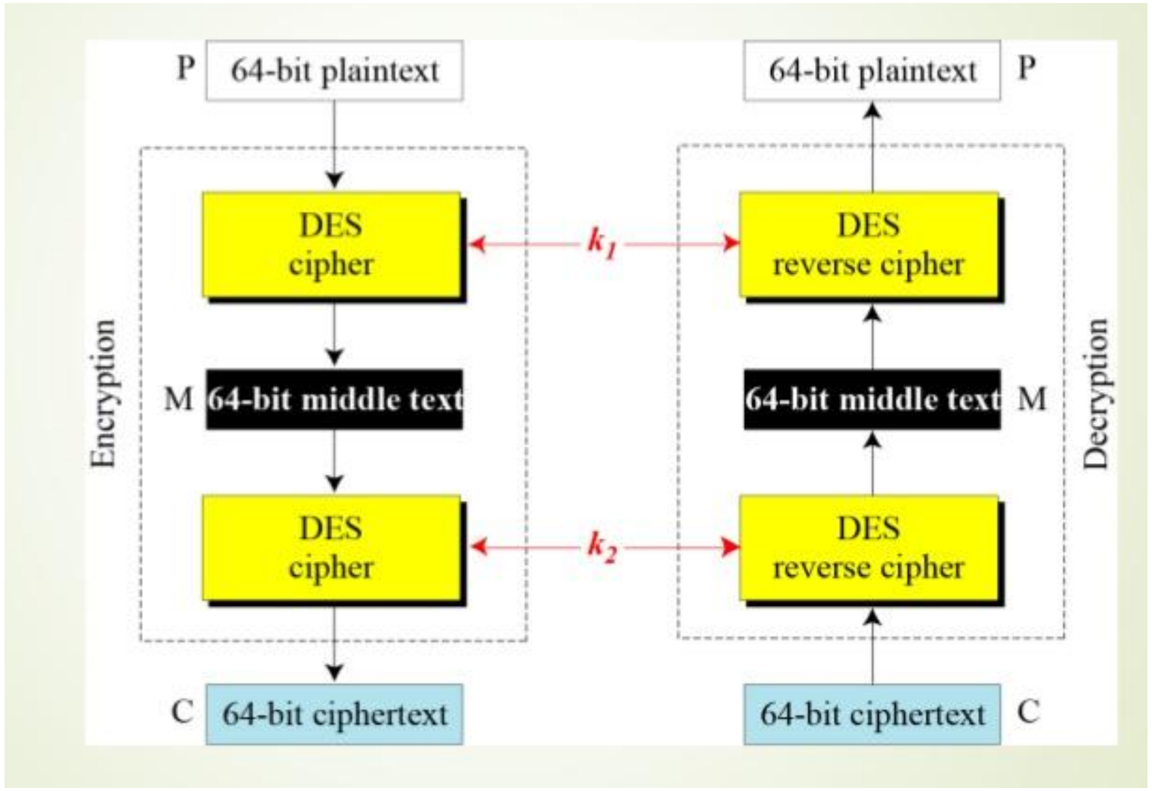
- Decryption requires that keys be applied in reverse order:
- P=D[K1, D[K2,C]]

Double DES



Double DES

- It is vulnerable to meet in the middle attack



Double DES

- The middle text , the text created by the first encryption or the first decryption
- $M = E_{K1}$ (P) $M = D_{K2}$ (C)
- Encrypt using all possible values of K1 and store for M
- Decrypt for all possible of K2 and store all values for M
- Compare for value of M in both tables

Triple DES

- Use three keys and three executions of the DES algorithm
- The 1st ,3rd stage use K_1 and second stage use K_2
- Effective key length of 168 bits: practically somewhat unwidely
- Triple DES method that uses only two keys such as
- (encrypt-decrypt-encrypt)

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

- For decryption uses (Decrypt-Encrypt-Decrypt)

