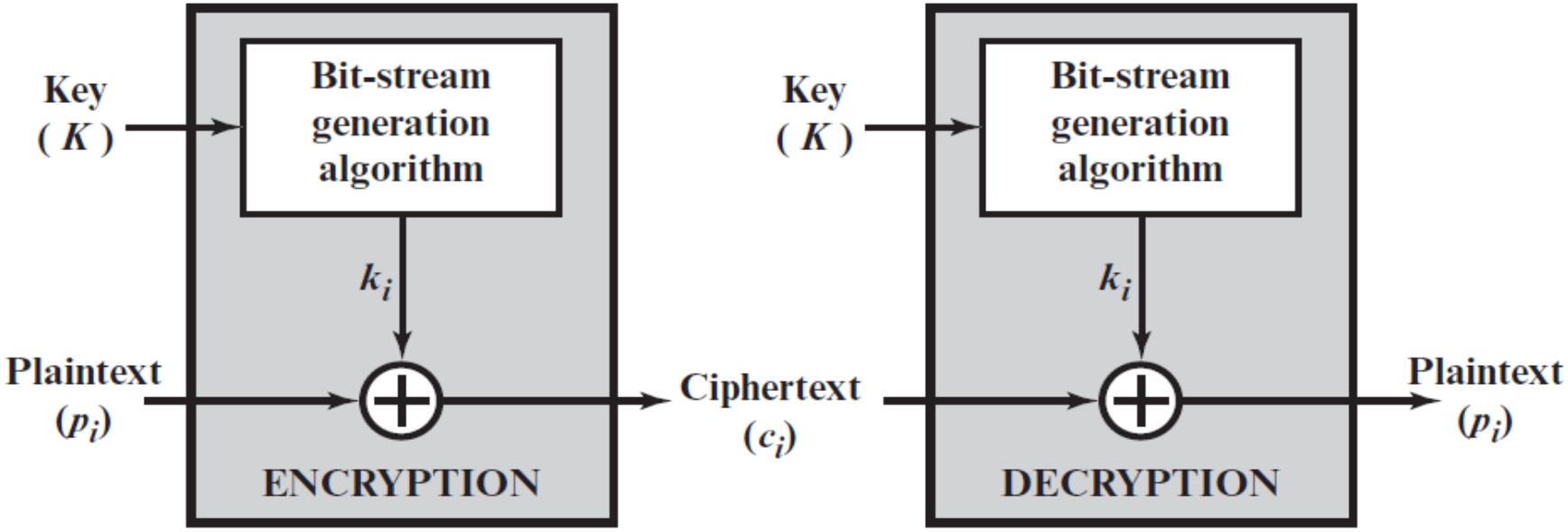
CRYPTOGRAPHY & CYBER SECURITY (RLMCA305) – MODULE 2

Bismi K. Charleys, Asst.professor in MCA,LMCST

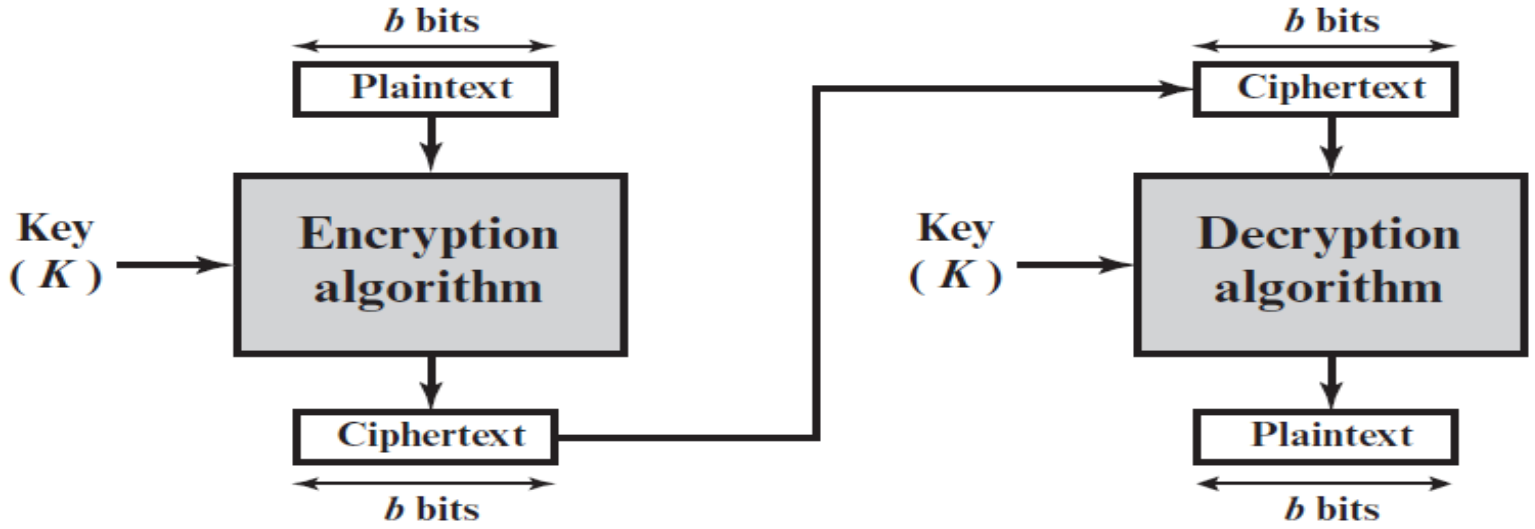
stream cipher

- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.



Block cipher

- A **block cipher is one in which a block of plaintext is treated as a whole and** used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used



(b) Block cipher

Block Cipher Design Principles

- **Block ciphers** are built in the ***Feistel cipher structure***. Block cipher has a specific number of rounds and keys for generating ciphertext. For defining the complexity level of an algorithm few design principles are to be considered.

They are:

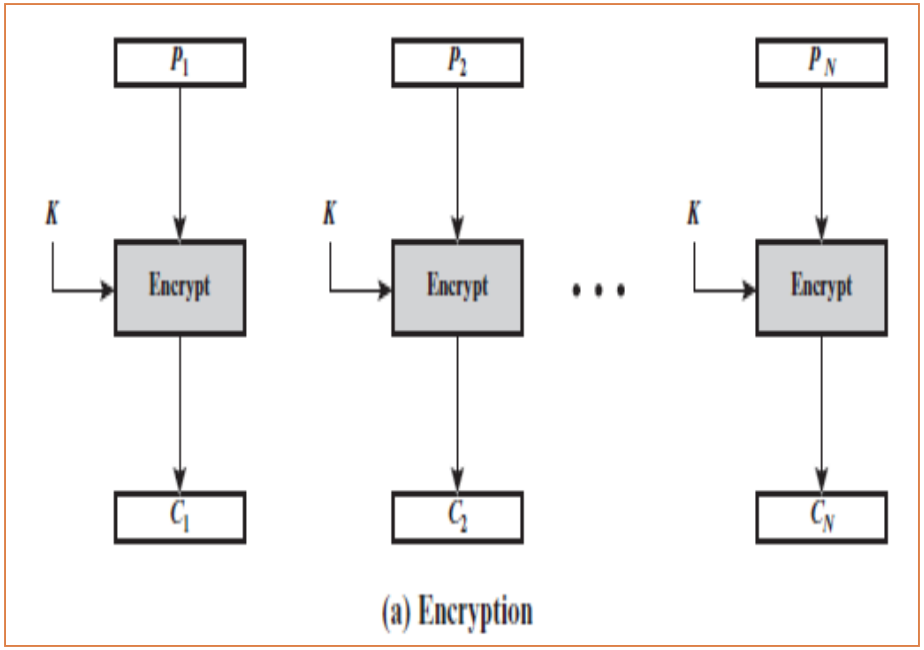
1. **Number of Rounds –**
   The number of Rounds is regularly considered in design criteria, it just reflects the number of rounds to be suitable for an algorithm to make it more complex..
2. **Design of Round function F –**
   The core part of the Feistel Block cipher structure is the Round Function. The complexity of cryptanalysis can be derived from the Round function i.e. the increasing level of complexity for the round function would be greatly contributing to an increase in complexity. To increase the complexity of the round function, the avalanche effect is also included in the round function, as the change of a single bit in plain text would produce a mischievous output due to the presence of avalanche effect.
3. **Subkeys count.** In Feistel Block cipher structure, each round would generate a sub-key for increasing the complexity of cryptanalysis. The Avalanche effect makes it more complex in deriving sub-key. Decryption must be done very carefully to get the actual output as the avalanche effect is present in it.
4. Key size
5. Block size

