Chap 2. Cryptography

Block Ciphers & Public Key Cryptography

Course Objectives- And explore the working principles and utilities of various cryptographic algorithms including secret key cryptography, hashes and message digests, and public key algorithms

Course Outcomes-Understand, compare and apply different encryption and decryption techniques to solve problems related to confidentiality and authentication

Topics

- Block cipher modes of operation
- Data encryption standard
- Advanced encryption standard
- RC5 algorithm
- Public key cryptography
- RSA algorithm
- Hashing techniques
- SHA 256, SHA 512, HMAC and CMAC
- Digital signature schemes RSA,DSS,
- Remote user authentication Protocols
- Kerberos
- Digital certificate X.509,PKI

Attributes of Strong Encryption:-Confusion and diffusion

- Confusion –
- 1) Change key values each round
- 2)Performed through substitution
- 3)Complicates plaintext/key relationship
- Diffusion –
- 1)Change location of plaintext in ciphertext
- 2) Done through transposition

BASIS FOR COMPARISON	CONFUSION	DIFFUSION
Basic	Utilized to generate vague cipher texts.	Utilized to generate obscure, plain texts.
Seeks to	Make a relation between statistics of the ciphertext and the value of the encryption key as complicated as possible.	The statistical relationship between the plaintext and ciphertext is made as complicated as possible.
Achieved through	Substitution algorithm	Transposition algorithm
Used by	Block cipher only.	Stream cipher and block cipher
Result in	Increased vagueness	Increased redundancy

Block cipher modes of operation

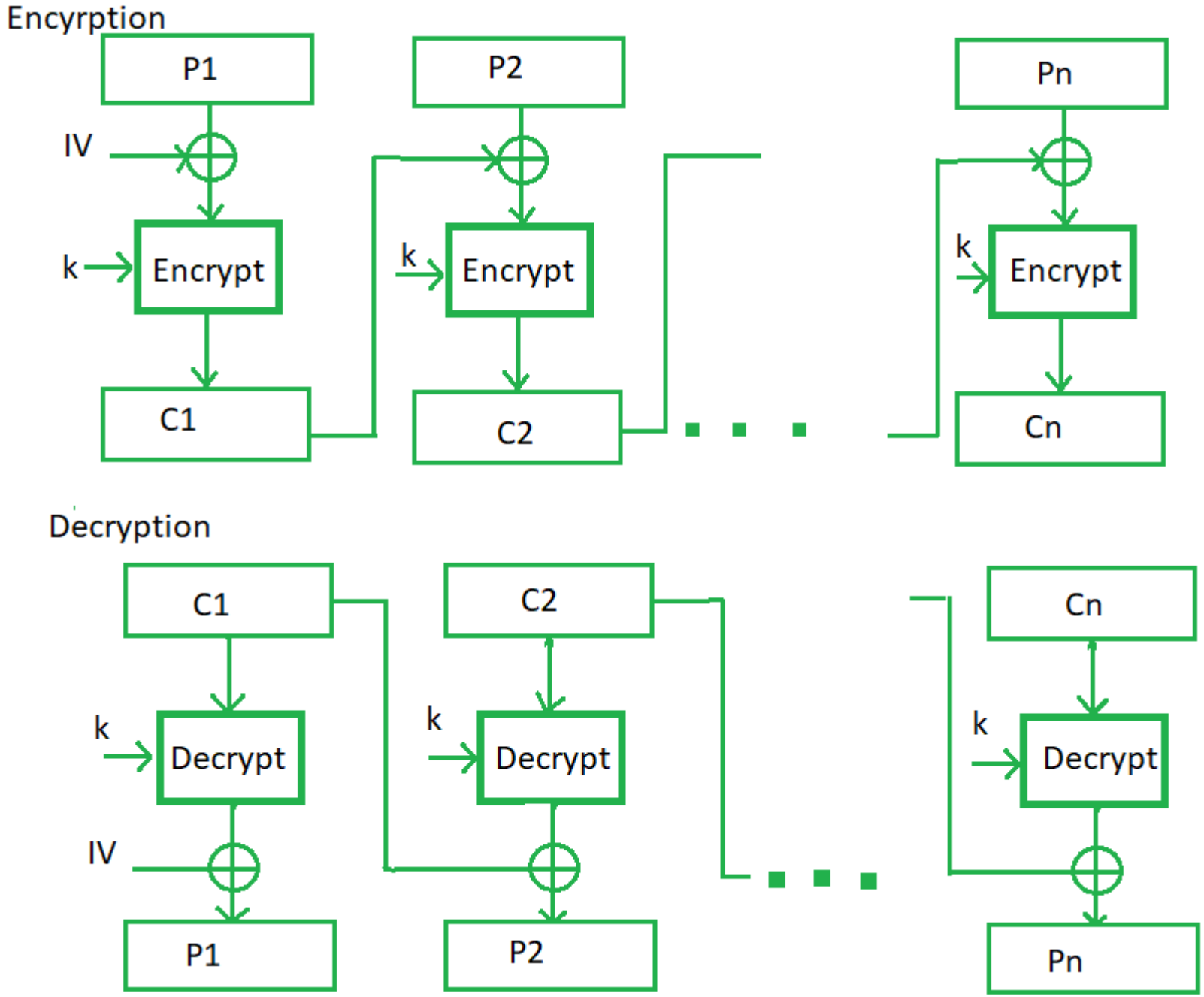
- Encryption algorithms are divided into two categories based on the input type, as a block cipher and stream cipher.
- **Block cipher** is an encryption algorithm that takes a fixed size of input say *b* bits and produces a ciphertext of *b* bits again.
- If the input is larger than *b* bits it can be divided further.
- For different applications and uses, there are several modes of operations for a block cipher.

Electronic Code Book (ECB)

- Simplest mode of operation of block cipher
- Processes a series of sequentially listed msgs blocks but 64 bit block at time. Each block is separately encrypted
- Same key is used for encryption and decryption

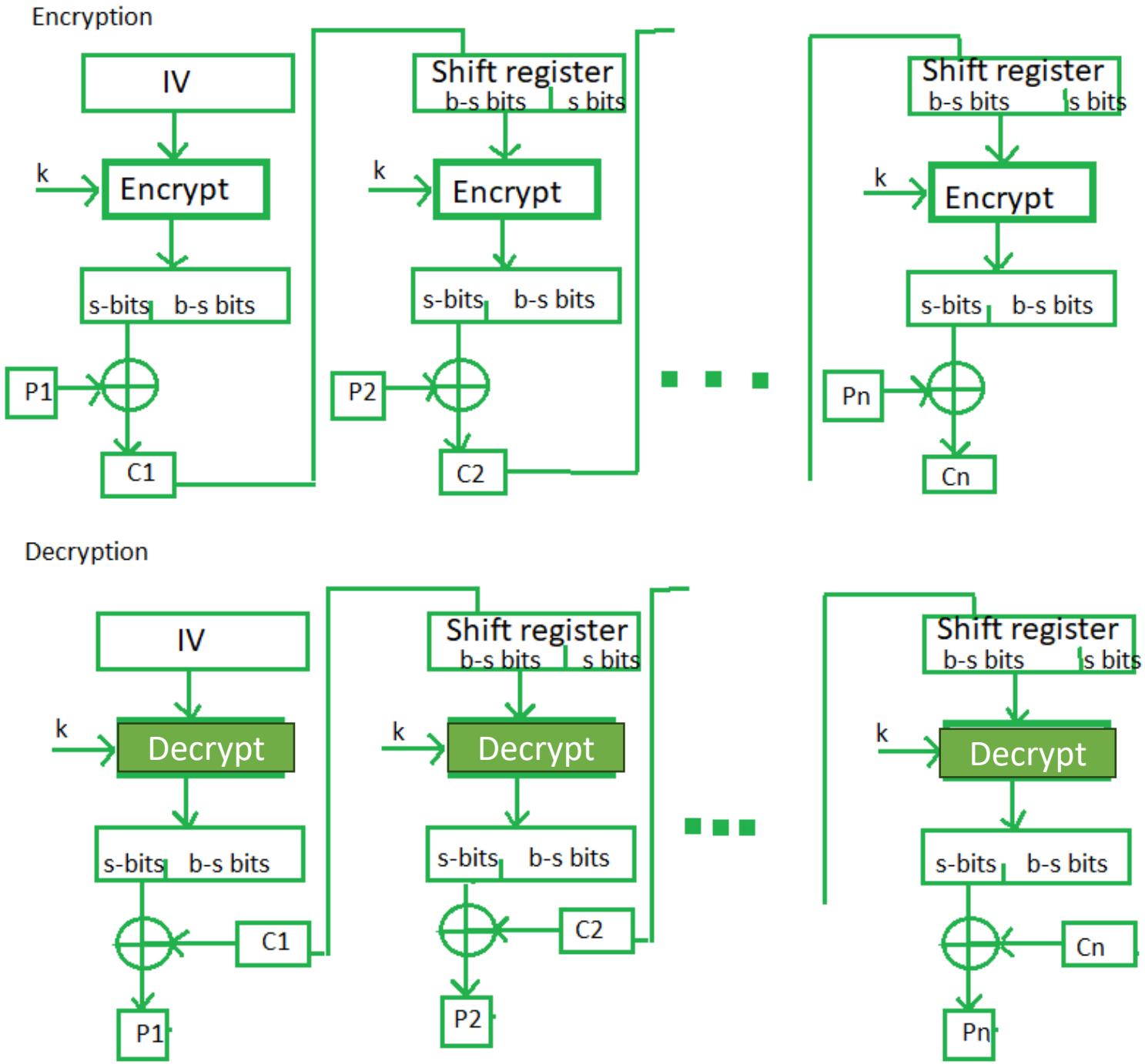
Cipher block chaining

- Cipher block chaining or CBC is an advancement made on ECB since ECB compromises some security requirements.
- Initialization vector is used
- In CBC, the previous cipher block is given as input to the next encryption algorithm after XOR with the original plaintext block.
- In a nutshell here, a cipher block is produced by encrypting an XOR output of the previous cipher block and present plaintext block.


Encyrption
P1
P2
Pn
IV
k
Encrypt
k
Encrypt
k
Encrypt
C1
C2
Cn
Decryption
C1
C2
Cn
k
Decrypt
k
Decrypt
k
Decrypt
IV
P1
P2
Pn

Cipher Feedback Mode

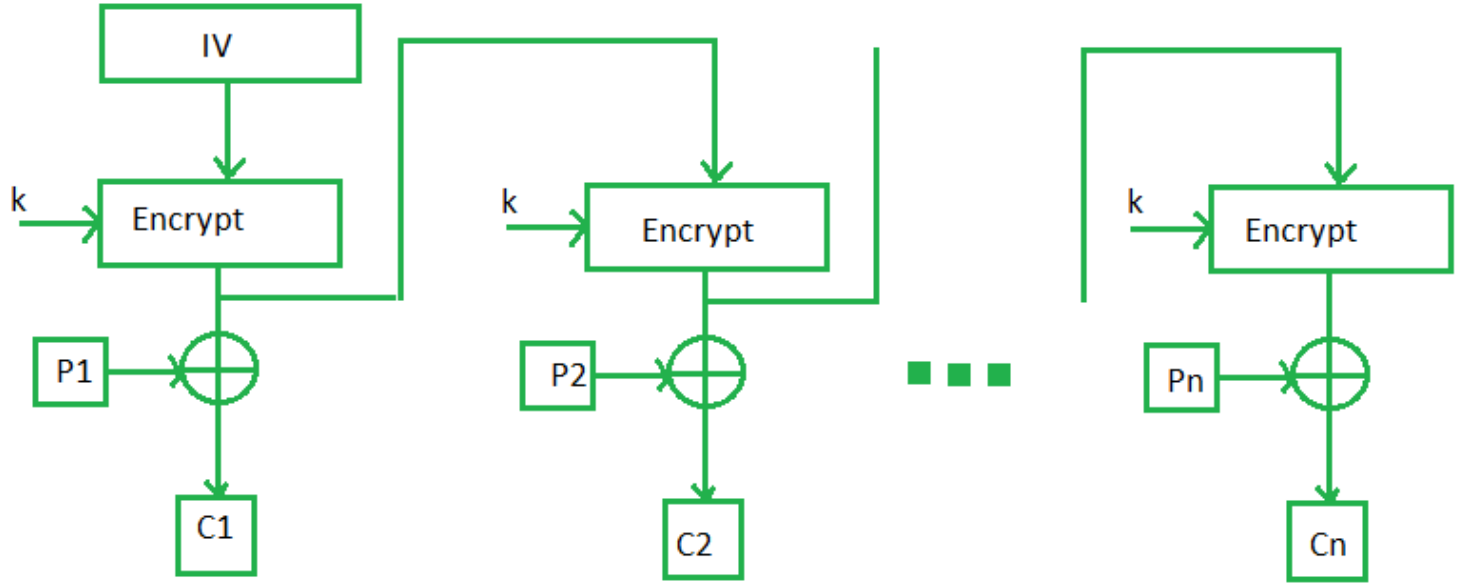
- Data is encrypted in the form of 8 bits/unit
- In this mode the cipher is given as feedback to the next block of encryption with some new specifications: first, an initial vector IV is used for first encryption and output bits are divided as a set of *s* and *b-s* bits.
- The left-hand side *s* bits are selected along with plaintext bits to which an XOR operation is applied.
- The result is given as input to a shift register having b-s bits to lhs,s bits to rhs and the process continues.
- The encryption and decryption process for the same is shown below, both of them use encryption algorithms.