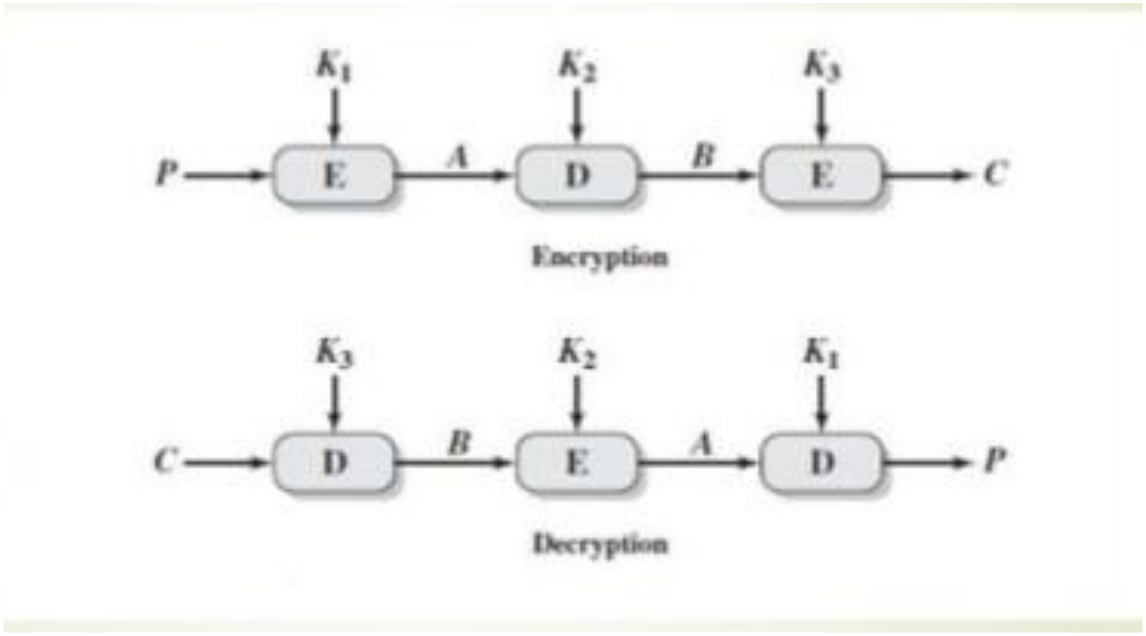
$$\boxed{P = D_{K1}[E_{K2}[D_{K1}[C]]]}$$

C = ciphertext

- P = Plaintext
- EK[X] = encryption of X using key K
- DK[Y] = decryption of Y using key K

Triple DES

- The three stages can use three different keys



Modes of operation

- A **mode of operation** is an algorithm that uses a block cipher to encrypt messages of arbitrary length with confidentiality or authenticity.
- A block cipher by itself is only suitable for the secure cryptographic transformation (encryption or decryption) of one fixed-length group of bits called a block
- A block cipher such as DES or AES only encrypts one block of a fixed size ,but often the data we want to encrypt (the plaintext) is larger than 56-bit or 128-bit.
- This problem is solved by using a mode of operation which describes how to repeatedly apply DES or AES (both encryption and decryption) on plaintexts (and ciphertexts) with size larger than one block

Block Cipher Modes of Operation

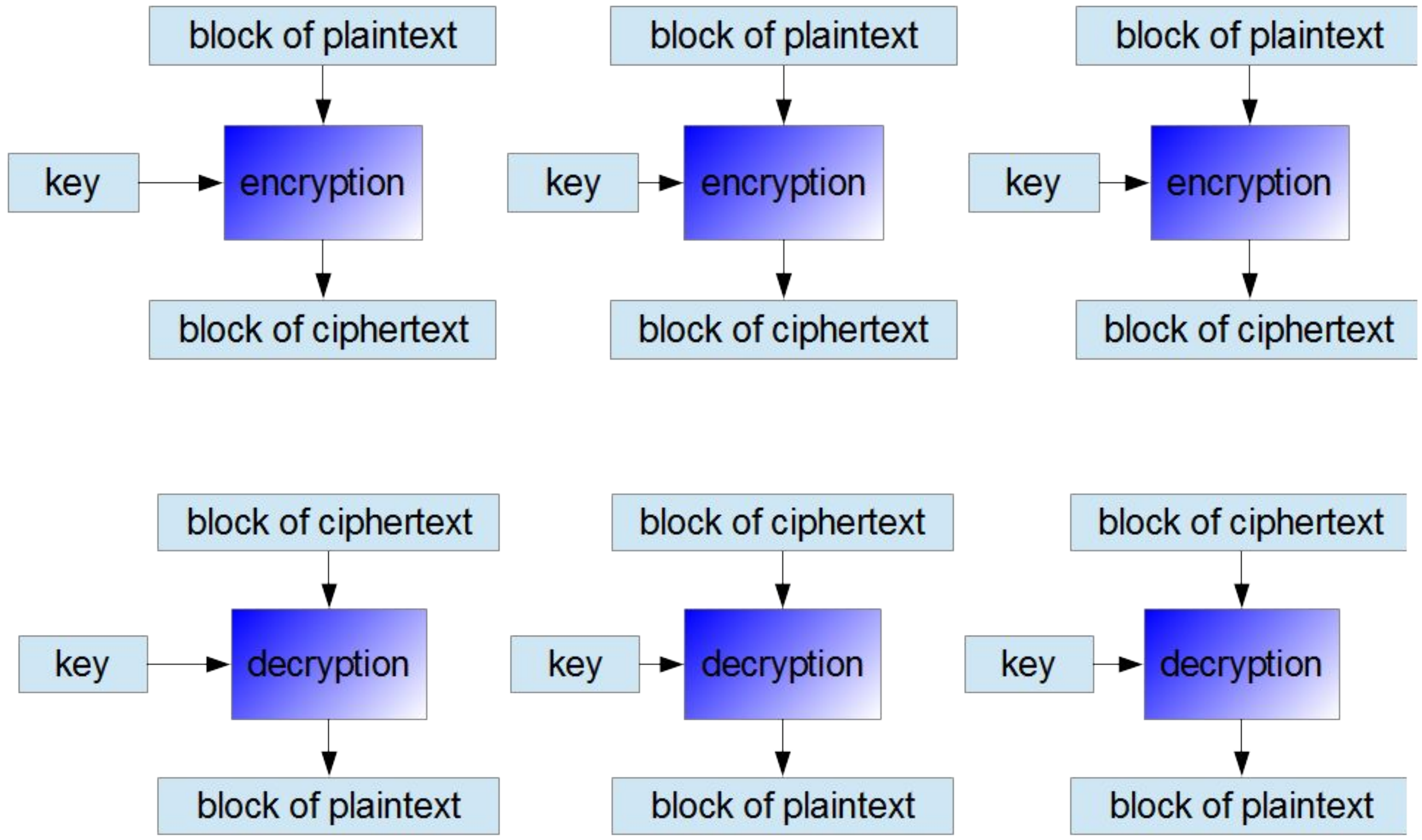
In order to apply DES in a variety of applications various modes of operation are defined