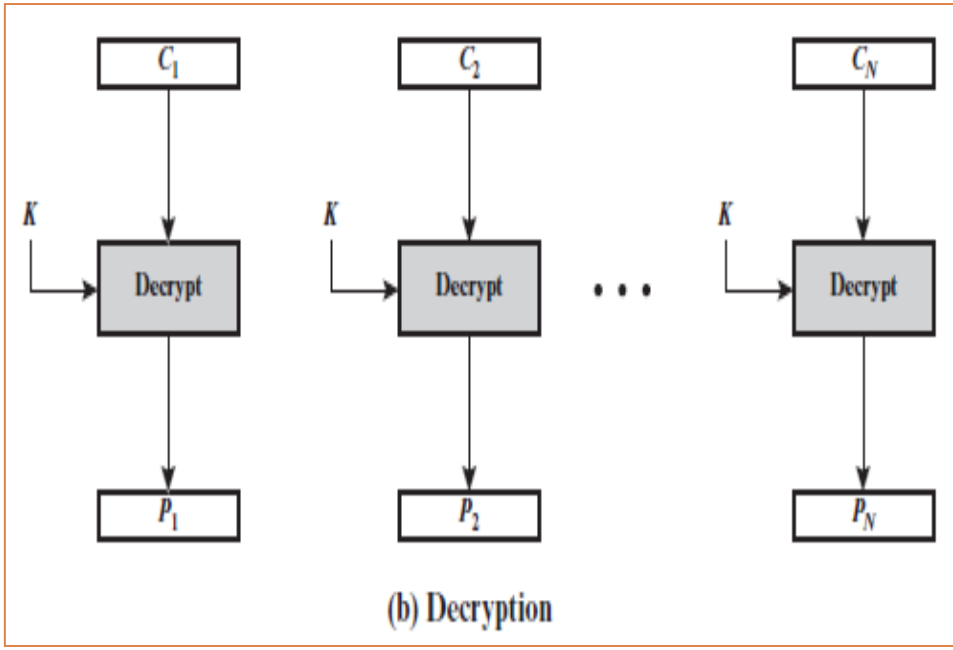
Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 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 block of plaintext and the preceding block 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 encryption output, and full blocks are used.	• 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

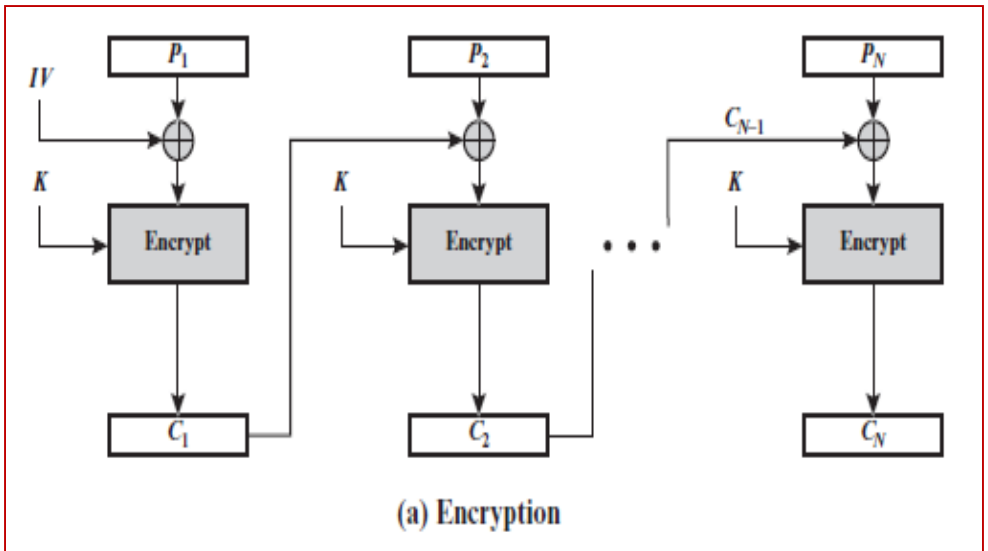
- The simplest mode is the **electronic codebook (ECB) mode, in which plaintext** is handled one block at a time and each block of plaintext is encrypted using the same key . The term *codebook is used because, for a given key, there is* a unique ciphertext for every *b-bit block of plaintext*.



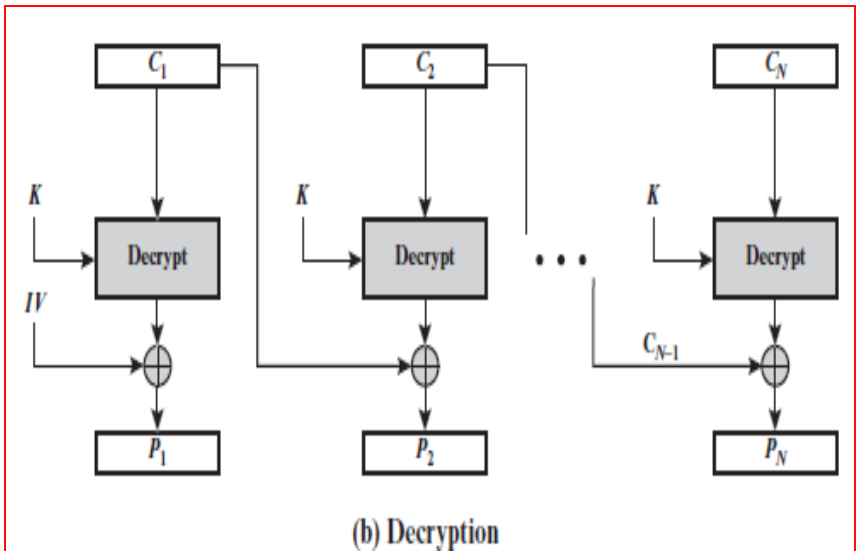


Cipher Block Chaining Mode(CBC)

- To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the **cipher block chaining (CBC) mode.** In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.