Encryption
IV
k
Encrypt
s-bits
b-s bits
P1
C1
Shift register
b-s bits
s bits
k
Encrypt
s-bits
b-s bits
P2
C2
Shift register
b-s bits
s bits
k
Encrypt
s-bits
b-s bits
Pn
Cn
Decryption
IV
k
Decrypt
s-bits
b-s bits
C1
P1
Shift register
b-s bits
s bits
k
Decrypt
s-bits
b-s bits
C2
P2
Shift register
b-s bits
s bits
k
Decrypt
s-bits
b-s bits
Cn
Pn

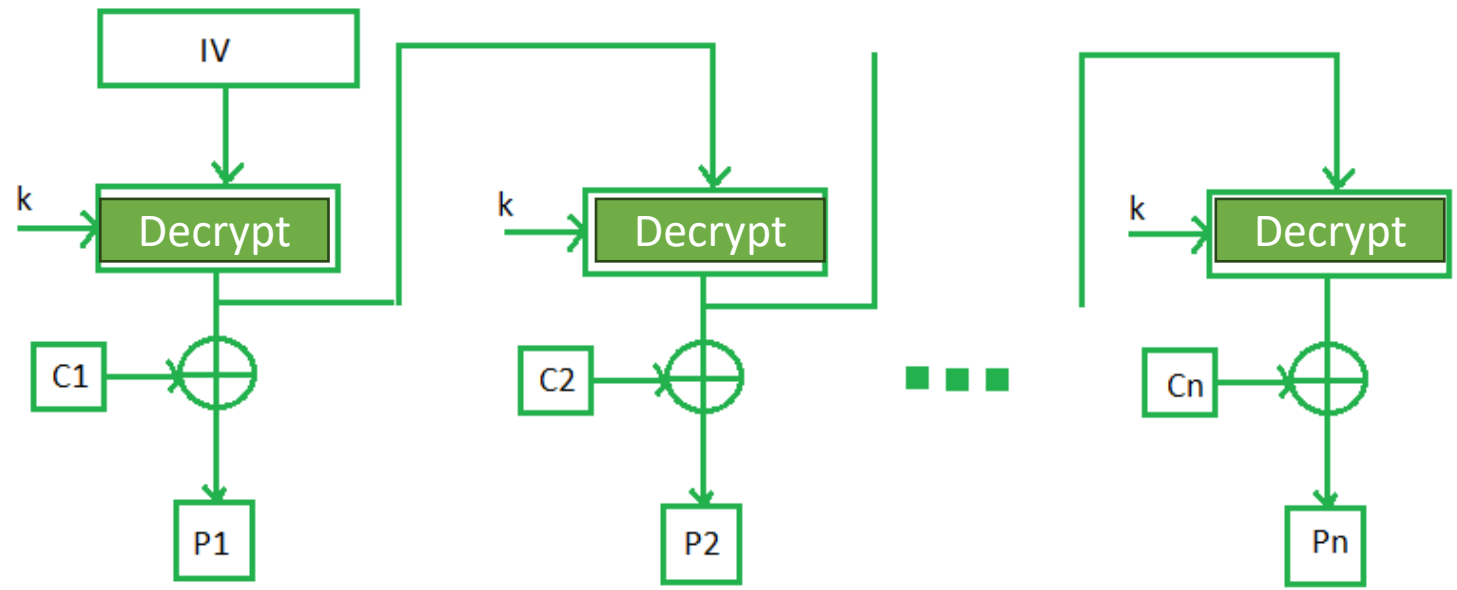
Output Feedback Mode

- The output feedback mode follows nearly the same process as the Cipher Feedback mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output.
- In this output feedback mode, all bits of the block are sent instead of sending selected *s* bits.
- The Output Feedback mode of block cipher holds great resistance towards bit transmission errors.
- It also decreases the dependency or relationship of the cipher on the plaintext.

Encryption

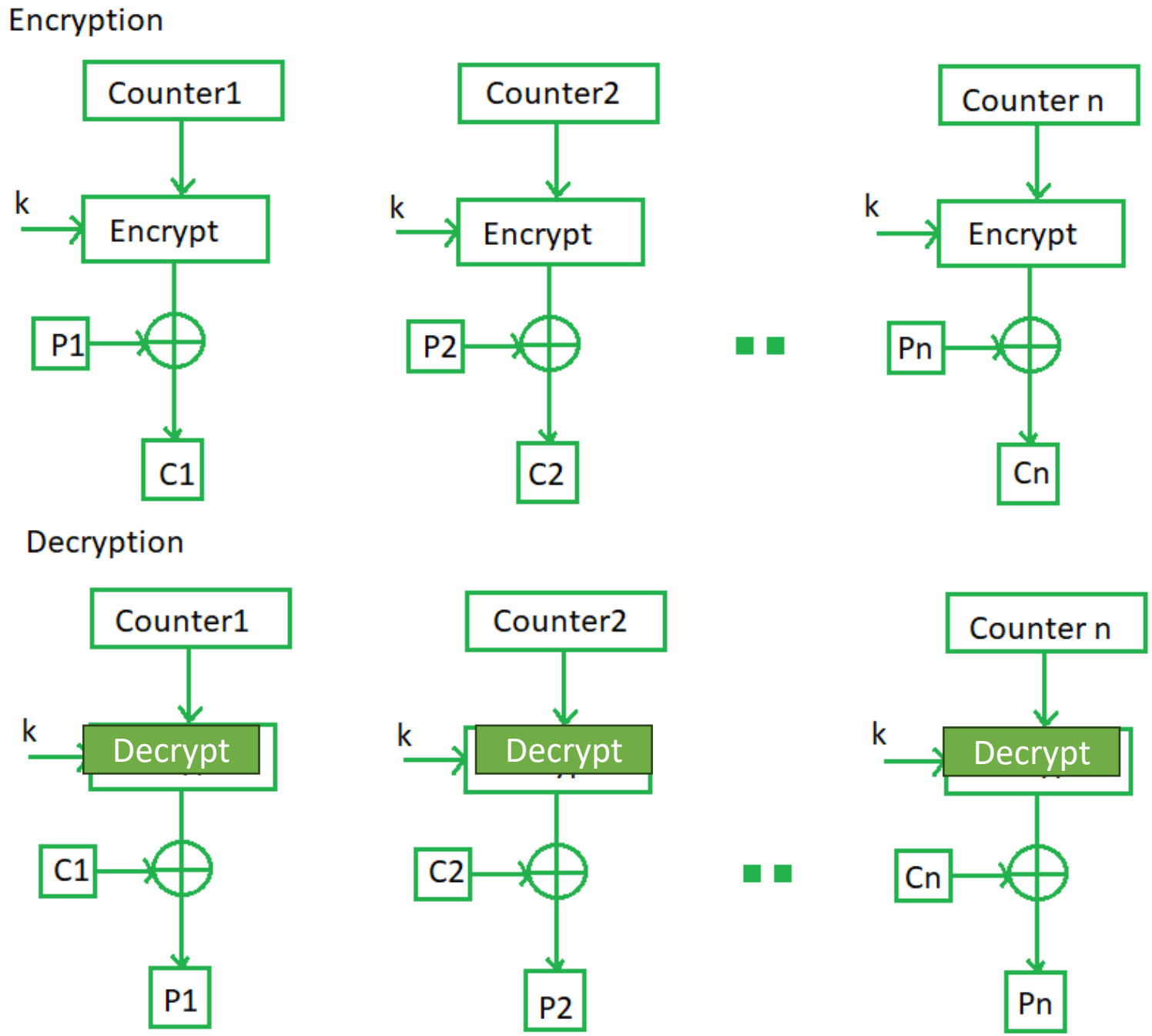


Decryption



Counter Mode

- The Counter Mode or CTR is a simple counter-based block cipher implementation.
- Every time a counter-initiated value is encrypted and given as input to XOR with plaintext which results in ciphertext block.
- The CTR mode is independent of feedback use and thus can be implemented in parallel.


Encryption
Counter1
k
Encrypt
P1
C1
Counter2
k
Encrypt
P2
C2
Counter n
k
Encrypt
Pn
Cn
Decryption
Counter1
k
Decrypt
C1
P1
Counter2
k
Decrypt
C2
P2
Counter n
k
Decrypt
Cn
Pn

Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

	ECB	CBC	CFB	OFB	CTR
Security	Low	High	High	High	medium
Parallelism	Yes	No	No	No	Yes
random access	Yes	No	No	No	Yes
Speed	Yes	No	No	No	Yes
Complexity	No	Yes	Yes	Yes	No
error propagation	No	Yes	Yes	Yes	No
Implementation cost	Low	high	High	High	Low

Table 1 Comparison of different operating modes.

Mode	Advantage	Disadvantage
Electronic CodeBook (ECB)	• Simple • fast • Support for parallel (encryption/decryption)	• Duplicate data in plaintext will be reflected in the ciphertext • The plaintext can be operated by deleting or replacing the ciphertext. • If the ciphertext packet is damaged, it will affect the plaintext. • Can't resist replay attacks • Should not be used
Cipher Block Chaining (CBC)	• Support for parallel computing (decryption) • Ability to decrypt any ciphertext packet • Duplicate data in plaintext will not be reflected in the ciphertext	• Do not Support for parallel computing (Encryption) • Wrong blocks affect all following blocks
Cipher-FeedBack (CFB)	• No padding • Support for parallel computing (decryption) • Ability to decrypt any ciphertext packet • Can be prepared for encryption and decryption first	• Do not Support for parallel computing (Encryption) • Can't resist replay attacks • Wrong blocks affect all following blocks
Output-FeedBack (OFB)	• No padding • Can prepare encryption and decryption in advance • Encryption and decryption use the same structure • Bad blocks only affect the current block	• Do not Support for parallel computing • Mallory can change some ciphertext damaged plaintext
CounYeR (CTR)	• No padding • Can prepare encryption and decryption in advance • Support for parallel computing • Encryption and decryption use the same structure • Bad blocks only affect the current block	• Mallory can change some ciphertext damaged plaintext

Data encryption standard (DES)

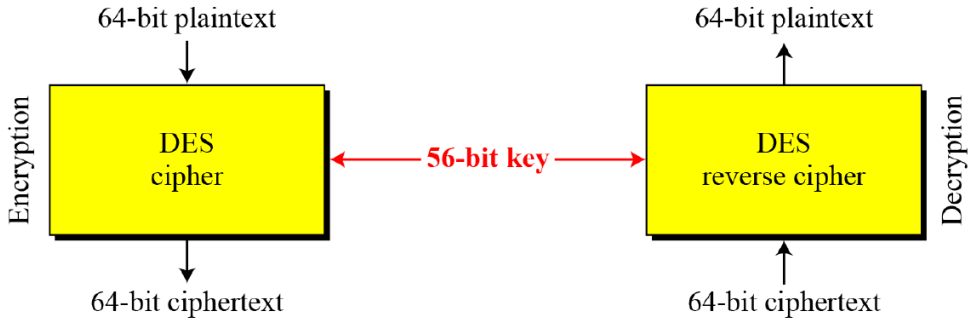
- DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences.
- DES uses a 56-bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded. Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key.

- DES is based on the two fundamental attributes of cryptography: substitution (also called confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called a round. Each round performs the steps of substitution and transposition.