1. Electronic codebook mode
2. Cipher Block Chaining mode
3. Cipher feedback mode
4. Output feedback mode
5. Counter mode

Electronic Codebook Mode (ECB)

- Message is broken into independent blocks which are encrypted
- In ECB mode plaintext is handled 64 bits at a time and each block of plaintext is encrypted using the same key
- **The term "code book" is used because for a given key there is a unique cipher text for every 64-bit block of plaintext.**
- For the message more than 64bits, producer simply breaks the message into 64 bit blocks, padding the last block if necessary.
- Decryption is done one block at a time, always using the same key.
- ECB method is ideal for short amount of data
- The most important characteristics of ECB is that, if same 64 bits appears more than once in message, it produces always the same cipher text
- Each block is encoded independently of the other blocks.

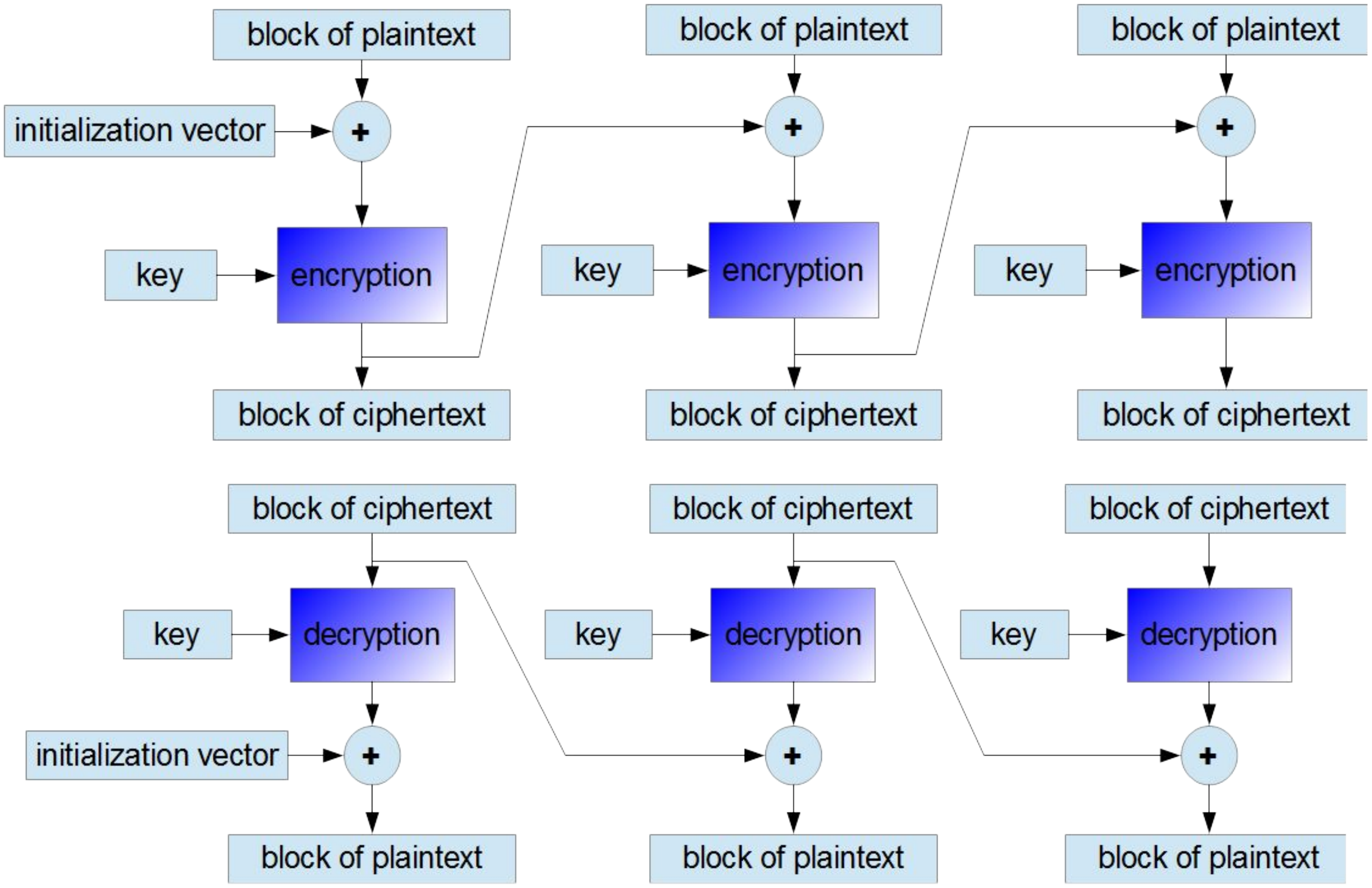
Electronic Codebook Mode (ECB)



