Topics Covered

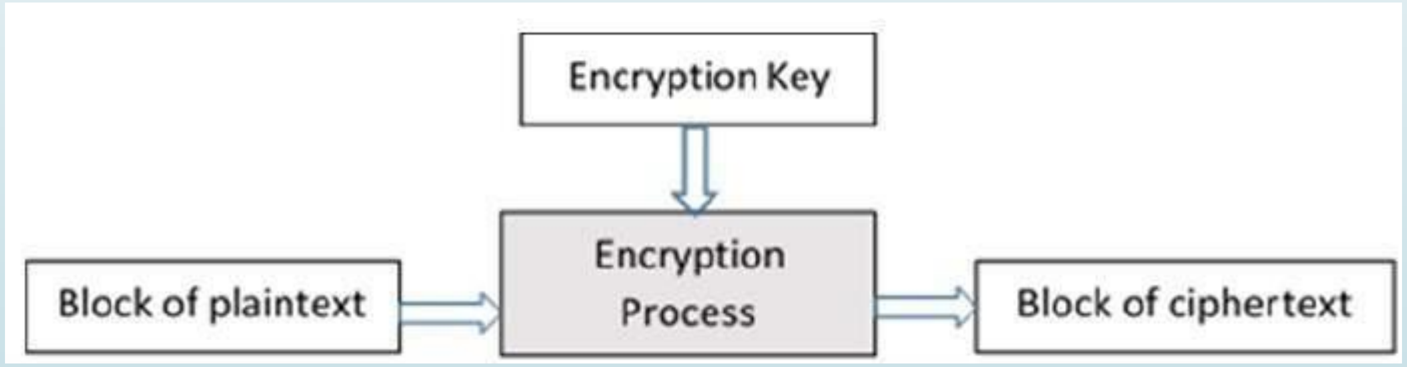
- **BLOCK CIPHER PRINCIPLES**
  - Stream ciphers and Block ciphers
  - Motivation for The Feistel Cipher Structure
  - The Feistel Cipher
- **THE DATA ENCRYPTION STANDARD**
  - Encryption
  - Decryption
  - The Avalanche Effect
- **THE STRENGTH OF DES**
  - The use of 56-Bit keys
  - The nature of DES algorithm
  - Timing attacks
- **TRIPLE DES**
  - Triple DES with 2 keys and 3 keys
- **DIFFERENTIAL AND LINEAR CRYPTANALYSIS**

GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)

BLOCK CIPHER PRINCIPLES

BLOCK CIPHER PRINCIPLES

- All symmetric block cipher algorithms are based on a structure known as **Feistel block cipher.**
- For that reason, it is important to understand the design principles of Feistel cipher.



STREAM CIPHERS AND BLOCK CIPHERS

- **Stream Cipher**
- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.
- Examples of classical stream ciphers are:
  a) the auto keyed Vigenère cipher and
  b) the Vernam cipher.
- If the cryptographic key stream is random, then this cipher is unbreakable by any means other than acquiring the key stream.

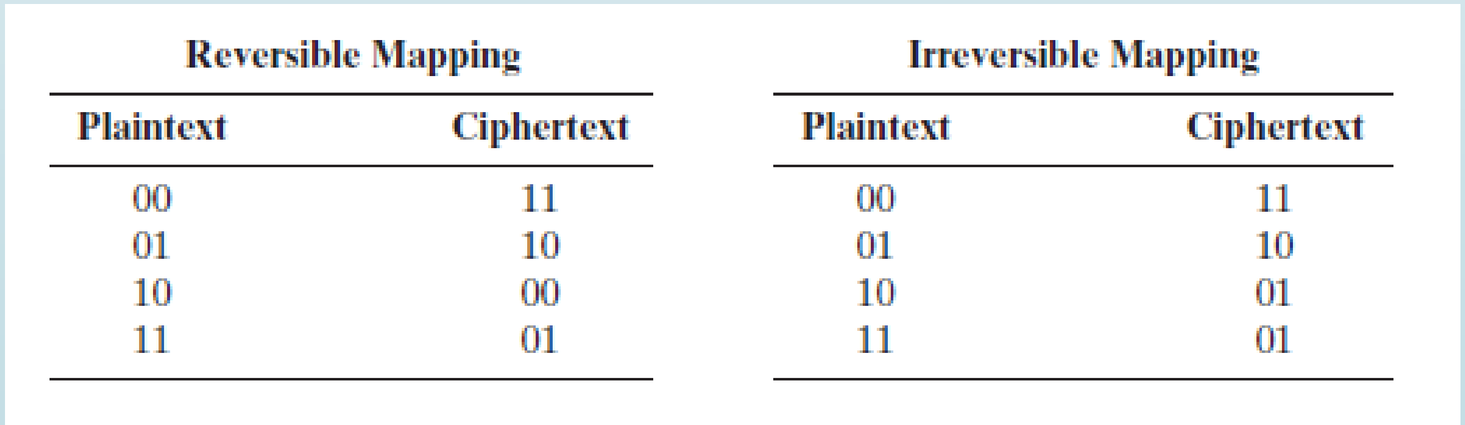
GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)

Stream Ciphers and Block Ciphers (conti..)

- **Block Ciphers**
- 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.
- A block size of 64 or 128 bits is used.
- The vast majority of network-based symmetric cryptographic applications make use of block ciphers.

MOTIVATION FOR THE FEISTEL CIPHER STRUCTURE

- A block cipher operates on a plaintext block of ***n* bits** to produce a ciphertext block of ***n* bits**.
- There are 2^n possible different plaintext blocks for the encryption to be reversible, each must produce a unique ciphertext block. Such a transformation is called **reversible or nonsingular**.
- Nonsingular and singular transformations for *n* = 2.
- If we limit ourselves to reversible mappings, the number of different transformations is $2^n!$



Reversible Mapping

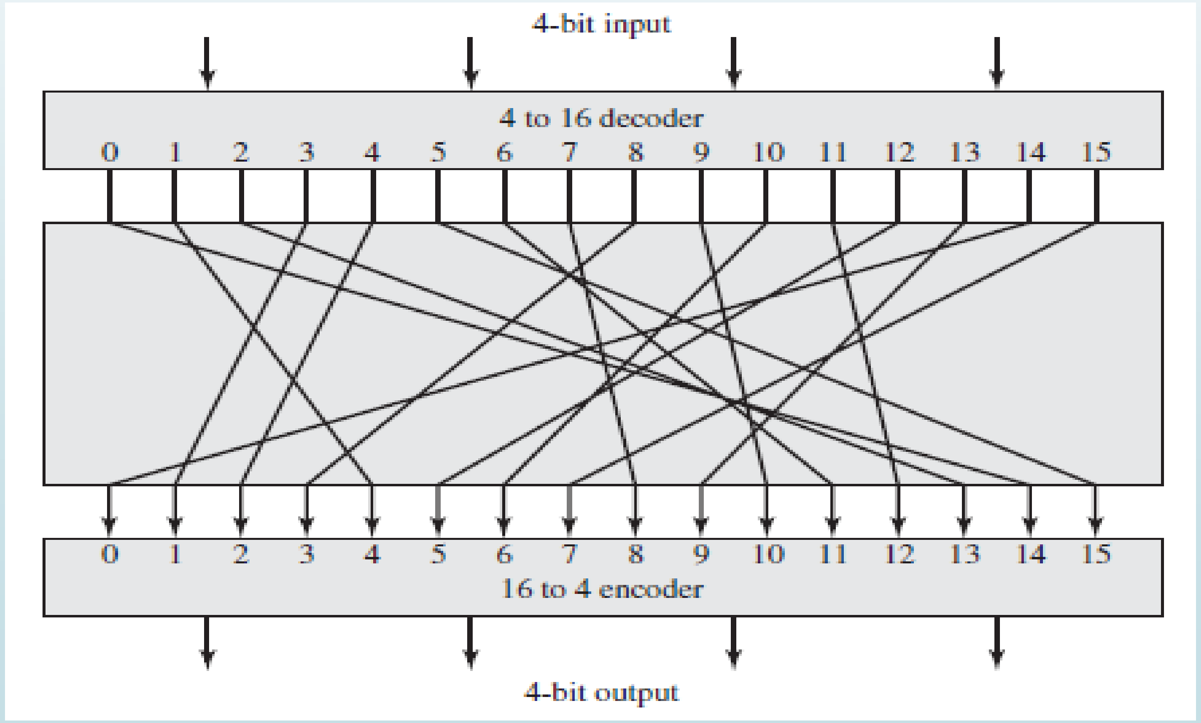
Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

Motivation for the Feistel Cipher Structure

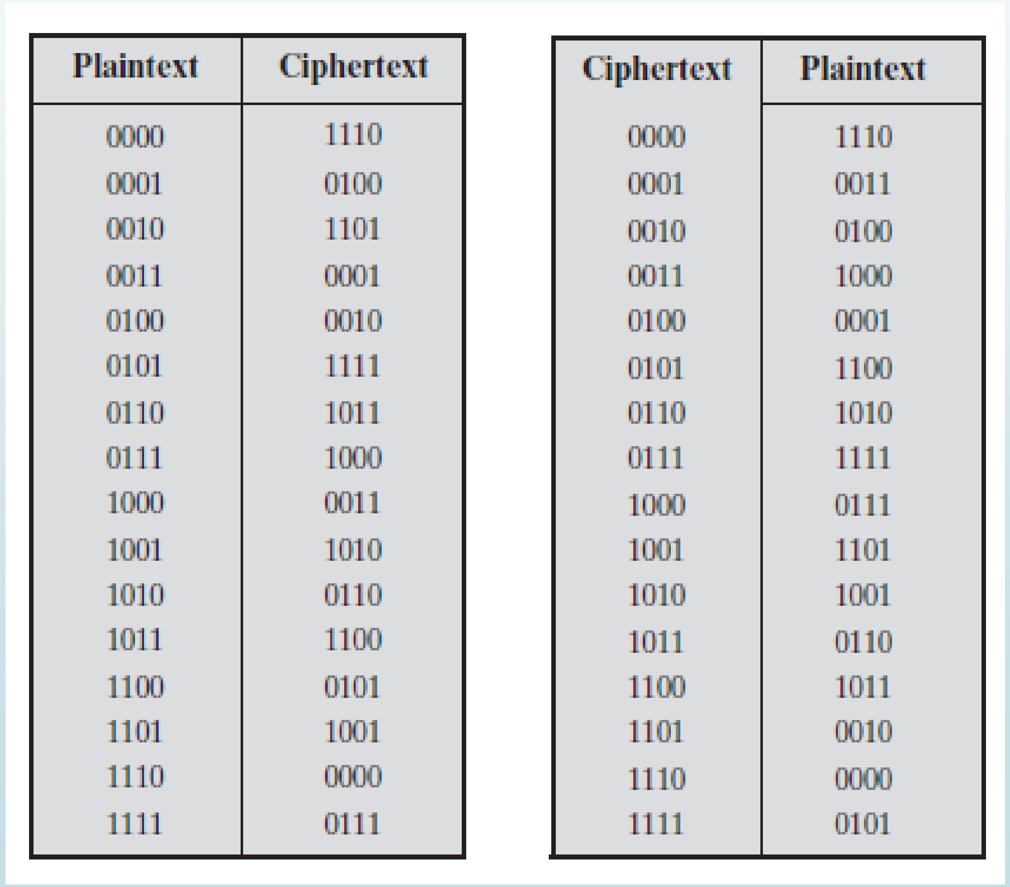
- General substitution cipher for n = 4.
- A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits.



GITAM (DEEMED TO BE UNIVERSITY) (Estd. u/s 3 of the UGC Act, 1956)

Motivation for the Feistel Cipher Structure

- Feistel refers to this as the *ideal block cipher*, because it allows for the maximum number of possible encryption mappings from the plaintext block.



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

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

Motivation for the Feistel Cipher Structure

- There is a **practical problem** with the ideal block cipher.
- If a small block size, such as **n = 4**, is used then the system is equivalent to a classical substitution cipher.
- Such systems are vulnerable to a **statistical analysis** of the plaintext.
- If **n** is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is **infeasible**.

THE FEISTEL CIPHER

- Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher.
- The essence of the approach is to develop a block cipher with a **key length of *k* bits** and a **block length of n bits**, allowing a total of **2^k possible transformations**, rather than the $2^n!$ transformations available with the ideal block cipher.

The Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations:
  - **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
  - **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order of sequence is changed.

- Feistel Cipher is a practical application of a proposal by Claude Shannon in 1945 to develop a product cipher that alternates *confusion* and *diffusion* functions.

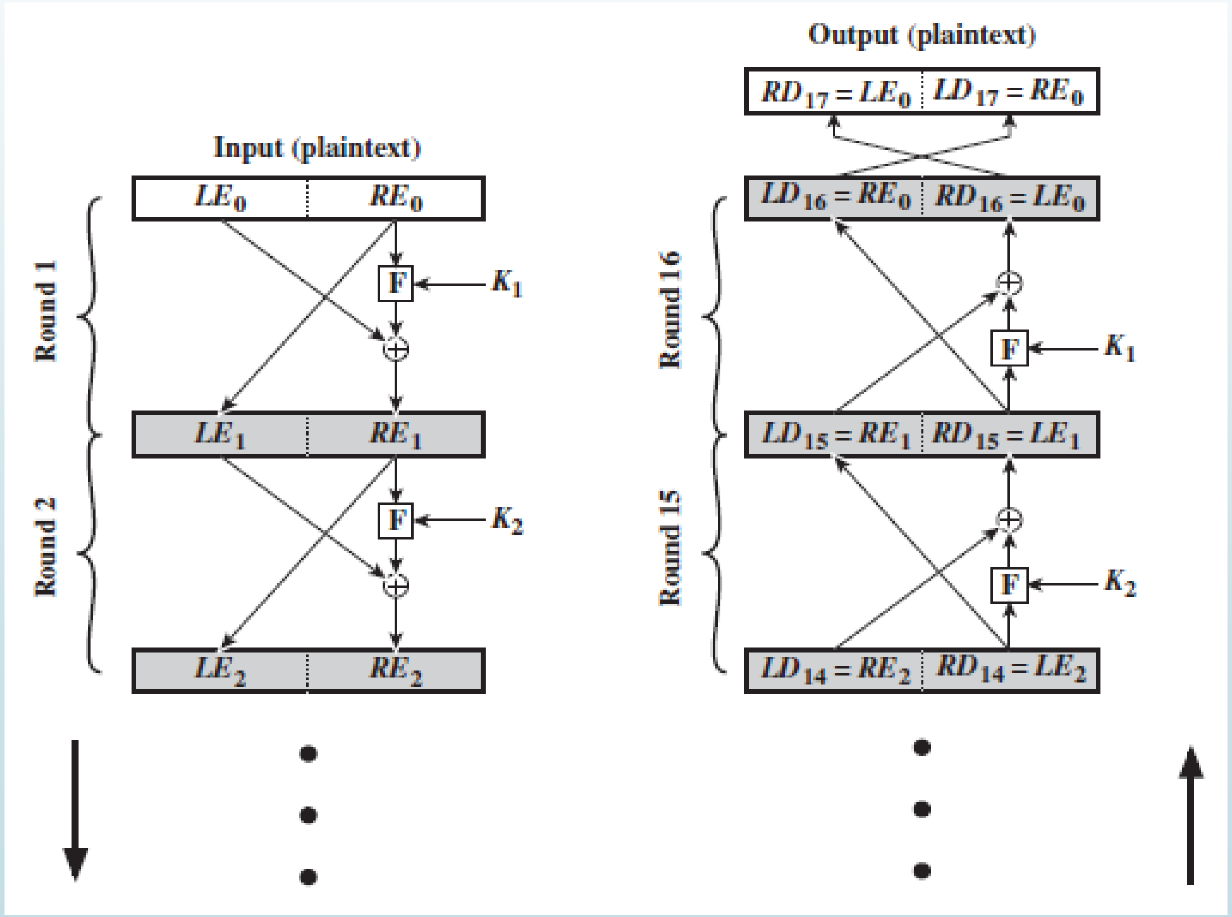
The Feistel Cipher

- **DIFFUSION and CONFUSION**
  - methods for frustrating statistical cryptanalysis.

- **Diffusion -** the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext.
- **Confusion -** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible.

The Feistel Cipher

- **FEISTEL CIPHER STRUCTURE**



The Feistel Cipher

- The inputs to the encryption algorithm are :
  - a plaintext block of 2w length bits and a key K.
- Plaintext block is divided into two halves, L_0 and R_0 .
- The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block.
- All rounds have the same structure.
- A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data.
- Following this substitution, a **permutation** is performed that consists of the interchange of the two halves of the data.
- This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)

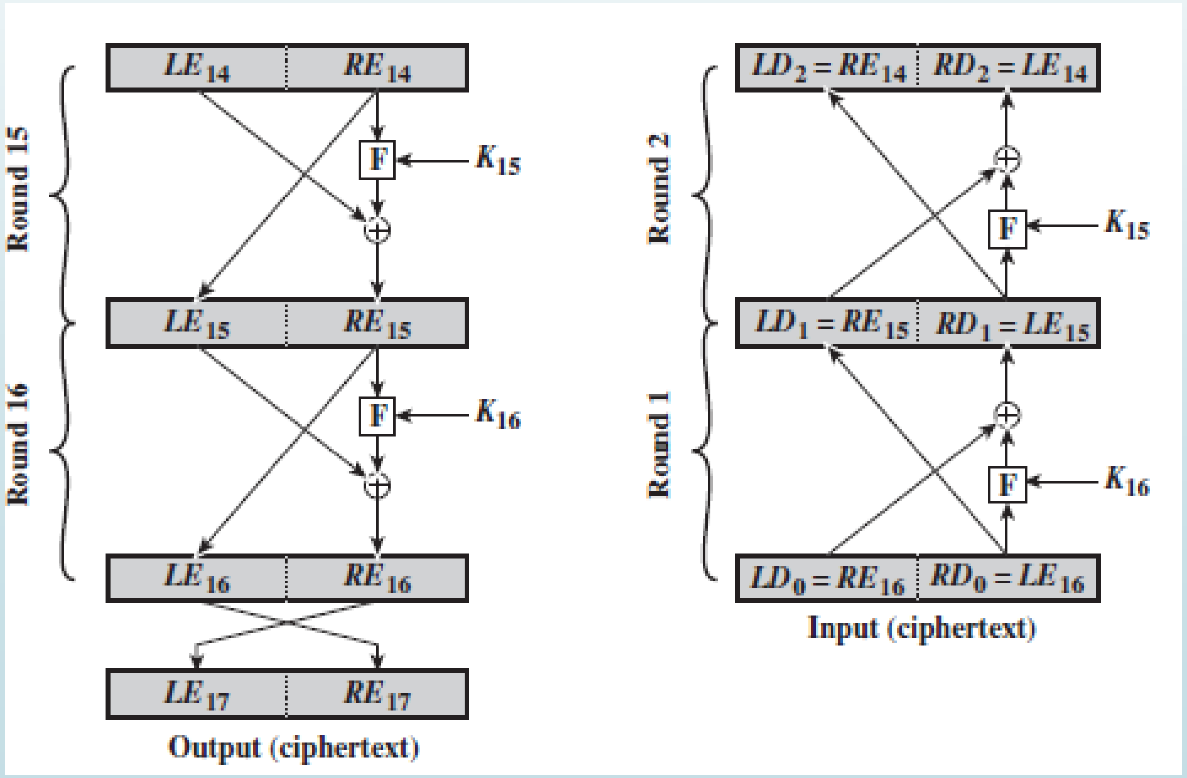
The Feistel Cipher

- **The exact realization of a Feistel network depends on the choice of the following parameters and design features:**

1. **Block size:** Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. AES uses a 128 bit block size.
2. **Key size:** Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. 128 bits has become a common size.
3. **Number of rounds:** Multiple rounds offer increasing security. A typical size is 16 rounds.
4. **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
5. **Round function F:** Greater complexity generally means greater resistance to cryptanalysis.

The Feistel Cipher

- **FEISTEL DECRYPTION ALGORITHM**
- The rule is as follows:
- Use the ciphertext as input to the algorithm, but use the subkeys K_i in reverse order.