(a) Encryption



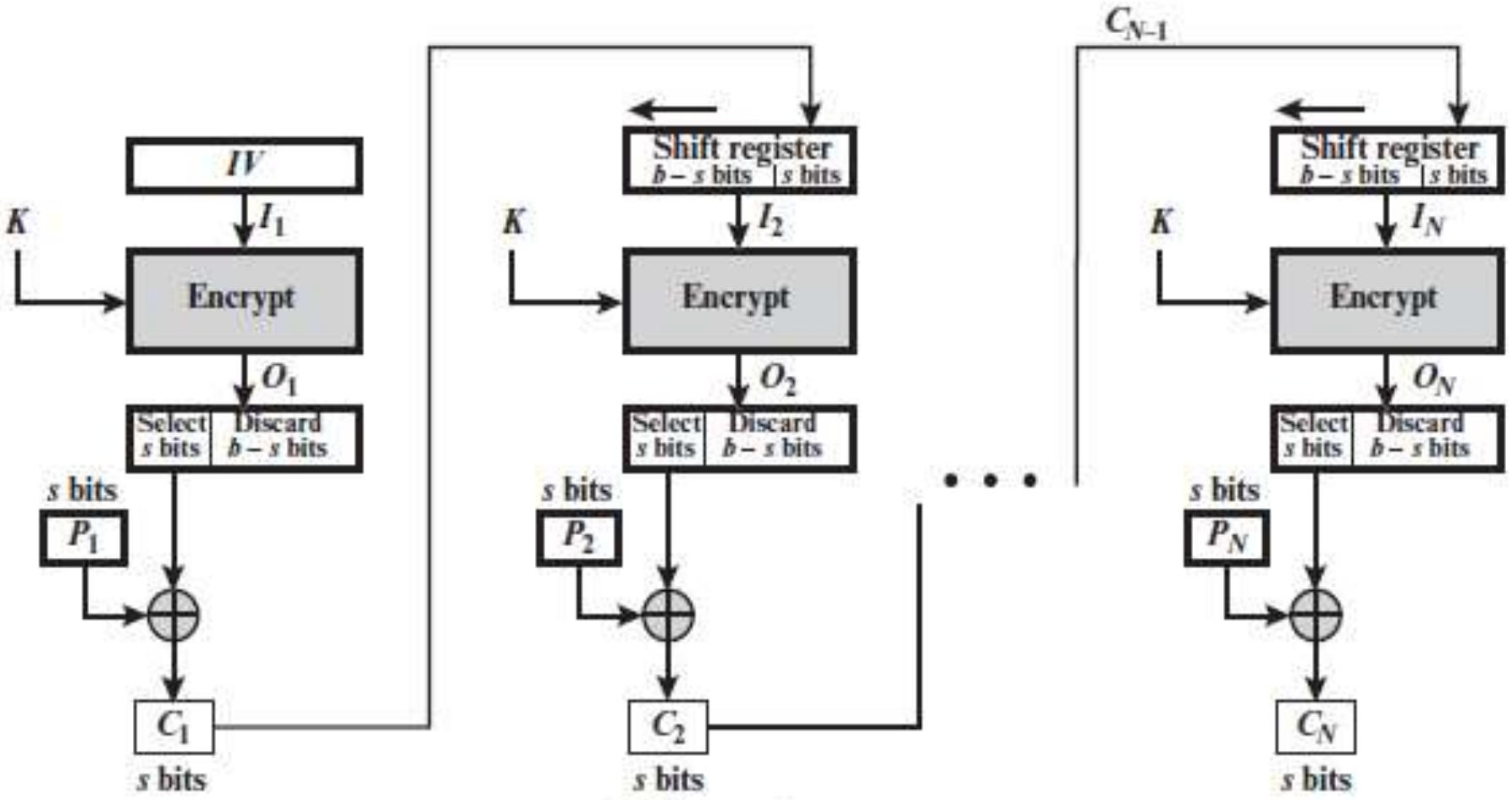
(b) Decryption

Cipher Feedback Mode

- The input to the encryption function is a *b-bit shift* register that is initially set to some initialization vector (IV). The leftmost (most significant) *s bits of the output of the encryption function are XORed with the first* segment of plaintext *P1 to produce the first unit of ciphertext C1, which is then* transmitted. In addition, the contents of the shift register are shifted left by *s bits,* and *C1 is placed in the rightmost (least significant) s bits of the shift register. This* process continues until all plaintext units have been encrypted.

C1 = P1 ⊕MSBs[E(K, IV)]

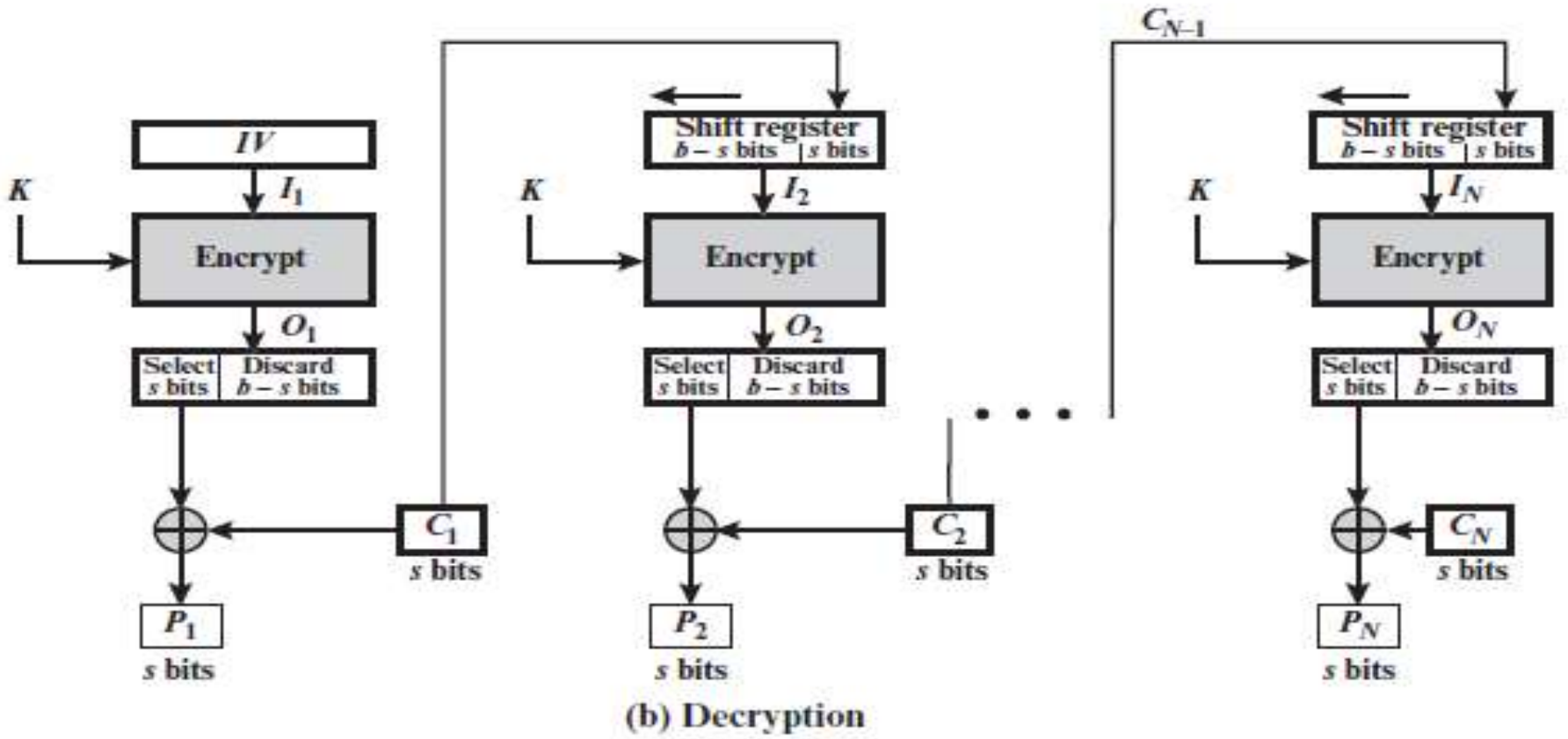
Cipher Feedback Mode



(a) Encryption

Cipher Feedback Mode

P1 = C1 ⊕MSBs[E(K, IV)]



Output Feedback Mode

- The **output feedback (OFB) mode is similar in structure to that of CFB. For OFB,** the output of the encryption function is fed back to become the input for encrypting the next block of plaintext . In CFB, the output of the XOR unit is fedback to become input for encrypting the next block. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, whereas CFB operates on an *s-bit subset*. OFB encryption can be expressed as