Remarks on ECB

- **Strength**:
  - it’s simple.
- **Weakness**:
  - Repetitive information contained in the plaintext may show in the cipher text, if aligned with blocks.
  - If the same message is encrypted (with the same key) and sent twice, their cipher texts are the same.
- Typical application: secure transmission of short pieces of information (e.g. a temporary encryption key)

Cipher Block Chaining (CBC)



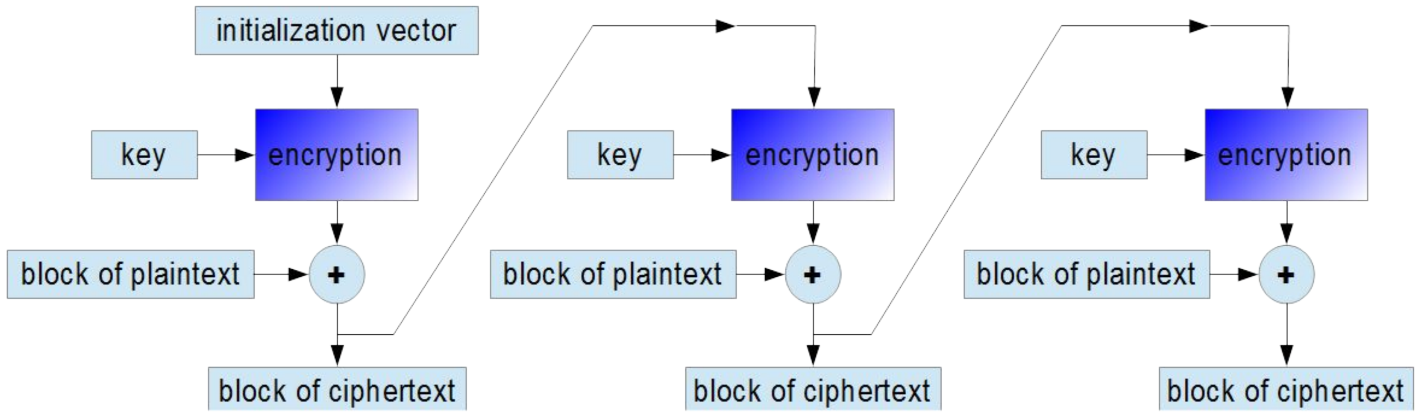
Advantages and Limitations of CBC

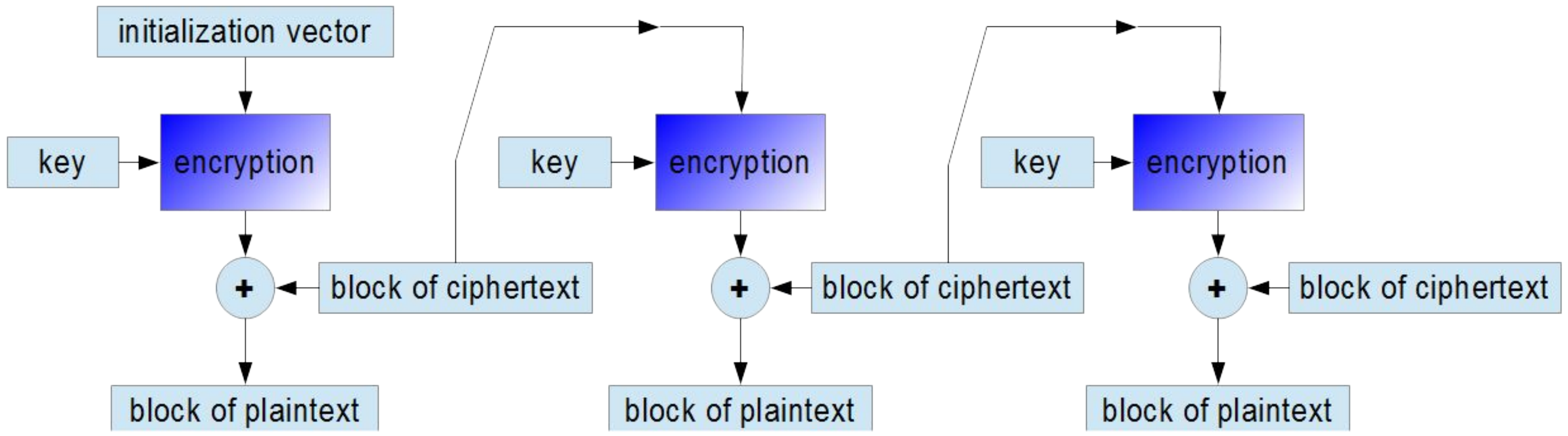
- The same key is used for all blocks.
- A cipher text block depends on **all** blocks before it.
- So, repeated plaintext blocks are encrypted differently.
- Any change to a block affects all following cipher text blocks
- Need **Initialization Vector** (IV)
  - which must be known to sender & receiver
  - hence IV must either be a fixed value
  - or must be sent encrypted in ECB mode before rest of message