GITAM (DEEMED TO BE UNIVERSITY) (Estd. u/s 3 of the UGC Act, 1956)

The Feistel Cipher

- Use the notation **LE_i and RE_i** for data traveling through the **encryption** algorithm and **LD_i and RD_i** for data traveling through the **decryption** algorithm.
- Consider the encryption process. We see that,

$$\begin{aligned} LE_{16} &= RE_{15} \\ RE_{16} &= LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$

- On the decryption side,

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$

The Feistel Cipher

- The XOR has the following properties:

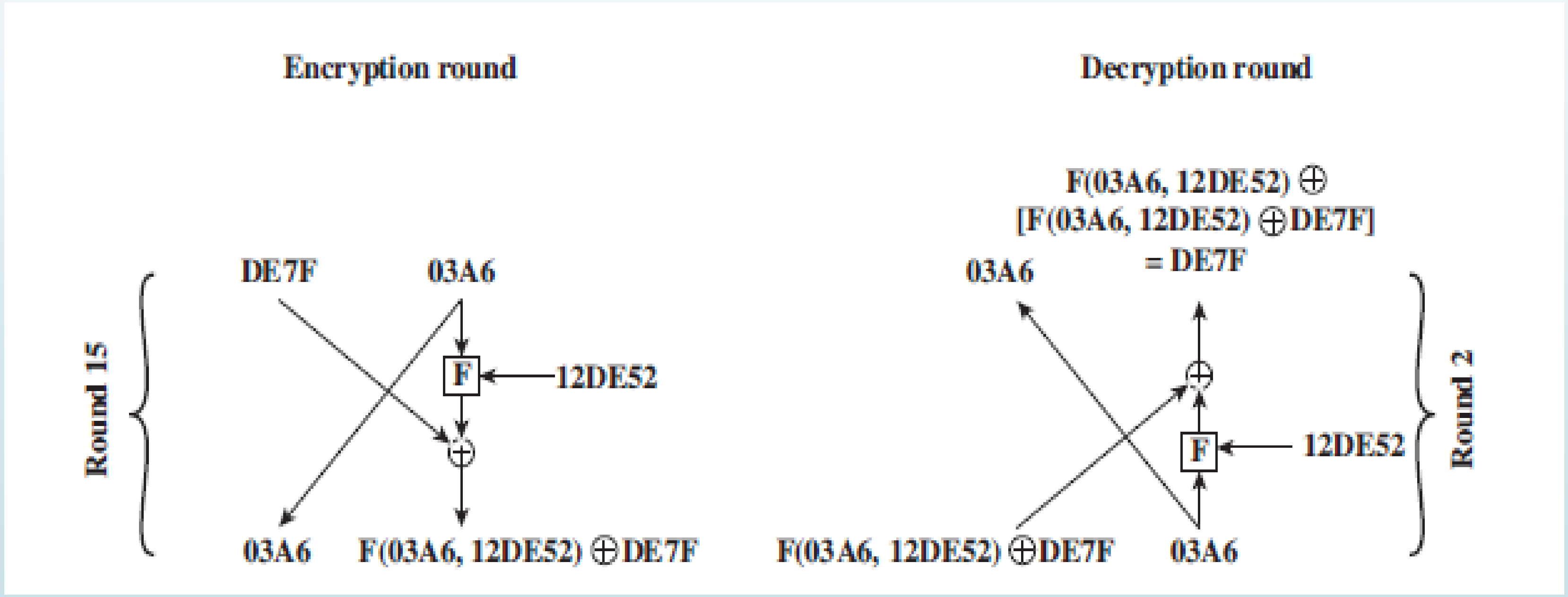
$$
\begin{aligned}
[A \oplus B] \oplus C &= A \oplus [B \oplus C] \\
D \oplus D &= 0 \\
E \oplus 0 &= E
\end{aligned}
$$

- We have $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$.
- The output of the first round of the decryption process is $RE_{15} \| LE_{15}$, which is the 32-bit swap of the input to the sixteenth round of the encryption.
- the *i*th iteration of the encryption algorithm,

$$
\begin{aligned}
LE_i &= RE_{i-1} \\
RE_i &= LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, K_i)
\end{aligned}
$$

The Feistel Cipher

- **Example**



The Feistel Cipher

- Suppose that the blocks at each stage are 32 bits (two 16-bit halves) and that the key size is 24 bits.
- **Encryption:**
- Suppose that at the end of encryption round fourteen, the value of the intermediate block in hexadecimal is DE7F03A6.

LE_{14} = DE7F $\quad\quad$ RE_{14} = 03A6

- Assume that key K_{15} is 12DE52.
- After round 15, we have,

LE_{15} = 03A6

RE_{15} = F(03A6, 12DE52) XOR DE7F

The Feistel Cipher

- Decryption:
- Assume

  $LD_1 = RE_{15}$ & $RD_1 = LE_{15}$

- **Demonstration of $\mathbf{LD_2 = RE_{14}}$ and $\mathbf{RD_2 = LE_{14}}$**
- Start with

  LD_1 = F(03A6, 12DE52) XOR DE7F

  RD_1 = 03A6

- From diagram it is clear that,

  LD_2 = 03A6 = RE_{14}

  RD_2 = F(03A6,12DE52) XOR [F(03A6, 12DE52) XOR DE7F

  = DE7F

  = LE_{14}

The Data Encryption Standard

DES – Data Encryption Standard

- A block cipher
- The algorithm is referred as the Data Encryption Algorithm (DEA).
- Data are encrypted in 64-bit blocks using a 56-bit key.
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output.
- The same steps, with the same key, are used to reverse the encryption.

DES History

- In the late 1960s, IBM set up a research project in computer cryptography led by **Horst Feistel**.
- The project was concluded in 1971 with the development of an algorithm known as **LUCIFER**.
- LUCIFER is a Feistel block cipher that operates on blocks of 64 bits, using a key size of 128 bits.
- **Walter Tuchman and Carl Meyer**, introduced a refined version of LUCIFER that was more resistant to cryptanalysis and had a **reduced key size of 56 bits**, in order to fit on a single chip.
- In 1973, the National Bureau of Standards (NBS) issued a request for proposals for a national cipher standard, Tuchman–Meyer's project was by far the best algorithm proposed and was **adopted in 1977** as the **Data Encryption Standard**.

DES Design Controversy

- There has been considerable controversy over design
  - – in choice of 56-bit key (vs Lucifer 128-bit)
  - – and because design criteria were classified
- Subsequent events and public analysis show in fact design was appropriate.
- DES has become widely used, especially in financial applications.
- Best known and widely used symmetric algorithm in the world.
- But, no longer is considered secure for highly sensitive applications.

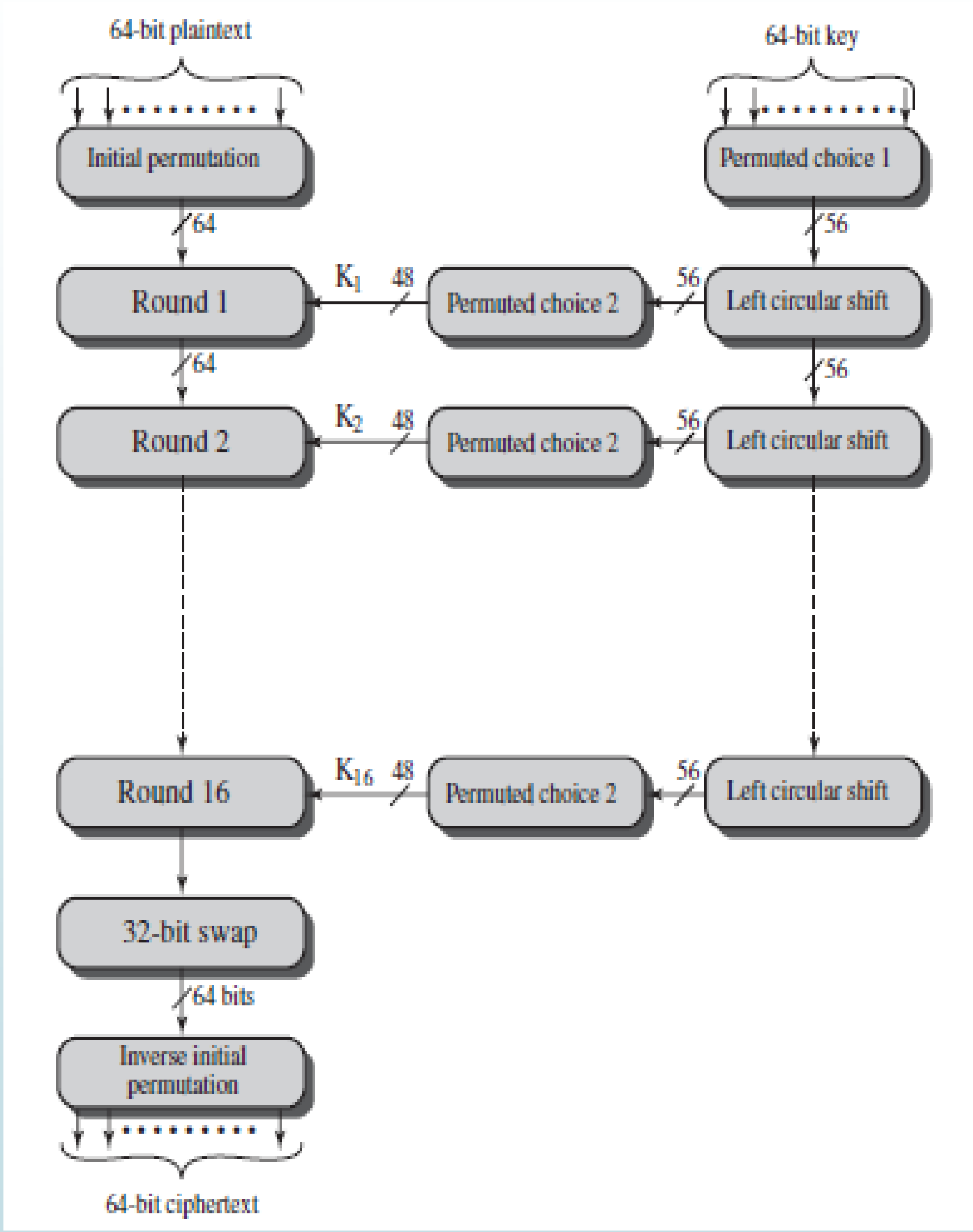
DES Encryption

- **Two inputs to the encryption function:**
  - the plaintext to be encrypted(64 bits) & the key (56 bits)
- **Left Hand Portion**
- **Processing of the plaintext proceeds in three phases:**

1. the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*.
2. sixteen rounds of the same function, which involves both permutation and substitution functions.
3. output of the 16th round consists of 64 bits that are a function of the input plaintext and the key.

The left and right halves of the output are swapped to produce the **preoutput**.

Finally, the preoutput is passed through an inverse of the initial permutation function, to produce the 64-bit ciphertext.

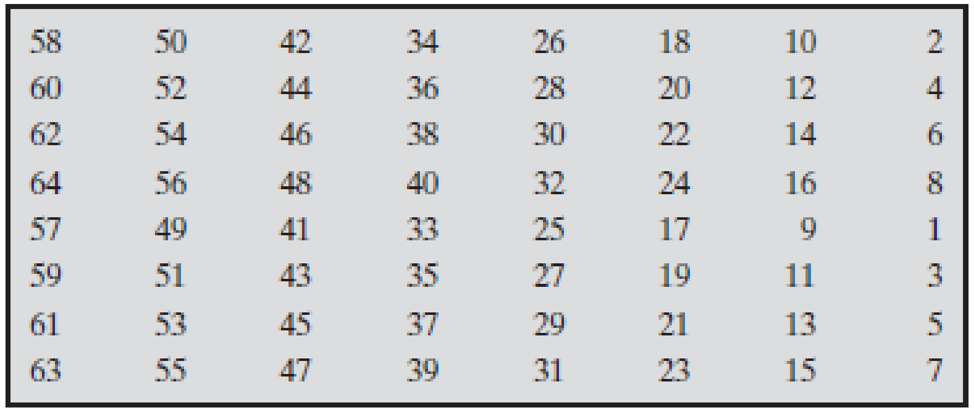


DES Encryption

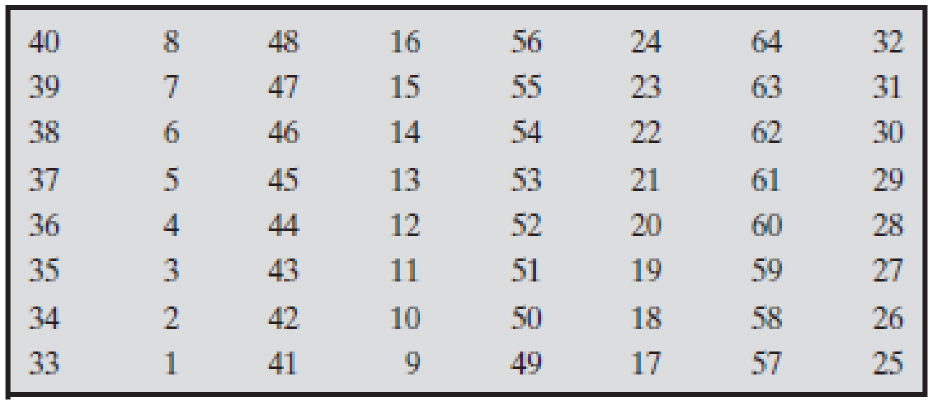
- **Right Hand Portion**
- 56-bit key is used.
- Initially, the key is passed through a permutation function
- Then, for each of the sixteen rounds, a *subkey* (*Ki*) is produced by the combination of a left circular shift and a permutation.
- The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

DES Encryption

- **INITIAL PERMUTATION**
- The initial permutation and its inverse are defined by tables.
- The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.


(a) Initial Permutation (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


(b) Inverse Initial Permutation (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

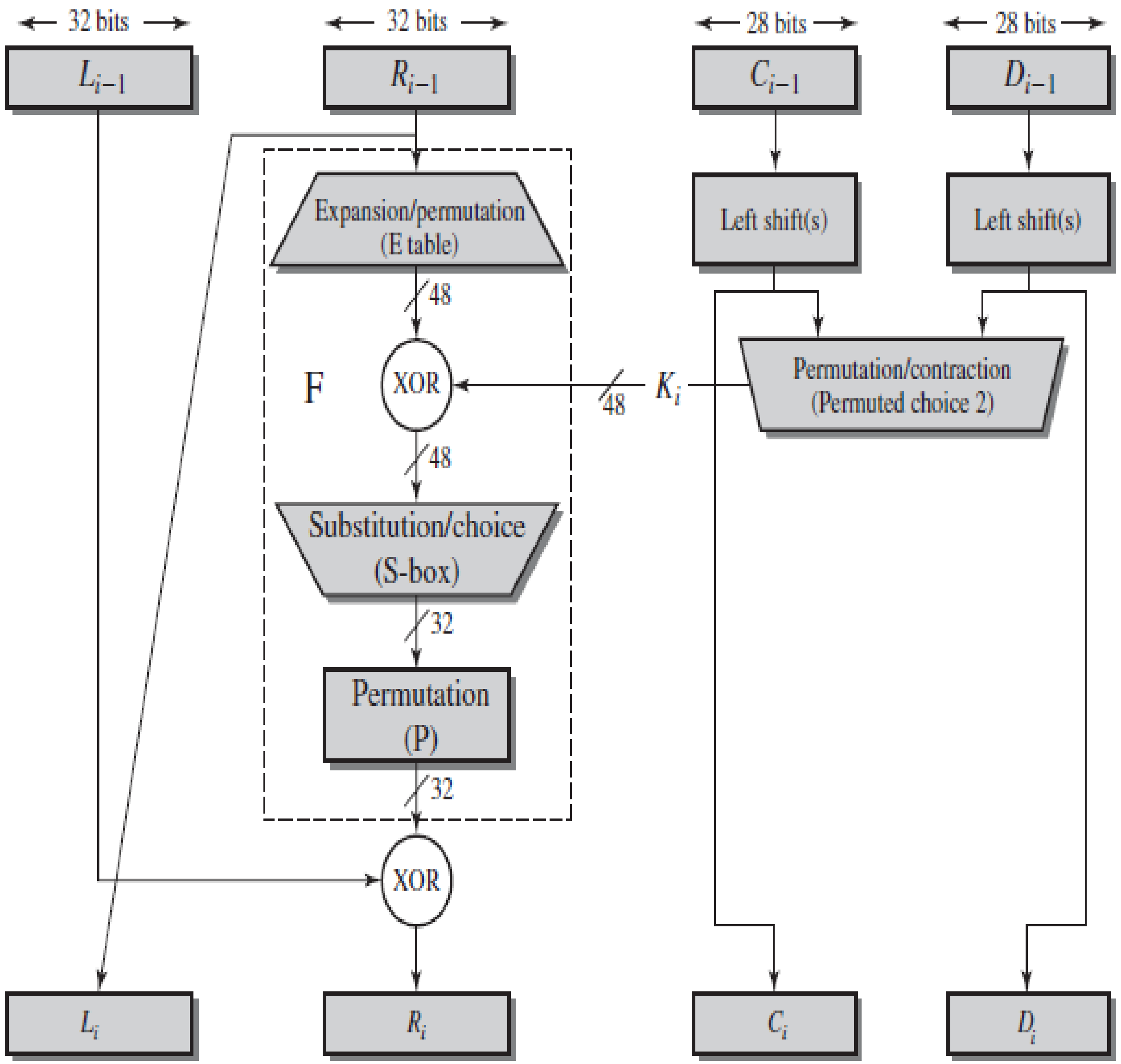
DES Encryption

- **DETAILS OF SINGLE ROUND**
- Internal structure of a single round.
- The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

- The round key K_i is 48 bits.
- The R input is 32 bits.

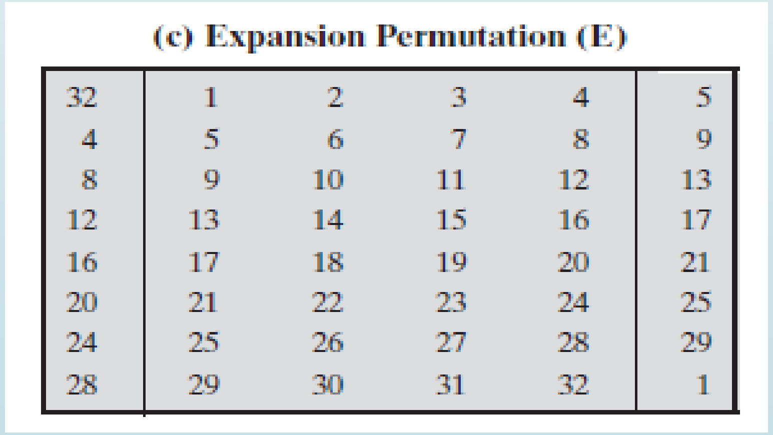


DES Encryption

- R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits.

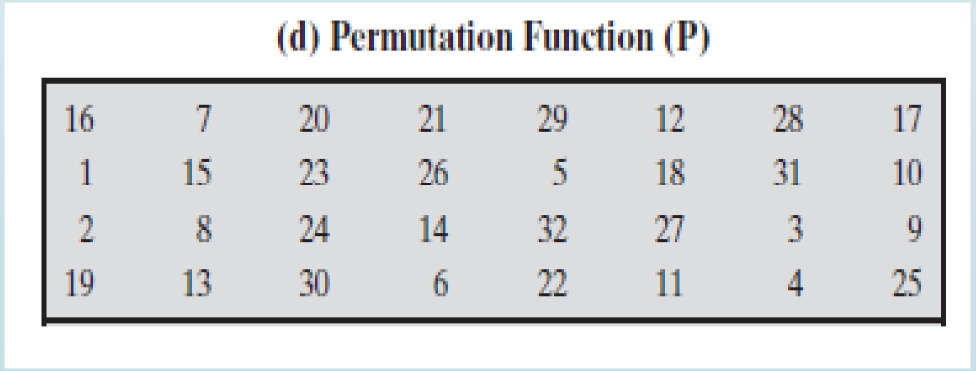
The resulting 48 bits are XORed with K_i.

This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as:



(c) Expansion Permutation (E)

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

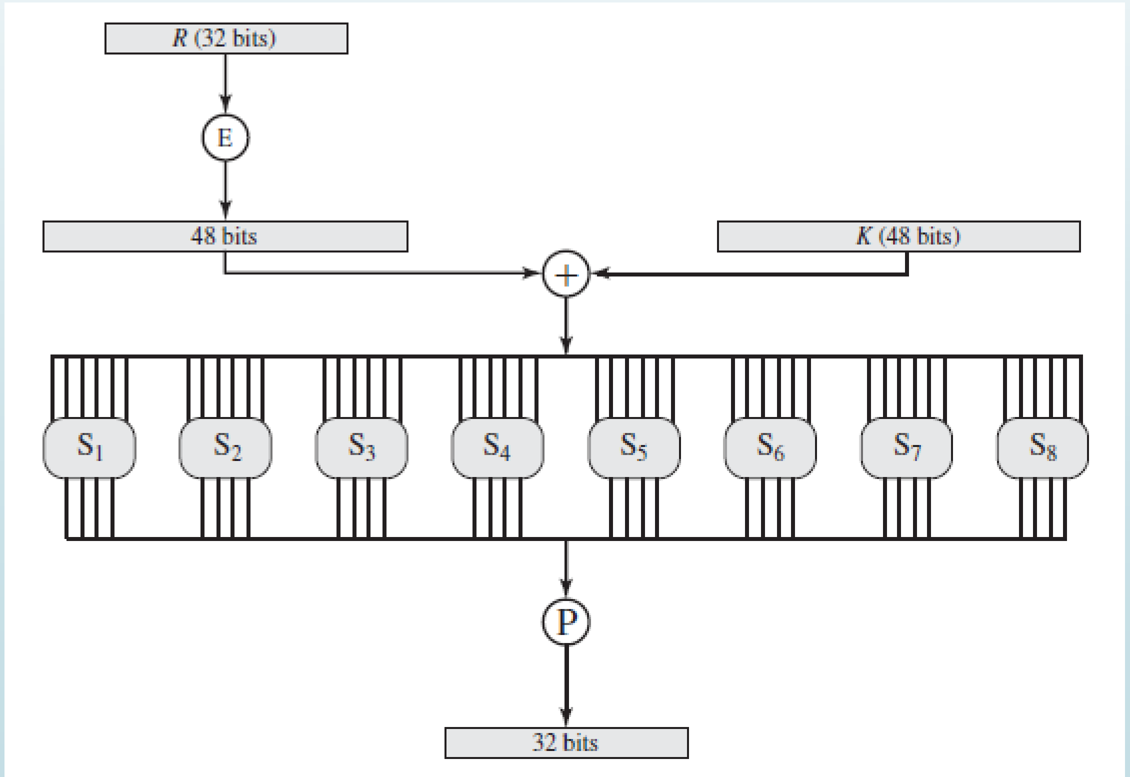


(d) Permutation Function (P)

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

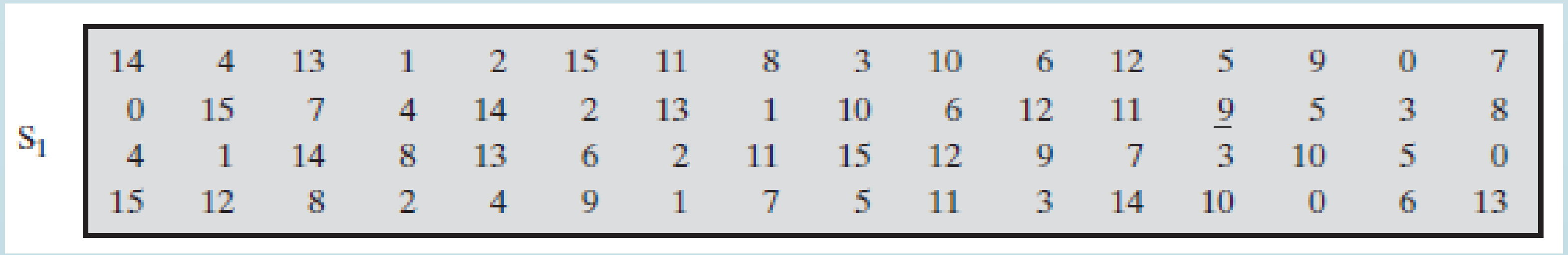
DES Encryption

- **S-Boxes**
- The role of the S-boxes in the function F is:



DES Encryption

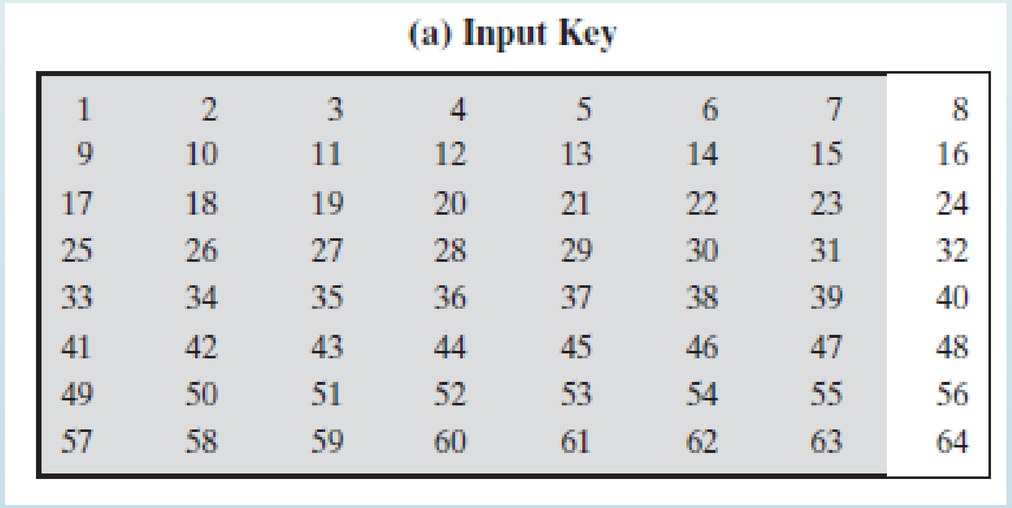
- The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.
- **Interpretation of S-box:**
- The first and last bits of the input to box S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for $S_{i.}$
- The middle four bits select one of the sixteen columns.
- The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output.
  - Example:
  - In S1, for input **011001**, the row is **01** (row 1) and the column is **1100** (column 12).
  - The value in row 1, column 12 is 9, so the output is 1001.



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	$\underline{9}$	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

DES Encryption

- **KEY GENERATION**
- 64-bit key is used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading.

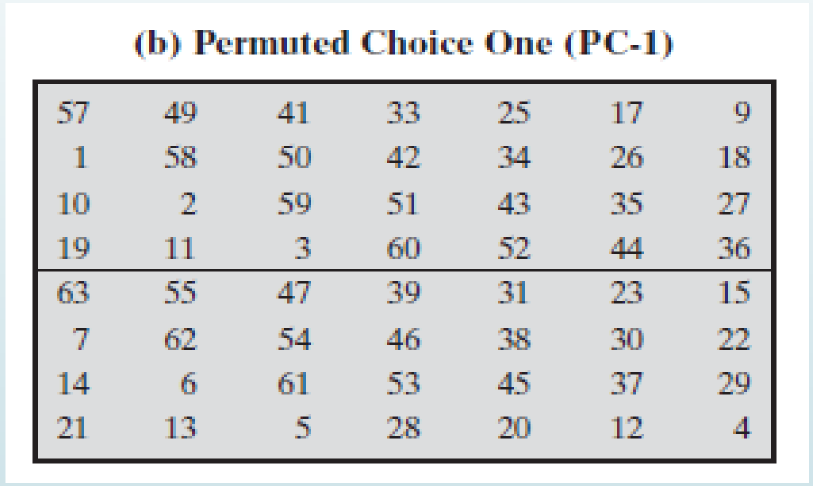


(a) Input Key

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

DES Encryption

- The key is first subjected to a permutation governed by table:

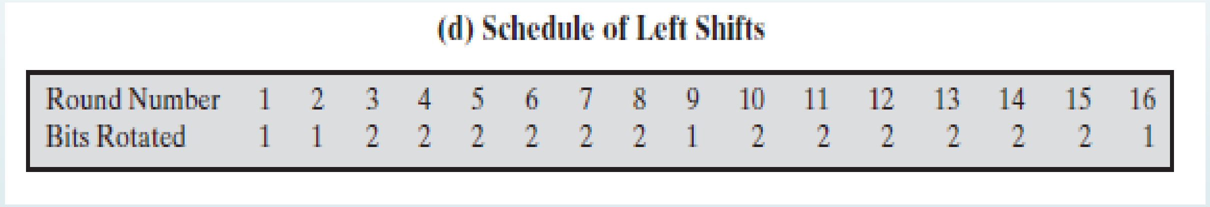


(b) Permuted Choice One (PC-1)

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

DES Encryption

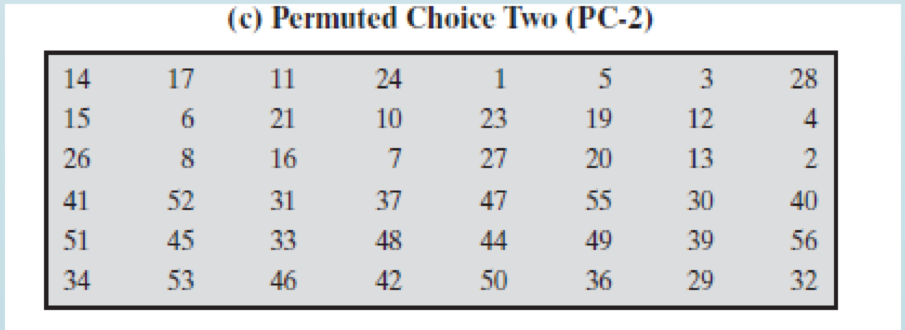
- The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0.
- At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by:



(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- These shifted values serve as input to the next round.
- They also serve as input to the part labeled Permuted Choice Two, which produces a 48-bit output that serves as input to the Function $F(R_{i-1}, K_i)$.



(c) Permuted Choice Two (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

DES Decryption

- As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

DES Example

The plaintext is a hexadecimal palindrome:

Plaintext:	**02468aceeca86420**
Key:	**0f1571c947d9e859**
Cipher text:	**da02ce3a89ecac3b**

Avalanche effect

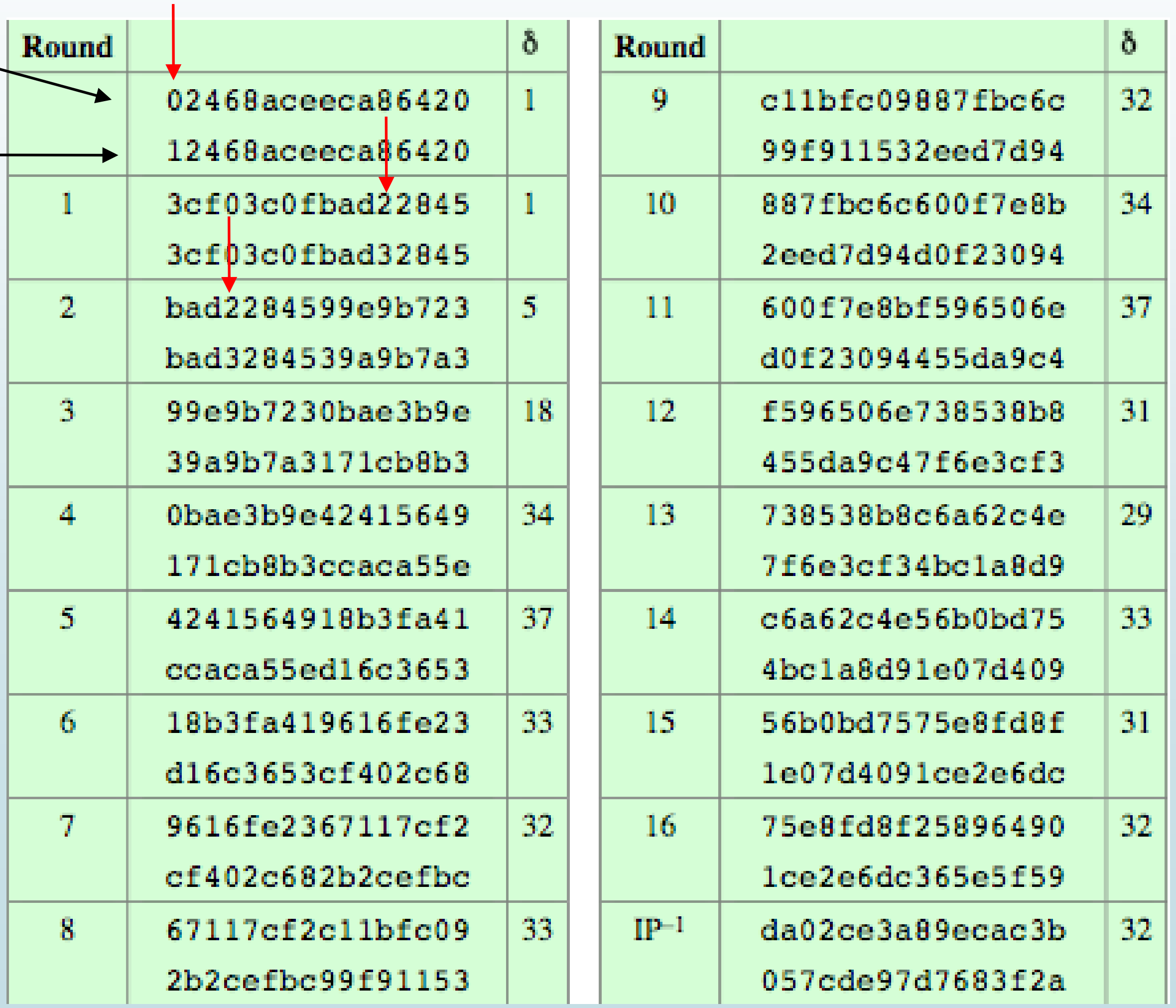
Avalanche Effect

- Key desirable property of encryption algorithm
- **Avalanche Effect** : a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.
- DES exhibits strong avalanche.

Avalanche Effect in DES: Change in Plaintext

original

modified



Round		δ	Round		δ
	02468aceeca86420	1	9	c11bfc09887fbc6c	32
	12468aceeca86420			99f911532eed7d94	
1	3cf03c0fbad22845	1	10	887fbc6c600f7e8b	34
	3cf03c0fbad32845			2eed7d94d0f23094	
2	bad2284599e9b723	5	11	600f7e8bf596506e	37
	bad3284539a9b7a3			d0f23094455da9c4	
3	99e9b7230bae3b9e	18	12	f596506e738538b8	31
	39a9b7a3171cb8b3			455da9c47f6e3cf3	
4	0bae3b9e42415649	34	13	738538b8c6a62c4e	29
	171cb8b3ccaca55e			7f6e3cf34bc1a8d9	
5	4241564918b3fa41	37	14	c6a62c4e56b0bd75	33
	ccaca55ed16c3653			4bc1a8d91e07d409	
6	18b3fa419616fe23	33	15	56b0bd7575e8fd8f	31
	d16c3653cf402c68			1e07d4091ce2e6dc	
7	9616fe2367117cf2	32	16	75e8fd8f25896490	32
	cf402c682b2cefbc			1ce2e6dc365e5f59	
8	67117cf2c11bfc09	33	IP^{-1}	da02ce3a89ecac3b	32
	2b2cefbc99f91153			057cde97d7683f2a	

Strength of DES

Strength of DES

- Since DES was adopted as a federal standard, there have been concerns about the level of security provided by DES.
- These concerns fall into two areas:

  a) key size and

  b) the nature of the algorithm

The Use of 56-Bit Keys

- 56-bit key ($2^{56} = 7.2 \times 10^{16}$ values), here, a brute force attack appears impractical.
- Have demonstrated breaks in

1. 1997

    by building a parallel machine with 1 million encryption devices, each of which could perform one encryption per microseconds.

    Would bring the average search time down to about 10 hours.

    Cost would be about $20 million dollars.

2. 1998 on dedicated H/W in a few days

    EFF (Electronic Frontier Foundation) announced that it had broken a DES encryption using a special purpose DES Cracker machine.

    1536 chips and search 88 billion keys/second

    built for less than $250,000 cost.

- There are a number of alternatives to DES, the most important of which are AES and triple DES.

The Nature of the DES Algorithm

- Another concern is that cryptanalysis is possible by exploiting the characteristics of the DES algorithm.
- The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration.
- There is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes.
- No one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.

Timing Attacks

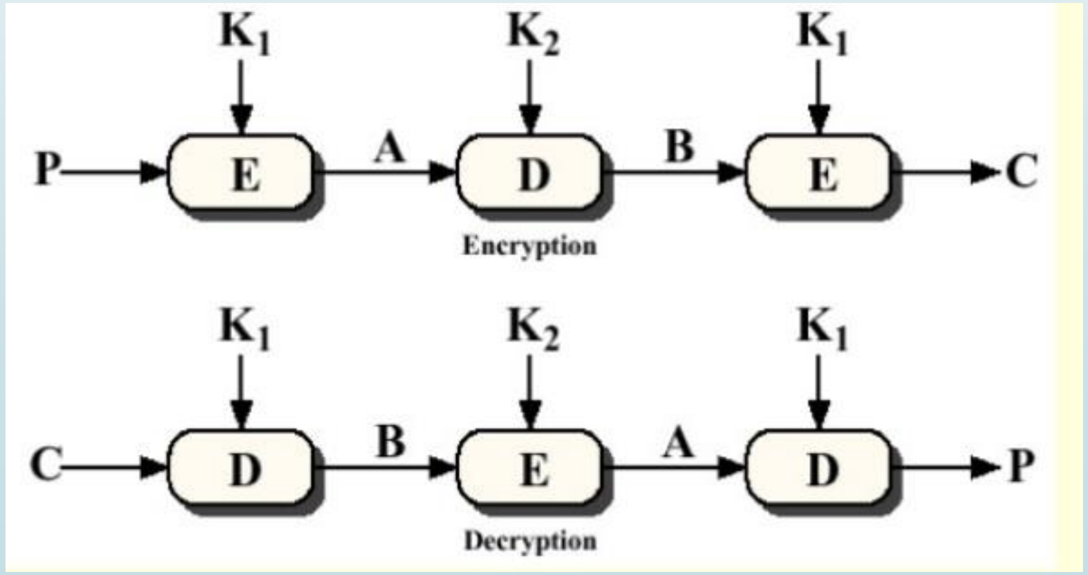
- Timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.
- A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
- DES appears to be fairly resistant to a successful timing attack.

Triple DES

GITAM (DEEMED TO BE UNIVERSITY) (Estd. u/s 3 of the UGC Act, 1956)

Triple DES with 2-Keys

- Use three stages of DES for encryption and decryption.
- The 1st, 3rd stage use K_1 key and 2nd stage use K_2 key.
- To make triple DES compatible with single DES, the middle stage uses decryption in the encryption side and encryption in the decryption side.
- It is much stronger than double DES.



Triple DES with 2-Keys

- The function follows an encrypt-decrypt-encrypt (EDE) sequence.

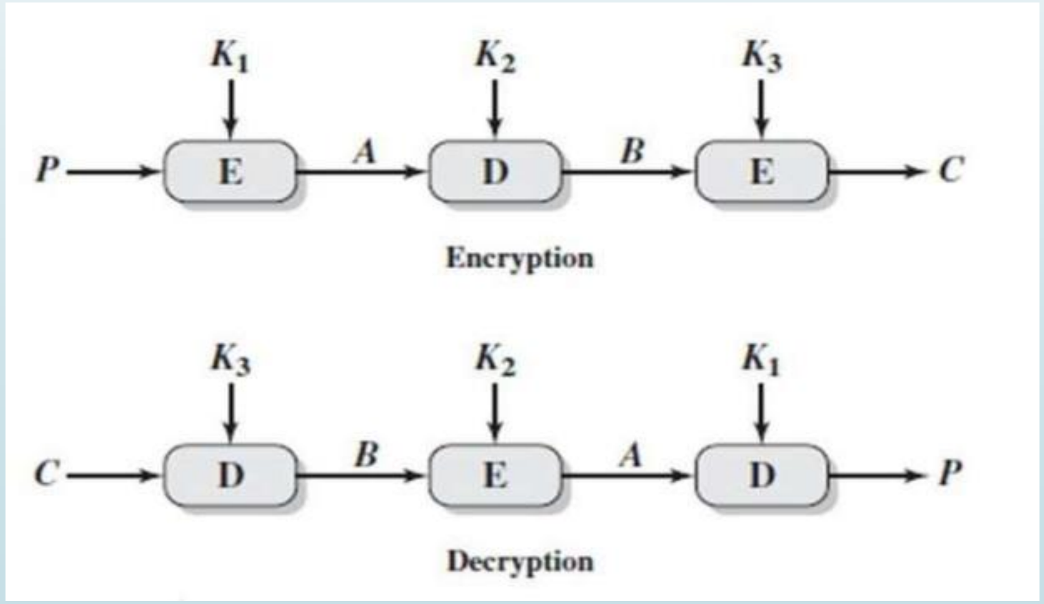
$$C = E(K_1, D(K_2, E(K_1, P)))$$

$$P = D(K_1, E(K_2, D(K_1, C)))$$

- By the use of triple DES with 2-key encryption, it raises the cost of meet-in-the-middle attack to 2^{112} .
- It has the drawback of requiring a key length of $56 \times 3 = 168$ bits, which may be somewhat unwieldy.

Triple DES with 3-Keys

- Although the attacks appear impractical, anyone using two key 3DES may feel some concern.
- Thus, many researches now feel that 3-key 3DES is the preferred alternative.
- Use three stages of DES for encryption and decryption with three different keys.



Triple DES with 3-Keys

- 3-key 3DES has an effective key length of 168 bits and is defined as,

$$C = E(K_3, D(K_2, E(K_1, P)))$$

$$P = D(K_1, E(K_2, D(K_3, C)))$$

- A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME.

Differential and Linear Cryptanalysis

Differential Cryptanalysis

- Most significant advances in cryptanalysis in recent years.
- The first published effort have been the cryptanalysis of a block cipher called FEAL by Murphy.
- Followed by a number of papers by Biham and Shamir, who demonstrated this form of attack on a variety of encryption algorithms and hash functions.
- Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 2^{55} encryptions.

Differential Cryptanalysis

- a statistical attack against Feistel ciphers.
- uses cipher structure not previously used.
- design of S-P networks has output of function *f* influenced by both input & key.
- hence cannot trace values back through cipher without knowing value of the key.
- differential cryptanalysis compares two related pairs of encryptions (differential).

Differential Cryptanalysis Compares Pairs of Encryptions

- Differential cryptanalysis compares two related pairs of encryptions.
- with known difference in the input $m_0||m_1$.
- searching for a known difference in output, when same subkeys are used.

$$m_{i+1} = m_{i-1} \oplus \mathrm{f}(m_i, K_i), \qquad i = 1, 2, \ldots, 16$$

$$\begin{aligned}
\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\
&= [m_{i-1} \oplus \mathrm{f}(m_i, K_i)] \oplus [m'_{i-1} \oplus \mathrm{f}(m'_i, K_i)] \\
&= \Delta m_{i-1} \oplus [\mathrm{f}(m_i, K_i) \oplus \mathrm{f}(m'_i, K_i)]
\end{aligned}$$

Differential Cryptanalysis

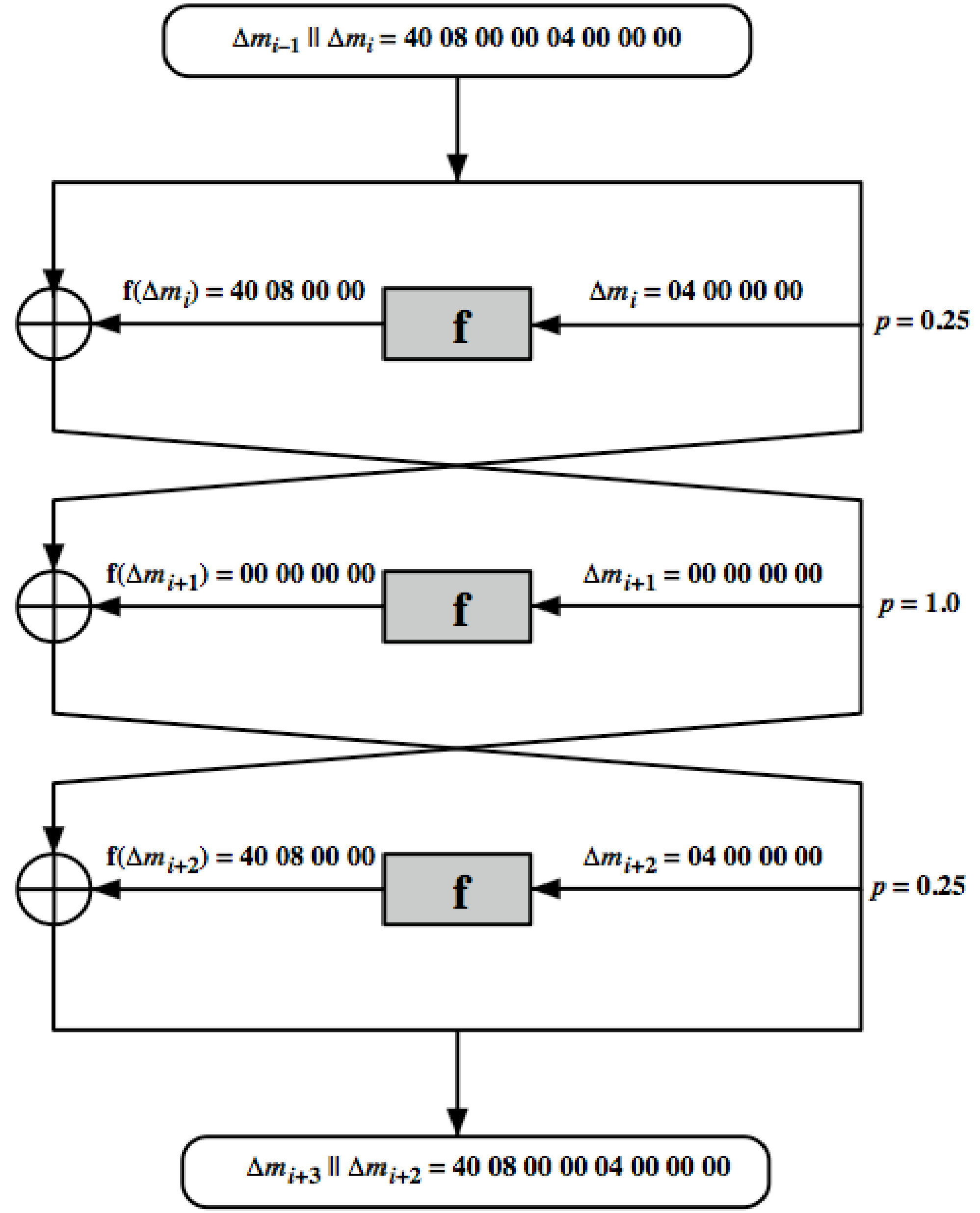
- have some input difference giving some output difference with probability p.
- if find instances of some higher probability input / output difference pairs occurring, can infer subkey that was used in round.
- then must iterate process over many rounds (with decreasing probabilities).

Differential Cryptanalysis

Input round i

Input round i+1

Overall probabilty of given output difference is (0.25)(1.0)(0.25) = 0.0625



$\Delta m_{i-1} \| \Delta m_i$ = 40 08 00 00 04 00 00 00
$f(\Delta m_i)$ = 40 08 00 00
f
Δm_i = 04 00 00 00
$p = 0.25$
$f(\Delta m_{i+1})$ = 00 00 00 00
f
Δm_{i+1} = 00 00 00 00
$p = 1.0$
$f(\Delta m_{i+2})$ = 40 08 00 00
f
Δm_{i+2} = 04 00 00 00
$p = 0.25$
$\Delta m_{i+3} \| \Delta m_{i+2}$ = 40 08 00 00 04 00 00 00

Differential Cryptanalysis

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR
- when found, assume intermediate deltas match
  - if intermediate rounds match required XOR have a **right pair**
  - if not then have a **wrong pair**, relative ratio is S/N for attack
- can then deduce keys values for the rounds
  - right pairs suggest same key bits
  - wrong pairs give random values
- for large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs
- Biham and Shamir have shown how a 13-round iterated characteristic can break the full 16-round DES

Linear Cryptanalysis

- another fairly recent development
- also a statistical method
- must be iterated over rounds, with decreasing probabilities
- developed by Matsui et al in early 90's
- based on finding linear approximations
- can attack DES with 2^{43} known plaintexts, easier but still in practice infeasible

GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)

Linear Cryptanalysis

- find linear approximations with prob p != ½
  - $P[i_1,i_2,\ldots,i_a] \oplus C[j_1,j_2,\ldots,j_b] = K[k_1,k_2,\ldots,k_c]$
  - where i_a,j_b,k_c are bit locations in P,C,K
- gives linear equation for key bits
- get one key bit using max likelihood algorithm
- using a large number of trial encryptions
- effectiveness given by: $|p-{}^1/_2|$

Thank You

Topics Covered

- **BLOCK CIPHER DESIGN PRINCIPLES**
  - Design criteria used in the DES effort.
  - Three critical aspects of block cipher design:
    - The number of rounds,
    - Design of the function F, and
    - Key scheduling.
- **BLOCK CIPHER MODES OF OPERATION**
- **ADVANCED ENCRYPTION STANDARD (AES)**
- **RC6**

Block Cipher Design Principles

Design Criteria of DES

- The criteria used in the design of DES, focused on the design of the S-boxes and on the P function that takes the output of the S-boxes.
- S-boxes are the only nonlinear part of DES.
- The criteria for the S-boxes are as follows:
  - No output bit of any S-box should be too close a linear function of the input bits.
  - Each row of an S-box should include all 16 possible output bit combinations.
  - If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits.
  - If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.
  - If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.
  - For any nonzero 6-bit difference between inputs, no more than eight of the 32 pairs of inputs exhibiting that difference may result in the same output difference.

Number of Rounds

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- It was observed that for 16-round DES, a differential cryptanalysis attack is slightly less efficient than brute force:
  - The differential cryptanalysis attack requires $2^{55.1}$ operations, whereas brute force requires 2^{55}.
- If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search.

Design of Function F

- The heart of a Feistel block cipher is the function F.

- **DESIGN CRITERIA FOR F**
  - The function F provides the element of confusion in a Feistel cipher.
  - Thus, it must be difficult to “unscramble” the substitution performed by F.
  - 1st criteria is that F be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be.
  - 2nd criteria is to have good avalanche properties.
  - 3rd criteria is **bit independence criterion (BIC)**.

Design of Function F

- **S-BOX DESIGN**
- Nyberg, who has written a lot about the theory and practice of S-box design, suggests the following approaches:
  - **Random:** Use some pseudorandom number generation or some table of random digits to generate the entries in the S-boxes.
  - **Random with testing:** Choose S-box entries randomly, then test the results against various criteria, and throw away those that do not pass.
  - **Human-made:** This is a more or less manual approach with only simple mathematics to support it. It is apparently the technique used in the DES design. This approach is difficult to carry through for large S-boxes.
  - **Math-made:** Generate S-boxes according to mathematical principles. By using mathematical construction, S-boxes can be constructed that offer proven security against linear and differential cryptanalysis, together with good diffusion.

Key Schedule Algorithm

- With any Feistel block cipher, the key is used to generate one subkey for each round.
- Subkeys are selected to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.
- No general principles for this have yet been promulgated.
- It is suggested that, at minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

Block Cipher Modes of Operation

Introduction to block cipher modes of operation

- NIST : National Institute of Standards and Technology
- NIST defines five block cipher modes of operations
  - Electronic code book
  - Cipher chaining block
  - Cipher feedback mode
  - Output feedback mode
  - Counter mode

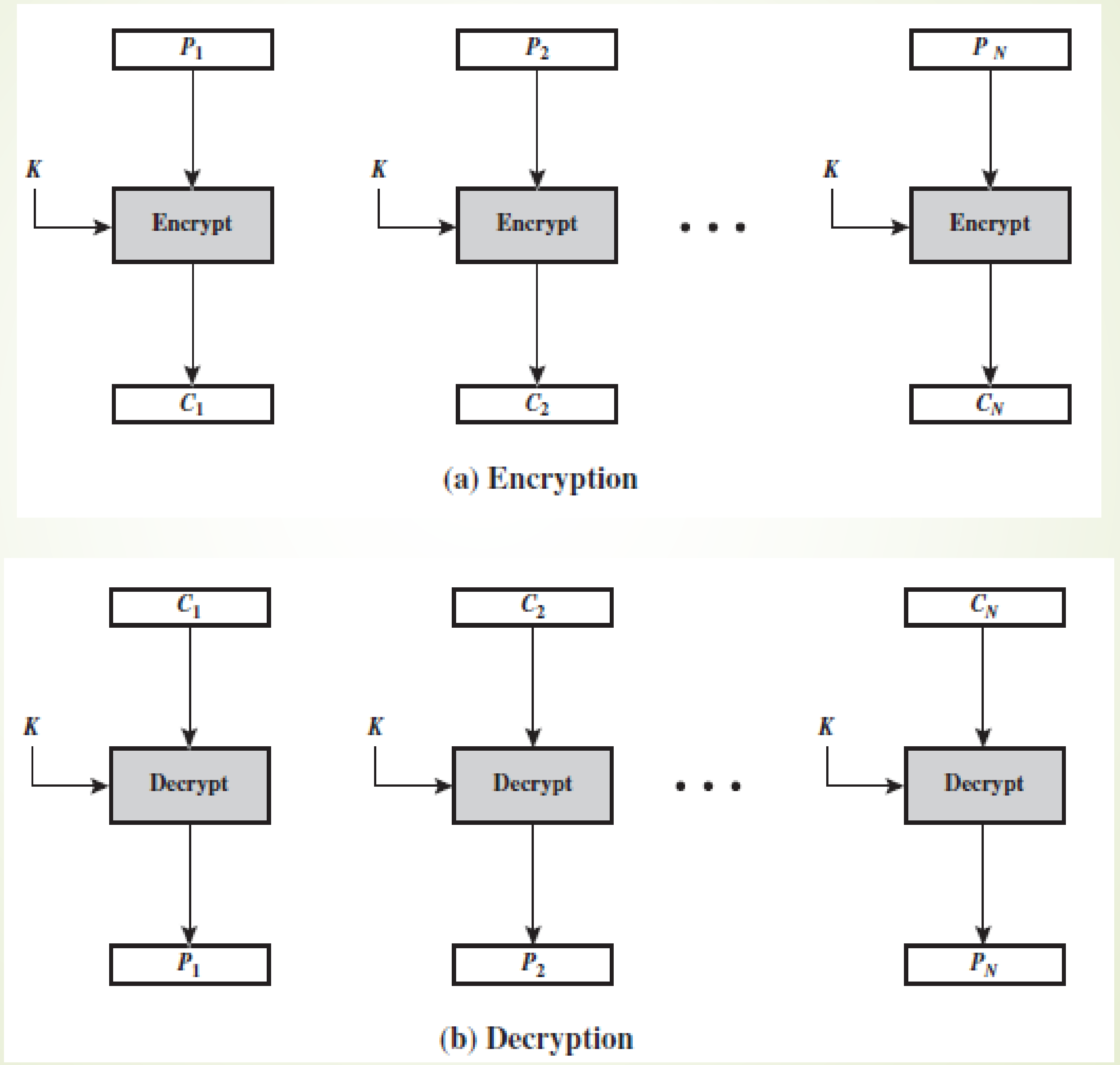
When we use block cipher modes of operation?

- Block cipher only allow to encrypt entire blocks.
- What if our message is longer/shorter than the block size?
- When message is longer/shorter than the block size , we use modes of operations.
- Algorithms that exploit a block cipher to provide a service (e.g. confidentiality).

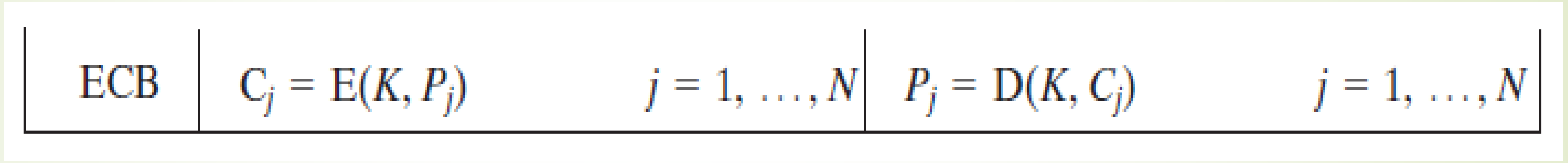
Electronic Code Book

- ECB is the simplest mode of operation.
- The plain text is divided into N blocks.
- The block size is n bits.
- If the plaintext size is not multiple of the block size , the text is padded to make the last block the same size other blocks.
- Same key is used to encrypt and decrypt each block.

Electronic codebook encryption & decryption process



Electronic codebook encryption/decryption



ECB	$C_j = E(K, P_j)$	$j = 1, \ldots, N$	$P_j = D(K, C_j)$	$j = 1, \ldots, N$

- **Security issues**
- Patterns at the block level are preserved.
- For example equal blocks in the plain text become equal block in the cipher text.
- If any person finds out the cipher text block 1,5 and 10 are
- the same ,that person knows that plaintext blocks 1, 5 and 10 are the same.
- This is a leak in security.

What is initialization vector (IV)?

- An initialization vector (IV) or starting variable is a block of bits that is used by several modes to randomize the encryption and hence to produce distinct cipher texts even if the same plain text is encrypted multiple times, without the need for a slower re-keying process.
- An initialization vector has different security requirements than a key, so the IV usually does not need to be secret.
- However, in most cases, it is important that an initialization vector is never reused under the same key.

GITAM (DEEMED TO BE UNIVERSITY) (Estd. u/s 3 of the UGC Act, 1956)

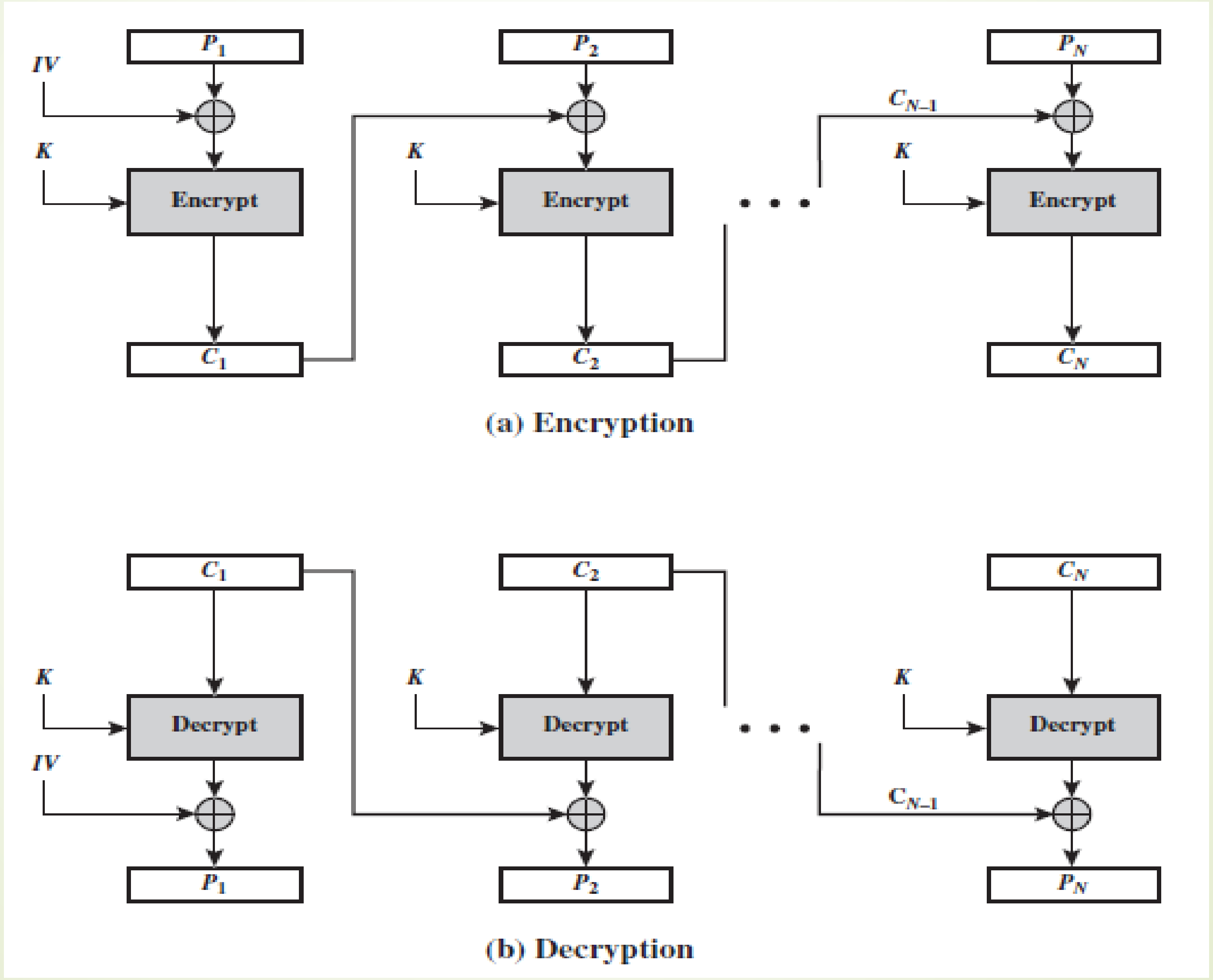
What is initialization vector? (continue…)

- For CBC and CFB, reusing an IV leaks some information about the first block of plaintext, and about any common prefix shared by the two messages.
- For OFB and CTR, reusing an IV completely destroys security.
- This can be seen because both modes effectively create a bit stream that is XORed with the plaintext, and this bit stream is dependent on the password and IV only. Reusing a bit stream destroys security.
- In CBC mode, the IV must, in addition, be unpredictable at encryption time; in particular, the (previously) common practice of re-using the last cipher text block of a message as the IV for the next message is insecure.

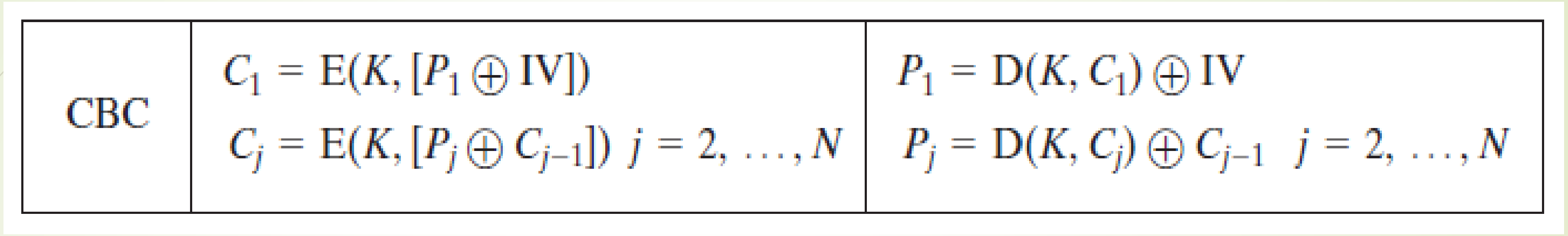
Cipher Block Chaining Mode

- IBM invented the Cipher Block Chaining (CBC) mode of operation in 1976.
- In CBC mode, each block of plaintext is XORed with the previous cipher text block before being encrypted.
- This way, each cipher text block depends on all plaintext blocks processed up to that point.
- To make each message unique, an initialization vector must be used in the first block.

Cipher block chaining mode encryption and decryption



Cipher block chaining mode



CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}])\ j = 2, \ldots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1}\ \ j = 2, \ldots, N$

- **Security issues**
- The patterns at the block level are not preserved.
- In CBC mode, equal plain text block belonging to the same message are enciphered into different cipher text block.
- However ,if two message are equal ,their encipherment is the same if they use the same IV.
- As a matter of fact ,if the first M blocks in two different message are equal , they are enciphered into equal blocks unless different IVs are used.
- For this reason , some people recommended the use of timestamp as an IV.
- Any person can add some cipher text blocks to the end of the cipher text stream.

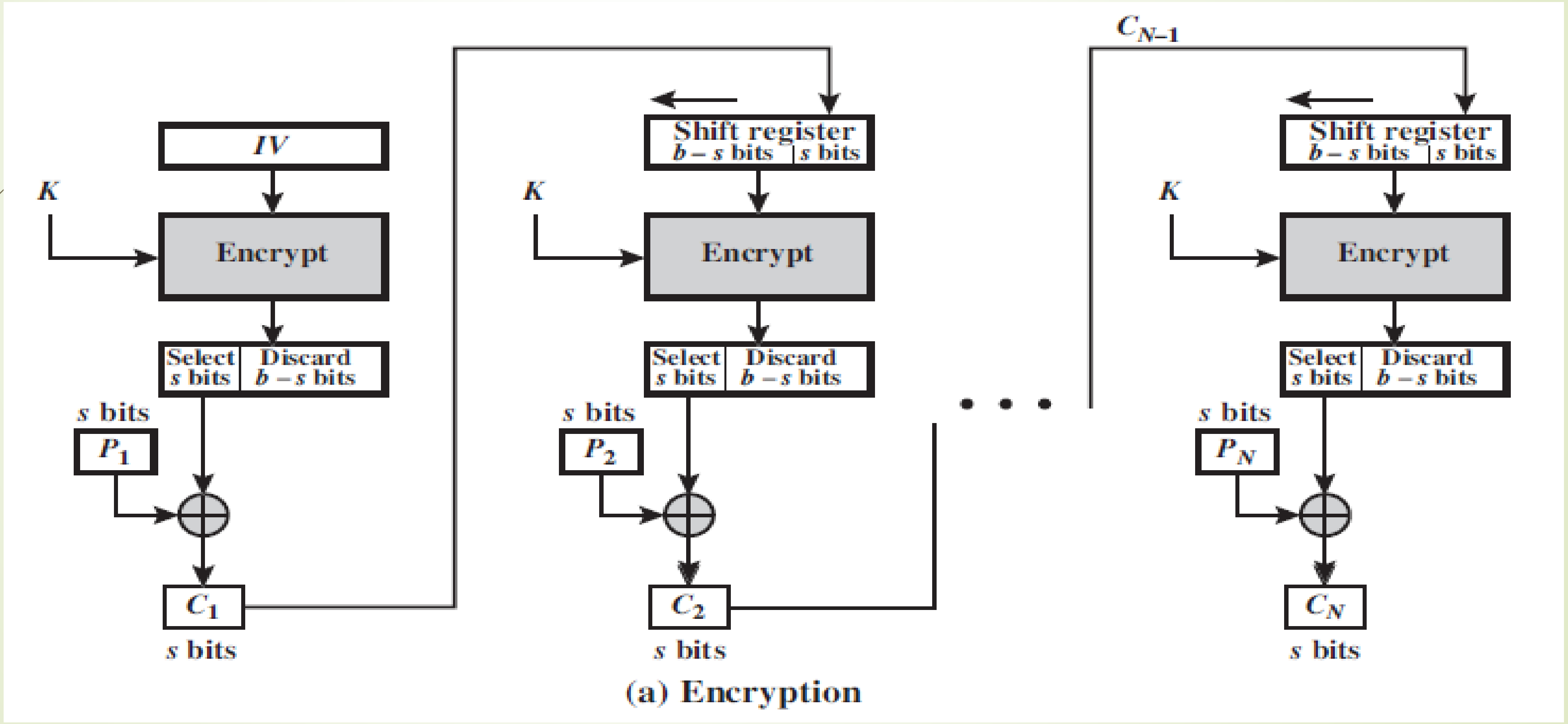
Cipher Feedback Mode

- ECB and CBC modes encrypt and decrypt blocks of the message.
- Block size n is predetermine by the underlying cipher,

  for example , for DES n = 64

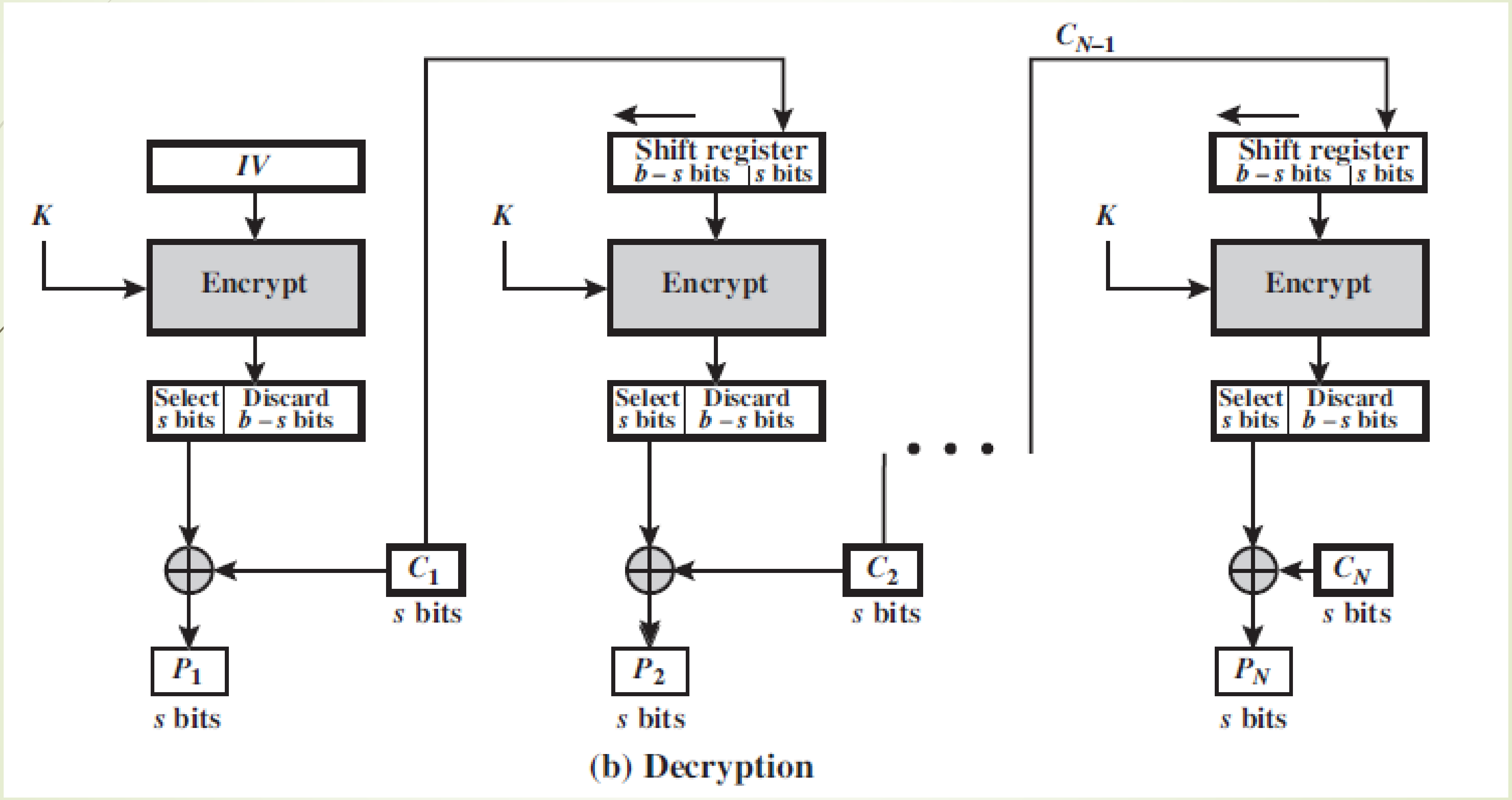
  for AES n =128
- In some situations, we need use DES or AES as secure cipher , but the plain text or cipher text block size are to be smaller.
- For example , to encrypt and decrypt 8-bit characters , you would not want to use one of the traditional cipher like Caesar cipher.
- The solution is to use DES or AES in cipher feedback mode.

Cipher Feedback Mode Encryption



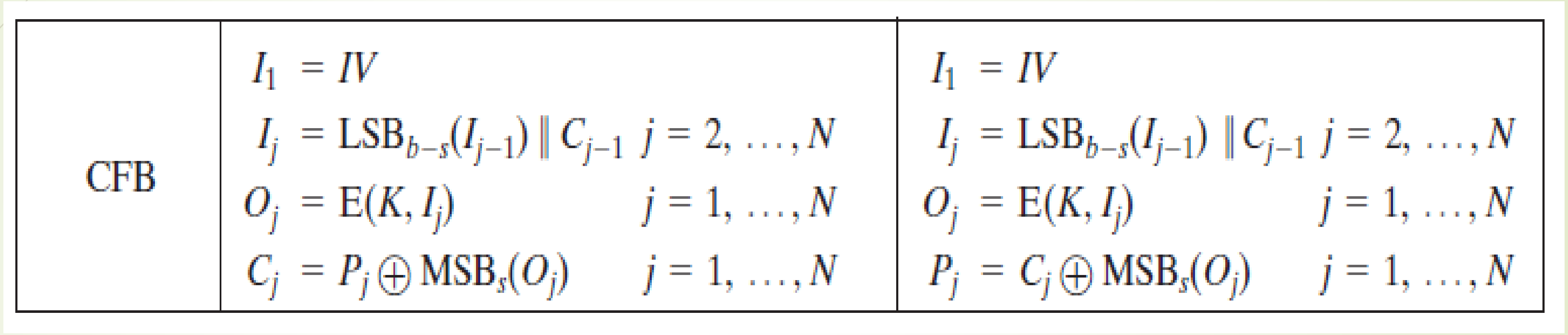
(a) Encryption

Cipher Feedback Mode Decryption



(b) Decryption

Cipher feedback mode



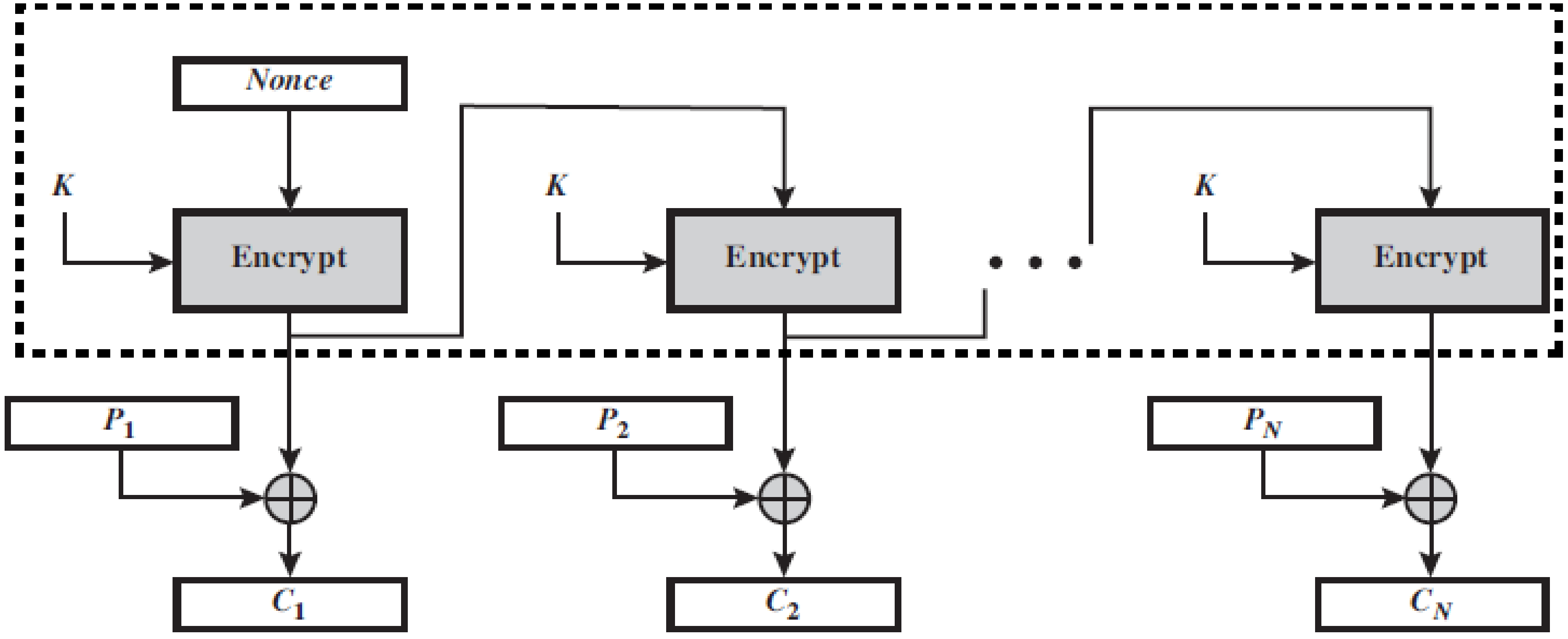
CFB	$I_1 = IV$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus \mathrm{MSB}_s(O_j) \quad j = 1, \ldots, N$	$I_1 = IV$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus \mathrm{MSB}_s(O_j) \quad j = 1, \ldots, N$

- **Security issues**
- Just like CBC , patterns at the block level are not preserved.
- More than one message can be encrypted with the same key, but the value of the IV should be changed for each message.
- This means that sender needs to use a different IV each time sender sends a message.
- Attacker can add some cipher text block to the end of the cipher text stream.

Output Feedback Mode

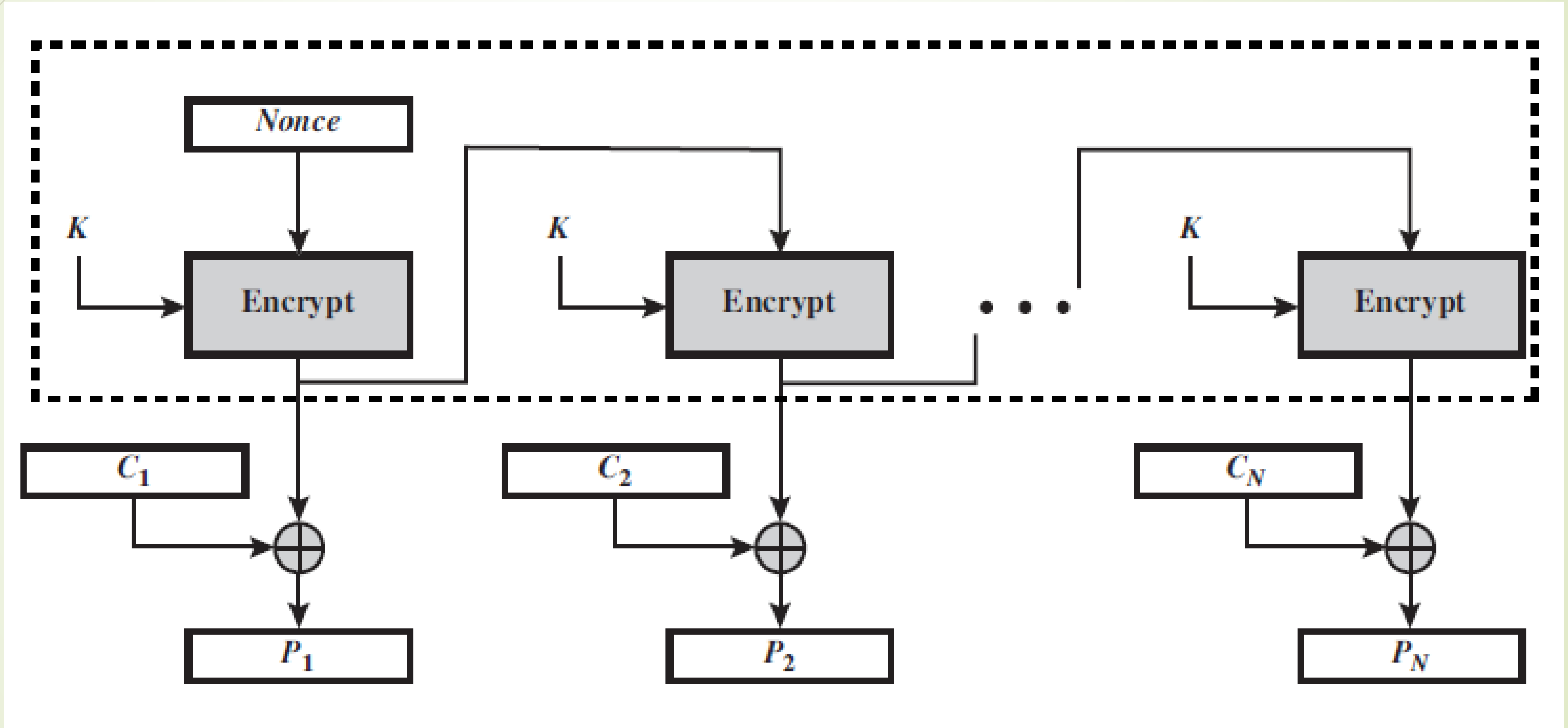
- Output feedback mode is very similar to CFB mode, with one difference: each bit in the cipher text is independent of the previous bit or bits.
- This avoids error propagation.
- If an error occur in transmission , it does not affect the bits that follow.
- Like cipher feedback mode , both the sender and the receiver use the encryption algorithm.

Output Feedback Mode Encryption



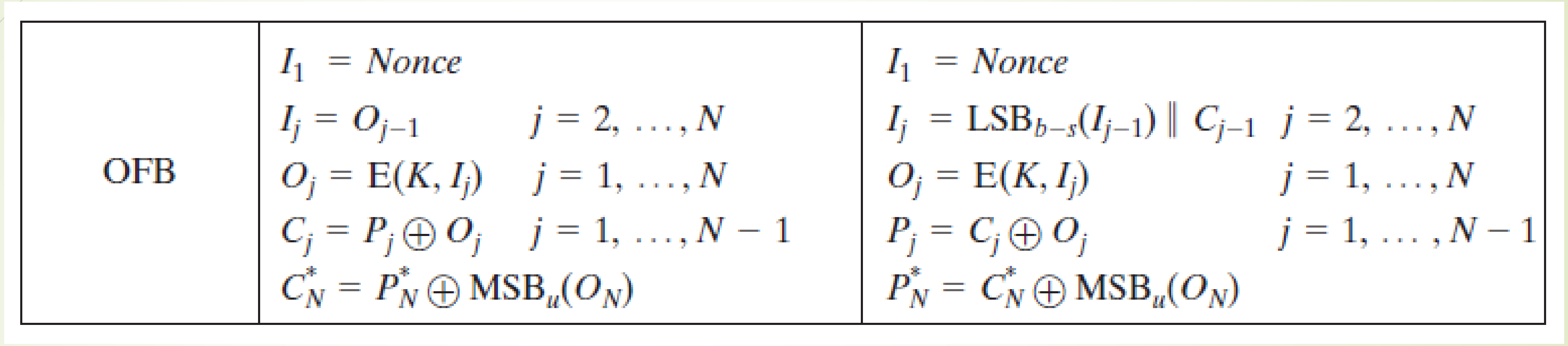
(a) Encryption

Output Feedback Mode Decryption



(b) Decryption

Output Feedback Mode



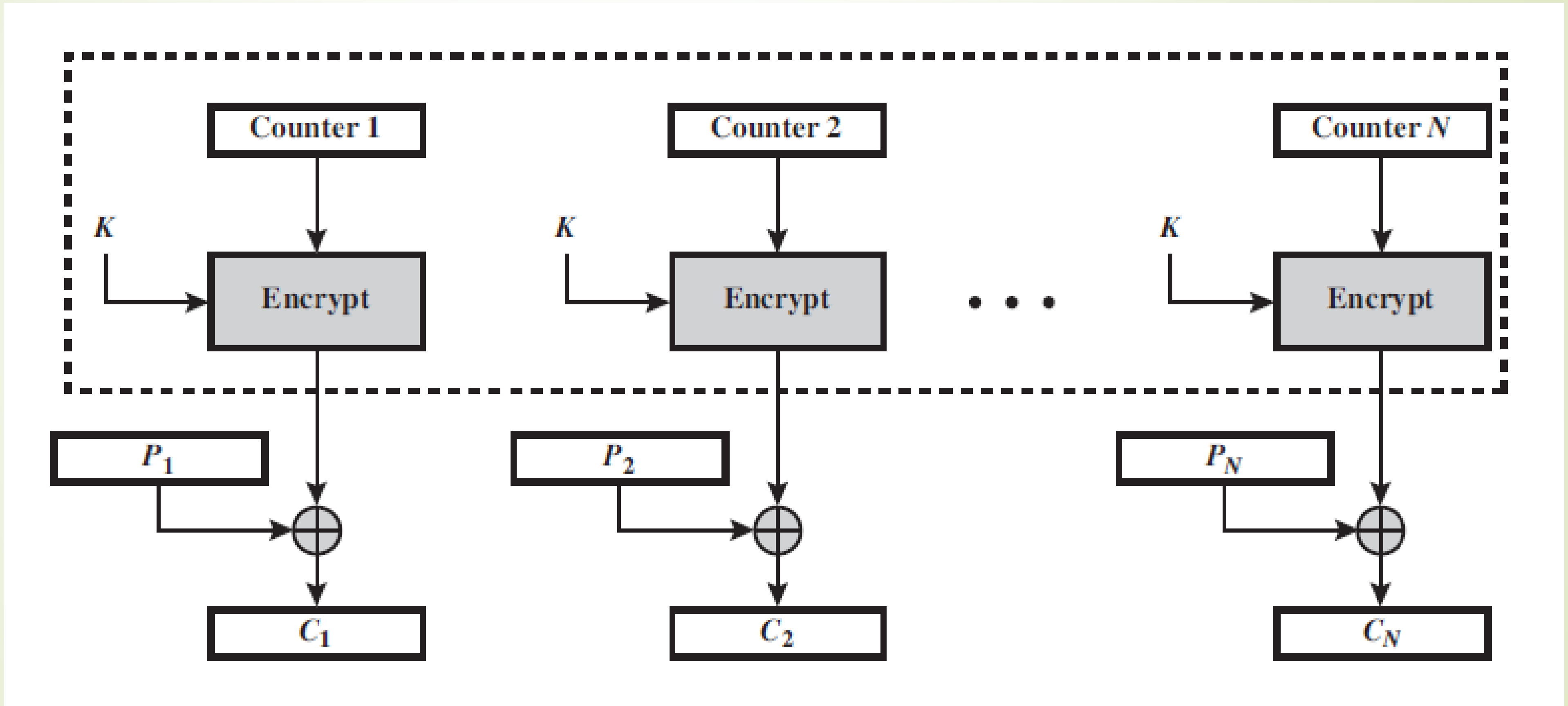
OFB	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$	$I_1 = Nonce$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$

- **Security Issues**
- Just like CBC , patterns at the block level are not preserved.
- Any change in the cipher text affects the plain text encrypted at the receiver side.

Counter Mode

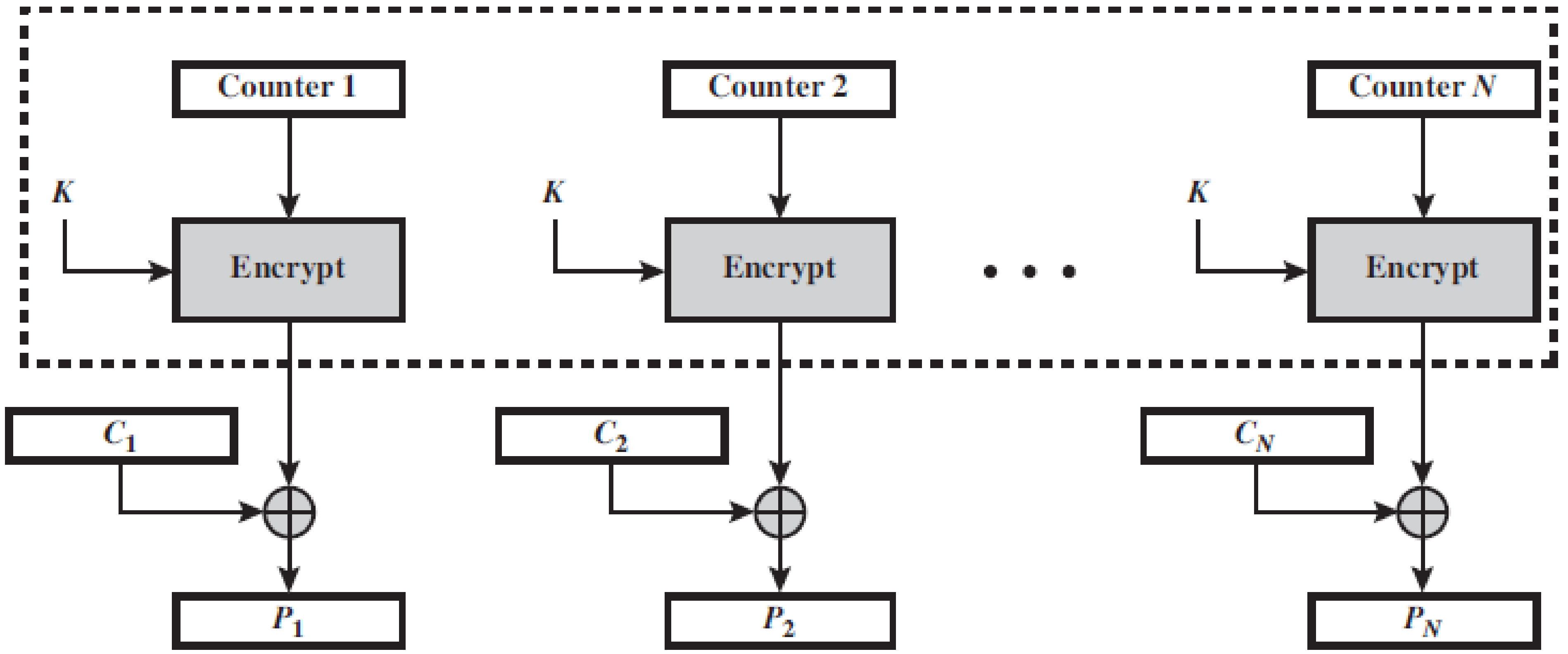
- In the counter mode , there is no feedback.
- The pseudo randomness in the key streams achieved using a counter.
- An n bit counter is initialized to a predetermined value(IV) and incremented based on a predefined rule(mod 2n).
- To provide a better randomness , the increment value can depend on the block numbers to be incremented.
- The plain text and cipher block text block have same block size as the underlying cipher.
- Both encryption and decryption can be performed fully in parallel on multiple blocks .
- Provides true random access to cipher text blocks.

Counter Mode Encryption process



(a) Encryption

Counter Mode Decryption process



(b) Decryption

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$

Advantages of Counter Mode

- **Hardware efficiency:** Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or cipherext.
- **Software efficiency:** Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features can be effectively utilized.
- **Preprocessing:** The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext.
- **Random access:** The ith 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.
- **Simplicity:** Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm.

Advanced Encryption Standard (AES)

What is AES?

- AES is an encryption standard chosen by the National Institute of Standards and Technology(NIST), USA to protect classified information.
- It has been accepted world wide as a desirable algorithm to encrypt sensitive data.
- It is a block cipher which operates on block size of 128 bits for both encrypting as well as decrypting.
- Each Round performs same operations.

Why AES?

- In 1990's the cracking of DES algorithm became possible.
- Around 50hrs of bruteforcing allowed to crack the message.
- NIST started searching for new feasible algorithm and proposed its requirement in 1997.
- In 2001 Rijndael algorithm designed by Rijment and Daemon of Belgium was declared as the winner of the competition.
- It met all Security, Cost and Implementation criteria.

How Does it works?

- AES basically repeats 4 major functions to encrypt data.
- It takes 128 bit block of data and a key and gives a ciphertext as output.
- The functions are:

  I. SubstituteBytes

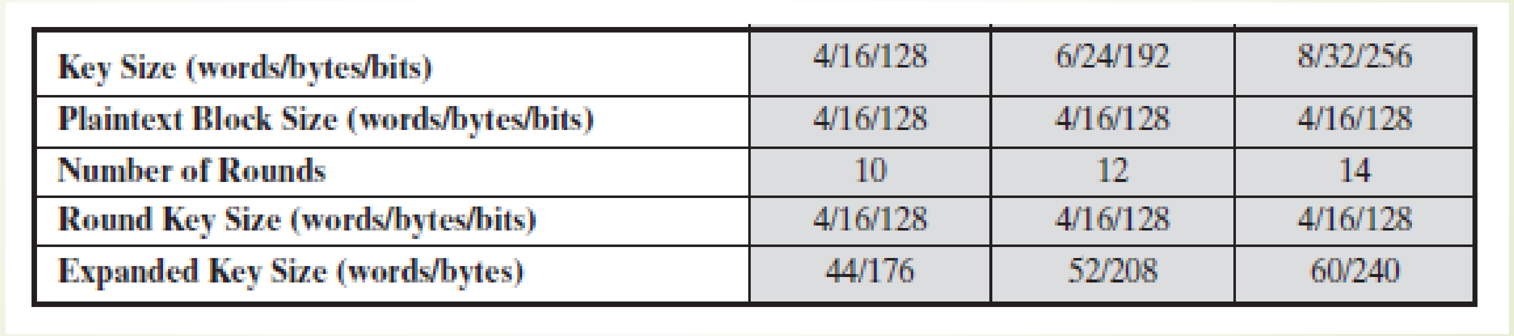
  II. ShiftRows

  III. MixColumns

  IV. AddRoundKey

How Does it works?

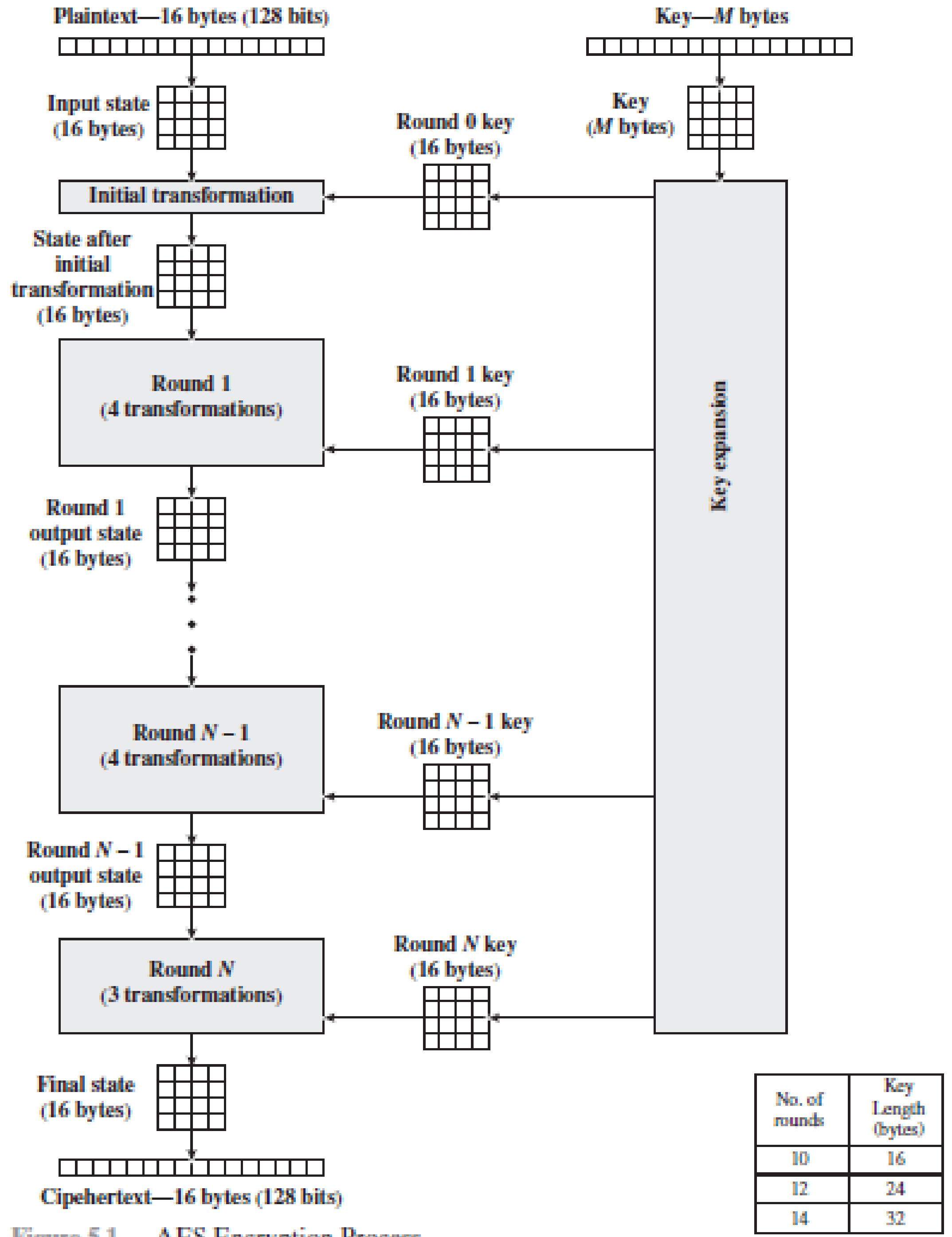
- The number of rounds performed by the algorithm strictly depends on the size of key.
- The following table gives overview of no. of rounds performed with the input of varying key lengths.
- AES Parameters:



Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

- The larger the number of keys the more secure will be the data.
- The time taken by software to encrypt will increase with no. of rounds.

AES Encryption Process


Plaintext—16 bytes (128 bits)
Key—M bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round N – 1 (4 transformations)
Round N – 1 key (16 bytes)
Round N – 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 5.1 AES Encryption Process

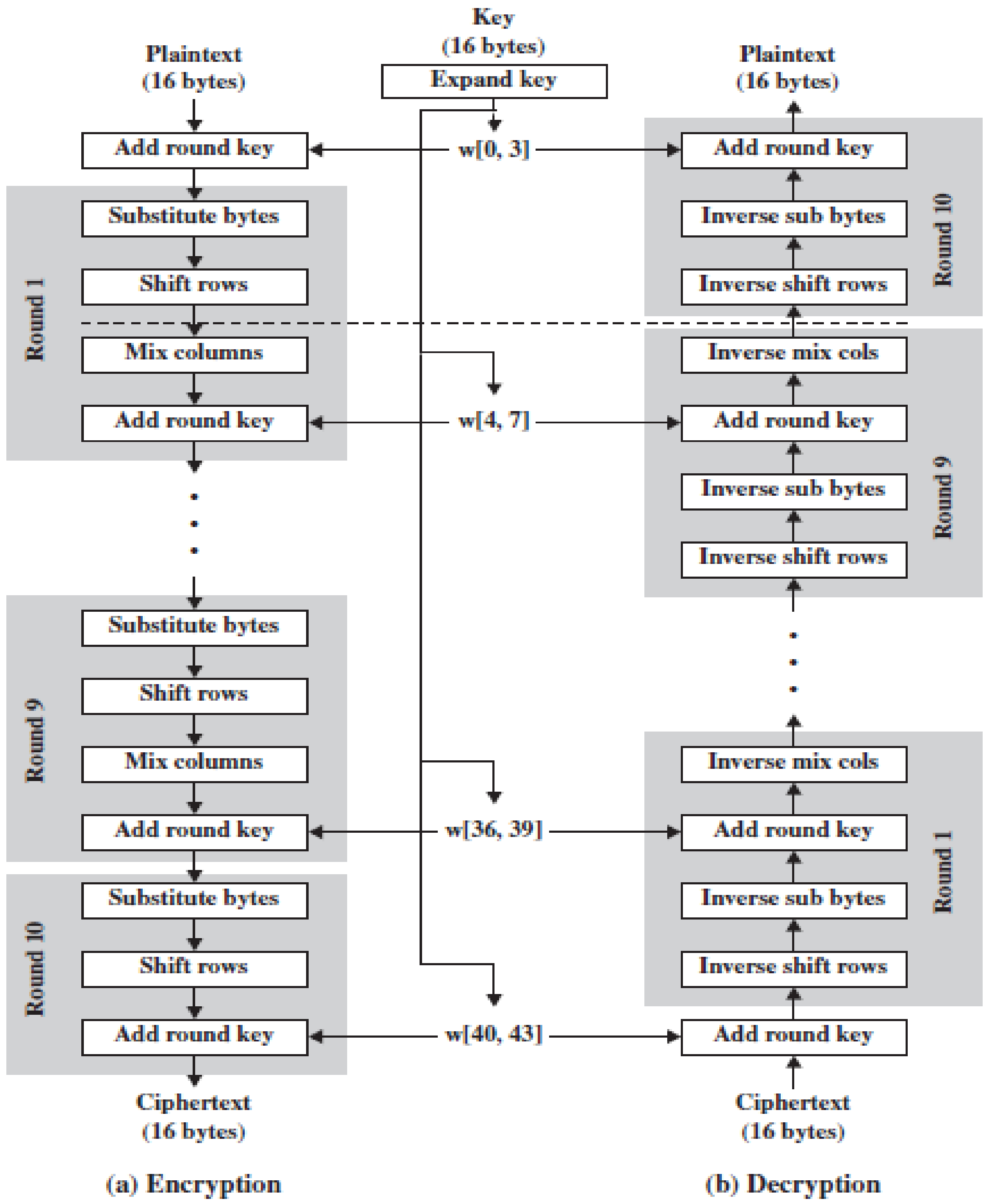
AES Encryption Process

- The input to the encryption and decryption algorithms is a single 128-bit block.
- this block is a 4 x 4 square matrix of bytes.
- This block is copied into the **State** array, which is modified at each stage of encryption or decryption.
- After the final stage, **State** is copied to an output matrix.
- The key is depicted as a square matrix of bytes.
- This key is then expanded into an array of key schedule words.
- Each word is four bytes, and the total key schedule is 44 words for the 128-bit key.
- Ordering of bytes within a matrix is by column.

AES cipher in detail

- AES processes the entire data block as a single matrix during each round using substitutions and permutation.
- The key that is provided as input is expanded into an array of forty-four 32-bit words, **w**[*i*]. Four distinct words (128 bits) serve as a round key for each round.
- Four different stages are used, one of permutation and three of substitution:
  - **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block.
  - **ShiftRows:** A simple permutation
  - **MixColumns:** A substitution that makes use of arithmetic over GF(2^8)
  - **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key
- For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.

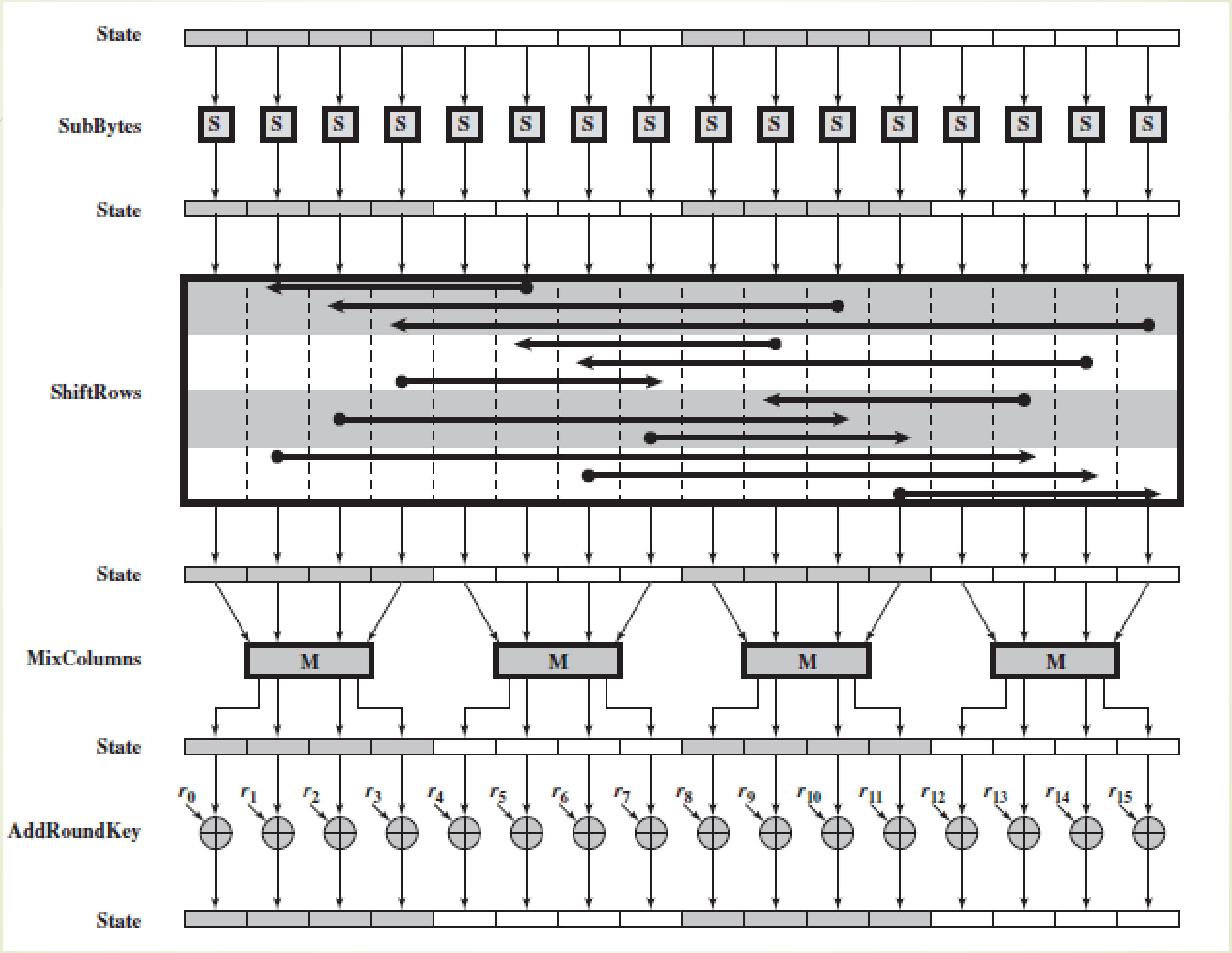
AES Encryption & Decryption Process



AES cipher in detail

- Only the AddRoundKey stage makes use of the key.
- The cipher is viewed as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on. This scheme is both efficient and highly secure.
- Each stage is easily reversible.
- decryption algorithm is not identical to the encryption algorithm.
- Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext.
- The final round of both encryption and decryption consists of only three stages.

AES Encryption Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

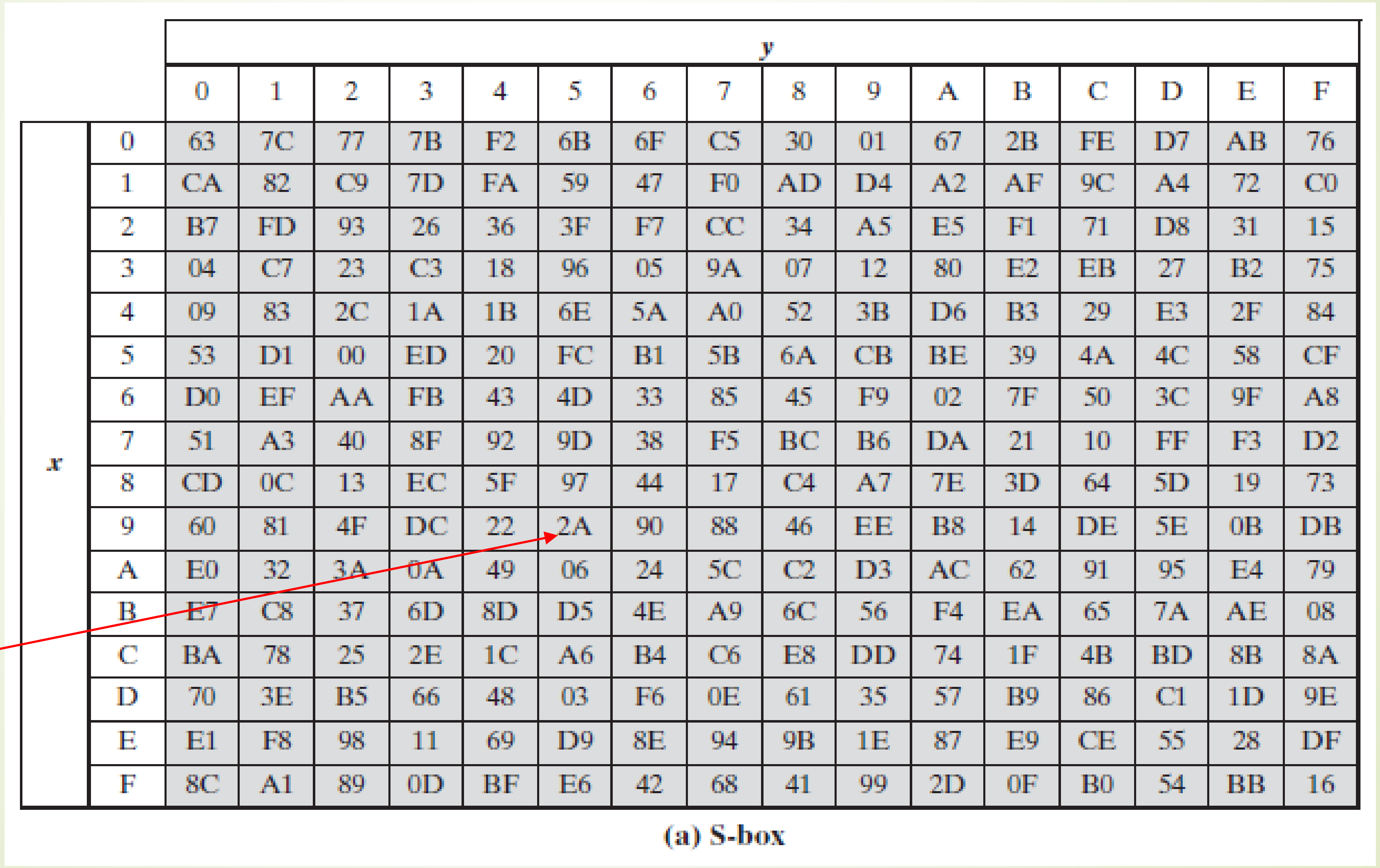
GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)

AES Transformation Functions

Substitute Bytes Transformation

- The **forward substitute byte transformation**, called SubBytes, is a simple table lookup.
- AES defines a 16 x 16 matrix of byte values, called an S-box that contains a permutation of all possible 256 8-bit values.
- Byte mapping:
    - The leftmost 4 bits of the byte are used as a row value.
    - The rightmost 4 bits are used as a column value.
    - Ex : the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}.

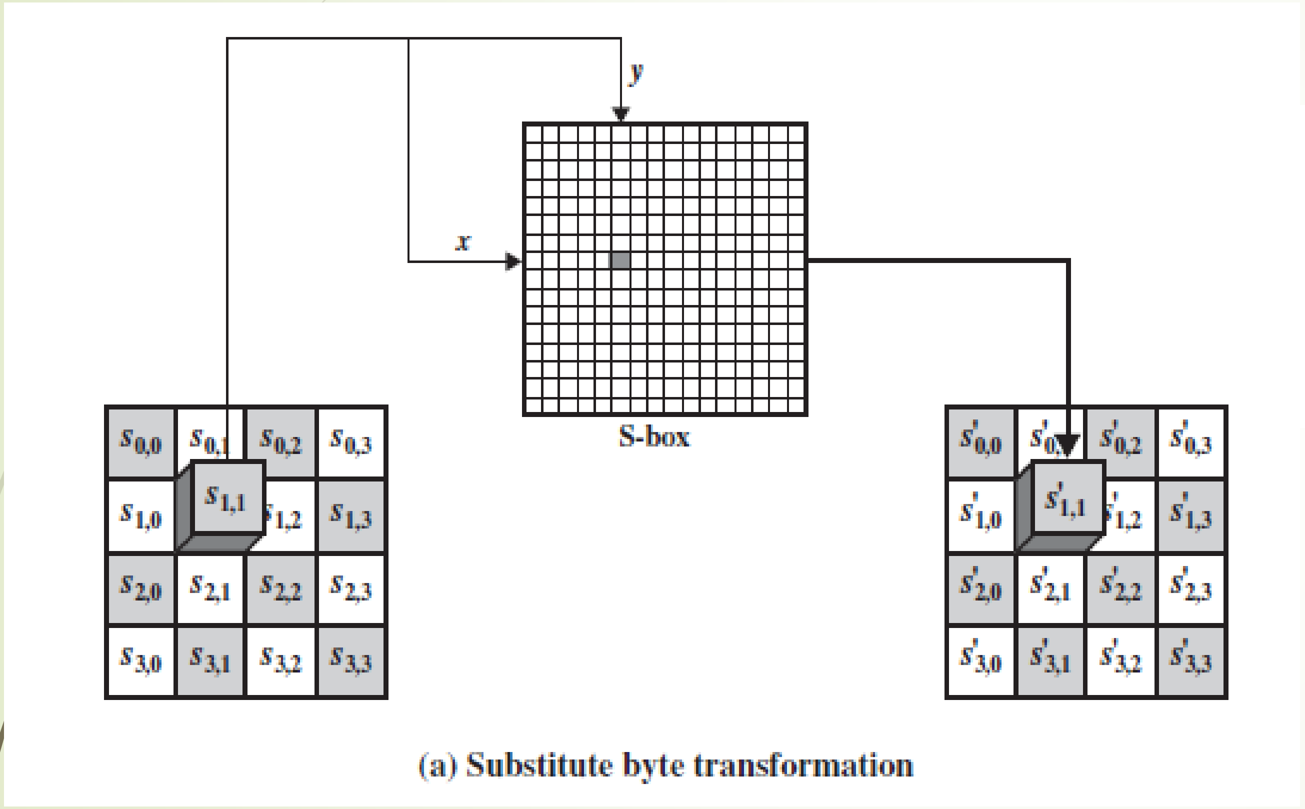
AES S-Box



	y															
x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

(a) S-box

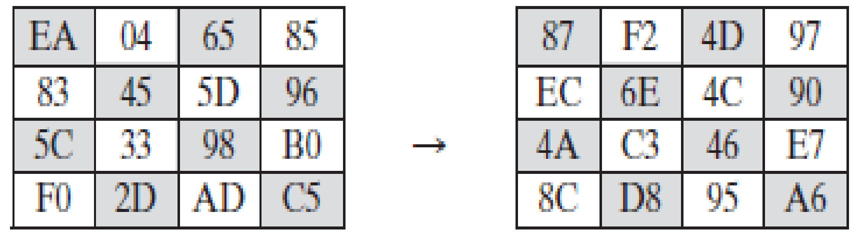
Substitute Byte Transformation



y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(a) Substitute byte transformation

- Example



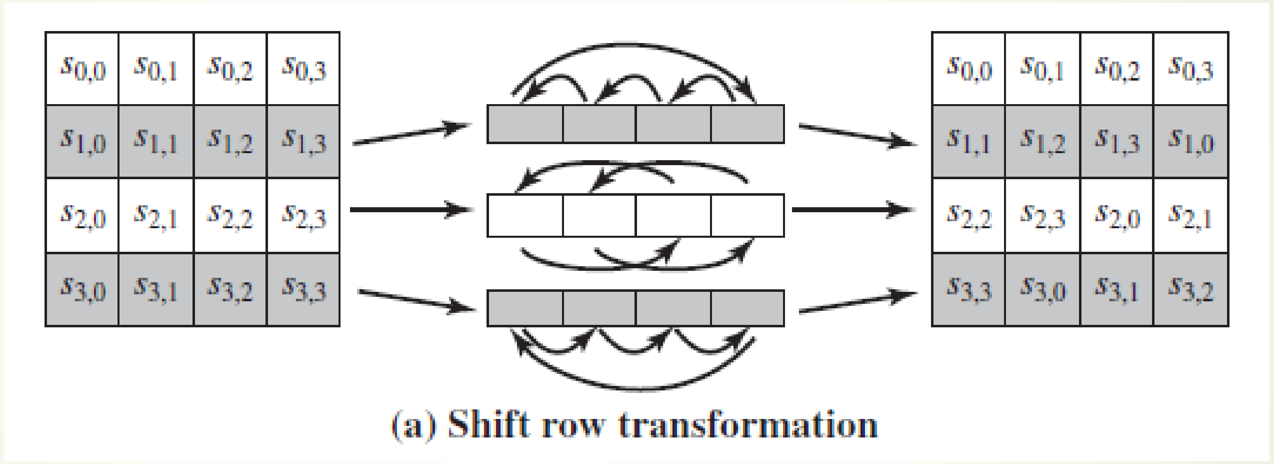
EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

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

ShiftRows Transformation

- **FORWARD AND INVERSE TRANSFORMATIONS**



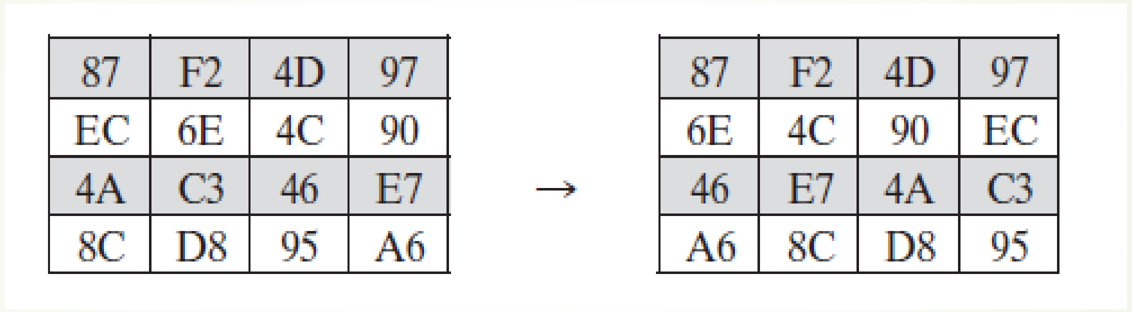
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

(a) Shift row transformation

- The first row of **State** is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.

ShiftRows Transformation

- Example:



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

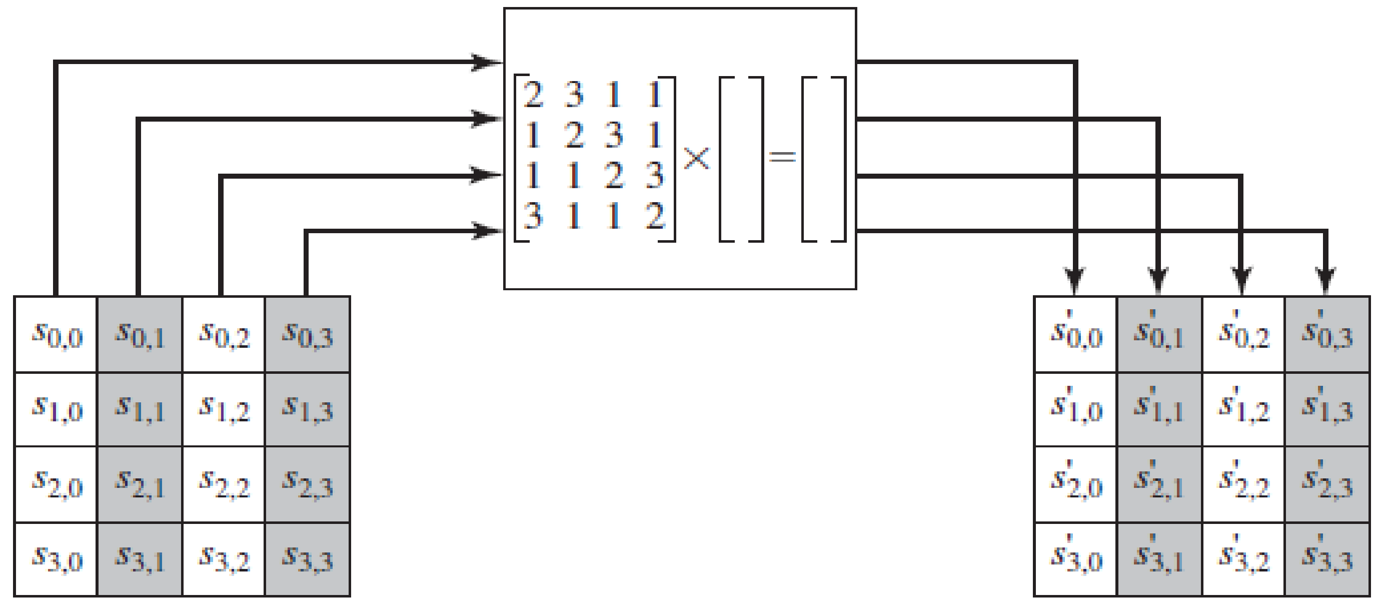
→

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

- The **inverse shift row transformation**, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.

MixColumns Transformation

- **FORWARD AND INVERSE TRANSFORMATIONS**
- The **forward mix column transformation** called MixColumns, operates on each column individually.
- The transformation can be defined by the following matrix multiplication on **State:**



$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} \ \end{bmatrix} = \begin{bmatrix} \ \end{bmatrix}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

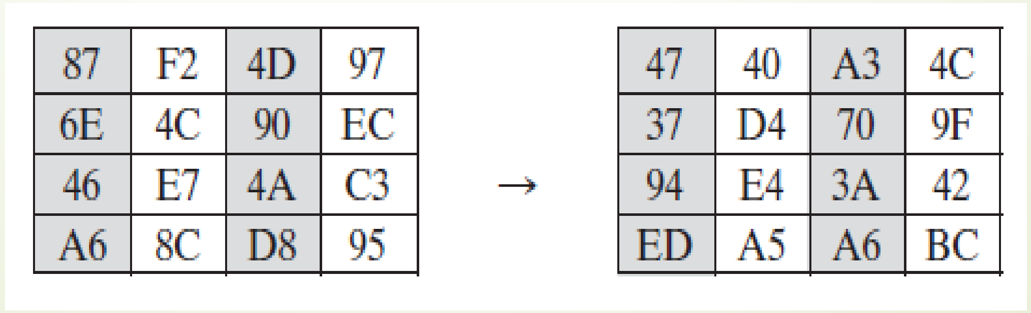
(b) Mix column transformation

MixColumns Transformation

- The MixColumns transformation on a single column of State can be expressed as:

$$
\begin{aligned}
s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\
s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\
s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j}) \\
s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})
\end{aligned}
$$

- Example :



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

$\rightarrow$

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Inverse MixColumn Transformation

- The **inverse mix column transformation** called InvMixColumns, is defined by the following matrix multiplication:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- We need to show

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

Inverse MixColumn Transformation

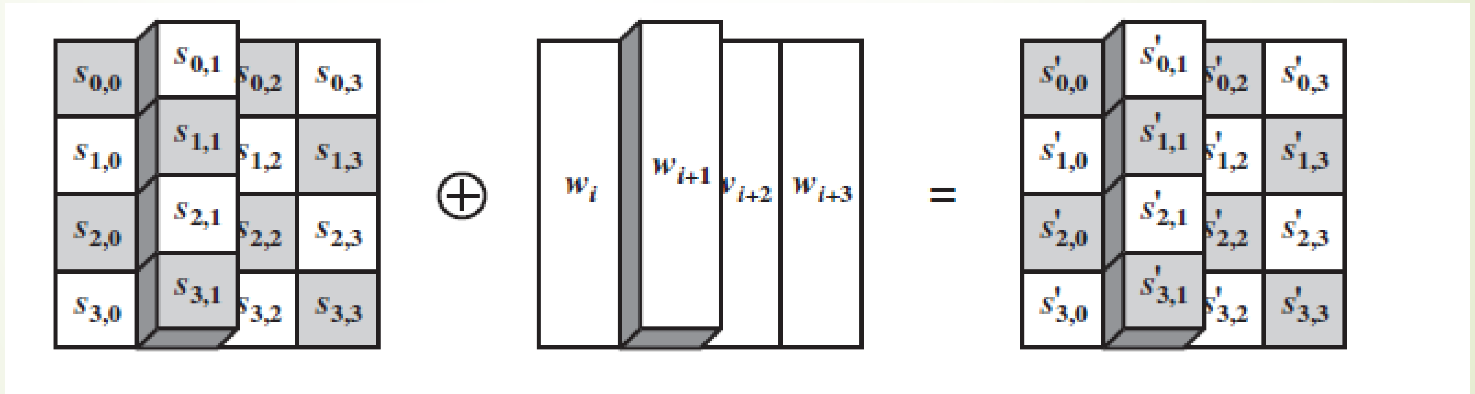
- which is equivalent to showing:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The inverse transformation matrix times the forward transformation matrix equals the identity matrix.

AddRoundKey Transformation

- **FORWARD AND INVERSE TRANSFORMATIONS**
- In the **forward add round key transformation**, called AddRoundKey, the 128 bits of **State** are bitwise XORed with the 128 bits of the round key.



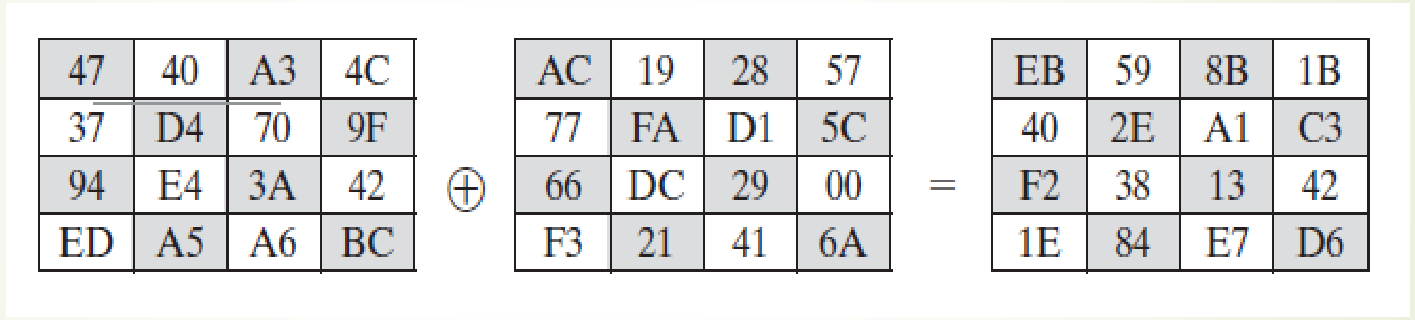
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$\oplus$
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(b) Add round key transformation

- Columnwise operation between the 4 bytes of a **State** column and one word of the round key.

AddRoundKey Transformation

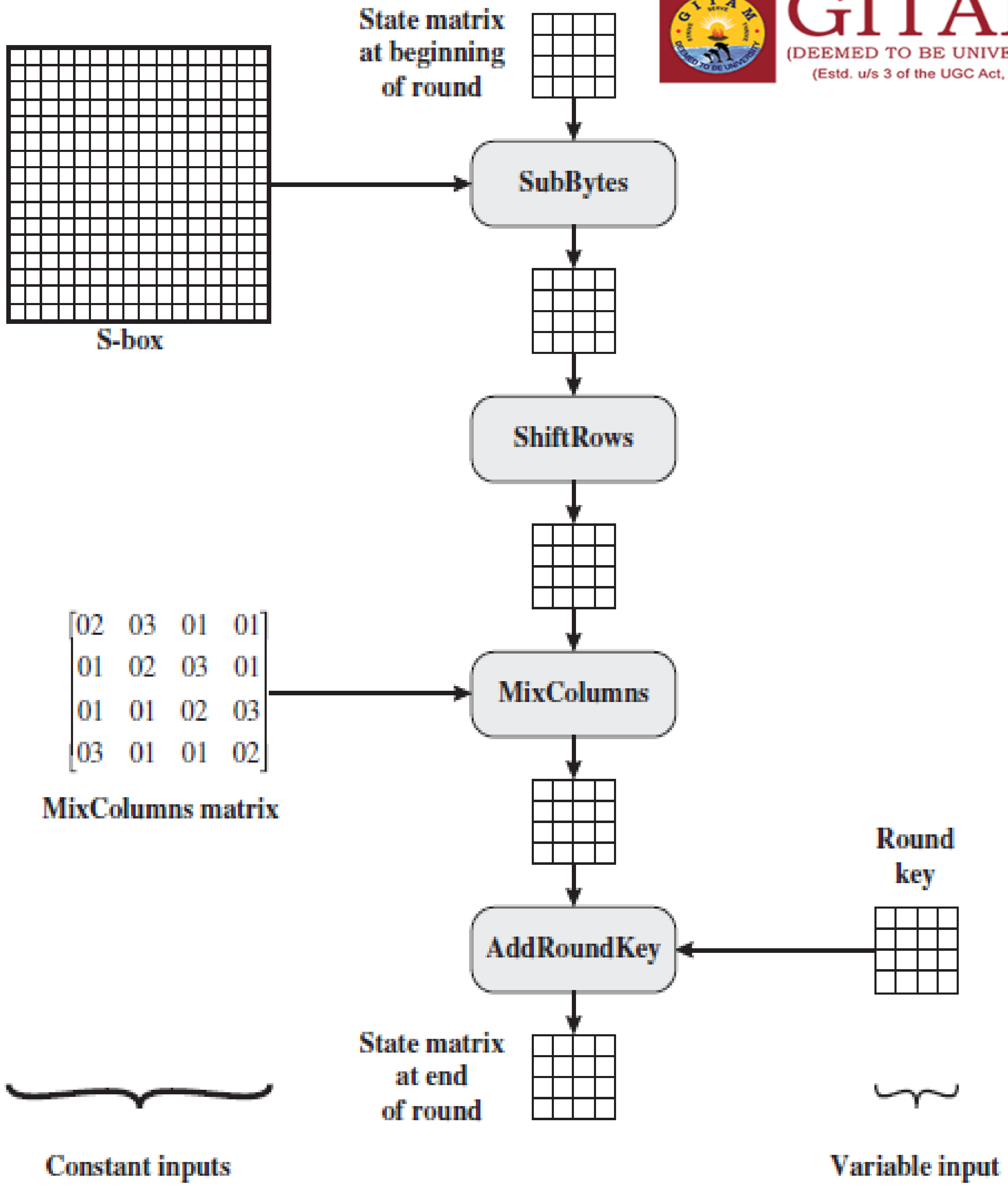
- Example:



$$\begin{bmatrix} 47 & 40 & A3 & 4C \\ 37 & D4 & 70 & 9F \\ 94 & E4 & 3A & 42 \\ ED & A5 & A6 & BC \end{bmatrix} \oplus \begin{bmatrix} AC & 19 & 28 & 57 \\ 77 & FA & D1 & 5C \\ 66 & DC & 29 & 00 \\ F3 & 21 & 41 & 6A \end{bmatrix} = \begin{bmatrix} EB & 59 & 8B & 1B \\ 40 & 2E & A1 & C3 \\ F2 & 38 & 13 & 42 \\ 1E & 84 & E7 & D6 \end{bmatrix}$$

- The first matrix is **State**, and the second matrix is the round key.
- The **inverse add round key transformation** is identical to the forward add round key transformation, because the XOR operation is its own inverse.

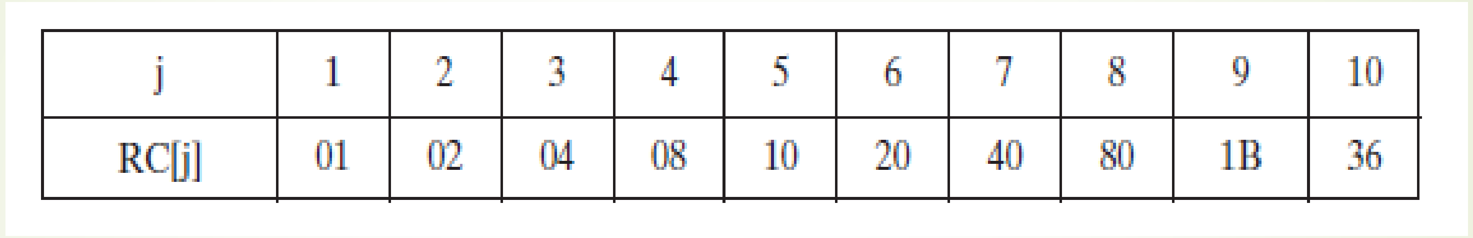
Inputs for Single AES Round



AES Key Expansion

- **Key Expansion Algorithm**
- takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).
- The function g consists of the following subfunctions.
- RotWord performs a one-byte circular left shift on a word. This means that an input word [B_0, B_1, B_2, B_3] is transformed into [B_1, B_2, $B_{3,}$ B_0].
- SubWord performs a byte substitution on each byte of its input word, using the S-box.
- The result of steps 1 and 2 is XORed with a round constant, Rcon[j].
- The round constant is a word in which the three rightmost bytes are always 0.