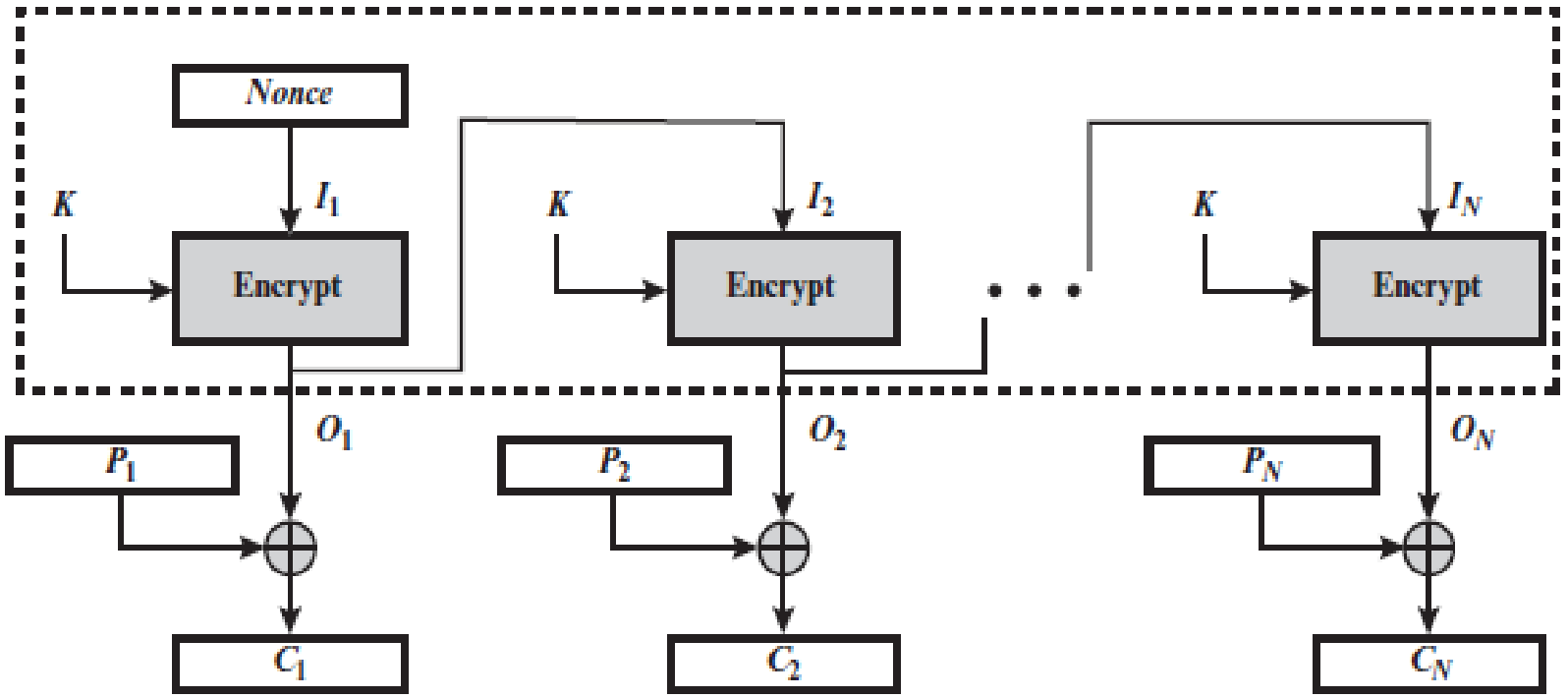
$C_j = P_j \oplus E(K, O_{j-1})$

Where

$O_{j-1} = E(K, O_{j-2})$

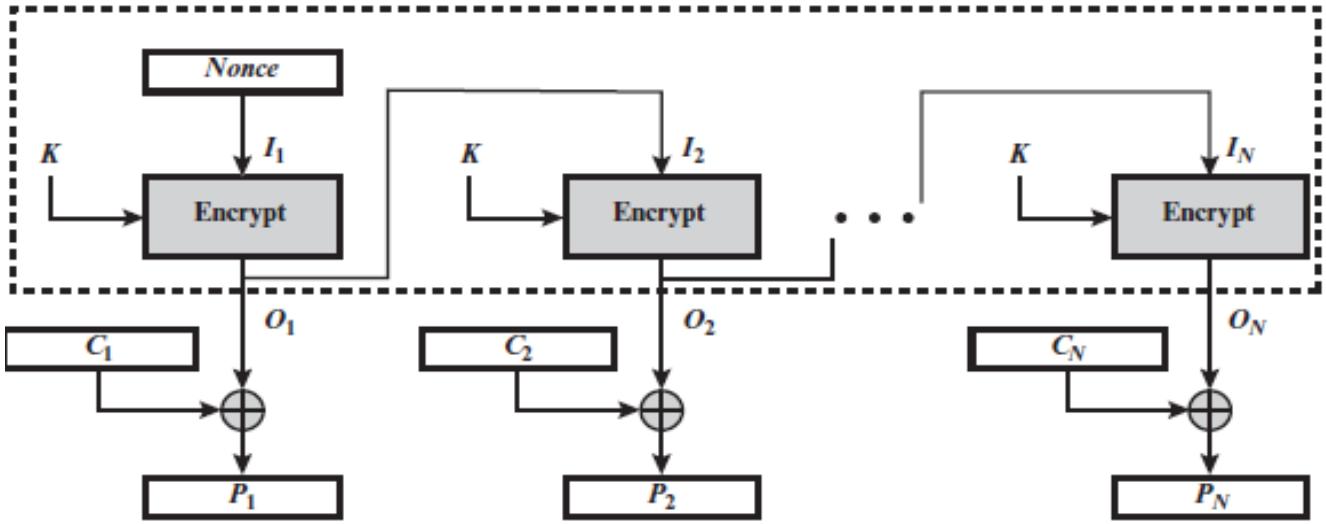
$C_j = P_j \oplus E(K, [C_{j-1} \oplus P_{j-1}])$

Output Feedback Mode



(a) Encryption

Output Feedback Mode



(b) Decryption

Data Encryption Standard (DES)

- Until the introduction of the Advanced Encryption Standard (AES) in 2001, the Data Encryption Standard (DES) was the most widely used encryption scheme.
- DES was issued in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA). For DEA, data are encrypted in 64-bit blocks using a 64-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. To do the encryption, DES uses "keys" where are also apparently 16 hexadecimal numbers long, or apparently **64 bits** long. However, every 8th key bit is ignored in the DES algorithm, so that the effective key size is **56 bits**.
- The same steps, with the same key, are used to reverse the encryption.

Data Encryption Standard (DES)

Block Size = 64 bit Plain Text

No of Rounds = 16 rounds

Key Size = 64 bit

No of subkeys =16 subkeys

Subkey size= 48 bit subkey

Cipher Text-=64 bit cipher text

DES Encryption

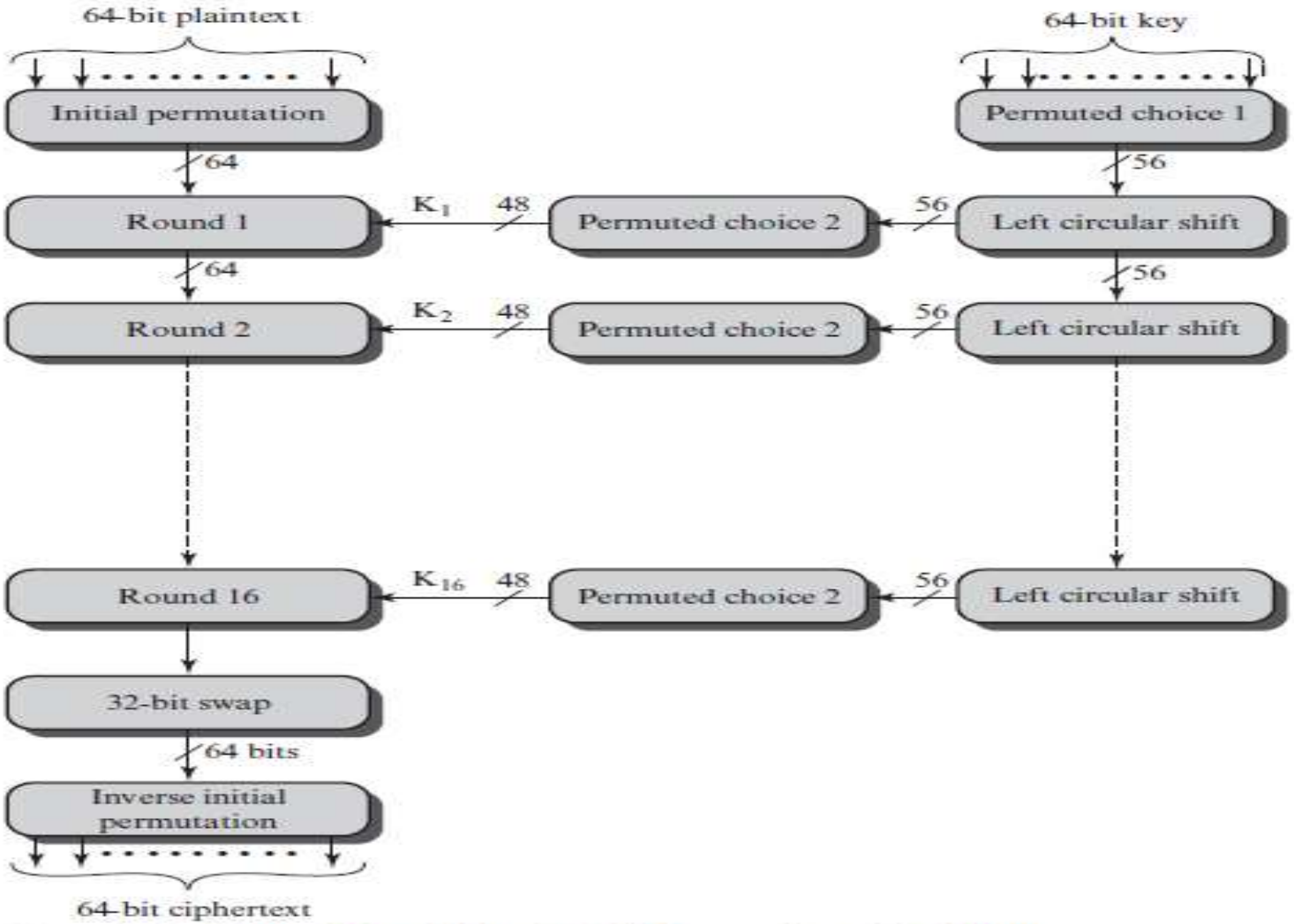


Figure 4.5 General Depiction of DES Encryption Algorithm

Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)
- no cryptographic value
- example:

```
IP(675a6967 5e5a6b5a) = (ffb2194d
004df6fb)
```

Initial & Final permutation

Table 6.1 *Initial and final permutation tables*

Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Permuted choice 1

Permuted choice1

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

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

Permuted Choice2

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

Permuted choice 2

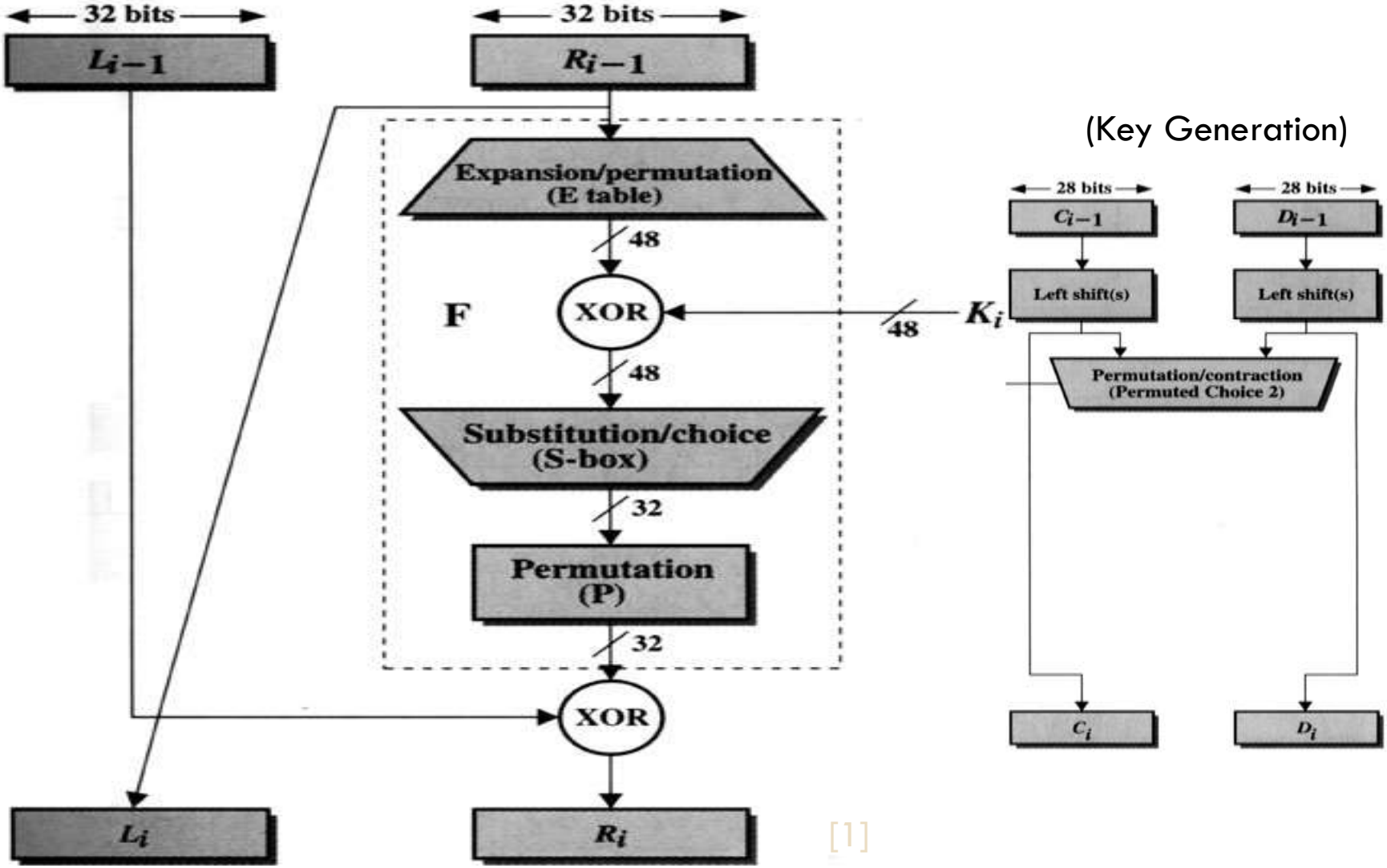
DES Round Structure

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \square F(R_{i-1}, K_i)$
- F takes 32-bit R half and 48-bit subkey:
  - expands R to 48-bits using perm E
  - adds to subkey using XOR
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes using 32-bit perm P

Encryption (Round)



[1]

Expansion P-Box

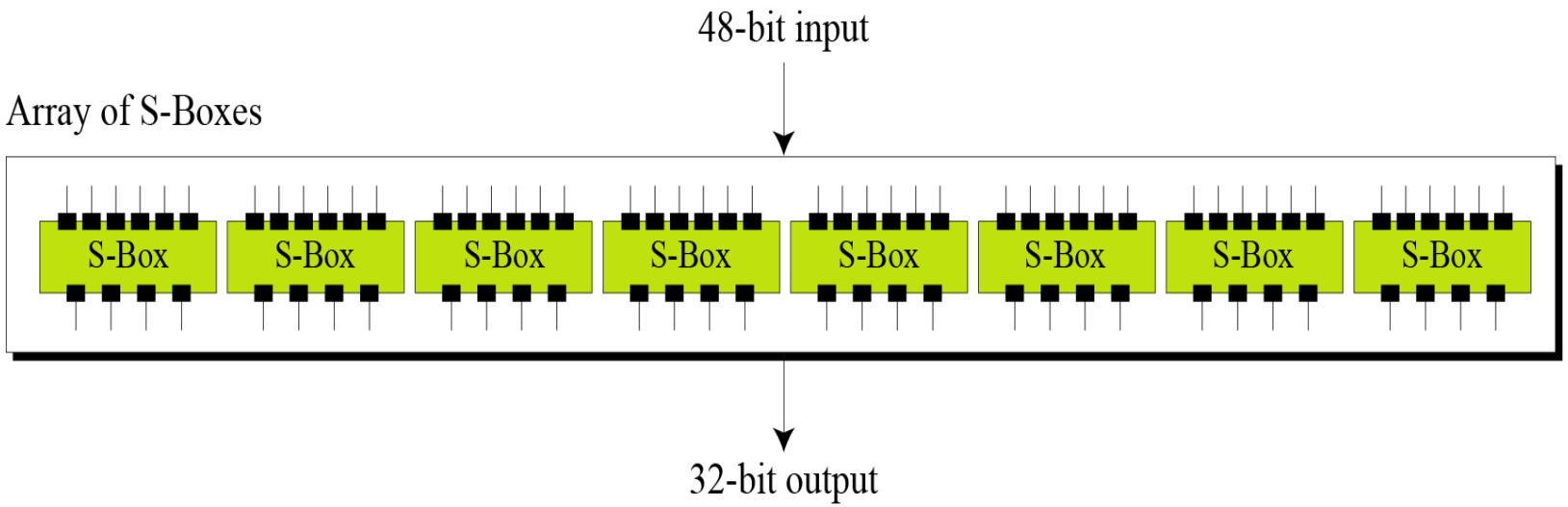
Expansion P-box table

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

S- Box

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

Figure of *S-boxes*

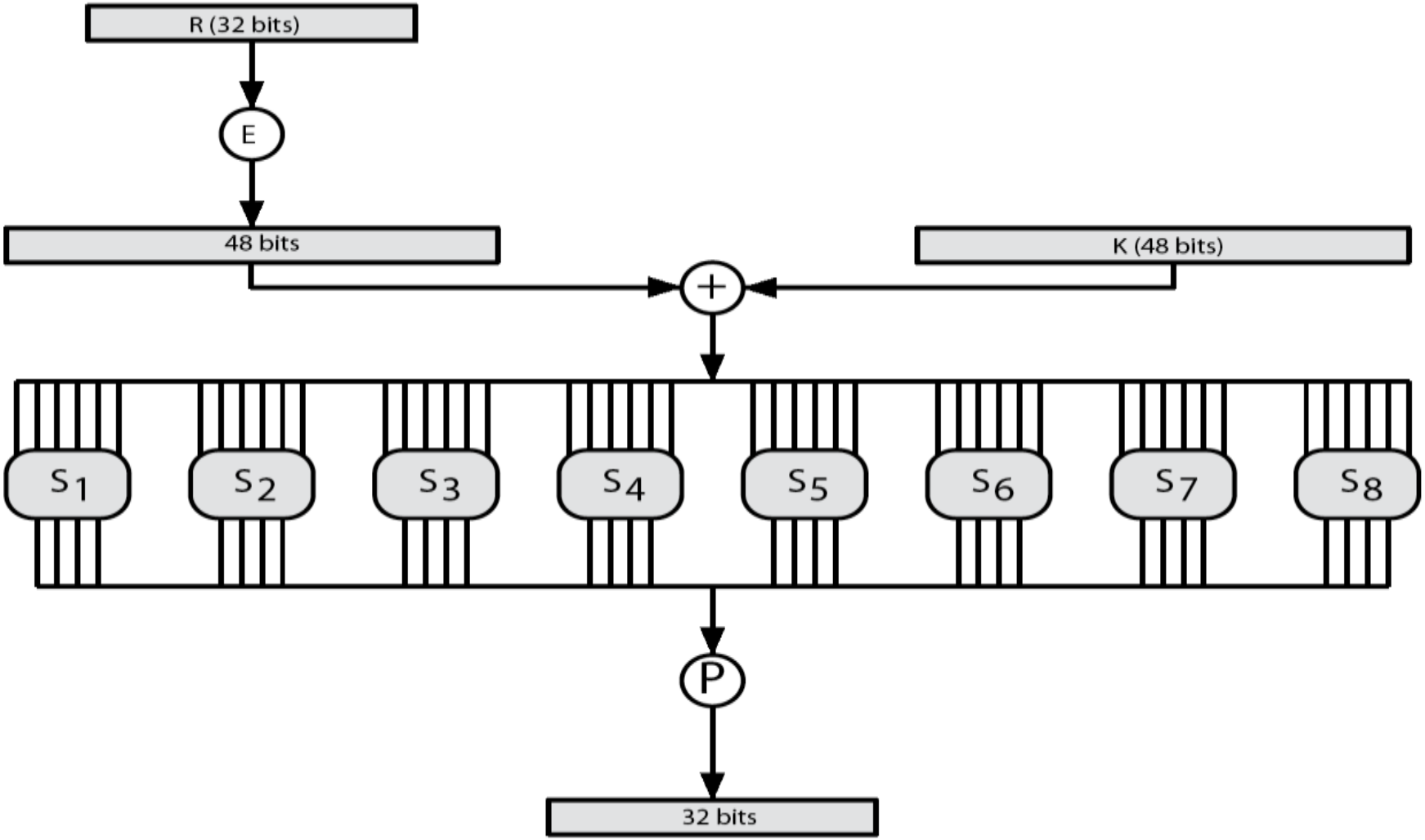


S-box

S-box1

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

DES Round Structure



Permutation table

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

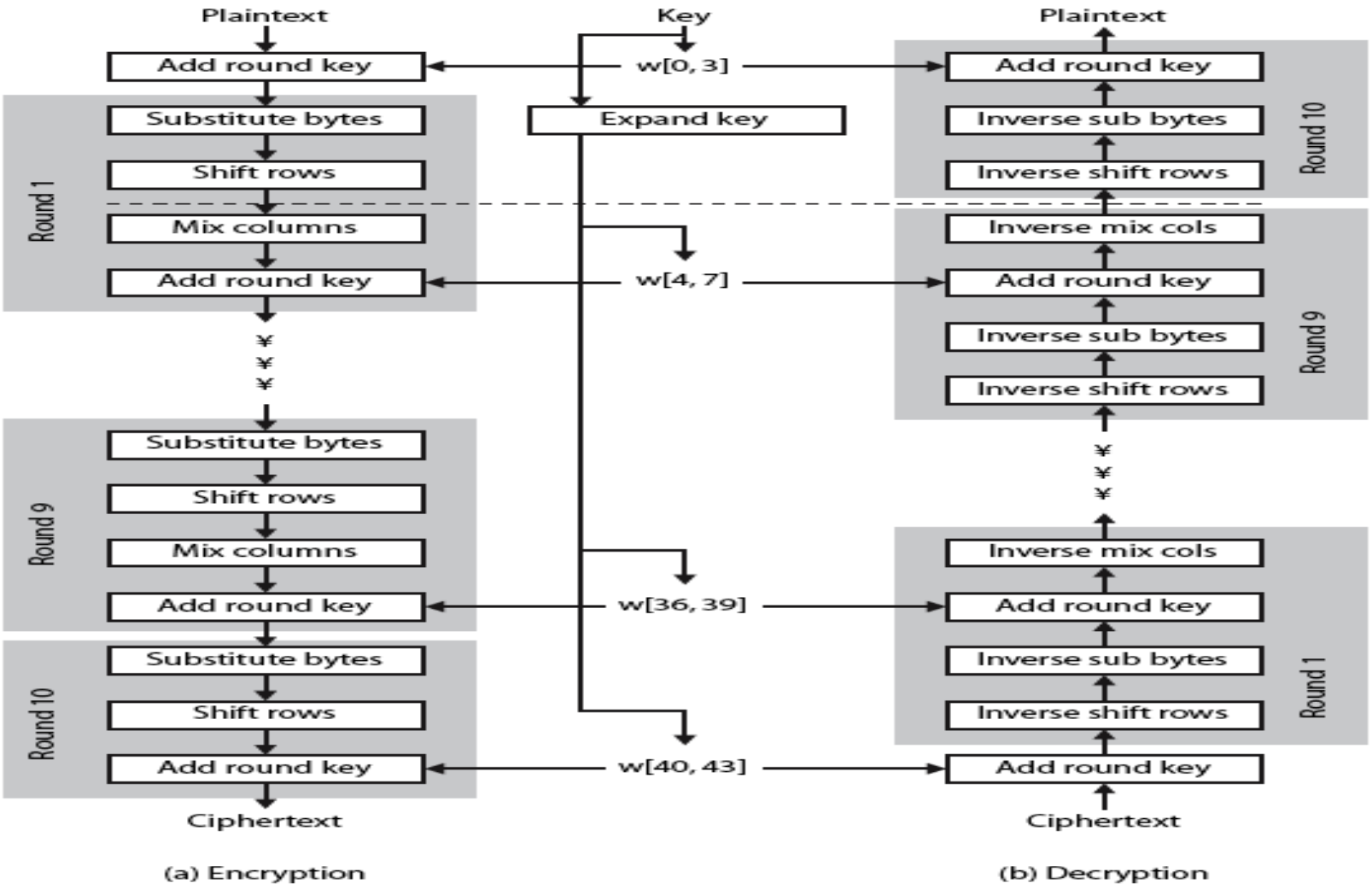
AES Alogorithm

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to have:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

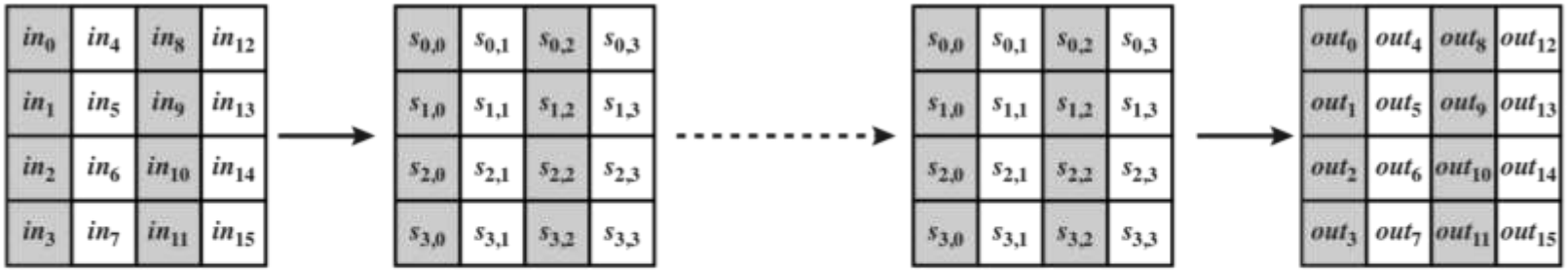
AES Alogorithm

- Block size=128 bit plaintext(4 words/16 bytes)
- No. of rounds= 10 rounds.
- Keysize=128 bits(4 words/16 bytes) : Size of 1 word= 32 bits.
- No.of subkeys= 44 subkeys.
- Each subkeysize=32 bit/ 1 word/ 4 bytes.
- Ecah round = 4 subkeys(128 bits/4 wortds/ 16 Bytes)
- Preround calculation= 4 subkeys(subkeys(128 bits/4 wortds/ 16 Bytes)
- Ciphertext=128 bits.

AES structure



AES Data Structures

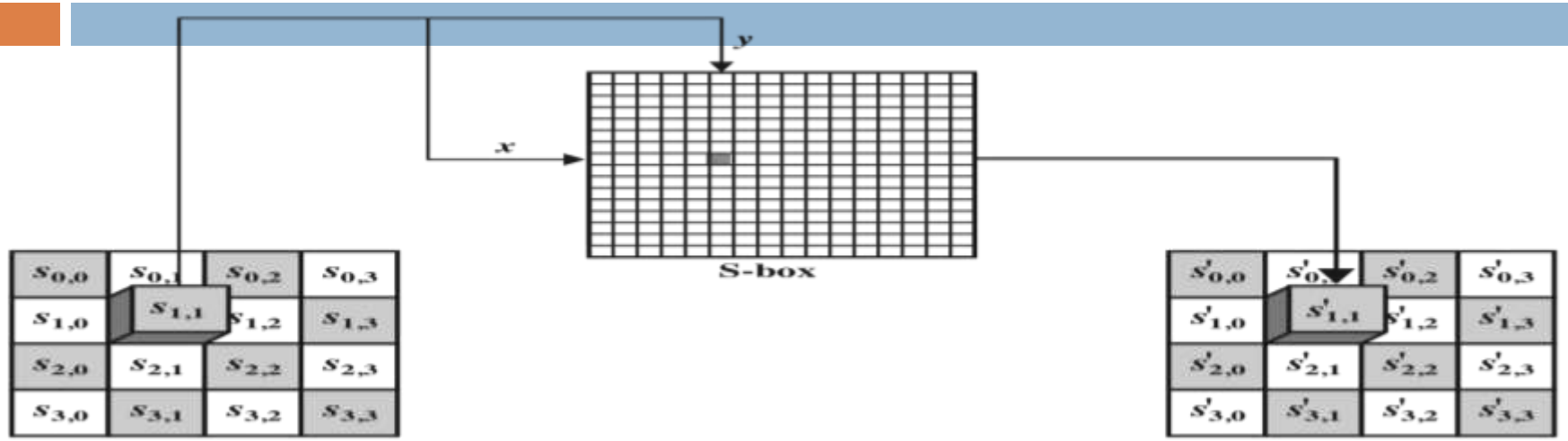


(a) Input, state array, and output

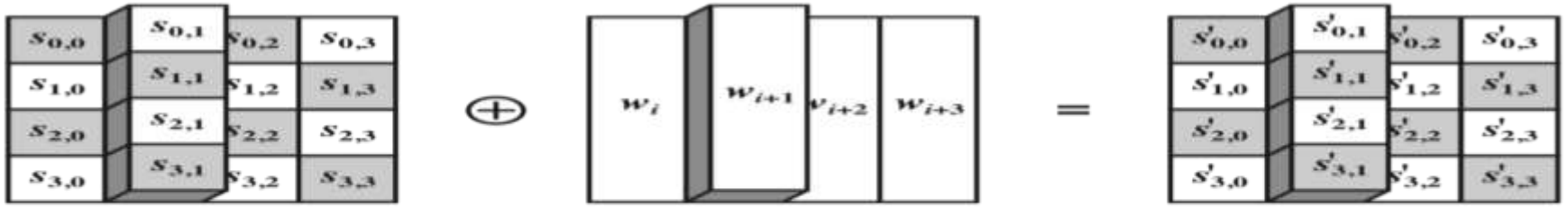
(b) Key and expanded key

Figure 5.2 AES Data Structures

Substitute Bytes



(a) Substitute byte transformation



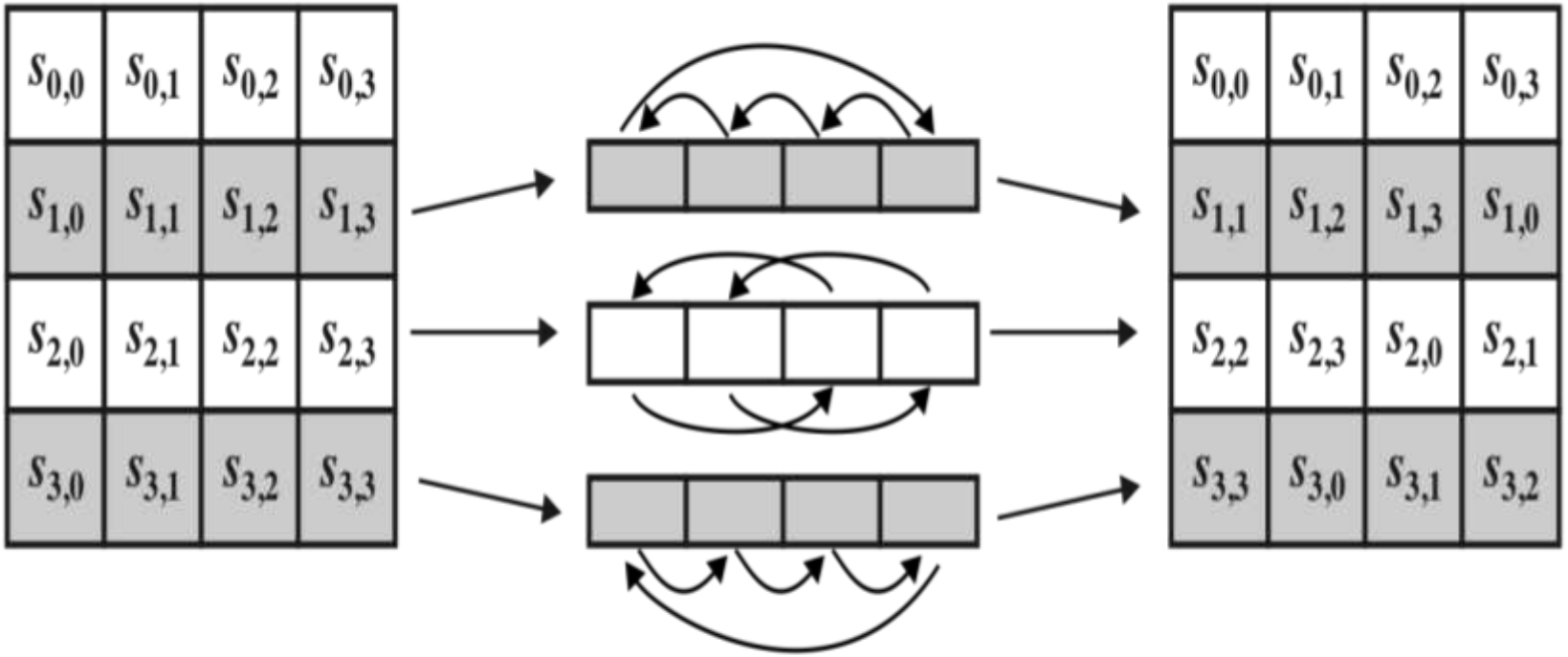
(b) Add round key Transformation

Figure 5.5 AES Byte-Level Operations

S-box

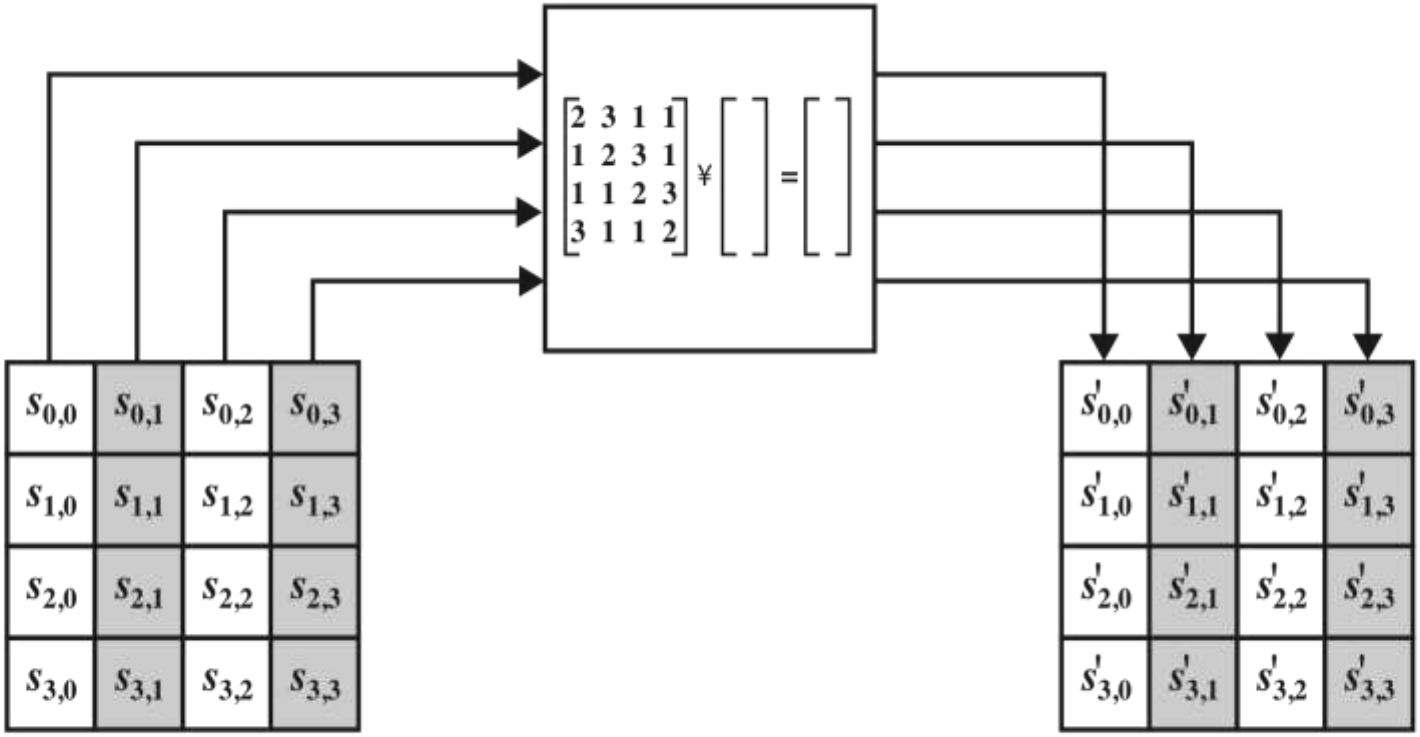
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Shift Row Transformation



(a) Shift row transformation

MixColumn Transformation



(b) Mix column transformation

RSA Algorithm(Asymmetric Key Encryption Algorithm)

Steps

- Consider two large prime numbers p,q
- Calculate n = p* q (n is product of two prime numbers).
- ϕ(n) = (p-1) * (q-1). [ϕ(n) is known as **Euler's totient function** , which counts the positive integers up to a given integer n that are relatively prime to 'n'].
- Select 'e'('e' is public key used in encryption side) such that gcd(e, ϕ(n)) =1. (that means 'e' should be *relatively prime* or *coprime* to ϕ(n))

RSA Algorithm(Asymmetric Key Encryption Algorithm)

- Select ‘d’ such that d*e mod ϕ(n)=1. (‘d’ is the private key, which is used in the decryption side).
- Now the public key is { e,n}
- Private key is {d,n}

Encryption

Consider plain text message ‘M’ whgich should be less than ‘n’.(M<n)

c =M^e mod n

Decryption

M=c^d mod n

RSA Algorithm(Asymmetric Key Encryption Algorithm)

Example:

P=3,q=5.

n = p*q =15.

$\phi(n) = (p-1) * (q-1) = 2 * 4 = 8$

$\phi(n)=8$.

- Select 'e' such that gcd(e, $\phi(n)$)=1. Here we can select e=3. According to RSA algorithm, **gcd(e, $\phi(n)$) must be equal to 1.** Here **3** satisfies the condition.
- Select 'd' such that d*e mod $\phi(n)$=1. Here we can select **d=3** . i.e **3* 3 mod 8=1.**

RSA Algorithm(Asymmetric Key Encryption Algorithm)

public key={e,n}={3,15}.

private key = { 3,15}

Select plain text M. Here M=4 . M<n. i.e = 4<15

Ciphertext c=M^e mod n

=4^3 mod 15

=64 mod 15 =4

so c=4

Decryption

M=c^d mod n

= 4^3 mod 15

M=4

RSA Algorithm(Asymmetric Key Encryption Algorithm

Example-2:

- Select primes p=11, q=3.
- n = pq = 11*3 = 33.
- $\phi(n) = (p-1) * (q-1) = 10 * 2 = 20$
- Choose 'e' = 3, which satisfies the condition $gcd(e, \varphi(n))=1$.
- Compute d such that d*e mod $\phi(n)=1$

  'd' = 7

- Public key = (e,n) = (3, 33)
  Private key = (d,n) = (7,33).

- Ciphertext c = M^e mod n **M=7** i.e 7^3 mod 33 = 343 mod 33 = 13. Hence the ciphertext ***c = 13.***
- M = c^d mod n = 13^7 mod 33 = 7.

Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that

a = bc mod n = (b mod n). (c mod n) mod n.

so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

One way of calculating M is as follows:-

Note that any number can be expressed as a sum of powers of 2. In particular 7 = 4 + 2 + 1.

So first compute values of 13^2 ,13^4, 13^8,... by repeatedly squaring successive values modulo 33.

13^1 mod 33=13

13^2 = 169 i.e 169 mod 33 = 4.(13^2 mod 33=4)

13^4 mod 33 = 4 * 4= 16

13^8 mod 33 = 16 * 16 =256 mod 33 = 25 etc…

Then, since 7 = 4 + 2 + 1, we have M = 13^7 = 13^(4+2+1) = 13^4 . 13^2 . 13^1=16 * 4 *13 = 832 mod 33= 7.

GCD of two numbers

- gcd(a,b) = gcd(b, a mod b) → [a mod b is the remainder when a is divided by b]
- Gcd(a,0) = a.

Example1:

Find gcd(3,8)

solution-:-

Here take a=3, b=8.

gcd(a,b) = gcd(b, a mod b)

gcd(3,8)=gcd(8 ,3 mod 8) =gcd(8,3) → since 3 mod 8 = 3.

GCD of two numbers

gcd(8,3) = gcd(3, 8mod 3)= gcd(3,2) → since 8 mod 3 = 2.

gcd(3,2) = gcd(2, 3 mod 2) = gcd(2,1) → since 3 mod 2 =1

gcd(2,1) = gcd(1, 2mod 1) = gcd(1,0) → 2 mod 1=0

gcd(1,0)=1 → since gcd(a,0) = a

so the gcd(3,8)=1.

Note: gcd of two numbers are 1 ,then those numbers are relatively prime.

GCD of two numbers

- gcd(a,b) = gcd(b, a mod b) → [a mod b is the remainder when a is divided by b]
- Gcd(a,0) = a.

Example:

Find gcd(8,5)

solution-:-

- Here take a=8, b=5.
- gcd(a,b) = gcd(b, a mod b)
- gcd(8,5)=gcd(5, 8 mod 5) =gcd(5,3) → since 8 mod 5 = 3.

GCD of two numbers

- gcd(5,3) = gcd(3, 5 mod 3)= gcd(3,2) → since 5 mod 3 = 2.
- gcd(3,2) = gcd(2,1)
- gcd(2,1) = gcd(1,0)
- gcd(1,0) =1. → since gcd(a,0)=a.
- **gcd(8,5)= 1.**

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

Important points

- Not an encryption algorithm
- Exchange secret/symmetric key between sender and receiver.
- To exchange key between sender and receiver asymmetric key encryption technique is used in Diffie Hellman key exchange. i.e public key and private key using.

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

Steps:

- Select prime number q
- Select α (alpha) such that α must be primitive root of q and $\alpha < q$.
- Assume X_A (Private key of user A) . $X_A < q$
- Calculate Y_A (public key of user A)
- $Y_A = \alpha^{\wedge} X_A \bmod q$.
- Assume X_B (private key of user B). $X_B < q$
- Calculate Y_B (public key of user B) i.e $Y_B = \alpha^{\wedge} X_B \bmod q$.

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

- $\{X_A , Y_A\}$ → private key , public key of user A
- $\{X_B , Y_B\}$ → private key, public key of user B.

Key generation(in sender ,receiver side)

user A | user B

$K=(Y_B)^{X_A} \bmod q = K=(Y_A)^{X_B} \bmod q$

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM (Example)

- q=11
- Select α , which is a primitive root of q.
- α is said to be the primitive root of q , if α^1 mod 11, α^2 mod 11, α^3 mod 11…… α^10 mod 11 gives the result = {1,2,3,4……10}.

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM (Example)

How to find primitive root of q ?

power →	1	2	3	4	5	6	7	8	9	10	(mod 11)
1	1	1	1	1	1	1	1	1	1	1	
2	2	4	8	5	10	9	7	3	6	1	
3	3	9	5	4	.....						
4											
5											
6											
7											
8											
9											
10											

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM (Example)

- Here $\alpha=2$.
- select X_A , which is less than q. Here $X_A = 8$ (private key)
- calculate $Y_A = \alpha \text{^} X_A \mod q. = 2\text{^}8 \mod 11 = 3$
- i.e $Y_A = 3.$
- Select X_B, which is less than q. Here $X_B = 4$ (private key)
- calculate $Y_B = = \alpha \text{^} X_B \mod q = 2\text{^}4 \mod 11$
- $Y_B = 5$
- User A $=\{X_A = 8, Y_{A=} 3\}$
- User B$= \{X_B = 4, Y_{B=} 5\}$
-

DIFFIE HELLMAN KEY EXCHANGE ALGORITHM (Example)

Sender (User A)

$K=(Y_B)^{X_A} \bmod q$

$=(5)^8 \bmod 11$

$=390625 \bmod 11$

$= 4$

Receiver(User B)

$K=(Y_A)^{X_B} \bmod q$

$= (3)^4 \bmod 11$

$= 81 \bmod 11$

$= 4$

Note that we don't have to calculate the full value of 5 to the power 8 here. We can make use of the fact that

$a = bc \bmod n = ((b \bmod n).(c \bmod n)) \bmod n.$

$5^8 = (5^2 . 5^2 . 5^4 \bmod 11) \bmod 11$

$5^2 \bmod 11 = 25 \bmod 11 = 3$

$(5^4 \bmod 11 = 625 \bmod 11 = 9$

$5^8 = (5^2 \cdot 5^2 \cdot 5^4 \bmod 11) \bmod 11$

So $(3 \cdot 3 \cdot 9) \bmod 11 = 81 \bmod 11 = 4$