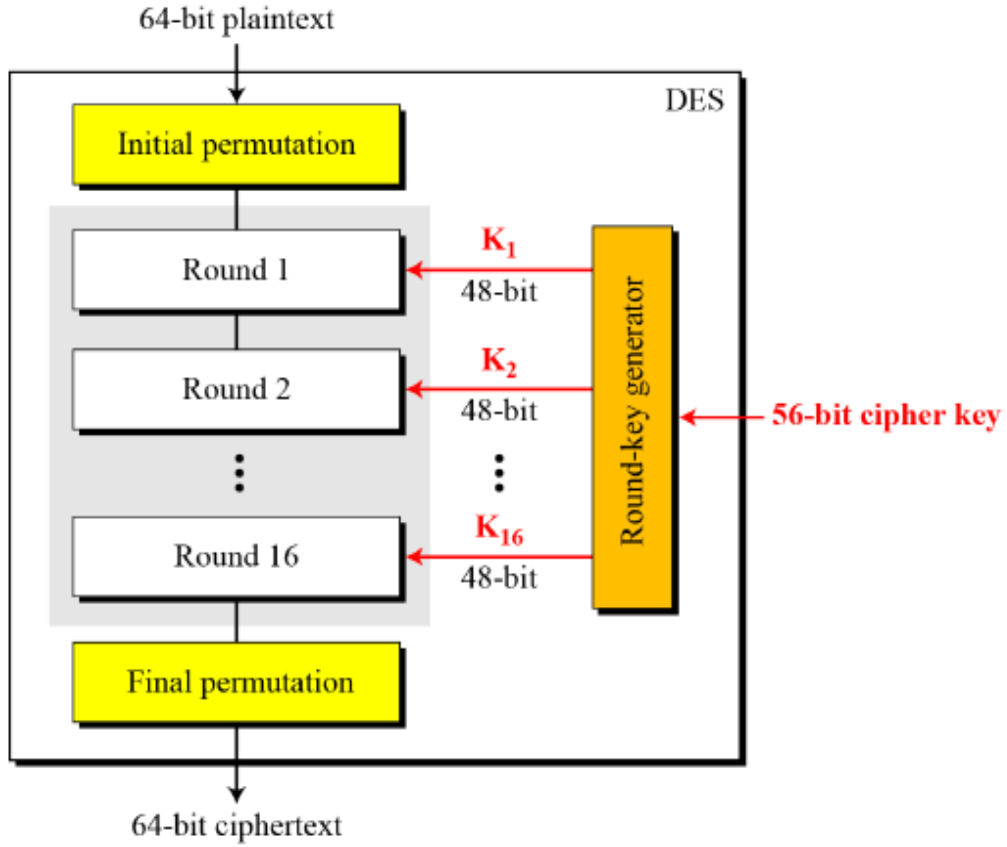
General structure of DES

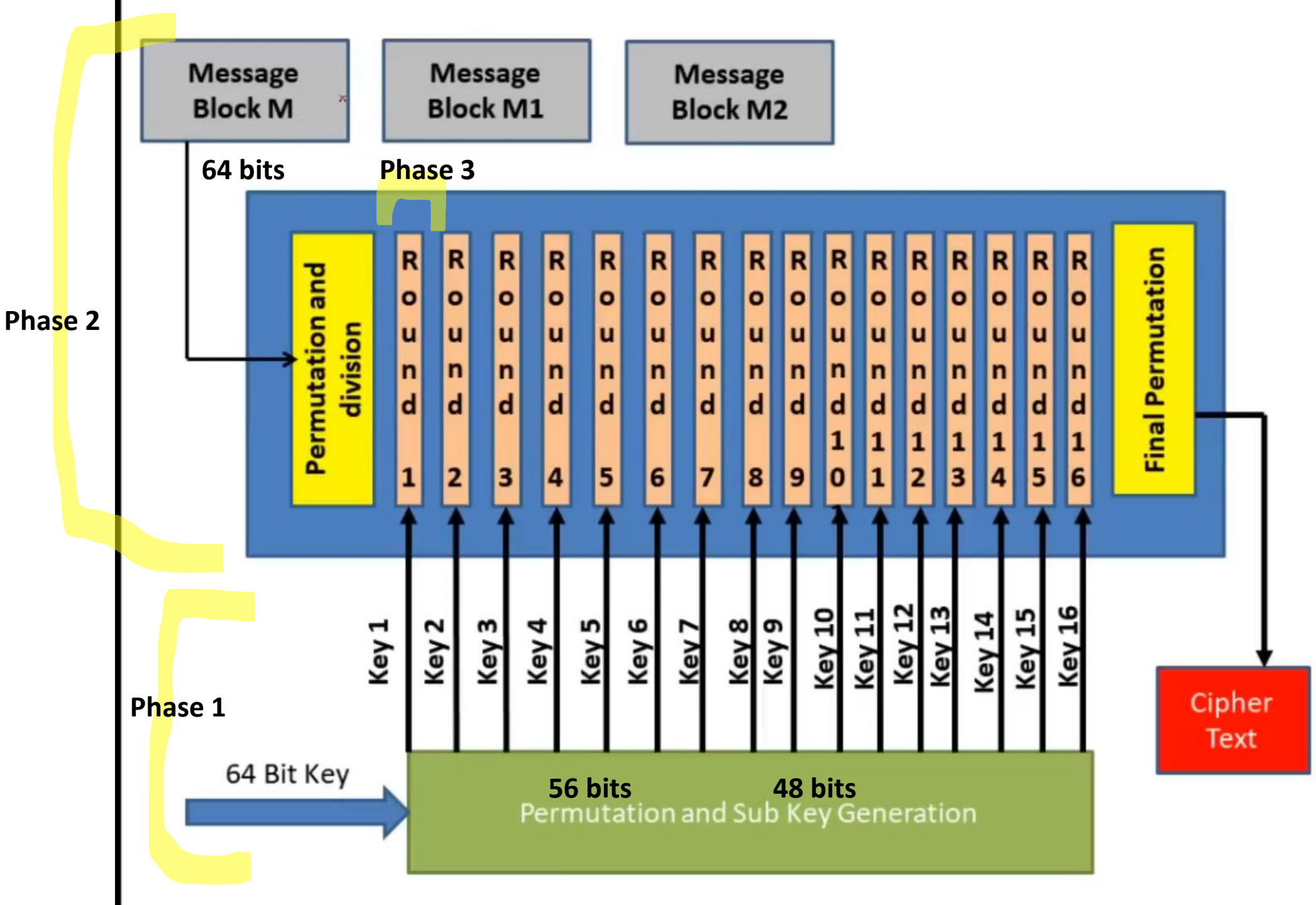
DES is a block cipher

Figure : *Encryption and decryption with DES*



The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds.




Message Block M
Message Block M1
Message Block M2
64 bits
Phase 3
Phase 2
Permutation and division
Round 1
Round 2
Round 3
Round 4
Round 5
Round 6
Round 7
Round 8
Round 9
Round 10
Round 11
Round 12
Round 13
Round 14
Round 15
Round 16
Final Permutation
Key 1
Key 2
Key 3
Key 4
Key 5
Key 6
Key 7
Key 8
Key 9
Key 10
Key 11
Key 12
Key 13
Key 14
Key 15
Key 16
Cipher Text
Phase 1
64 Bit Key
56 bits
48 bits
Permutation and Sub Key Generation

Phase 1

Key in Hexadecimal = 133457799BBCDFF1

- The 64-bit key is permuted according to the following table, **PC-1**.
- Note only 56 bits of the original key appear in the permuted key.

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

7*8=56 bit keys

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	21	4

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, C0 and D0, where each half has 28 bits.

From the permuted key K+, we get

C0 = 1111000 0110011 0010101 0101111

D0 = 0101010 1011001 1001111 0001111

From the previous slide

C0 = 11110000110011001010101011111

D0 = 01010101011001100111100011110

C_1 = 11100001100110010101010111111

D_1 = 10101010110011001111000111101

C_2 = 11000011001100101010101111111

D_2 = 01010101100110011110001111010

C_3 = 00001100110010101010111111111

D_3 = 01010110011001111000111101010

C_4 = 00110011001010101011111111000

D_4 = 01011001100111100011110101010

C_5 = 11001100101010101111111100000

D_5 = 01100110011110001111010101010

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_6 = 001100101010101111111000011
D_6 = 100110011110001111010101010101

C_7 = 110010101010111111100001100
D_7 = 011001111000111101010101010110

C_8 = 001010101011111111000110011
D_8 = 100111100011110101010101011001

C_9 = 010101010111111110000110011001
D_9 = 001111000111101010101011001100

C_{10} = 010101011111111000011001100110
D_{10} = 111100011110101010101100110011

C_{11} = 010101111111100001100110010101
D_{11} = 110001111010101010110011001111

C_{12} = 010111111110000110011001010101
D_{12} = 000111101010101011001100111111

C_{13} = 011111111000011001100101010101
D_{13} = 011110101010101100110011110000

C_{14} = 111111100001100110010101010101
D_{14} = 111010101010110011001111000101

C_{15} = 111110000110011001010101010111
D_{15} = 101010101011001100111110000111

C_{16} = 111100001100110010101010111111
D_{16} = 010101010110011001111000111111

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

C_1 = 11100001100110010101011111

D_1 = 10101010110011001111100011110

28+28=56 bits

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

48 bits

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

8*6= 48 bits

48 bits

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
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

Phase 2

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

64 bits

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

32 bits each L0 and R0

$L_n = R_{n-1}$

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

Let + denote XOR addition

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

$$f(R_{n-1}, K_n)$$

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

- To calculate f, we first expand each block R_0 from 32 bits to 48 bits.
- This is done by using a selection table that repeats some of the bits in R_0.
- We'll call the use of this selection table the function **E**.
- Thus $\mathbf{E}(R_0)$ has a 32 bit input block, and a 48 bit output block.

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

48 bits

8*6 =48 bits

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

$f(R_0, K_1)$

We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

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

K1+ E(R0) = B1 B2 B3 B4 B5 B6 B7 B8

We now calculate

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

B1= 011000

1st bit and last bit=00 →0

Consider 0th as row

2nd bit to 5th bit →1100 →12

Consider 8th column

Intersection of 0th row and 12th column

Value is 5

5→ 0101

$S_1(B_1)$ 0101

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

= 0101 1100 1000 0010 1011 0101 1001 0111

32 bits

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

Final stage of permutation $f(R_0,K_1)$

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

$$= 0101\ 1100\ 1000\ 0010\ 1011\ 0101\ 1001\ 0111$$

32 bits

$$f(R_0,K_1) = 0010\ 0011\ 0100\ 1010\ 1010\ 1001\ 1011\ 1011$$

8*4 =32 bits

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

$L_n = R_{n-1}$

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

Let + denote XOR addition

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

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

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

Finding output of Round 1

$f(R_0,K_1)$ = 0010 0011 0100 1010 1010 1001 1011 1011

L0 = 1100 1100 0000 0000 1100 1100 1111 1111
R0 = 1111 0000 1010 1010 1111 0000 1010 1010

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

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

= 1100 1100 0000 0000 1100 1100 1111 1111
+ 0010 0011 0100 1010 1010 1001 1011 1011

R1 = 1110 1111 0100 1010 0110 0101 0100 0100

L1 = 1111 0000 1010 1010 1111 0000 1010 1010

Output for round 1
32 bits each

Finally after 16 Rounds

Output of 16 Rounds

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

Cyphertext which in hexadecimal format is

85E813540F0AB405.