- The round constant is different for each round and is defined as Rcon[j] = (RC[j], 0, 0, 0), with RC[1] = 1 RC[j] = 2*RC[j-1] and with multiplication defined over the field GF(2^8).
- The values of RC[j] in hexadecimal are :

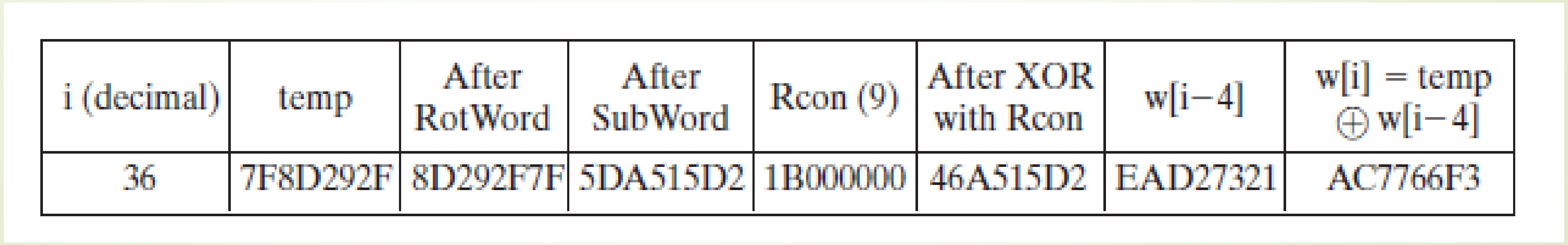


j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

- For example, suppose that the round key for round 8 is:

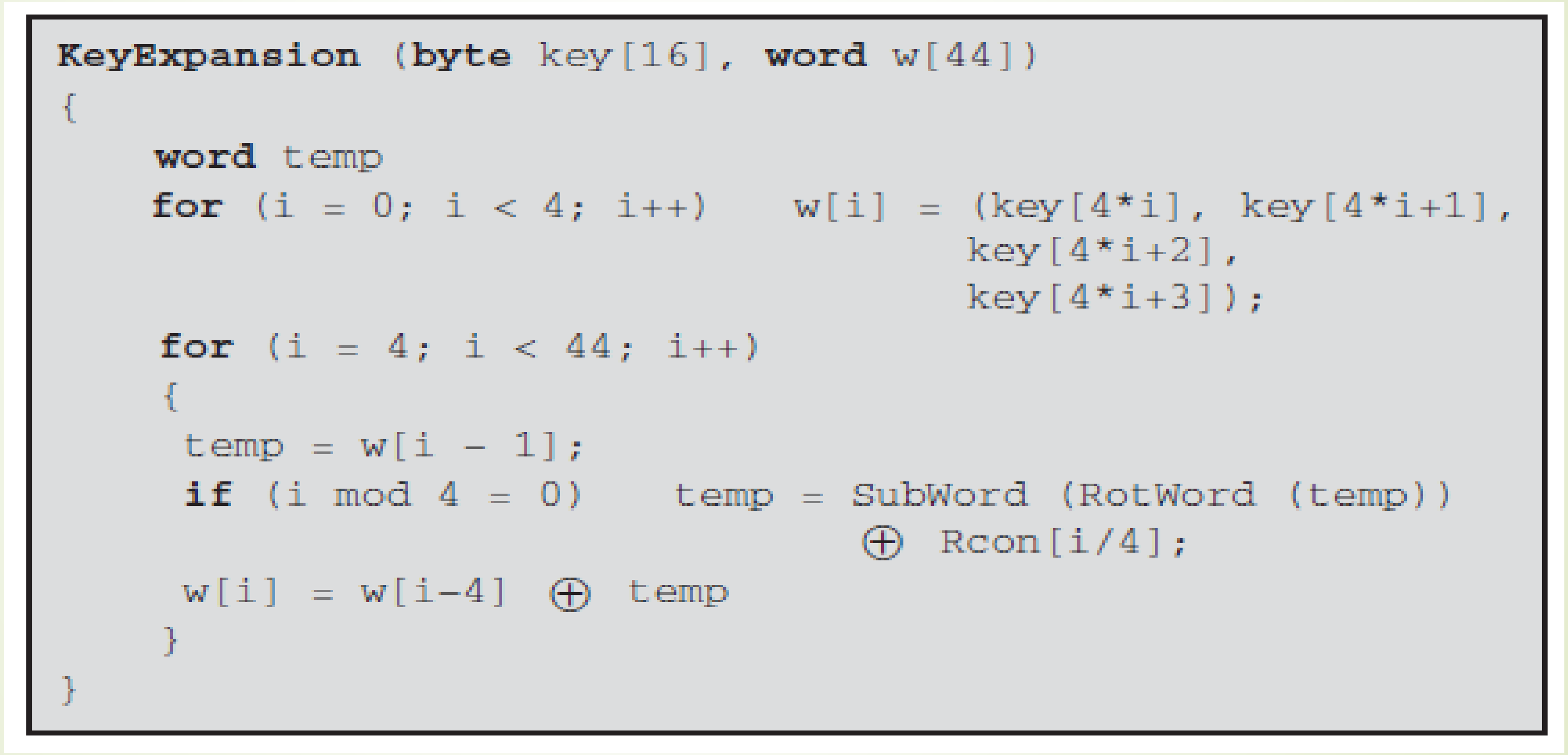
EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

- Then the first 4 bytes (first column) of the round key for round 9 are calculated as follows:



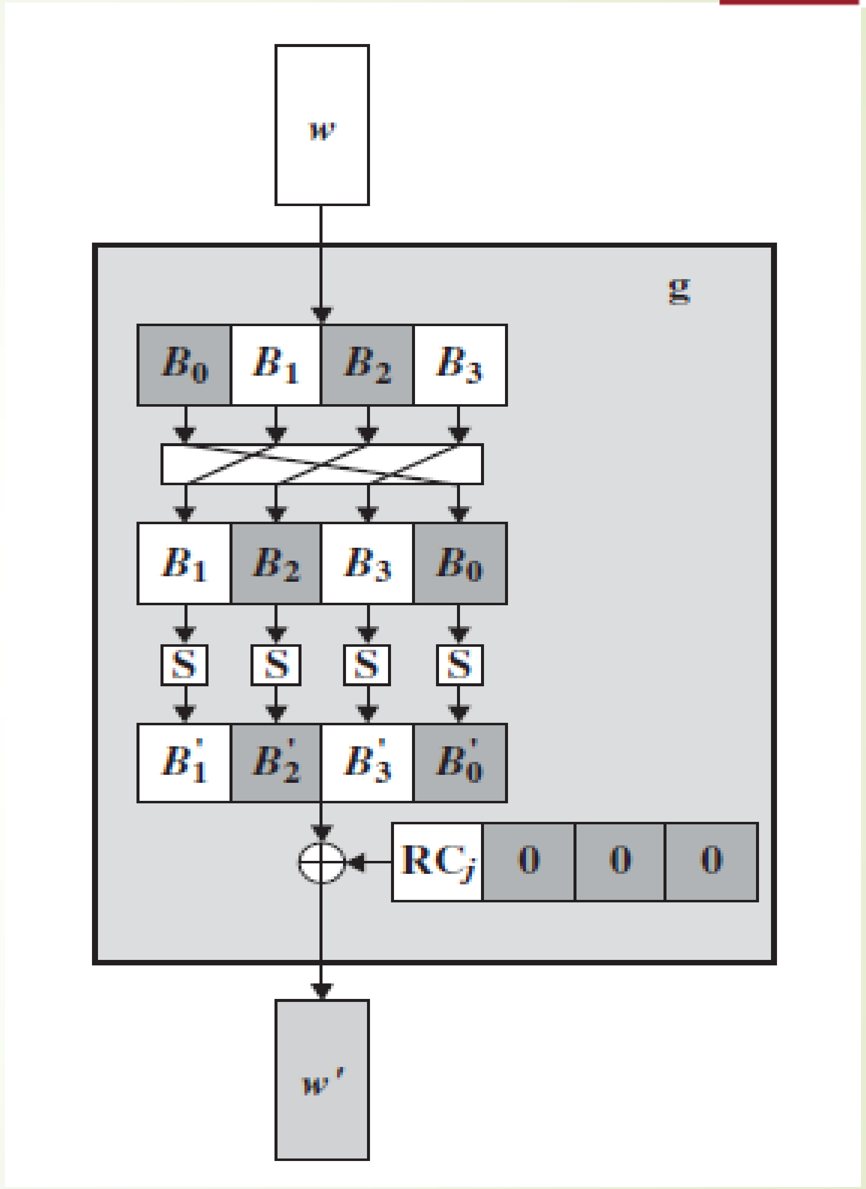
i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i−4]	w[i] = temp ⊕ w[i−4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Key Expansion Algorithm

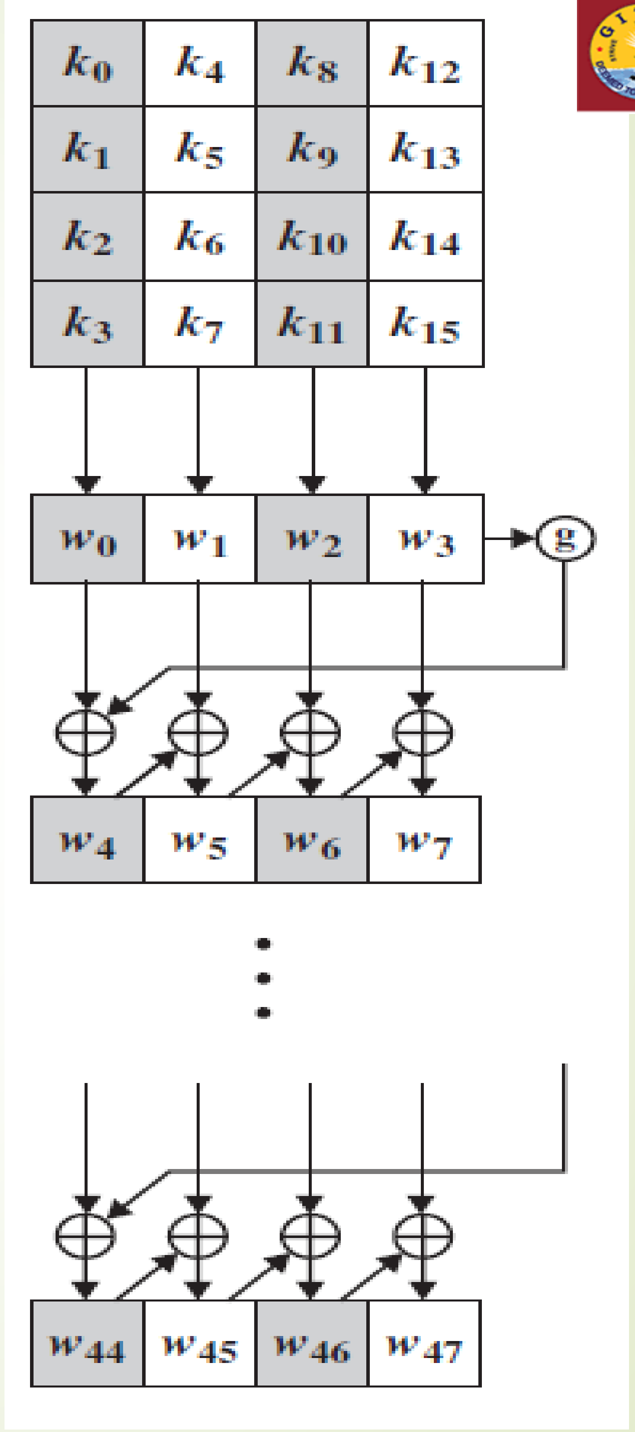


```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i], key[4*i+1],
                                       key[4*i+2],
                                       key[4*i+3]);
    for (i = 4; i < 44; i++)
    {
     temp = w[i − 1];
     if (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                                ⊕ Rcon[i/4];
     w[i] = w[i−4] ⊕ temp
    }
}
```

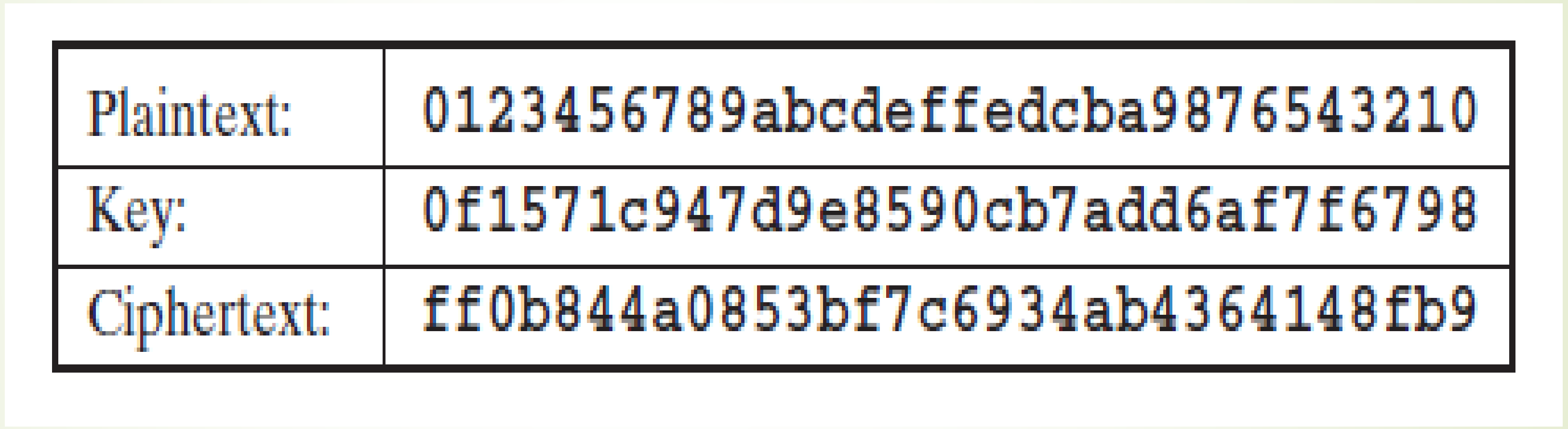
Function g



Overall algorithm



AN AES EXAMPLE



Plaintext:	0123456789abcdeffedcba9876543210
Key:	0f1571c947d9e8590cb7add6af7f6798
Ciphertext:	ff0b844a0853bf7c6934ab4364148fb9

RC6

Introduction

- RC6 (Rivest cipher 6) is a symmetric key block cipher derived from RC5.
- It was designed by Ron Rivest, Matt Robshaw, Ray Sidney, and Yiqun Lisa Yin Aug 20- 1998.
- Designed to meet the requirements of the Advanced Encryption Standard (AES).
- RC6 has a block size of 128 bits and supports key sizes of 128, 192, and 256 bits up to 2040-bits
- Improvements over RC5
  - Uses four w-bit registers
  - Integer multiplication
  - Quadratic equation
  - Fixed bit shifting

Introduction

- **Advantages** –
- RC6 is a secure, compact and simple block cipher.
  - It offers good performance and considerable flexibility.
- **Disadvantages** –
  - Null.
- **Consists of three components :**
  - Key expansion algorithm
    - Identical to RC5 version
  - Block encryption algorithm
  - Block decryption algorithm

RC6 Structure

- Specification: RC6-w/r/b
  - w is the word size in bits
  - r is the number of non-negative rounds
  - b is the key size in bits
- Round Stages
  - Pre-whitening
  - r rounds
  - Post-whitening

RC6 Structure

- Pre-whitening
  - Removes inference of part of the input to the first round of encryption
- r rounds
  - Uses integer multiplication
  - Uses a quadratic equation
    - $f(x) = x(2x + 1)(\text{mod } 2^w)$
  - Uses fixed bit shifting
  - All of the above are required for sufficient diffusion
- Post-whitening
  - Removes inference of part of the input to the last round of encryption

Key Setup

- The user supplies a key of b bytes, where $0 \leq b \leq 255$
- The key bytes are zero-padded and stored in little-endian order
  - $(2r+4)$ words are derived and stored in a round key array S

Diffusion

- Integer multiplication ensures that the bits used for rotation amounts depend on the bits of *x*, which is a word or register
- The quadratic equation increases the avalanche of changes per round
- The bit shift complicates more advanced cryptanalytic attacks

RC6 Encryption

- B and D are pre-whitened
- The loop controls the rounds defined by r
- A and C are post-whitened

Input:
Plaintext stored in four w-bit input registers A,B,C,D
Number r of rounds
w-bit round keys S[0,…,2r + 3]

Output:
Ciphertext stored in A,B,C,D

Procedure:

```
B = B + S[0]
D = D + S[1]
for i = 1 to r do
{
  t = (B x (2B + 1)) <<< $\log_2 w$
  u = (D x (2D + 1)) <<< $\log_2 w$
  A = ((A  t) <<< u) + S[2i]
  C = ((C  u) <<< t) + S[2i+ 1]
  (A,B,C,D) = (B,C,D,A)
}
A = A + S[2r + 2]
C = C + S[2r + 3]
```

RC6 Decryption

- C and A are pre-whitened
- The loop runs in reverse for r rounds
- D and B are post-whitened

Input:

Ciphertext stored in four w-bit input registers A,B,C,D

Number r of rounds

w-bit round keys $S[0,\dots,2r + 3]$

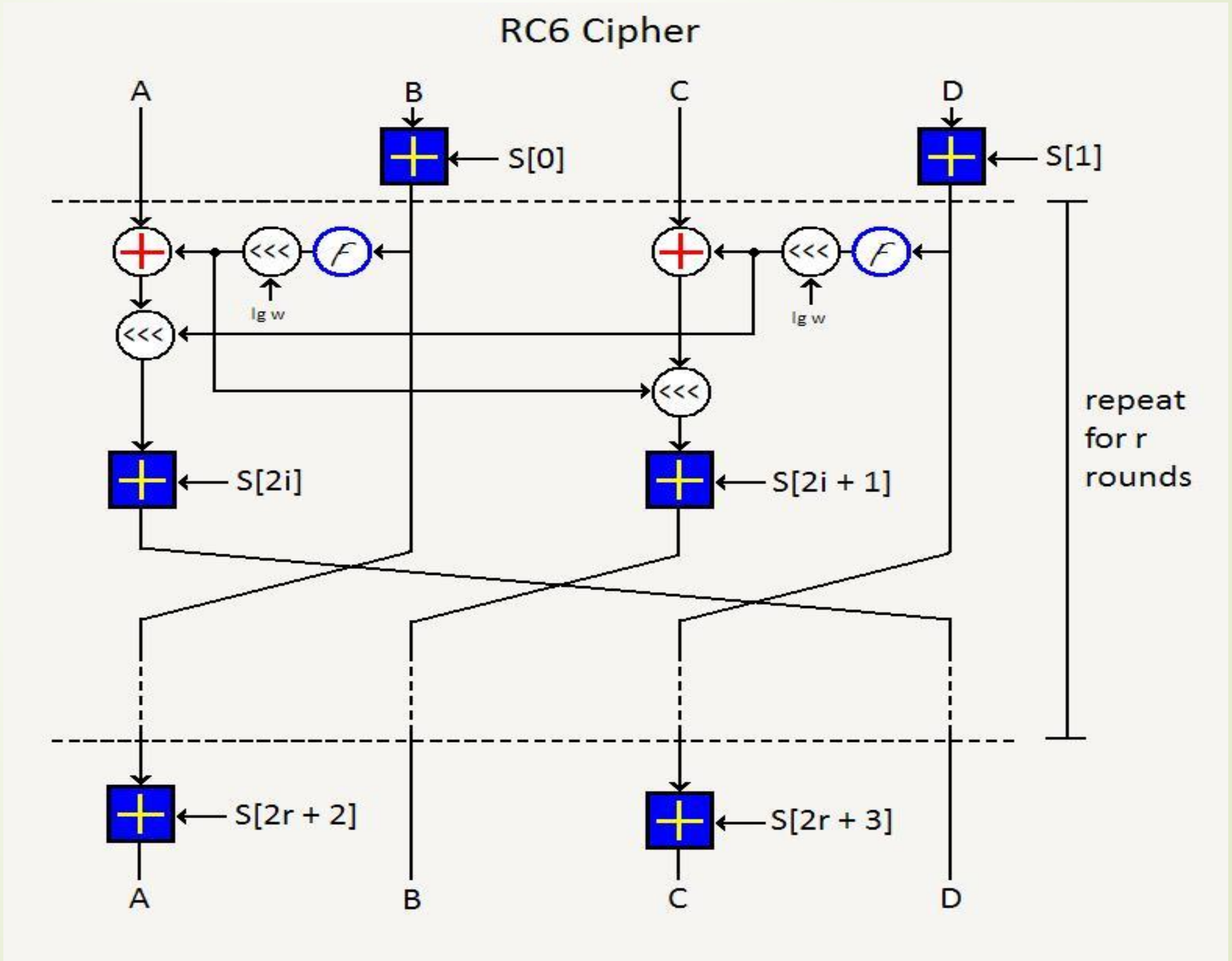
Output:

Plaintext stored in A,B,C,D

Procedure:

```
  $C = C - S[2r + 3]$
  $A = A - S[2r + 2]$
for $i = r$ downto 1 do
{
  $(A,B,C,D) = (D,A,B,C)$
  $u = (D \times (2D + 1)) <<< \log_2 w$
  $t = (B \times (2B + 1)) <<< \log_2 w$
  $C = ((C - S[2i + 1]) >>> t)\ u$
  $A = ((A - S[2i]) >>> u)\ t$
}
$D = D - S[1]$
$B = B - S[0]$
```

RC6 Algorithm



RC6 for AES

- RC6-32/20/[16,24,32]
  - 32-bit words
    - 128-bit block size / 4 registers
  - 20 rounds
  - 16, 24, and 32-bit keys are available

Input:
Plaintext stored in four w-bit input registers A,B,C,D
20 rounds
32-bit round keys S[0,…,43]

Output:
Ciphertext stored in A,B,C,D

Procedure:

```
B = B + S[0] //Pre-whitening
D = D + S[1]
for i = 1 to 20 do
{
  t = (B x (2B + 1)) <<< 5
  u = (D x (2D + 1)) <<< 5
  A = ((A  t) <<< u) + S[2i]
  C = ((C  u) <<< t) + S[2i+ 1]
  (A,B,C,D) = (B,C,D,A)
}
A = A + S[42] //Post-whitening
C = C + S[43]
```

AES Candidacy

- To meet the architectural constraints, the use of four w-bit registers permitted better 32-bit implementations
  - RC5 uses 64-bit operations and the constraints do not involve proper implementations
- The 20 rounds were chosen from linear cryptanalysis results
  - 16 rounds could be compromised with linear cryptanalysis

GITAM (DEEMED TO BE UNIVERSITY) (Estd. u/s 3 of the UGC Act, 1956)

Attacks

- All known attacks are theoretical
- Brute Force
  - Not feasible due to plaintext limit of 2^{128}
- Linear Cryptanalysis
  - Effective up to 16 rounds
- Differential Cryptanalysis
  - Effective up to 12 rounds
- Statistical Attack
  - Analyze distributions to discover round keys

Thank You