Cipher Feed-back (CFB)

- It is possible to convert DES into a stream cipher using either the cipher feedback (CFB) or the output feedback mode (OFB)
- Message is treated as a stream of bits
- **The stream cipher eliminates the need to pad a message to be an integral number of blocks.**
- Cipher text is the same length as the plain text.
- Plaintext is divided into segments of **s** bits.
- The i/p to the encryption function is a 64 bit shift register that is initially set to some initialization vector(IV).
- The left most bit (MSB)s bits of the o/p encryption function are X-ORed with the first segment of plaintext P_1 to produce C_1.
- In addition, the contents of shift register are shifted left by s bits and C_1 is placed in the right most s bits of the shift register.

Cipher Feedback block

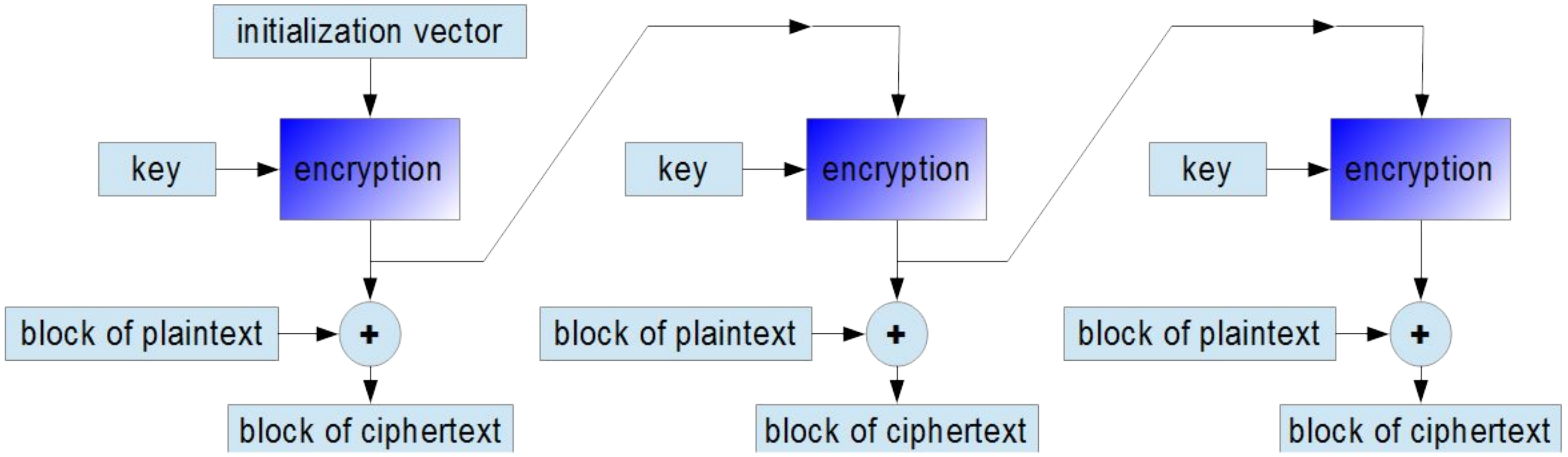


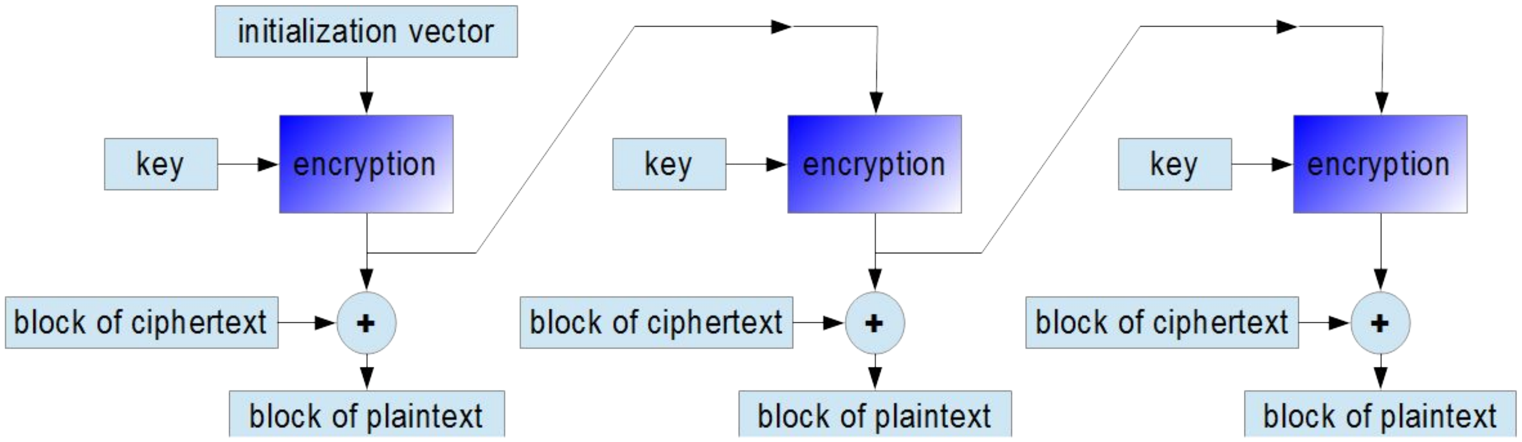


Output Feedback (OFB) Mode

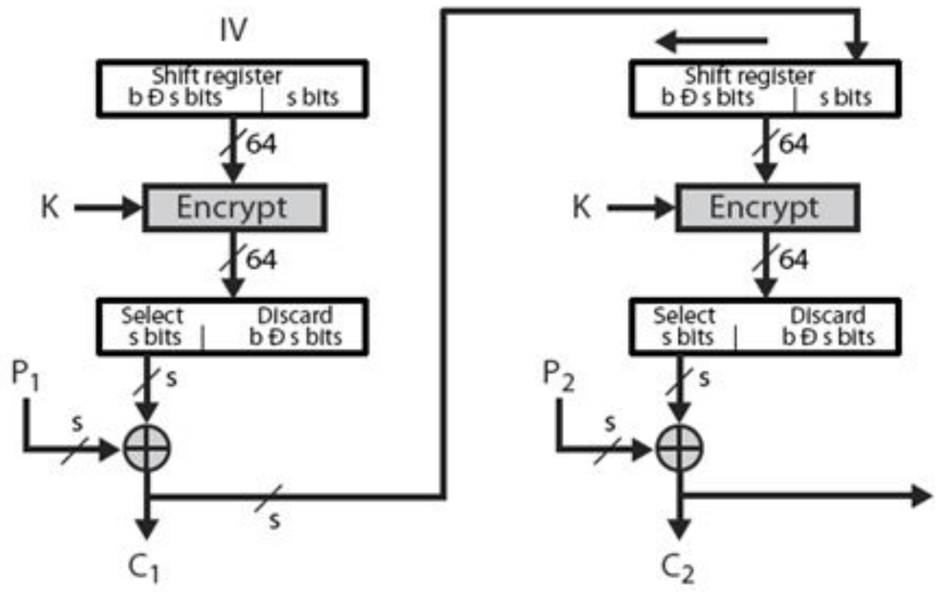
- Very similar to Cipher Feedback in structure
- But K_{i-1} rather than C_{i-1} is fed back to the next stage
- As in CFB , the input to the block cipher is a shift register x
- Initially x1 = IV
- For x >1, x_i = shift –left-s-bits(x_{i-1}) $\oplus$ K_{i-1}