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

Inverse Initial Permutation

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

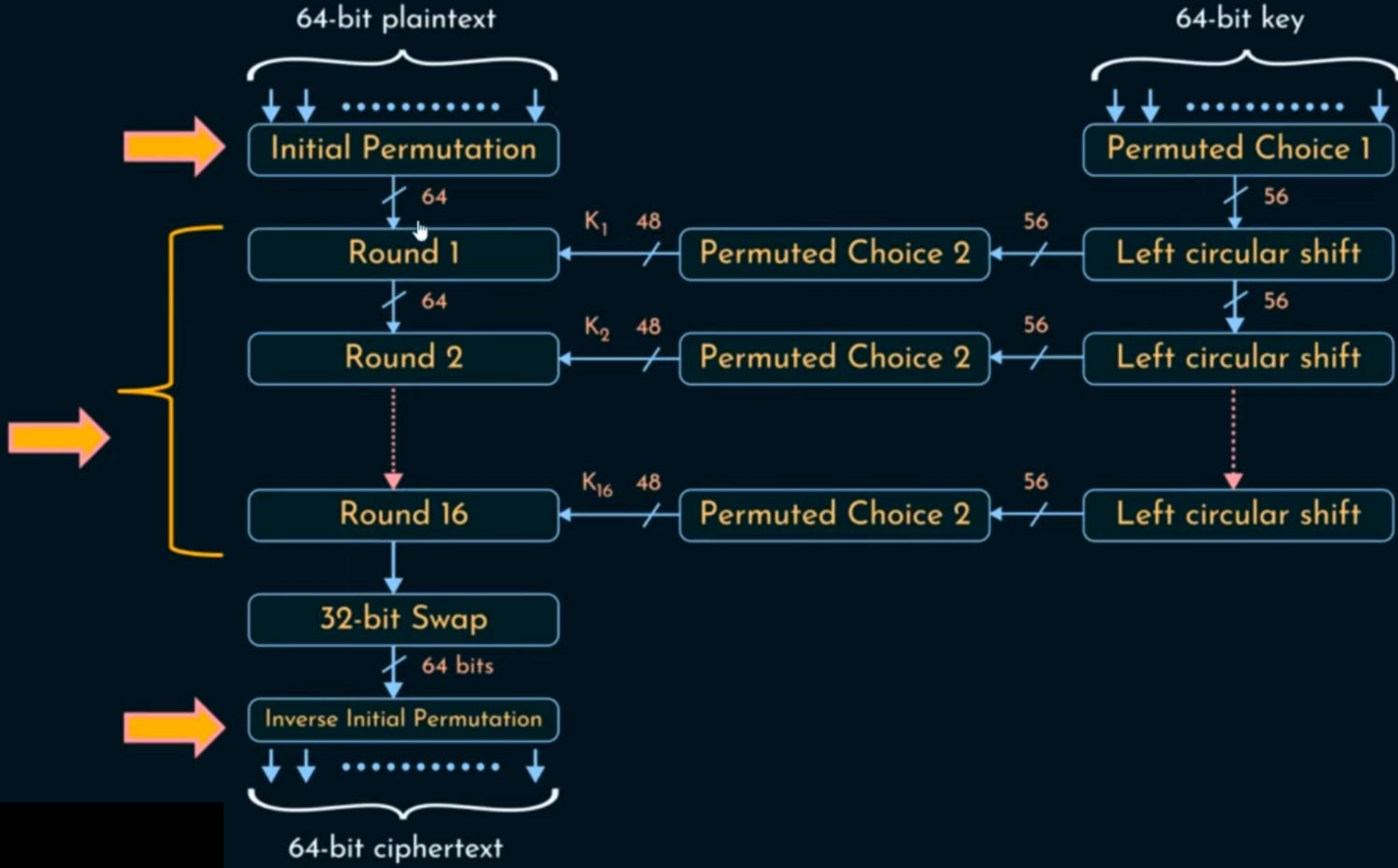
Initial Position

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

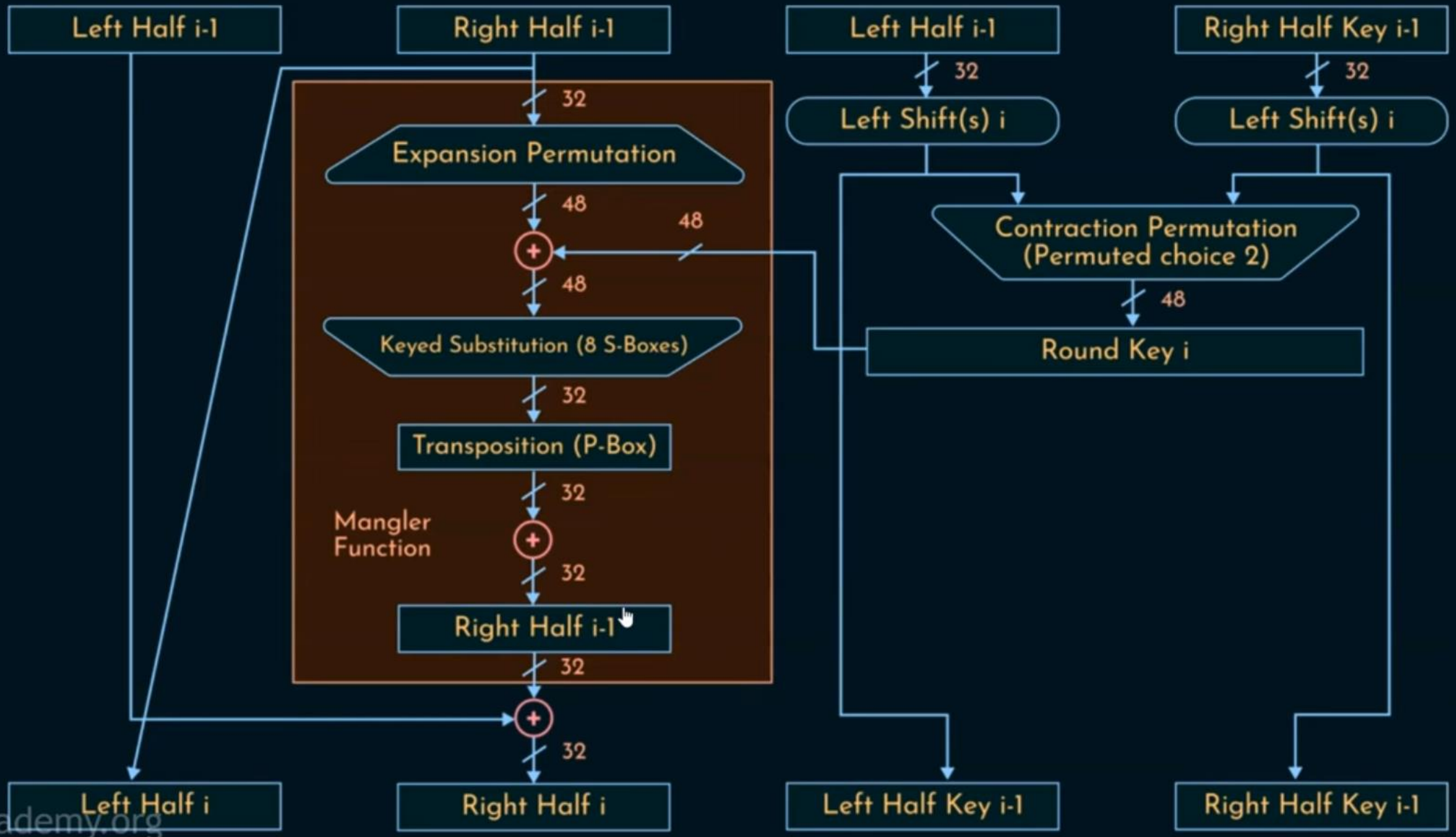
Initial permutation table

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

DES Encryption Algorithm



Single Round of DES Algorithm



Rounds and DES function

- DES uses 16 Feistel rounds Every round takes left 32 bits (L I-1) and right 32 bits (R I-1) from previous round (initial permutation box for the very first time) to produce L1 and R1 which go to the next round (to final permutation box) each round uses 2 cipher elements mixer and swapper.
- The swapper is invertible the mixer is invertible because of XOR operation.

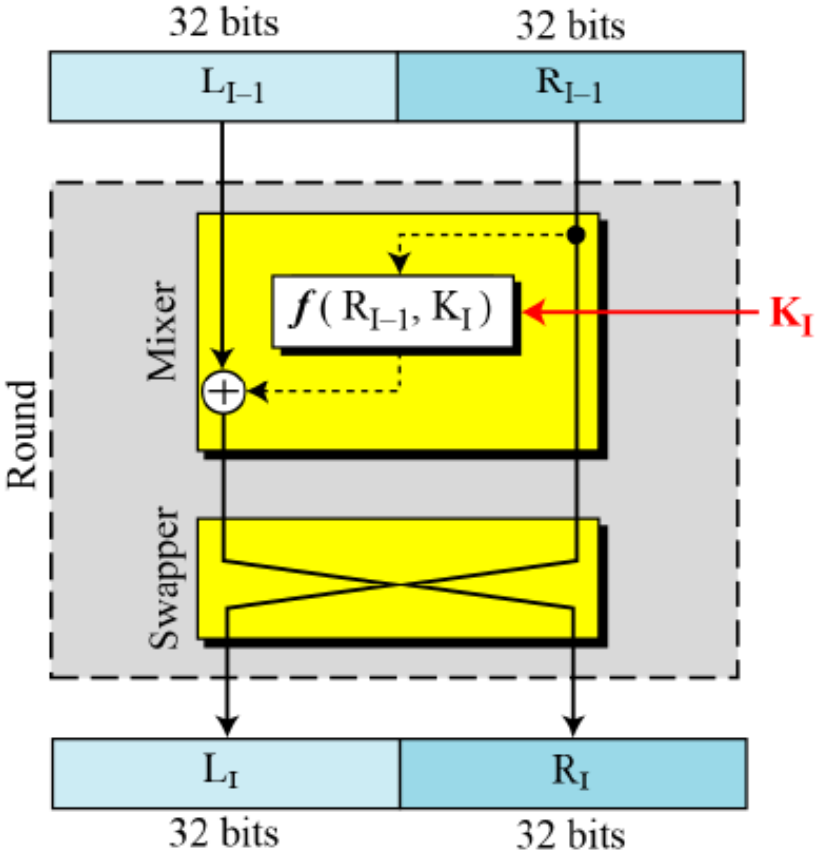


Figure 6.4
A round in DES (encryption site)

- Expansion box
- Whitener XOR
- Group of S box
- Straight P box

The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

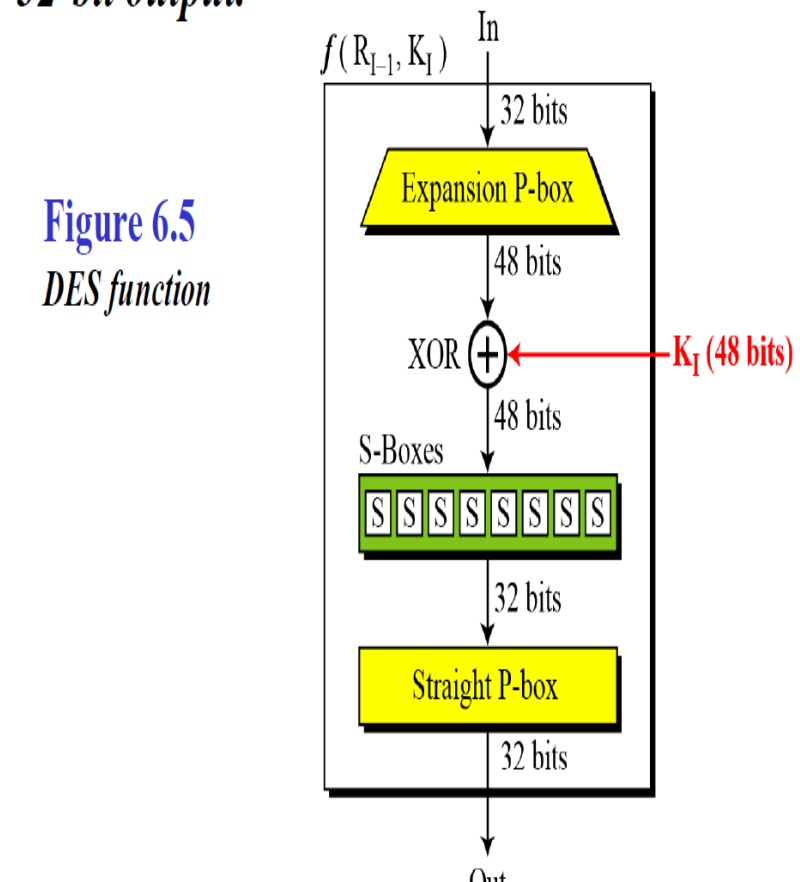
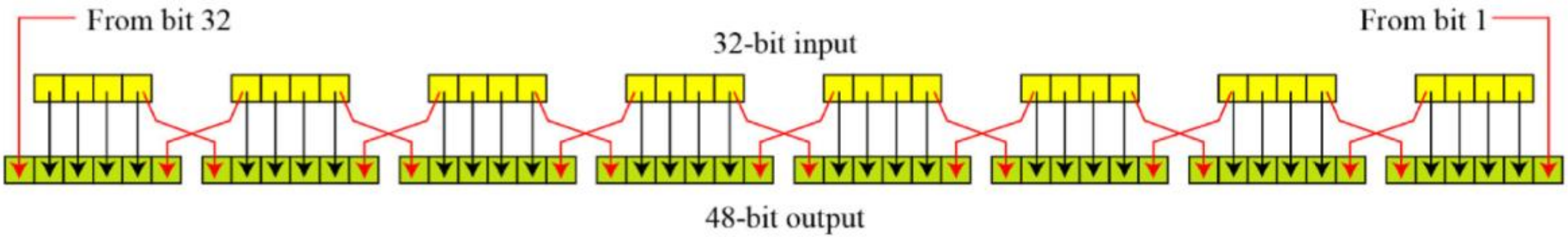


Figure 6.5
DES function

Expansion P-box

Since R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits.

Expansion permutation



Although the relationship between the input and output can be defined mathematically,

Expansion P-box table

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

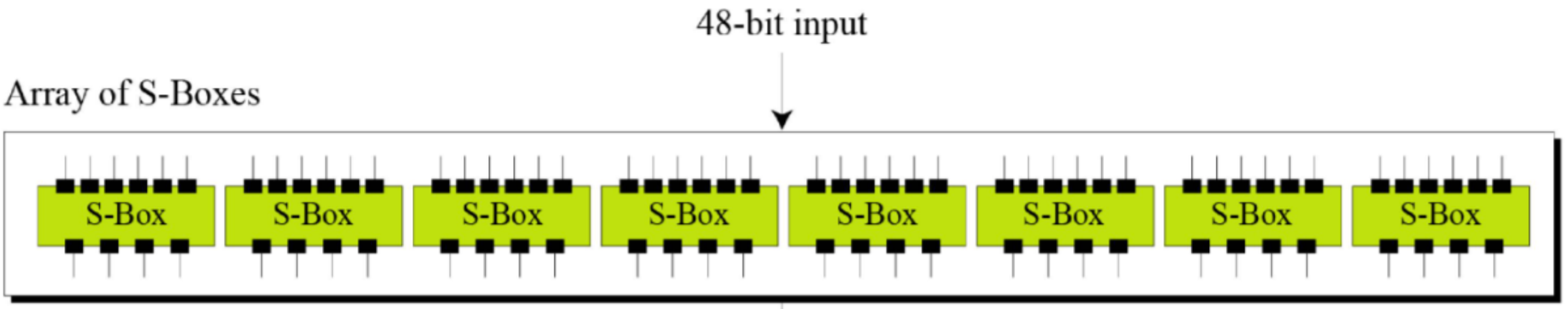
Whitener (XOR)

After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key. Note that both the right section and the key are 48-bits in length. Also note that the round key is used only in this operation.

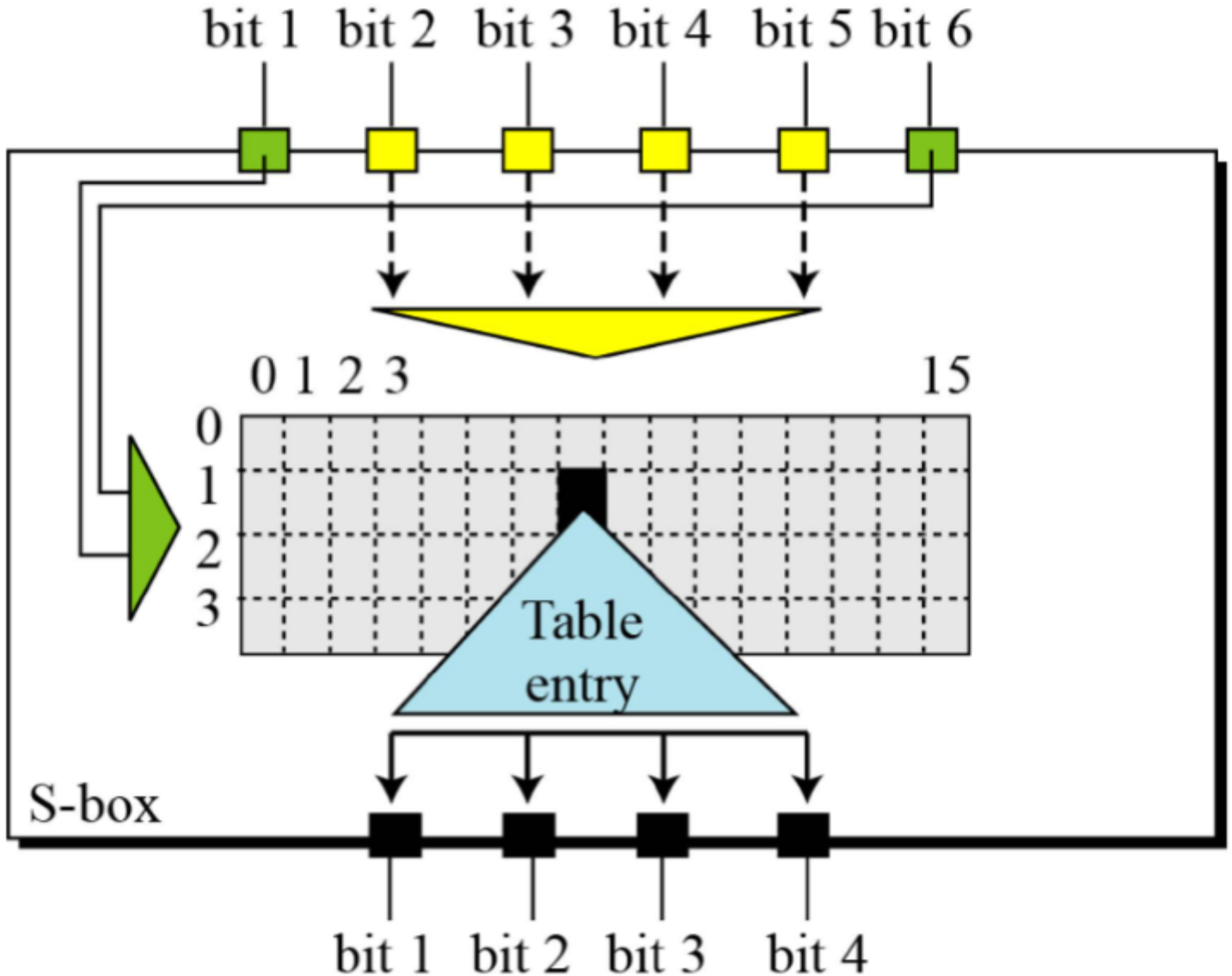
S-Boxes

The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.

S-boxes



S-box rule


bit 1
bit 2
bit 3
bit 4
bit 5
bit 6
0 1 2 3
15
0
1
2
3
Table
entry
S-box
bit 1
bit 2
bit 3
bit 4

- 32 bit output of s boxes is then subjected to the straight permutation to get 32 bit output with rule below

Straight Permutation

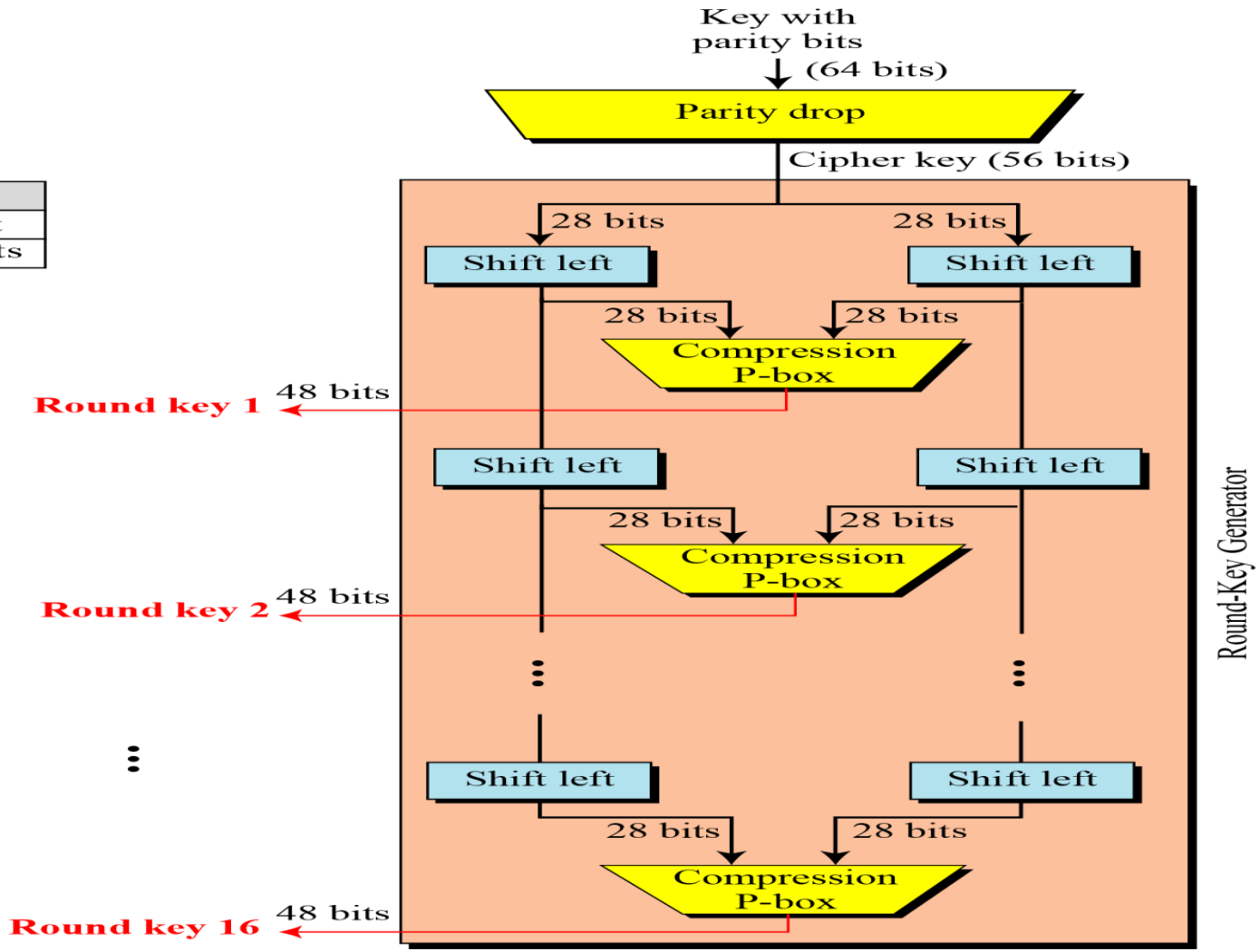
Straight permutation table

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

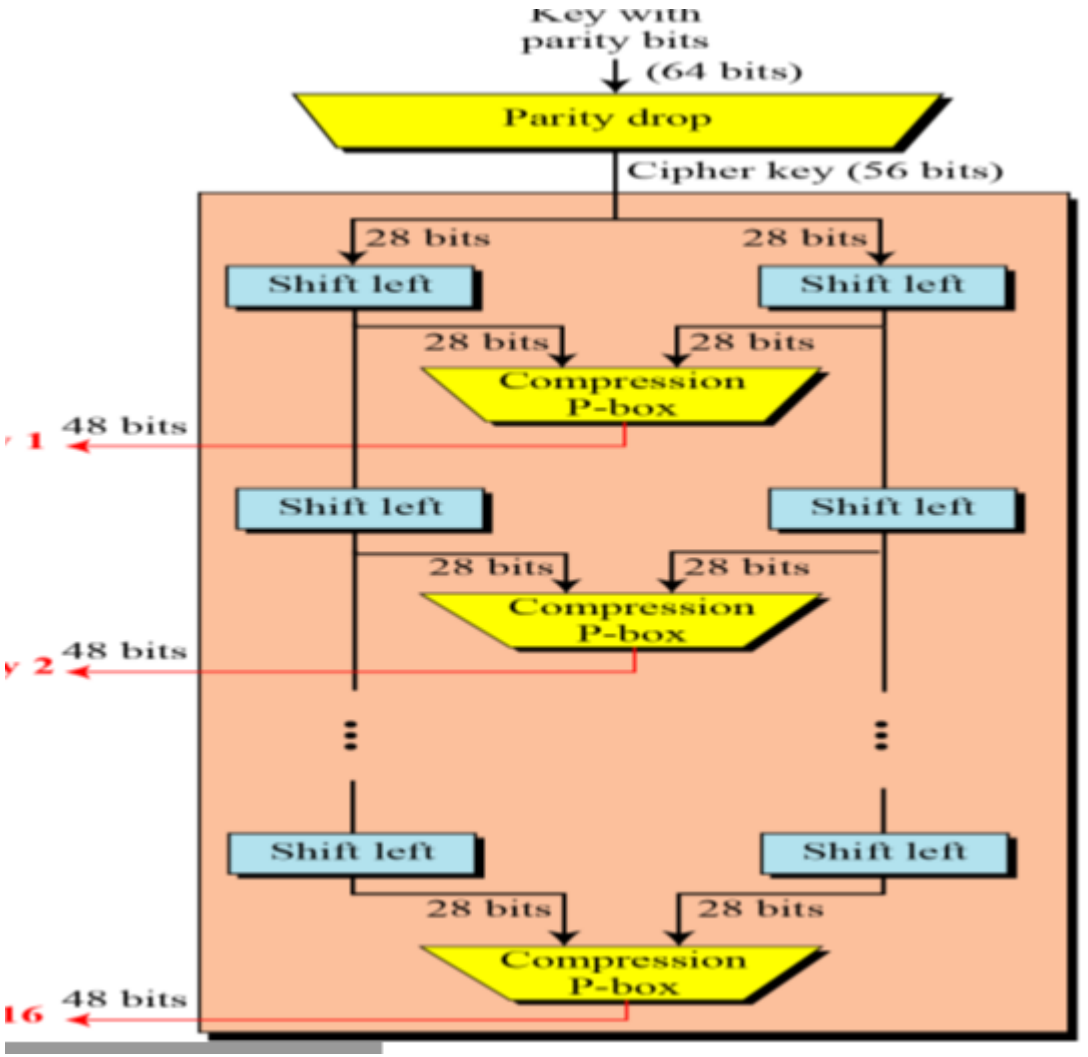
Key generation in DES

Shifting

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



- Cipher key is given as 64 bit key out of which 8 extra bits (parity bits are dropped). Round key generator creates sixteen 48 bits out of 56 bit cipher key.
- Parity drop drops parity bits(8,16,24,…64) from 64 bit key and permutes the rest 56 bits acc to the table. Then 56 bits are divided into two 28 bit parts. In every round each part is shifted left one or two bits. The compression P box then changes 56 bits to 48 bits which Is the key for the round.



Shifting

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

Parity-bit drop table

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

Table 6.13 *Number of bits shifts*

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 6.14 *Key-compression table*

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Disadvantage of DES

- The 56 bit key size is the largest defect of DES and the chips to implement one million of DES encrypt or decrypt operations a second are applicable (in 1993).
- Hardware implementations of DES are very quick.
- DES was not designed for application and therefore it runs relatively slowly

Triple DES

- Triple Data Encryption Standard (DES) is a type of computerized cryptography where block cipher algorithms are applied three times to each data block.
- The key size is increased in Triple DES to ensure additional security through encryption capabilities.
- Each block contains 64 bits of data.
- Encrypt-Decrypt-Encrypt (EDE). It works by taking three 56-bit keys (K1, K2 and K3), and encrypting first with K1, decrypting next with K2 and encrypting a last time with K3.

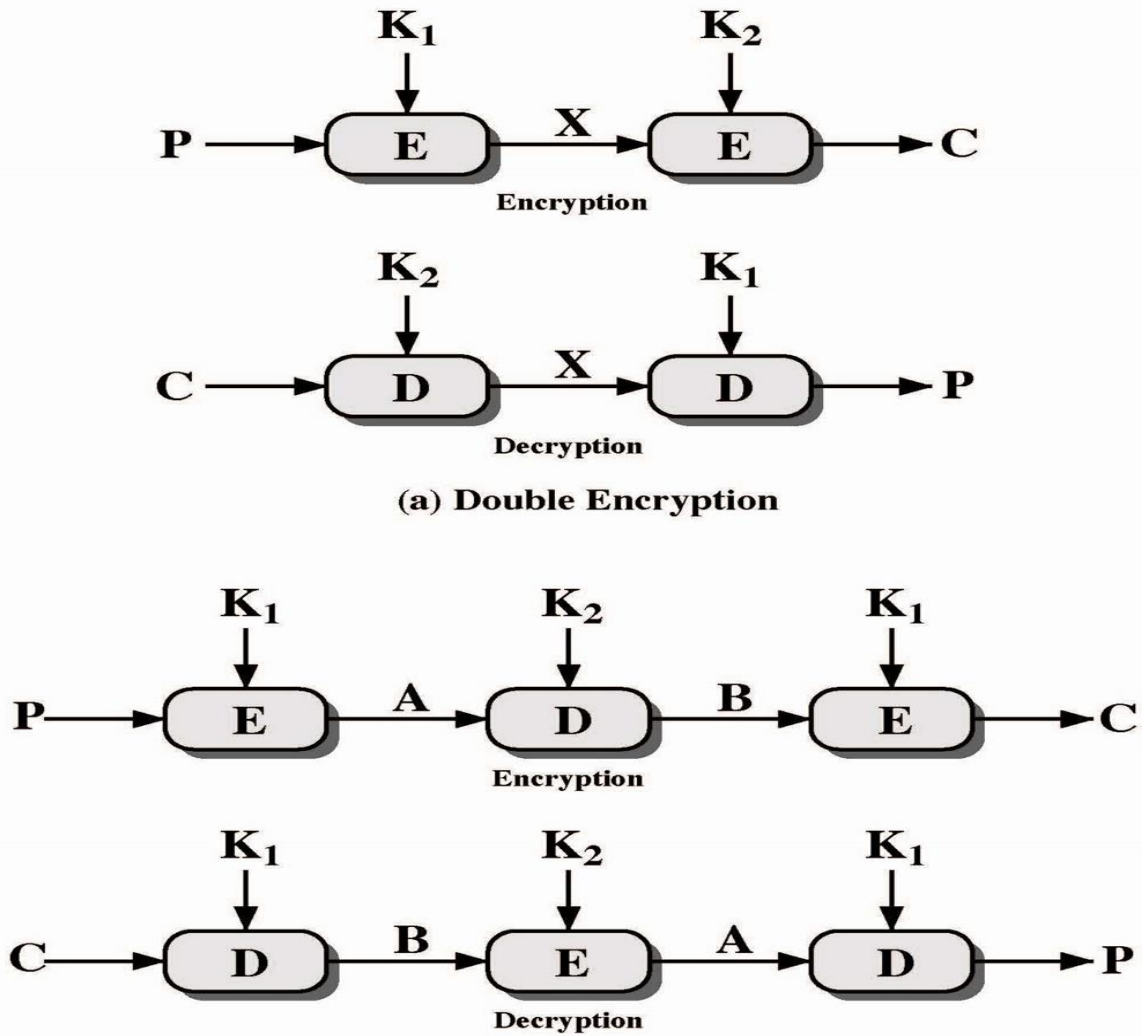


Figure 6.1 Multiple Encryption

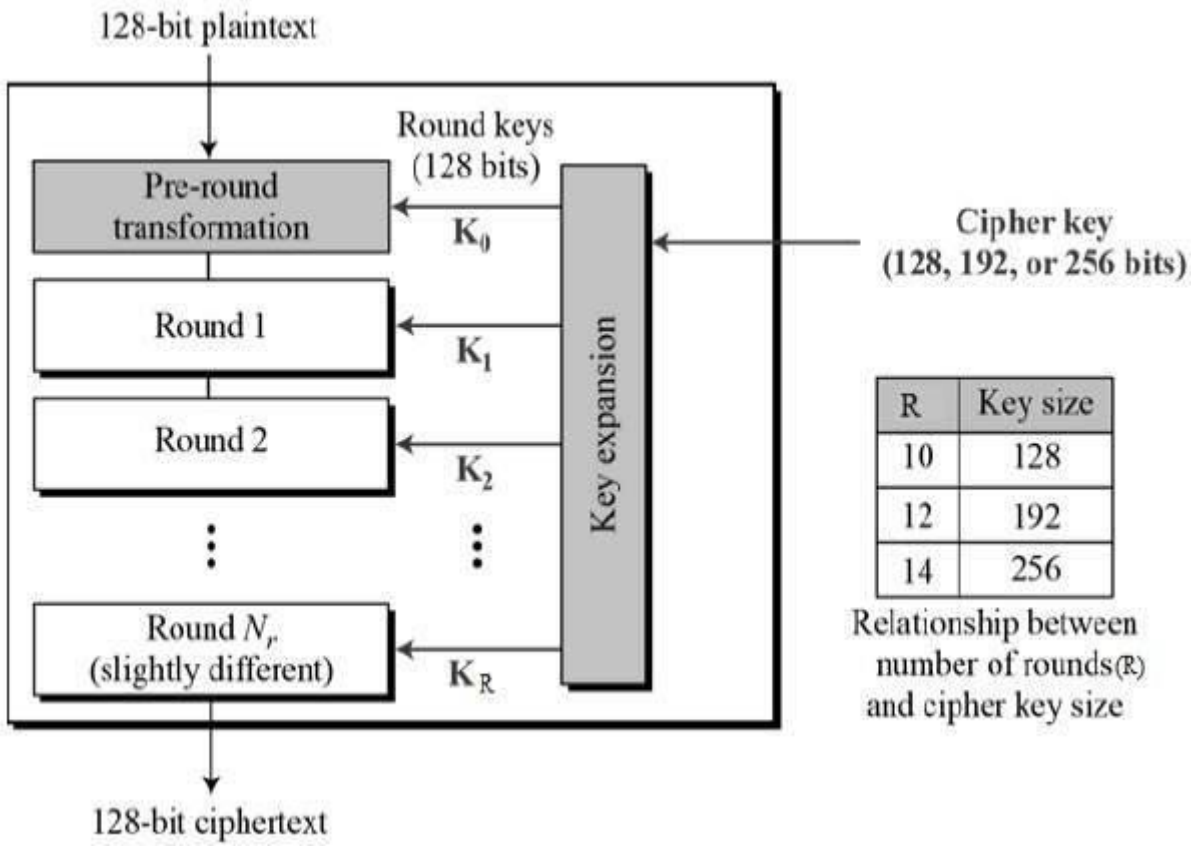
Triple DES Modes

MODE	ENCRYPTION SEQUENCE
DDD	Decrypt-Decrypt-Decrypt
DDE	Decrypt-Decrypt-Encrypt
DED	Decrypt-Encrypt-Decrypt
DEE	Decrypt-Encrypt-Encrypt
EDD	Encrypt-Decrypt-Decrypt
EDE	Encrypt-Decrypt-Encrypt
EED	Encrypt-Encrypt-Decrypt
EEE	Encrypt-Encrypt-Encrypt

Advanced Encryption Standard (AES)

- Symmetric Key Block cipher
- AES is non feistel cipher that encrypts and decrypts a data block of 128 bits.
- The number of rounds in AES is variable and depends upon length of key. It has defined 3 versions with 10,12,14 rounds
- Every key uses different key size which can be 128,192,256 bits depending on the number of rounds
- 10 rounds for 128 bit, 12 for 192 bit and 14 rounds for 256 bit keys

Design of AES encryption



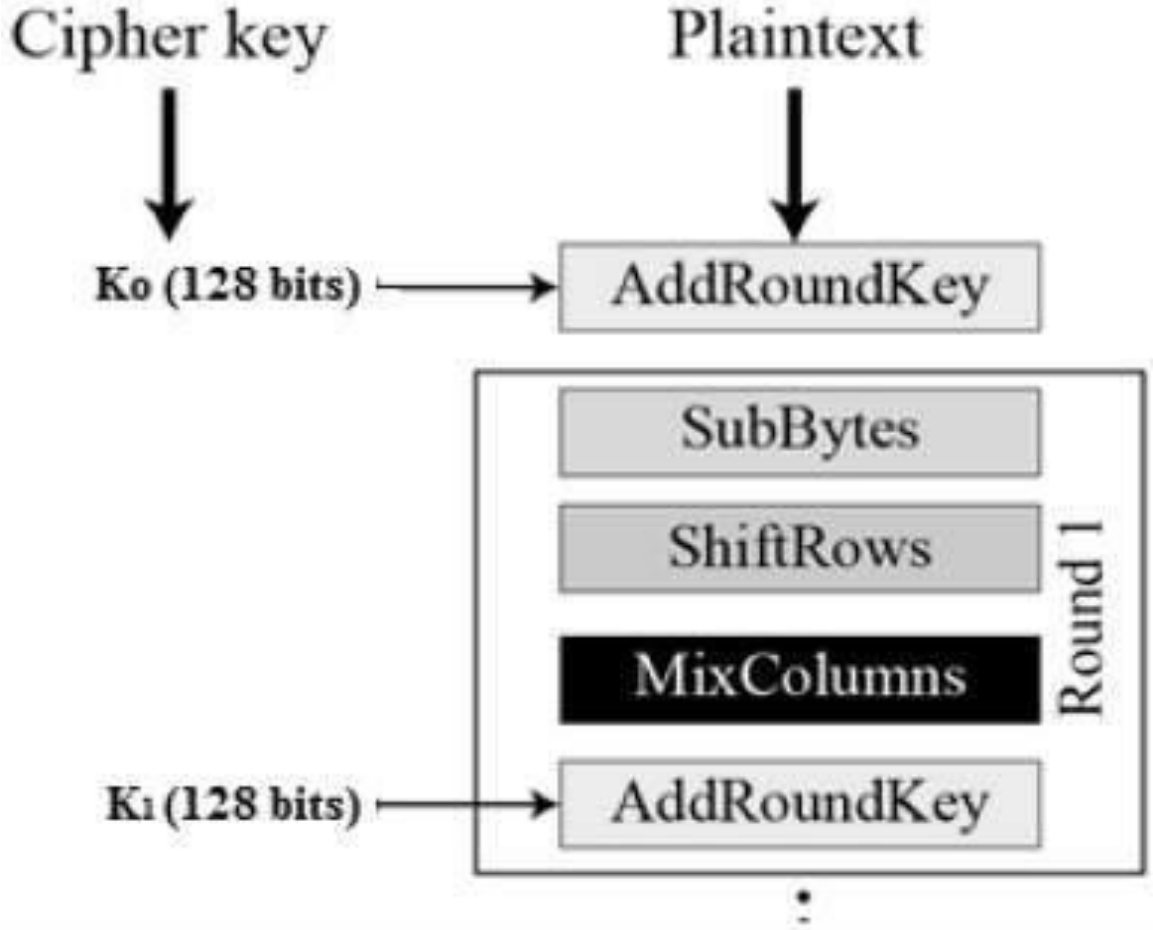
R	Key size
10	128
12	192
14	256

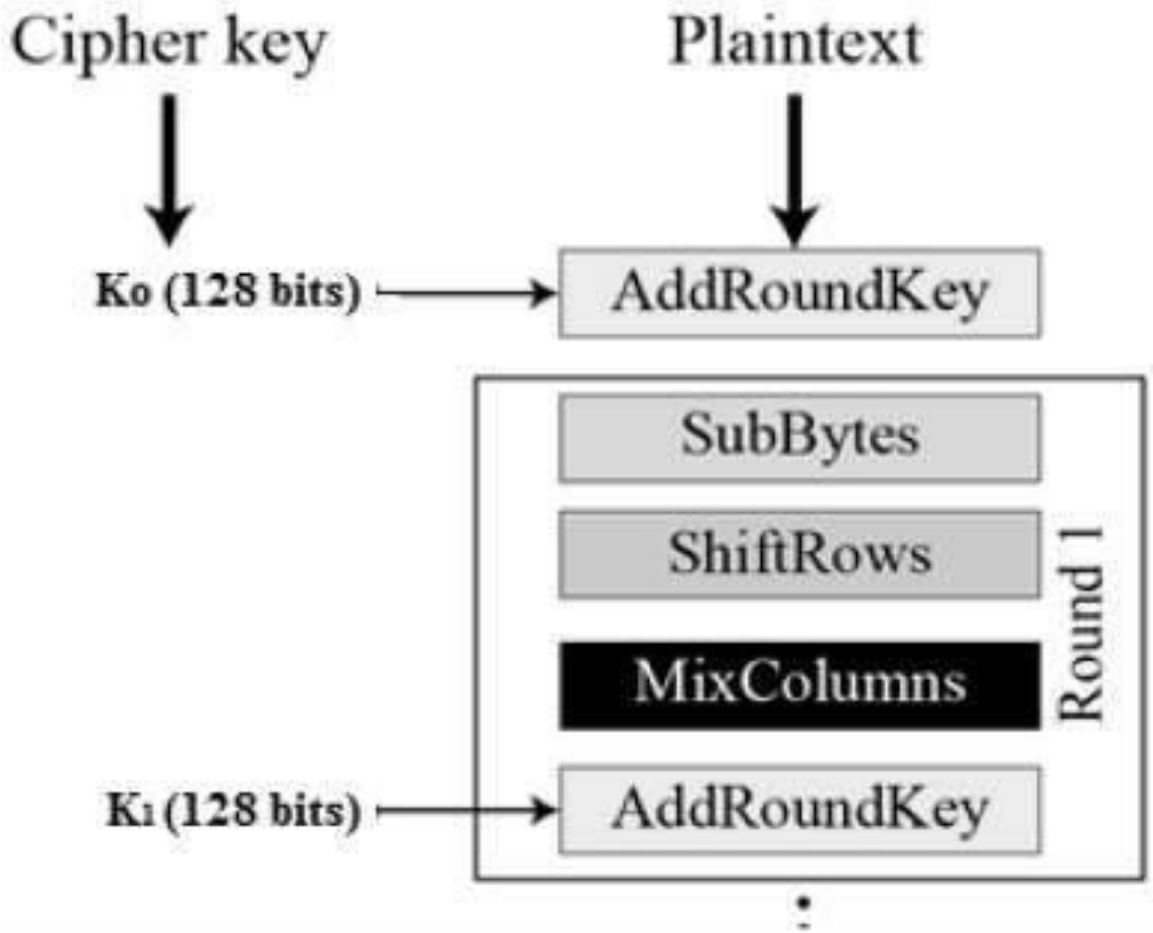
Relationship between number of rounds(R) and cipher key size

Encryption process

- Comprises of 4 transformations
- SubBytes (substitution)
- shiftRows (permutation)
- MixColumns (Mixing)
- AddRoundKey (key adding)

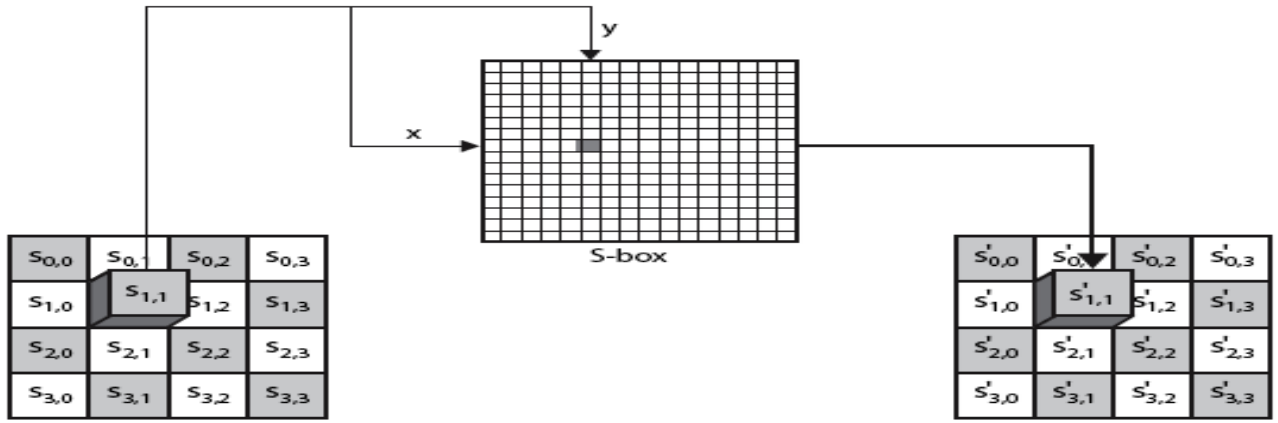
- Pre round uses one transformation
- Last round has only 3 transformations, mix columns transformation is not used




Cipher key
Plaintext
K0 (128 bits)
AddRoundKey
SubBytes
ShiftRows
MixColumns
Round 1
K1 (128 bits)
AddRoundKey

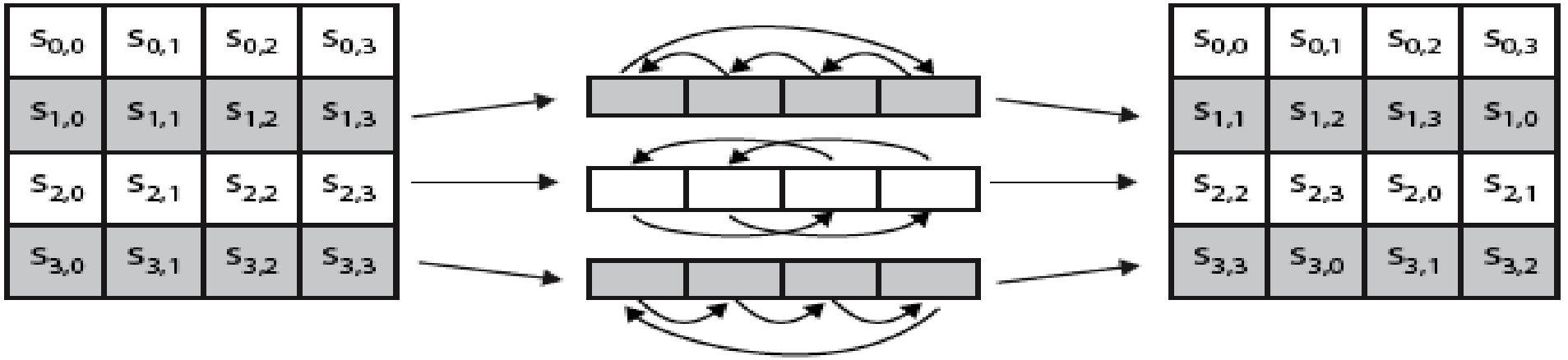
SubBytes (substitution)

- The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns



shiftRows (permutation)

- Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row. Shift is carried out as follows –
  - First row is not shifted.
  - Second row is shifted one (byte) position to the left.
  - Third row is shifted two positions to the left.
  - Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.



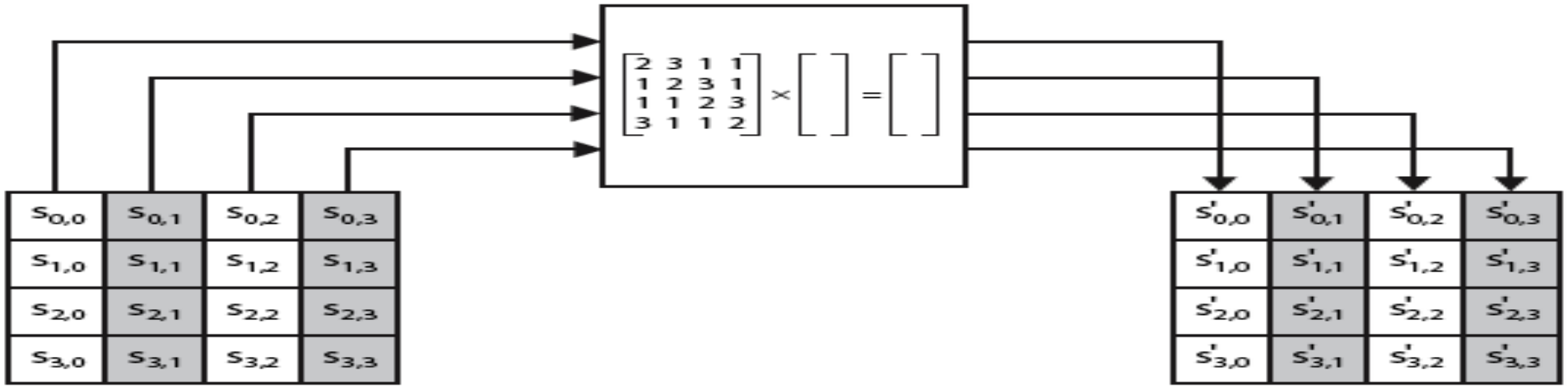
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

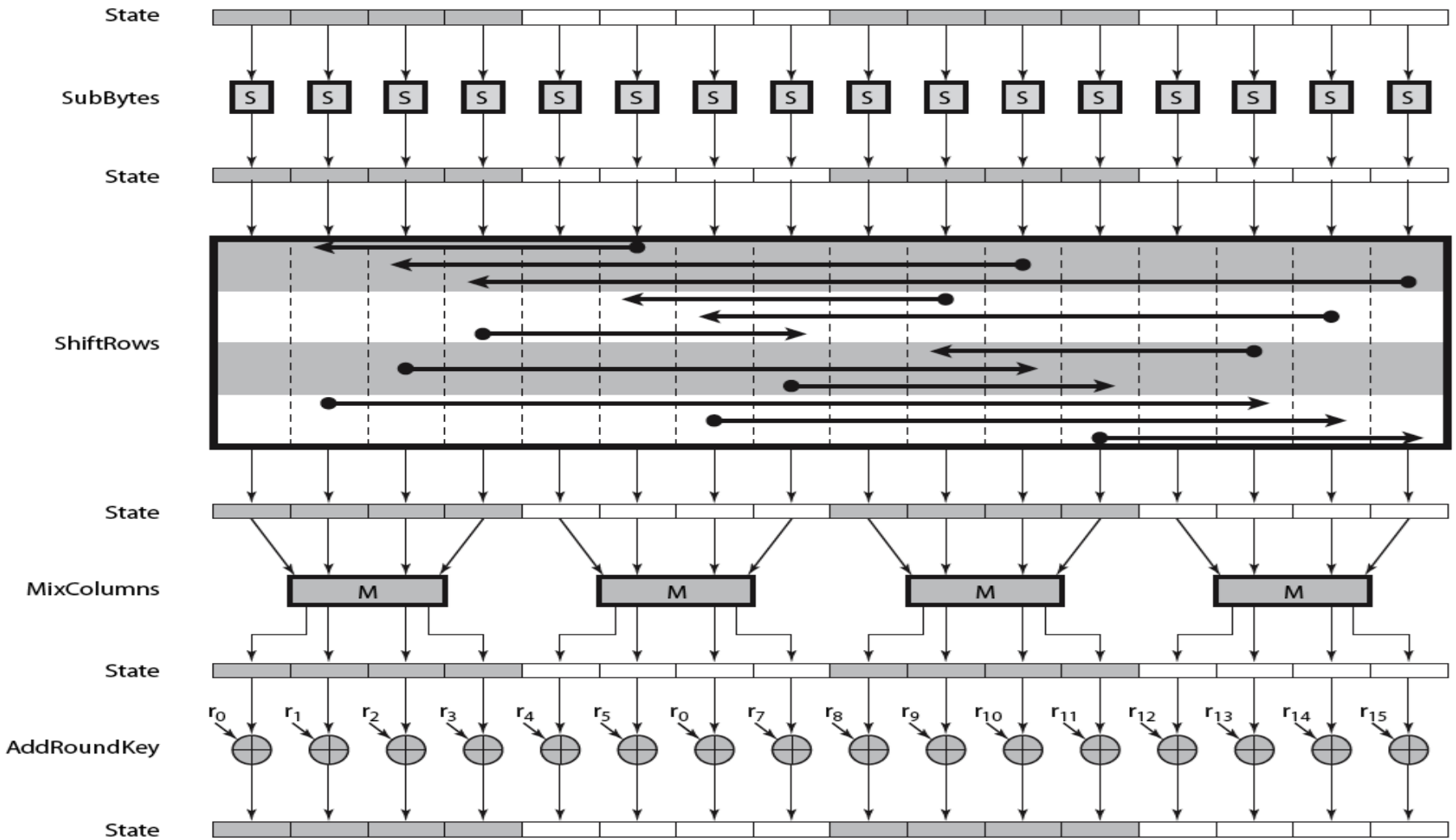
MixColumns (Mixing)

- Each column of four bytes is now transformed using a special mathematical function.
- This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column.
- The result is another new matrix consisting of 16 new bytes.
- It should be noted that this step is not performed in the last round.



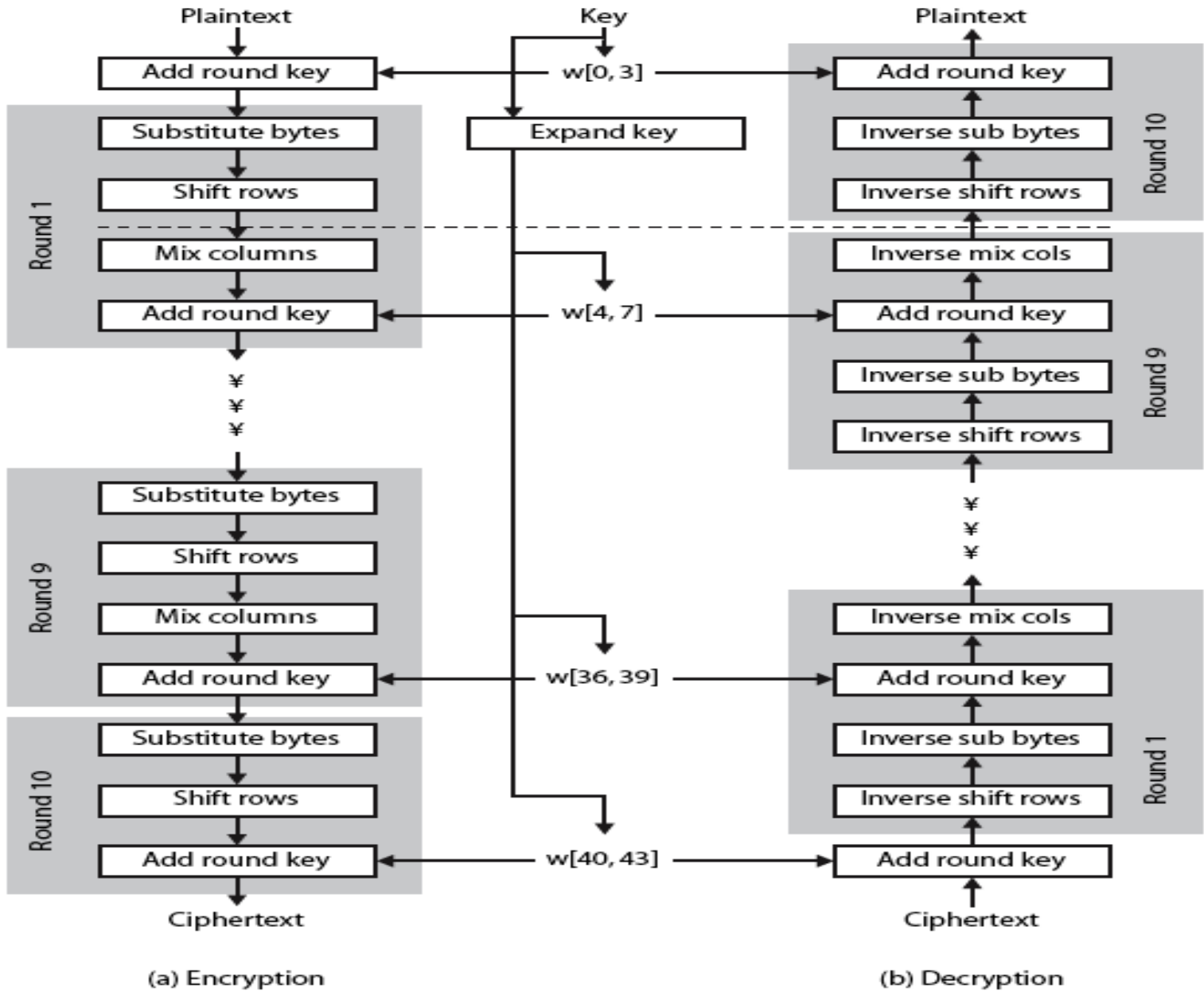
AddRoundKey (key adding)

- The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key.
- If this is the last round then the output is the ciphertext.
- Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.
- AES is widely adopted and supported in both hardware and software.
- No practical cryptanalytic attacks against AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of ‘future-proofing’ against progress in the ability to perform exhaustive key searches


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0
r1
r2
r3
r4
r5
r0
r7
r8
r9
r10
r11
r12
r13
r14
r15
State

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key



(a) Encryption

(b) Decryption

Analysis of AES

- an **iterative** rather than **Feistel** cipher
- key expanded into array of 32-bit words
  - four words form round key in each round
- 4 different stages are used as shown
- has a simple structure
- only AddRoundKey uses key
- AddRoundKey a form of Vernam cipher
- each stage is easily reversible
- decryption uses keys in reverse order
- decryption does recover plaintext
- final round has only 3 stages

AES VS DES

Basis of Comparison	DES	AES
Developed	DES was developed in 1977	AES was developed in 2001
Full-Form	DES stands for Data Encryption Standard	AES stands for Advanced Encryption Standard
Principle	DES follows the principle of Feistel Structure	AES s based on the principle of Substitution and Permutation
Plaintext	The plaintext is of 64 bits.	The plaintext can be 128, 192, 256 bits.
Ciphertext	Generate Ciphertext of 64 bits	Can Generate Ciphertext of 128, 192, 256 bits
Key Length	The key length is 56 bits.	Key length can be 128, 192, 256 bits.
Rounds	DES contains a fixed number of rounds, i.e. 16	AES contains a variable number of rounds depending on the size of the input, i.e. 10 rounds for 128 bit, 12 rounds for 192 bit and 14 rounds for 256 bits
Security	DES is less secure and hardly used now	AES is much more secure than DES and is widely used nowadays.
Speed	DES is comparatively slower than AES	AES is faster than DES

Public key cryptography

- RC5
- RSA
- DIFFIE HELLMAN KEY EXCHANGE
- EIGamal Encryption
- Elliptical Curve Cryptography(ECC)

RC5

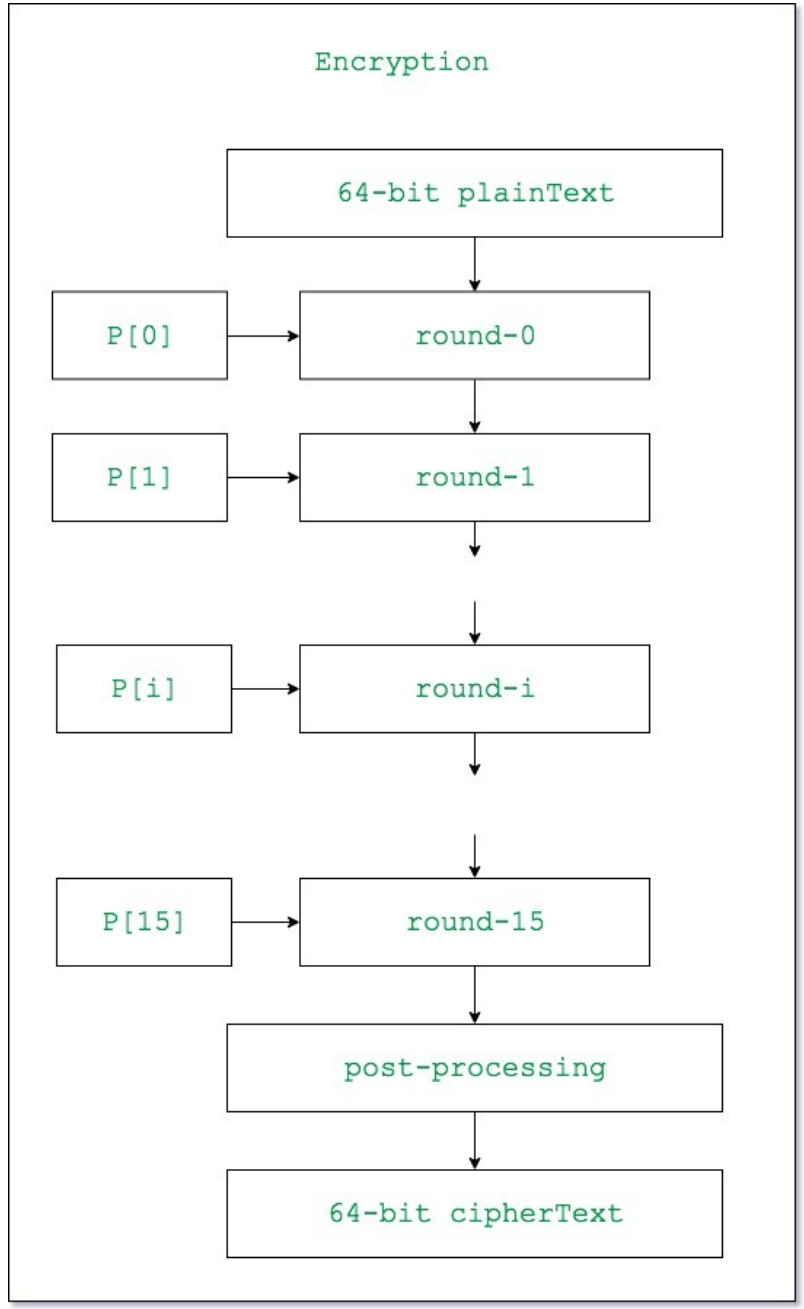
- RC5 is a symmetric key block encryption algorithm designed by Ron Rivest in 1994.
- It is notable for being simple, fast (on account of using only primitive computer operations like XOR, shift, etc.) and consumes less memory.
- RC5 is a block cipher and addresses two word blocks at a time. Depending on input plain text block size, number of rounds and key size

Blowfish algorithm

- Blowfish is a **variable-length, symmetric, 64-bit block cipher.**
- Designed by **Bruce Schneier in 1993** as a "general-purpose algorithm,"
- It was intended to provide a **fast, free, drop-in alternative to the aging Data Encryption Standard (DES)**

Understanding Blowfish

- Blowfish features a 64-bit block size and takes a variable-length key, from 32 bits to 448 bits.
- It consists of 16 Feistel-like iterations, where each iteration operates on a 64-bit block that's split into two 32-bit words.
- Blowfish uses a single encryption key to both encrypt and decrypt data.


Encryption
64-bit plainText
P[0]
round-0
P[1]
round-1
P[i]
round-i
P[15]
round-15
post-processing
64-bit cipherText

Block size 64 bit

Key size 32 to 448 bit

No of subkeys 18 (P array)

No of rounds 16

No of S boxes 4 having 256 entries and each block is 32 bit

Fast, compact, simple and secure

Generation of subkeys

- 18 subkeys (P[0] TO P[17])
- These 18 subkeys are stored in P array each array element is 32 bit
- It contents hexadecimal representation of each subkey

32-bit hexadecimal representation of initial values of sub-keys

P[0]	243f6a88	P[9]	38d01377
P[1]	85a308d3	P[10]	be5466cf
P[2]	13198a2e	P[11]	34e90c6c
P[3]	03707344	P[12]	c0ac29b7
P[4]	a4093822	P[13]	c97c50dd
P[5]	299f31d0	P[14]	3f84d5b5
P[6]	082efa98	P[15]	b5470917
P[7]	ec4e6c89	P[16]	9216d5d9
P[8]	452821e6	P[17]	8979fb1b

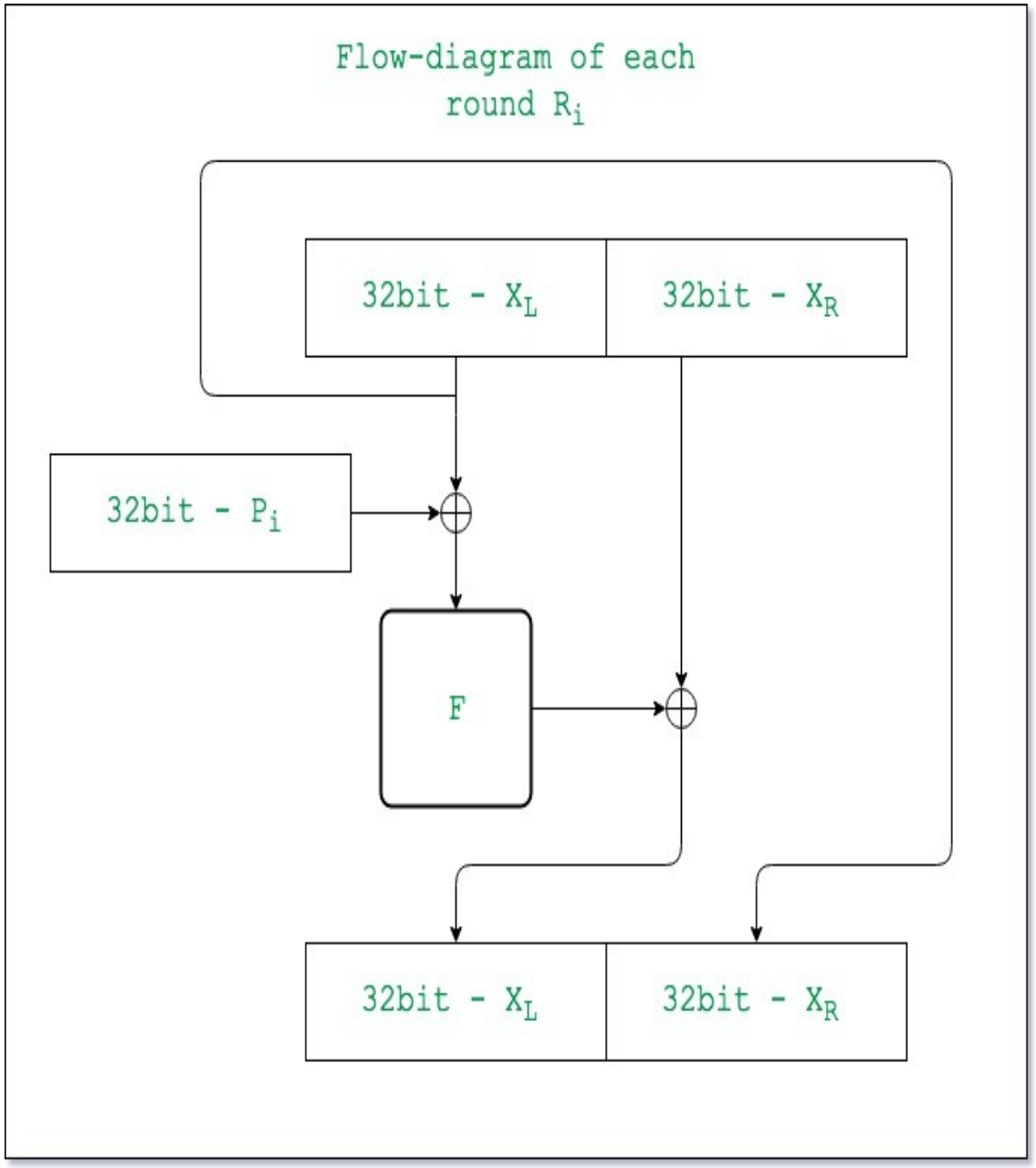
- Now each of the subkey is changed with respect to the input key as: (32-448)divide 32
- P[0] = P[0] xor 1st 32-bits of input key
- P[1] = P[1] xor 2nd 32-bits of input key
- .
- .
- .
- P[i] = P[i] xor (i+1)th 32-bits of input key
- (roll over to 1st 32-bits depending on the key length)
- .
- .
- .
- P[17] = P[17] xor 18th 32-bits of input key
- (roll over to 1st 32-bits depending on key length)
- The resultant P-array holds 18 subkeys that is used during the entire encryption process

Initialise Substitution Boxes

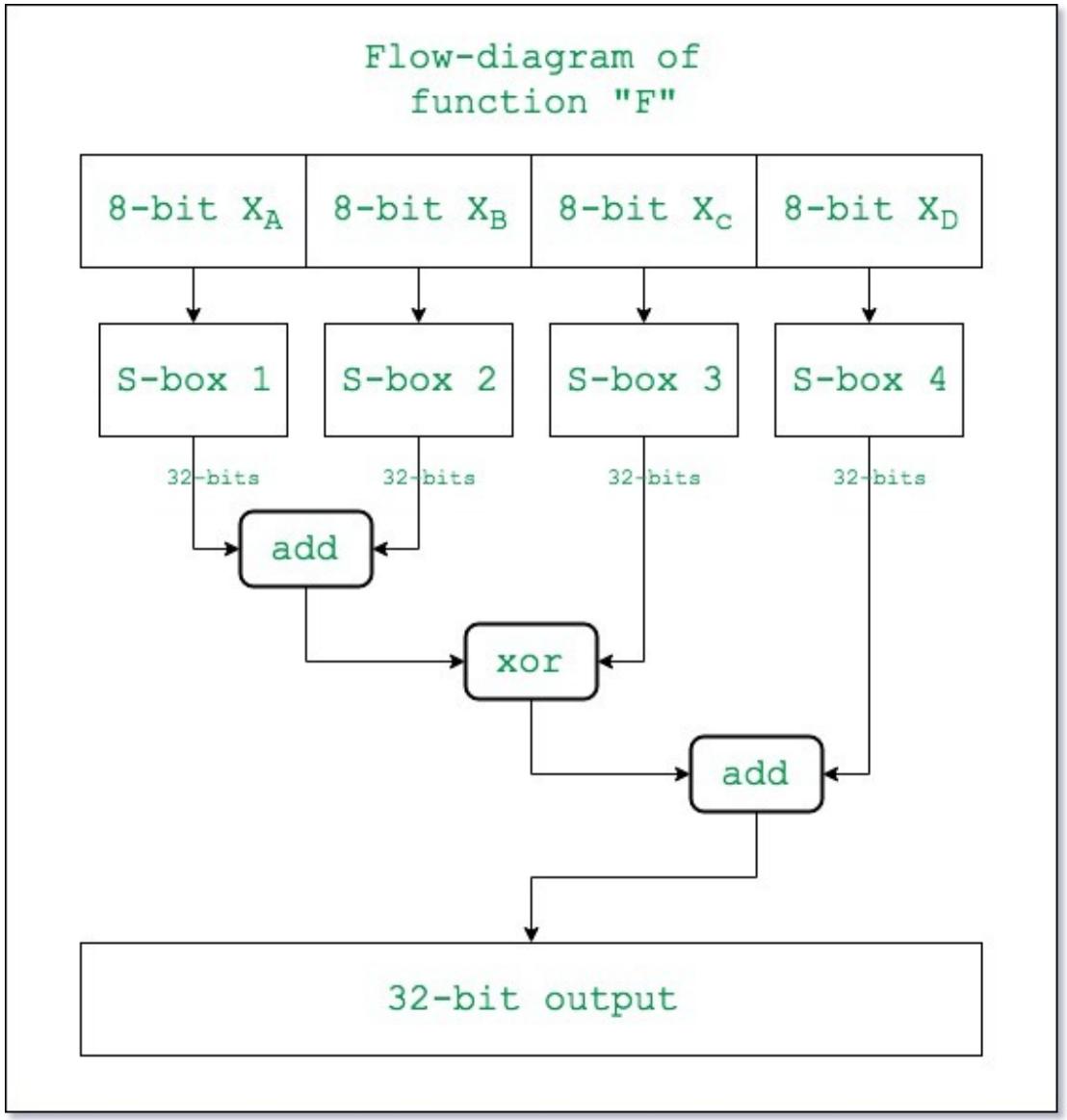
- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption aswell as decryption process
- each S-box having 256 entries{S[i][0]…S[i][255], 0&lei&le4} where each entry is 32-bit.

Encryption:

- The encryption function consists of two parts:
- a. Rounds: The encryption consists of 16 rounds with each round(Ri) taking inputs the plainText(P.T.) from previous round and corresponding subkey(Pi). The description of each round is as follows:



The description of the function ” F ” is as follows:



Post-processing