Output Feedback mode

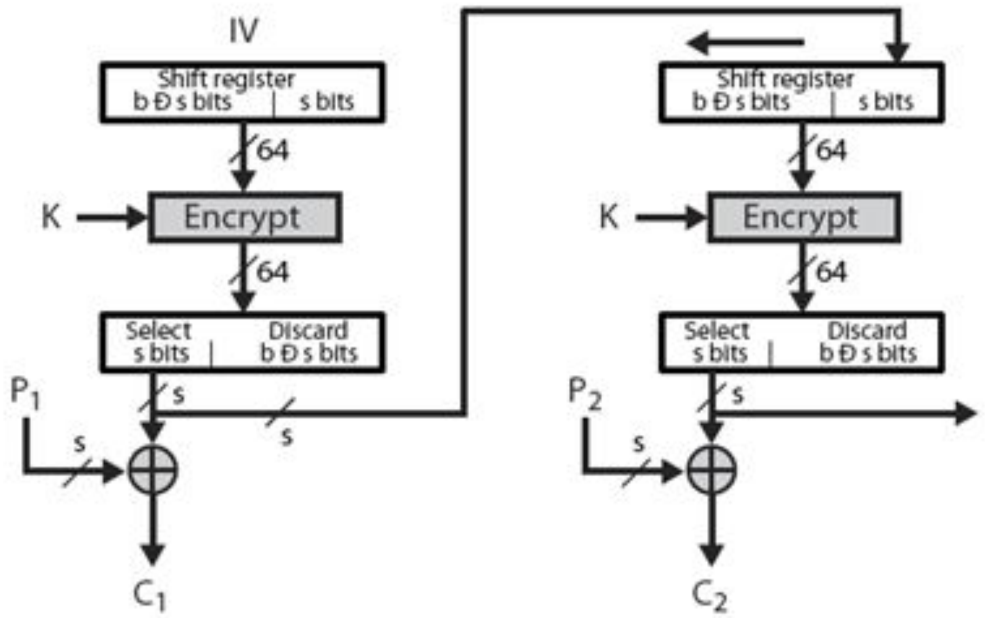




Cipher Feedback



Output Feedback



OFB Con…

- Adv:
   - Bit errors in transmission do not propagate
- Drawbacks:
   - It is more vulnerable to message stream modification attack than CFB.

Counter Mode (CTR)

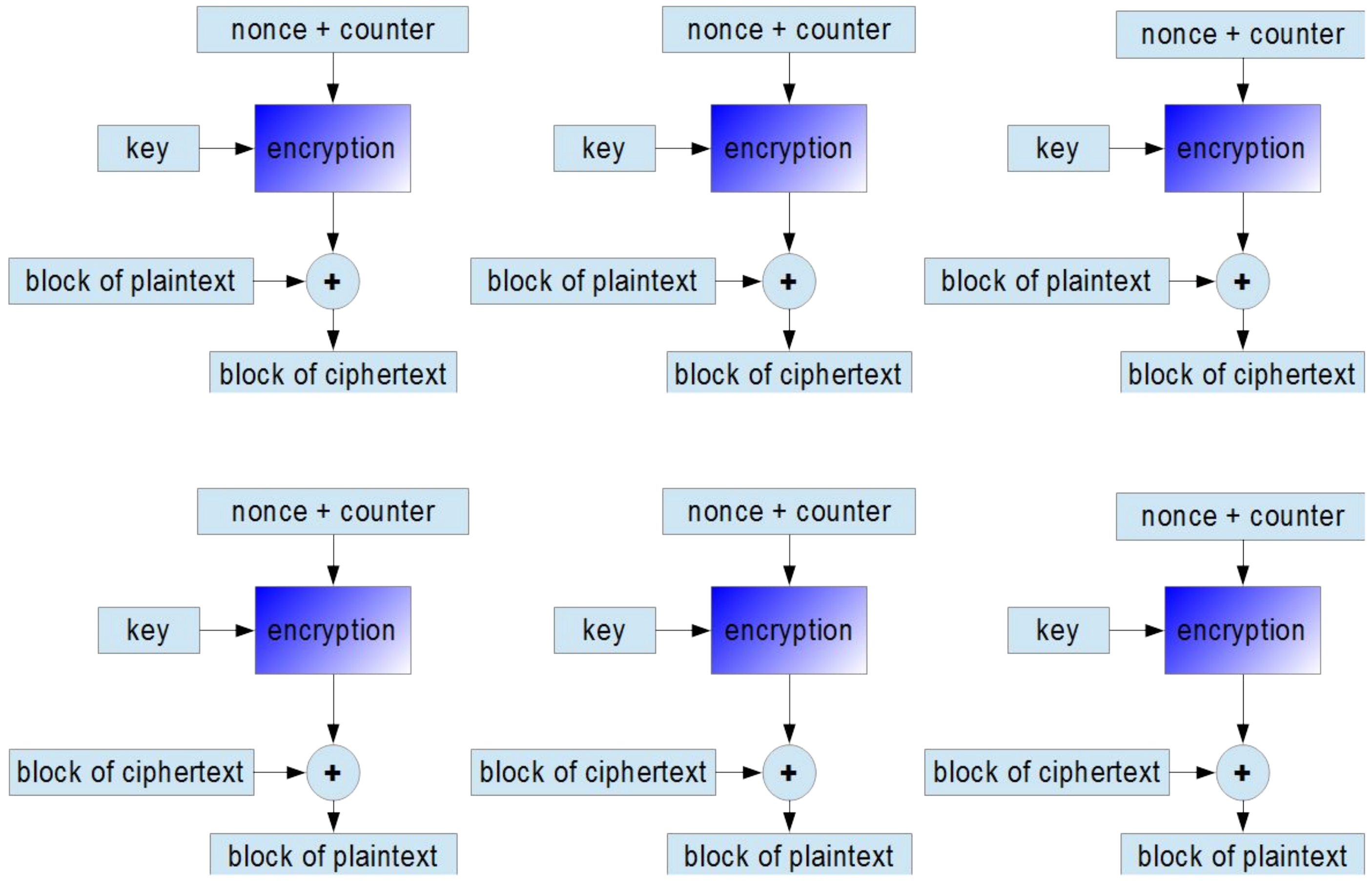
- A Counter equal to the plaintext block size is used.
- Only one requirement is that the counter value must be different for each plaintext block that is encrypted
- A counter T is initialized to some value and then incremented by 1 for each subsequent plaintext block.
- Encryption:

$T_1 = IV$

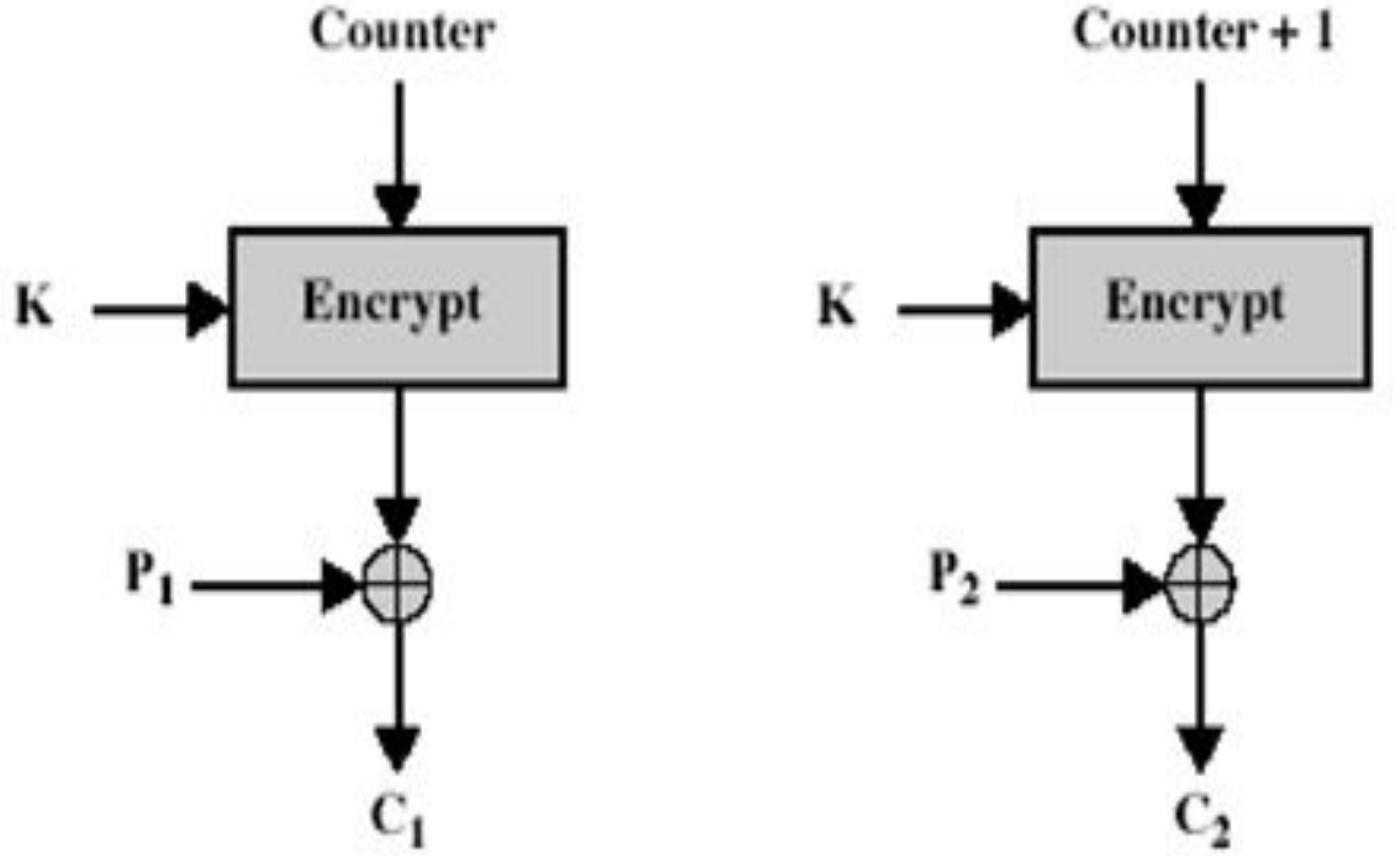
$T_i = T_{i-1} + 1$

$C_i = P_i \text{ XOR } E_K(T_i)$

Counter Mode



CTR Cont..



Remark on CTR

- Strengths:
- Encryption can be done in parallel on multiple blocks of plaintext or cipher text. So maximum throughput.
  - Needs only the encryption algorithm (so do CFB and OFB)
  - Fast encryption/decryption; blocks can be processed (encrypted or decrypted) in parallel; good for high speed links (ATM Networks)
  - Random access to encrypted data blocks
- As in OFB, IV should not be reused.

Advantages of CTR

- **Hardware efficiency:**
  - Unlike the chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext.
  - For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block.
- **Software efficiency:**
  - because of the opportunities for parallel execution in CTR mode, processors that support parallel features (such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions) can be effectively utilized.

Advantages of CTR

- **Random access:**
  - The *i*th block of plaintext or ciphertext can be processed in random-access fashion
- **Provable security:**
  - It can be shown that CTR is at least as secure as the other modes discussed in this section.
- **Simplicity:**
  - Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm