CRYPTOGRAPHY & NETWORK SECURITY(BCS703)

INTRODUCTION

- **Cryptography and Network Security** are crucial for protecting information in the digital age.

- Cryptography, at its core, **uses mathematical algorithms** to transform readable data (plaintext) into an unreadable format (ciphertext), ensuring **confidentiality and integrity** during transmission and storage.

- Network security encompasses **various measures to safeguard entire networks from unauthorized access, threats, and data breaches.**

- Together, they form a robust defense system for sensitive information.

Importance of subject

- There are two types of security covered in this syllabus:
  - Information (Data) Security
  - Network Security

- **Data Security:** Protect data from unauthorized user, it is called data security.
  - Data Security Algorithm: Symmetric & Asymmetric Cipher techniques.

- **Network Security:** Protect data during communication (Send data through secure channel), it is known as network security.
  - How to provide security during data transmission from server to client and vice-versa.
  - Use of Digital Signature and Digital certificate to achieve Security Goals (CIA).
  - Key distribution through secure channel.
  - Web Security and threats.

Course objectives:

1. Understand the basics of Cryptography concepts, Security and its principle
2. To analyse different Cryptographic Algorithms
3. To illustrate public and private key cryptography
4. To understand the key distribution scenario and certification
5. To understand approaches and techniques to build protection mechanism in order to secure computer networks

Course outcome

At the end of the course, the student will be able to :

CO1: Explain the basic concepts of Cryptography and Security aspects

CO2: Apply different Cryptographic Algorithms for different applications

CO3: Analyze different methods for authentication and access control.

CO4: Describe key management, key distribution and Certificates.

CO5: Explain about Electronic mail and IP Security.

	Module-1 10 hours
	A model for Network Security, Classical encryption techniques: Symmetric cipher model, Substitution ciphers-Caesar Cipher, Monoalphabetic Cipher, Playfair Cipher, Hill Cipher, Polyalphabetic Ciphers, One time pad, Steganography. Block Ciphers and Data Encryption Standards: Traditional Block Cipher structures, data Encryption Standard (DES), A DES Example, The strength of DES, Block cipher design principles. Chapter 1: 1.8 Chapter 3: 3.1, 3.2, 3.5 Chapter 4: 4.1, 4.2, 4.3, 4.4, 4.5
	Module-2 10 hours
	Pseudorandom number Generators: Linear Congruential Generators, Blum Blum Shub Generator. Public key cryptography and RSA: Principles of public key cryptosystems-Public key cryptosystems, Applications for public key cryptosystems, Requirements for public key cryptography, Public key Cryptanalysis, The RSA algorithm: Description of the Algorithm, Computational aspects, The Security of RSA. Diffie-Hellman key exchange: The Algorithm, Key exchange Protocols, Man-in-the-middle Attack, Elliptic Curve Cryptography: Analog of Diffie-Hellman key Exchange, Elliptic Curve Encryption/Decryption, Security of Elliptic Curve Cryptography. Chapter 8: 8.2 Chapter 9: 9.1, 9.2 Chapter 10: 10.1, 10.4

	Module-3 10 hours
	Applications of Cryptographic Hash functions, Two simple Hash functions, Key management and distribution: Symmetric key distribution using symmetric encryption, Symmetric key distribution using asymmetric encryption, Distribution of public keys, X.509 Certificates, Public Key Infrastructures Chapter 11: 11.1, 11.2 Chapter 14: 14.1, 14.2, 14.3, 14.4, 14.5
	Module-4 10 hours
	User Authentication: Remote user authentication principles, Kerberos, Remote user authentication using asymmetric encryption. Web security consideration, Transport layer security. Email Threats and comprehensive email security, S/MIME, Pretty Good Privacy. Chapter 15: 15.1, 15.3, 15.4 Chapter 17: 17.1, 17.2 Chapter 19: 19.3, 19.4, 19.5
	Module-5 10 hours
	Domainkeys Identified Mail. IP Security: IP Security overview, IP Security Policy, Encapsulating Security Payload, Combining security associations, Internet key exchange.

Books

Text Books:

William stallings, "Cryptography and Network Security", Pearson Publication, Seventh Edition.

References:

1. Keith M Martin, "Everyday Cryptography", Oxford University Press
2. V.K Pachghare,"Cryptography and Network Security", PHI, 2nd Edition

Course Outcomes:

CO	Description	RBT
C403.1	Analyze Classical and Symmetric Encryption Techniques for security applications.	L4
C403.2	Apply Public Key Cryptography and Key Exchange Protocols in modern secure communication systems.	L3
C403.3	Develop Secure Systems Using Cryptographic Hash Functions and Key Management techniques.	L3
C403.4	Analyze and evaluate the user authentication and Web Security Mechanisms.	L3
C403.5	Examine the different Network Security Protocols.	L4

Continuous Internal Evaluation:

- For the Assignment component of the CIE, there are 25 marks and for the Internal Assessment Test component, there are 25 marks.
- The first test will be administered after 40-50% of the syllabus has been covered, and the second test will be administered after 85-90% of the syllabus has been covered
- Any two assignment methods mentioned in the 22OB2.4, if an assignment is project-based then only one assignment for the course shall be planned. The teacher should not conduct two assignments at the end of the semester if two assignments are planned.
- For the course, CIE marks will be based on a scaled-down sum of two tests and other methods of assessment.

Internal Assessment Test question paper is designed to attain the different levels of Bloom's taxonomy as per the outcome defined for the course.

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

- Group assignment (TWO) to implement Cryptographic Algorithms (15 + 10 marks)

Cryptography:

- Cryptography is the practice of securing communication by converting readable information into an unreadable format, making it incomprehensible to unauthorized parties.

Key Concepts:

- **Encryption:** The process of converting plaintext into ciphertext using an algorithm and a key.
- **Decryption:** The reverse process of transforming ciphertext back into plaintext using the same key and algorithm.
- **Key:** A piece of information used in both encryption and decryption processes.

Types of Cryptography:

- **Symmetric-key cryptography:** Uses the same key for both encryption and decryption.
- **Asymmetric-key cryptography:** Employs separate keys for encryption and decryption (public and private keys).
- **Hash functions:** Create unique fingerprints of data, used for integrity checks.

Network Security:

- Network security refers to the measures implemented to protect a network and its resources from unauthorized access, misuse, or damage.

Key Concepts:

- **Confidentiality:** Ensuring that information is accessible only to authorized individuals.
- **Integrity:** Guaranteeing that data remains accurate and hasn't been tampered with.
- **Availability:** Ensuring that authorized users can access network resources when needed.
- **Authentication:** Verifying the identity of users and devices accessing the network.

Components of Network Security:

- **Firewalls:** Act as barriers between the network and the outside world, filtering traffic based on predefined rules.
- **Intrusion Detection Systems (IDS):** Monitor network traffic for suspicious activity and alert administrators.
- **Access control:** Restricts access to network resources based on user roles and permissions.
- **Virtual Private Networks (VPNs):** Create secure, encrypted connections over public networks.

CRYPTOGRAPHY & NETWORK SECURITY(BCS703)

MODULE-1

A model for Network Security,

Classical encryption techniques: Symmetric cipher model, Substitution ciphers-Caesar Cipher, Monoalphabetic Cipher, Playfair Cipher, Hill Cipher, Polyalphabetic Ciphers, One time pad, Steganography.

Block Ciphers and Data Encryption Standards: Traditional Block Cipher structures, data Encryption Standard (DES), A DES Example, The strength of DES, Block cipher design principles.

A MODEL FOR NETWORK SECURITY

- A message is to be transferred from one party to another across some sort **of Internet service.**

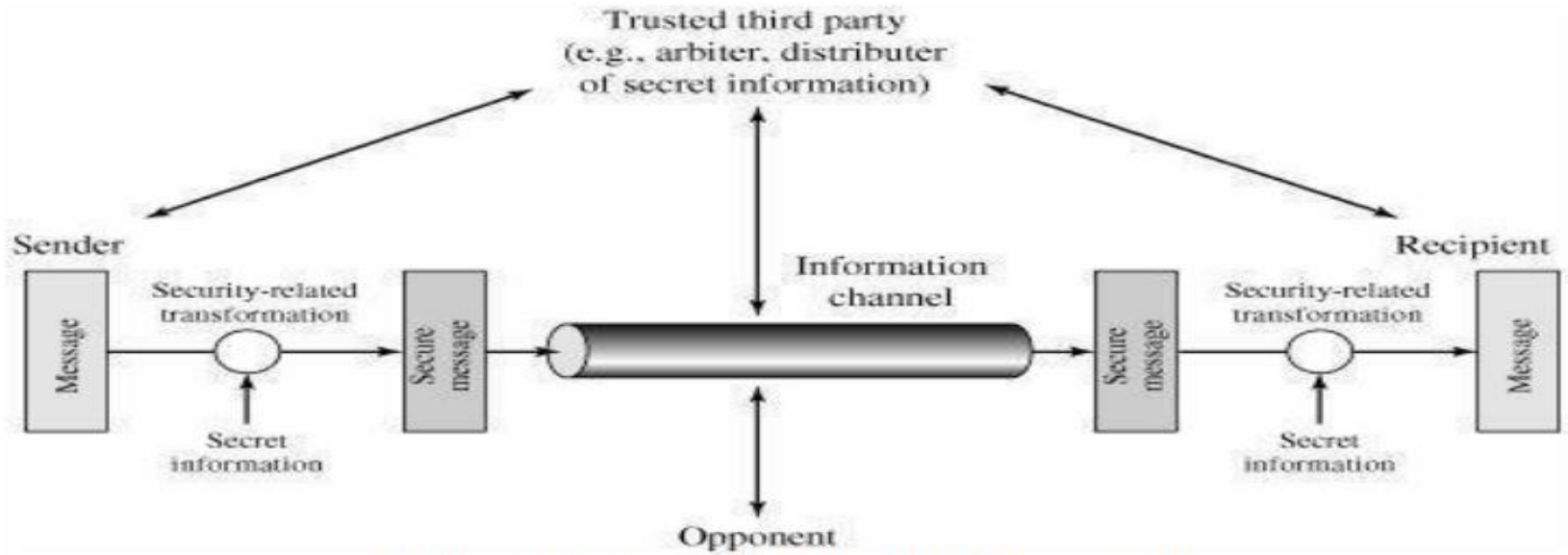


Figure 1.3 Model for Network Security

A **security-related transformation** on the information to be sent, Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the **addition of a code based on the contents of the message, which can be used to verify the identity of the sender.**

Some **secret information** shared by the two principals and, it is hoped, unknown to the opponent. An example is an **encryption key** used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

All the techniques for providing security have two components:

This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation.

The algorithm should be such that an opponent cannot defeat its purpose.

2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of the secret information.

4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service

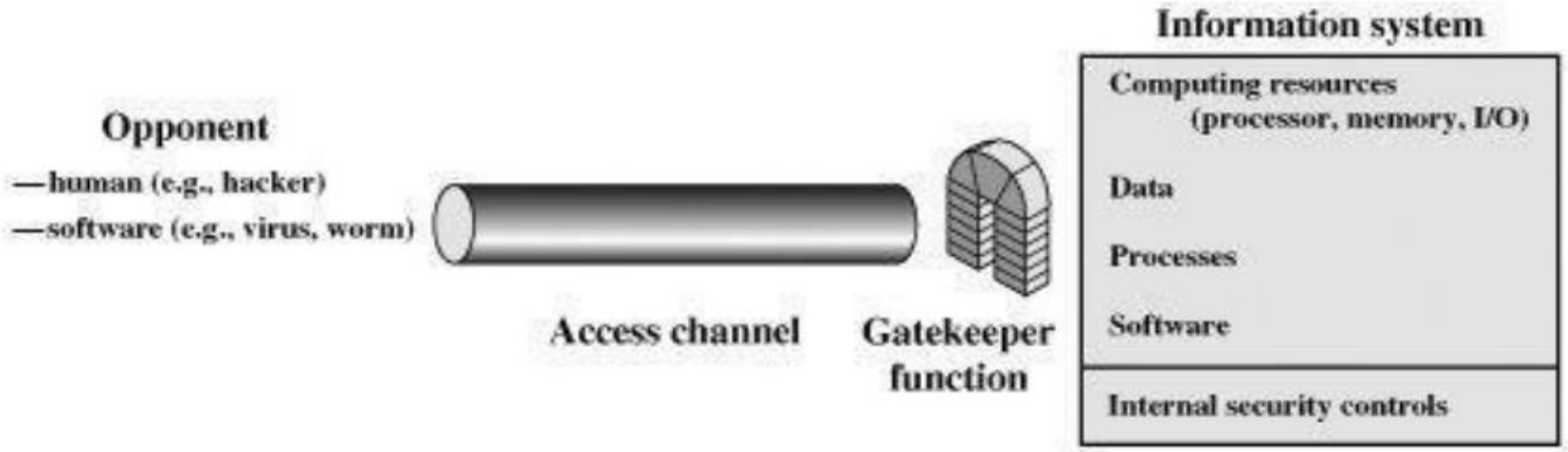


Figure 1.4 Network Access Security Model

A general model(Figure 1.4), which reflects a **concern for protecting an information system from unwanted access**. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

- The **security mechanisms** needed to cope with unwanted access fall into two broad categories (Figure 1.4).
- The first category might be termed a **gatekeeper function**. It includes **password-based login procedures** that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access,
- The second line of defense consists of a variety of **internal controls** that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

CLASSICAL ENCRYPTION TECHNIQUES

Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption.

- Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- The two types of attack on an encryption algorithm are cryptanalysis, based on properties of the encryption algorithm, and brute-force, which involves trying all possible keys.
- Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements.

- Rotor machines are sophisticated precomputer hardware devices that use substitution techniques.
- Steganography is a technique for hiding a secret message within a larger one in such a way that others cannot discern the presence or contents of the hidden message.

An original message is known as the **plaintext**, while the coded message is called the **ciphertext**. The process of converting from plaintext to ciphertext is known as **enciphering** or **encryption**; restoring the plaintext from the ciphertext is **deciphering** or **decryption**. The many schemes used for encryption constitute the area of study known as **cryptography**.

Such a scheme is known as a **cryptographic system** or a **cipher**. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls "breaking the code" The areas of cryptography and cryptanalysis together are called **cryptology**.

SYMMETRIC CIPHER MODEL

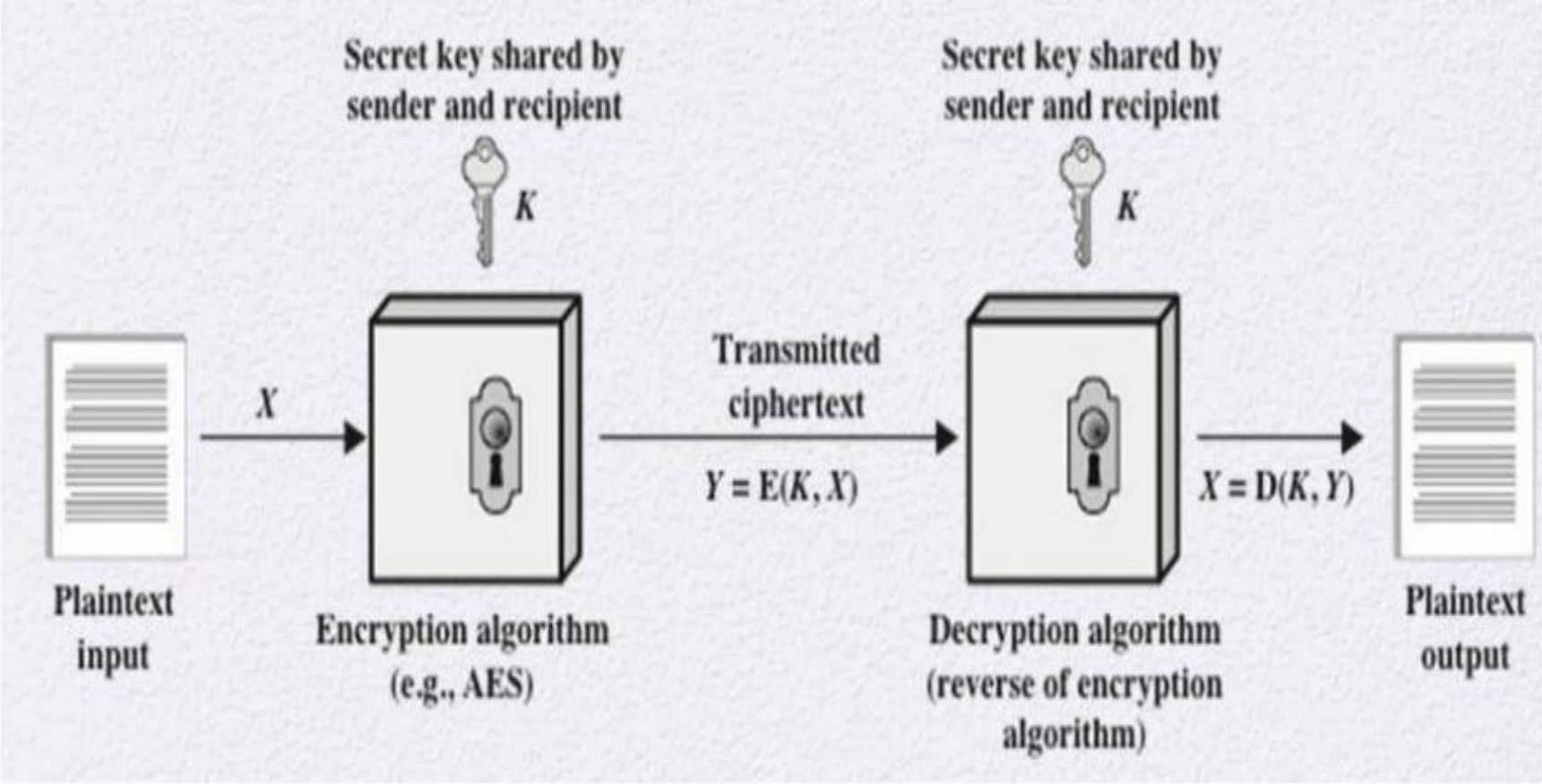


Figure 1.7 Simplified Model of Symmetric Encryption

A symmetric encryption scheme has five ingredients (Figure 1.7):

- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plaintext.

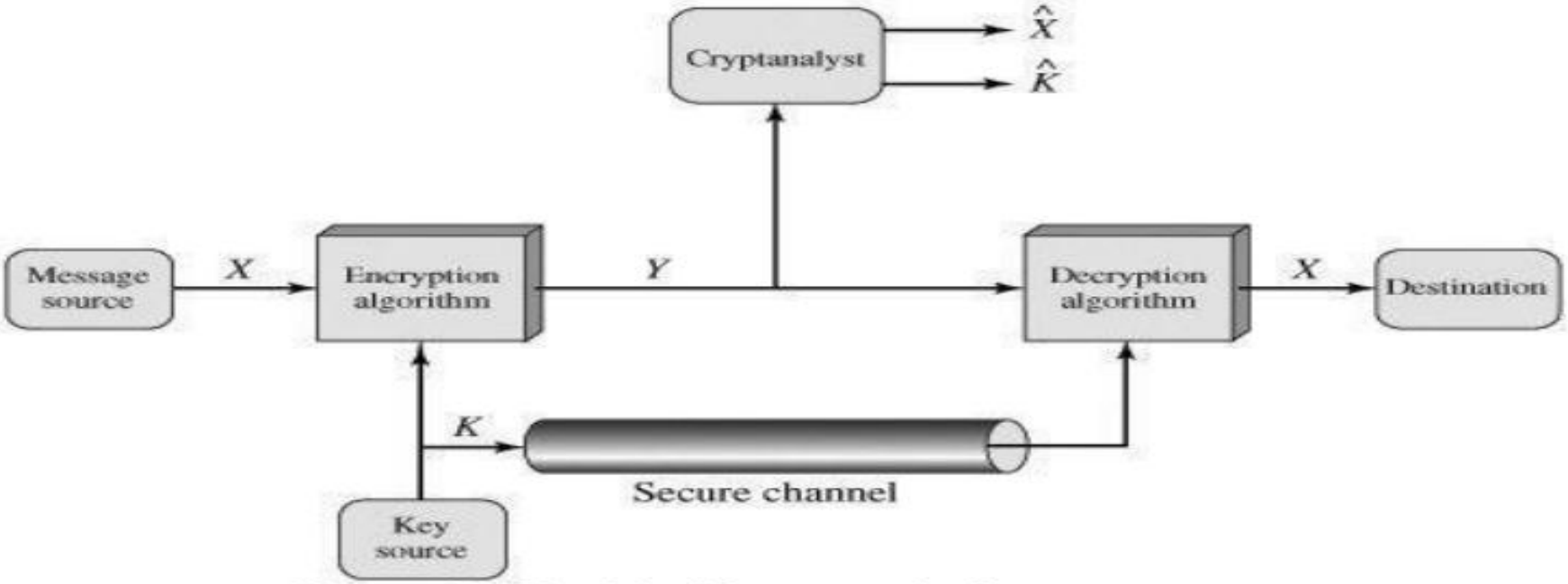


Figure 1.8 Model of Symmetric Cryptosystem

With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext Y=[Y1,Y2,.....Y_N] .We can write this as Y=E(K,X)This notation indicates that is produced by using encryption algorithm E as a function of the plaintext X , with the specific function determined by the value of the key K .

The intended receiver, in possession of the key, is able to invert the transformation:

X=D(K,Y)

An opponent, observing Y but not having access K to X or, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate X. Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate K.

There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

Cryptanalysis and Brute-Force Attack

- The objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single ciphertext.
- There are two general approaches to attacking a conventional encryption scheme:
  - **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.
  - **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

Substitution Techniques

- The two basic building blocks of all encryption techniques are substitution and transposition.

- A substitution technique is one in which the **letters of plaintext are replaced by other letters or by numbers or symbols**.
- If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

Caesar Cipher

- The earliest known, and the simplest, use of a substitution cipher was by **Julius Caesar.** The Caesar cipher involves replacing each letter of the alphabet **with the letter standing three places further down the alphabet**.

```
plain:  a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

```
plain:   meet me after the toga party
cipher:  PHHW PH DIWHU WKH WRJD SDUWB
```

To cipher a given text, we need an integer value, known as a shift/key. The shift indicates the number of spots each letter of the text has been moved down. The encryption can be represented using modular arithmetic by first transforming the letters into numbers

Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter, substitute the cipher text letter:

$$C = E(3, p) = (p + 3) \bmod 26$$

A shift may be of any amount, so that the general Caesar algorithm is

$$C = E(k, p) = (p + k) \bmod 26$$

where takes on a value in the range 1 to 25.The decryption algorithm is simply

$$p = D(k, C) = (C - k) \bmod 26$$

Let's encrypt the phrase "Go to Valley" using the Caesar Cipher with a key of 9.

Plain Text: Go to Valley

G : position of G is 6

C=(6+9)mod 26=15 i.e p

o: =(14+9)mod 26 =23mod 26=23 i.e x

t: =(19+9)mod26= 28 mod 26= (28-26)=2 ie c.

Continuing the same for the remaining characters we get cipher text as:

Px Cx EJUUNH

Monoalphabetic Cipher

- A type of simple substitution cipher in which each letter of the plaintext is replaced with the other 25 characters and the order of replaced is not fixed.

Plaintext Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M
Ciphertext Alphabet	Q	W	E	R	T	Y	U	I	O	P	A	S	D
Plaintext Alphabet	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Ciphertext Alphabet	F	G	H	J	K	L	Z	X	C	V	B	N	M

Ex:

- Encrypt the word ATTACK using monoalphabetic cipher

Cipher text: QZZQEA

As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English, such as is shown in Figure 1.9. If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match. In any case, the relative frequencies of the letters in the ciphertext (in percentages) are as follows:

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

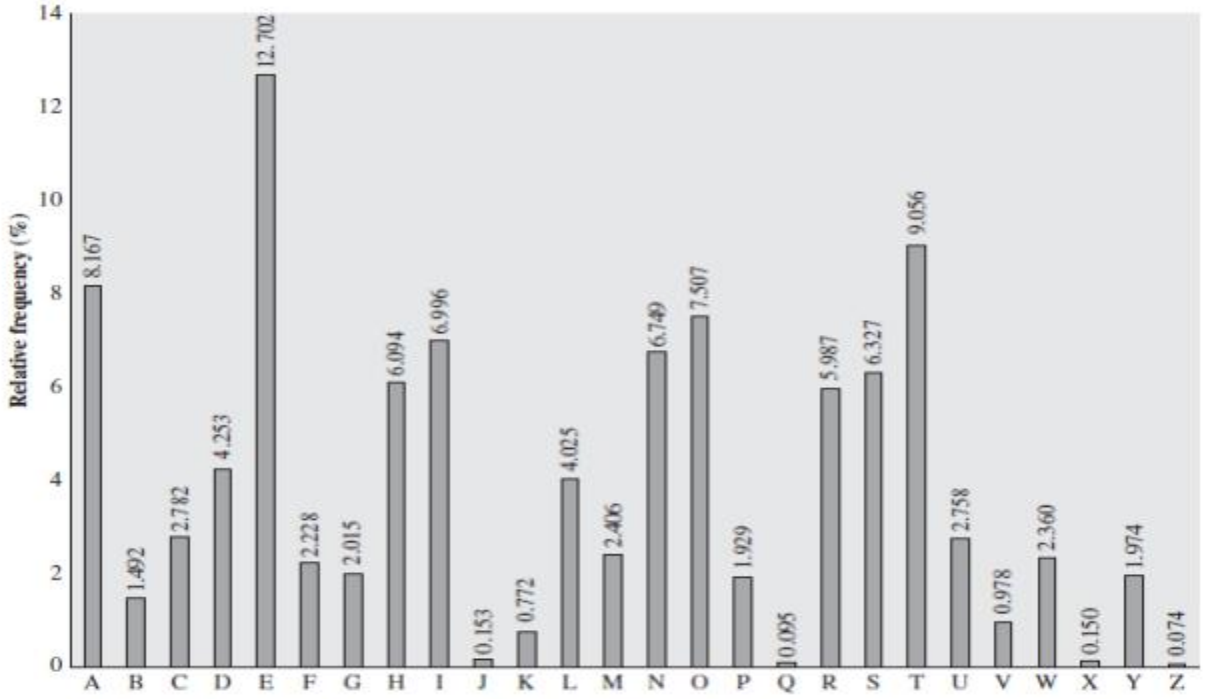


Figure 1.10 Relative Frequencies of Letters in English Text

Playfair Cipher

- The best-known **multiple-letter encryption cipher is the Playfair**, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams.

Encryption Technique

The algorithm consists of 2 steps:

1. ***Generate the key Square (5x5):***
   - *The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.*
   - *The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.*

2. **Algorithm to encrypt the plain text:** *The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.*

***PlainText**: "instruments"*
***After Split:** 'in' 'st' 'ru' 'me' 'nt' 'sz'*
***Explanation:** Pair cannot be made with same letter. Break the letter in single and add a bogus letter to the previous letter. Here **'z'** is the bogus letter.*

***Plain Text:** "hello"*
***After Split:** 'he' 'lx' 'lo'*
***Explanation:** Here **'x'** is the bogus letter.*

***Plain Text:** "helloe"*
***After Split:** 'he' 'lx' 'lo' 'ez'*
***Explanation:** If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter. Here **'x'** and **'z'** are the bogus letters.*

Rules for Encryption

- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last

For example:
Diagraph: *"me"*
Encrypted Text: *cl*
Encryption: *m -> c e -> l*

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.

For example:
Diagraph: *"st"*
Encrypted Text: *tl*
Encryption: *s -> t t -> l*

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

***If neither of the above rules is true**: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.*

For example:
***Diagraph:** "nt"*
***Encrypted Text:** rq*
***Encryption:** n -> r t -> q*

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

For example:

Plain Text: "instrumentsz"

Encrypted Text: gatlmzclrqtx

Encryption:

i -> g
n -> a
s -> t
t -> l
r -> m
u -> z
m -> c
e -> l
n -> r
t -> q
s -> t
z -> x

in:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

st:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

ru:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

me:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

nt:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

sz:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Explain Playfair Cipher Algorithm. Find the Ciphertext for plaintext = "instruments" with key = "MONARCHY". **(10 Marks)**

A message received at an Australian wireless station in play fair code : KXJEY UREBE key used was ROYAL NEW ZEALAND NAVY. Decrypt the message. **(08 Marks)**

Encrypt he plaintext "ELECTRONICS" using a playfair cipher with a key "INDIA". **(04 Marks)**

Hill Cipher

- Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26.
- the Hill Cipher uses a polygraphic substitution cipher, which means homogeneous substitution over many levels of blocks.
- This polygraphic substitution cipher allows Hill Cipher to function easily with digraphs (two-letter blocks), trigraphs (three-letter blocks), or any other multiple-sized blocks to create a uniform cipher.
- Hill Cipher is based on linear algebra, advanced matrices (matrix multiplication and matrix inverses), and modulo arithmetic principles.

Encryption

- Encrypting using the Hill cipher depends on the following operations –

E(K, P) = (K*P) mod 26

- Here K is our key matrix, and P is the vectorized plaintext.

$$(c_1\ c_2\ c_3) = (p_1\ p_2\ p_3)\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

$$\mathbf{C} = \mathbf{PK} \bmod 26$$

- For example, consider the plaintext “paymoremoney” and use the encryption key

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

- Check the key matrix size and split the plain text
- The first three letters of the plaintext “pay” are represented by the vector (15 0 24).

C = PK mod 26

Plaintext:

P	15
A	0
y	24

M	12
O	14
r	17

E	4
M	12
o	17

N	13
E	4
Y	24

Calculate the C = KP mod 26

$$Cpay = \begin{bmatrix} 15 & 0 & 24 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 255 + 0 + 48 \\ 255 + 0 + 48 \\ 75 + 0 + 456 \end{bmatrix} = \begin{bmatrix} 303 \\ 303 \\ 531 \end{bmatrix} \text{ take mod 26.}$$

C = (17 17 11) ➔ (R R L)

$$Cmor = \begin{bmatrix} 12 & 14 & 17 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 204 + 294 + 34 \\ 204 + 252 + 34 \\ 60 + 294 + 323 \end{bmatrix} = \begin{bmatrix} 532 \\ 490 \\ 677 \end{bmatrix} \text{ take mod 26.}$$

C = (12 22 1) ➔ (M W B)

$$Cemo = \begin{bmatrix} 4 & 12 & 14 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 68 + 252 + 28 \\ 68 + 216 + 28 \\ 20 + 252 + 266 \end{bmatrix} = \begin{bmatrix} 348 \\ 312 \\ 538 \end{bmatrix} \text{ take mod 26.}$$

C = (10 0 18) ➔ (K A S)

$$Cney = \begin{bmatrix} 13 & 4 & 24 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 221 + 84 + 48 \\ 221 + 72 + 48 \\ 65 + 84 + 456 \end{bmatrix} = \begin{bmatrix} 353 \\ 341 \\ 605 \end{bmatrix} \text{ take mod 26.}$$

C = (15 3 7) ➔ (P D H)

CIPHERTEXT : (RRL MWB KAS PDH)

- Decryption requires using the inverse of the matrix K $P = CK^{-1} \bmod 26$

Hill Cipher Decryption with Steps

Step 1: Find the Inverse of the Key Matrix

- Given key matrix:

```
K = 3 2
    5 7

Adjoint_K = 7 -2
           -5  3
```

Calculate the determinant:

```
det(K) = (3 * 7) - (2 * 5) = 21 - 10 = 11
```

The modular inverse of 11 modulo 26 is 19.

Inverse matrix:

To find the inverse of matrix K modulo 26:

Given matrix K:

$$K = \begin{bmatrix} 3 & 2 \\ 5 & 7 \end{bmatrix}$$

1. Calculate the determinant:

$$\det(K) = (3 * 7) - (2 * 5) = 21 - 10 = 11$$

2. Find the modular multiplicative inverse of the determinant modulo 26:

$$11 * x \equiv 1 \pmod{26}$$

The modular inverse of 11 modulo 26 is 19, because:

$$11 * 19 \equiv 1 \pmod{26}$$

3. Compute the adjoint (or adjugate) matrix of K:

$$\text{Adj}(K) = \begin{bmatrix} 7 & -2 \\ -5 & 3 \end{bmatrix}$$

4. Calculate K^(-1) modulo 26:

$$K^{-1} = (1/\det(K)) * \text{Adj}(K) = 19 * \begin{bmatrix} 7 & -2 \\ -5 & 3 \end{bmatrix}$$

Perform the multiplication:

$$19 * \begin{bmatrix} 7 & -2 \\ -5 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 19 * 7 & 19 * (-2) \\ 19 * (-5) & 19 * 3 \end{bmatrix}$$

$$= \begin{bmatrix} 133 & -38 \\ -95 & 57 \end{bmatrix}$$

Reduce each element modulo 26:
Now we reduce each element modulo 26 to find their values within the range from 0 to 25:

For 133 mod 26:
133 ÷ 26 = 5 is quotient 3 is remainder
So, 133 mod 26 = 3.

For -38 mod 26:
-38 ÷ 26 = -1 is quotient -12 is remainder
To get a positive remainder, add 26:
-12 + 26 = 14
So, -38 mod 26 = 14.

For -95 mod 26:
-95 ÷ 26 = -3 is quotient -17 is remainder

To get a positive remainder, add 26:
-17 + 26 = 9

For 57 mod 26:
57 ÷ 26 = 2 is quotient and remainder is 5
So, 57 mod 26 = 5.

So, the inverse of K modulo 26 is:

$$K^{-1} \equiv \begin{bmatrix} 3 & 14 \\ 9 & 5 \end{bmatrix} \pmod{26}$$

- **Using Hill Cipher to encipher and decipher the message “Hi”. Use the key (03 02**

  05 07)

use A =0,B =1,...,Z =25

H = 7,I = 8

plaintext vector:P =[7

8]

Encryption :

Formula:

$$C = K \cdot P \pmod{26}$$

$$\begin{bmatrix} 3 & 2 \\ 5 & 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 3 \cdot 7 + 2 \cdot 8 \\ 5 \cdot 7 + 7 \cdot 8 \end{bmatrix} = \begin{bmatrix} 21 + 16 \\ 35 + 56 \end{bmatrix} = \begin{bmatrix} 37 \\ 91 \end{bmatrix}$$

Take mod 26:

- 37 mod 26 = 11 → **L**
- 91 mod 26 = 13 → **N**

So ciphertext = **"LN"**

Decryption :

We need $K^{-1} \pmod{26}$.

1. **Determinant:**

$$\det(K) = (3 \cdot 7 - 2 \cdot 5) = 21 - 10 = 11$$

2. **Find inverse of determinant mod 26:**

   We need $11^{-1} \pmod{26}$.

   Since $11 \times 19 = 209 \equiv 1 \pmod{26}$,

$$11^{-1} \equiv 19 \pmod{26}$$

3. **Adjugate of K:**

$$\text{adj}(K) = \begin{bmatrix} 7 & -2 \\ -5 & 3 \end{bmatrix}$$

4. **Multiply by determinant inverse (19) mod 26:**

$$K^{-1} \equiv 19 \cdot \begin{bmatrix} 7 & -2 \\ -5 & 3 \end{bmatrix} \pmod{26}$$

$$= \begin{bmatrix} 133 & -38 \\ -95 & 57 \end{bmatrix}$$

Reduce mod 26:

- 133 mod 26 = 3
- -38 mod 26 = 14
- -95 mod 26 = 15
- 57 mod 26 = 5

$$K^{-1} = \begin{bmatrix} 3 & 14 \\ 15 & 5 \end{bmatrix}$$

5. **Decrypt ciphertext (L=11, N=13):**

$$C = \begin{bmatrix} 11 \\ 13 \end{bmatrix}$$

$$P = K^{-1} \cdot C \pmod{26} = \begin{bmatrix} 3 & 14 \\ 15 & 5 \end{bmatrix} \begin{bmatrix} 11 \\ 13 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \cdot 11 + 14 \cdot 13 \\ 15 \cdot 11 + 5 \cdot 13 \end{bmatrix} = \begin{bmatrix} 33 + 182 \\ 165 + 65 \end{bmatrix} = \begin{bmatrix} 215 \\ 230 \end{bmatrix}$$

Modulo 26:

215 mod 26 = 7 → H

230 mod 26 = 8 → I

plaintext = **"HI"**

Explain Hill Cipher Algorithm. Using Hill-Cipher perform encryption and decryption for plaintext = "paymoremoney" using key $K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$. **(10 Marks)**

Encrypt the message "Meet me at the usual place at ten rather than eight O'clock". Using the hill cipher with key $\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$. Show your calculation and result. **(10 Marks)**

Encrypt the message "MAM" using Hill cipher with key

$$A = \begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}.$$

Further show the calculations for corresponding decryption of Cipher text to recover plain text. **(08 Marks)**

Encrypt the plaintext "CRYPTOGRAPHY" using HILL CIPHER technique with key matrix $K = \begin{bmatrix} 9 & 4 \\ 5 & 7 \end{bmatrix}$ and decrypt the same. **(10 Marks)**

Polyalphabetic Ciphers

- One of the way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message.

Vigenere Cipher

- The vigenere cipher is an algorithm that uses a number of linked caesar ciphers to encrypt an alphabetic text.
- It is based on the alphabets of a keyword. This cipher is a representation of a polyalphabetic substitution.
- It is simple to understand and use this algorithm.

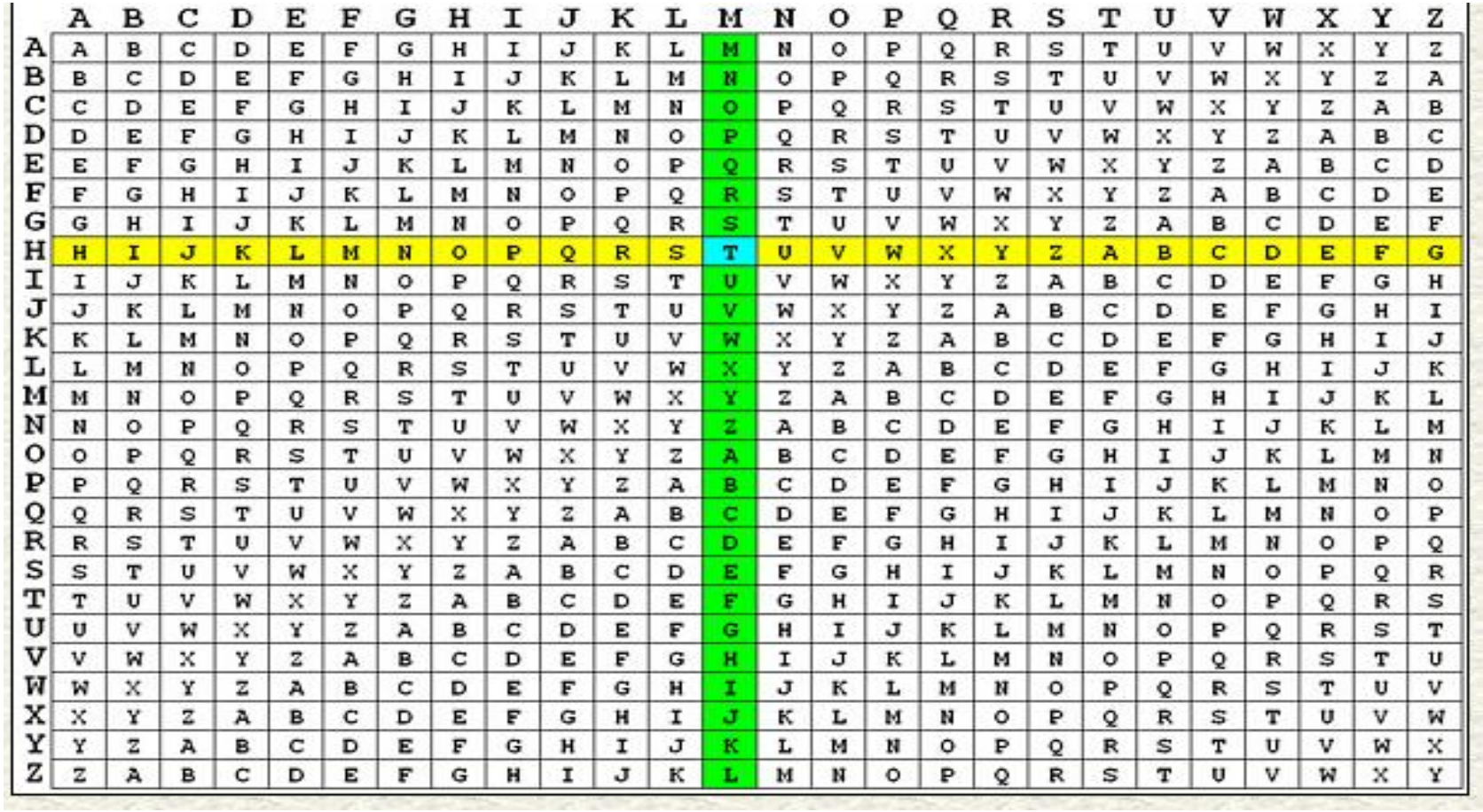
- The Vigenère cipher uses a 26×26 table with A to Z as the row heading and column heading This table is usually referred to as the **Vigenère Tableau, Vigenère Table or Vigenère Square**. We shall use Vigenère Table.

- The first row of this table has the 26 English letters. Starting with the second row, each row has the letters shifted to the left one position in a cyclic way.

- For example, when B is shifted to the first position on the second row, the letter A moves to the end.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

For example, suppose the plaintext is **MICHIGAN TECHNOLOGICAL UNIVERSITY** and the keyword is **HOUGHTON**. Then, the keyword must be repeated as follows:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

MICHI GANTE CHNOL OGICA LUNIV ERSIT Y
HOUGH TONHO UGHTO NHOUG HTONH OUGHT O
TWWNP ZOAAS WNUHZ BNWWG SNBVC SLYPM M

We can express the Vigenère cipher in the following manner. Assume a sequence of plaintext letters $P = p_0, p_1, p_2, \ldots, p_{n-1}$ and a key consisting of the sequence of letters $K = k_0, k_1, k_2, \ldots, k_{m-1}$, where typically $m < n$. The sequence of ciphertext letters $C = C_0, C_1, C_2, \ldots, C_{n-1}$ is calculated as follows:

$$C = C_0, C_1, C_2, \ldots, C_{n-1} = \mathrm{E}(K, P) = \mathrm{E}[(k_0, k_1, k_2, \ldots, k_{m-1}), (p_0, p_1, p_2, \ldots, p_{n-1})]$$
$$= (p_0 + k_0) \bmod 26, (p_1 + k_1) \bmod 26, \ldots, (p_{m-1} + k_{m-1}) \bmod 26,$$
$$(p_m + k_0) \bmod 26, (p_{m+1} + k_1) \bmod 26, \ldots, (p_{2m-1} + k_{m-1}) \bmod 26, \ldots$$

Thus, the first letter of the key is added to the first letter of the plaintext, mod 26, the second letters are added, and so on through the first m letters of the plaintext. For the next m letters of the plaintext, the key letters are repeated. This process continues until all of the plaintext sequence is encrypted. A general equation of the encryption process is

$$C_i = (p_i + k_{i \bmod m}) \bmod 26 \qquad \textbf{(3.3)}$$

Compare this with Equation (3.1) for the Caesar cipher. In essence, each plaintext character is encrypted with a different Caesar cipher, depending on the corresponding key character. Similarly, decryption is a generalization of Equation (3.2):

$$p_i = (C_i - k_{i \bmod m}) \bmod 26 \qquad \textbf{(3.4)}$$

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is *deceptive*, the message "we are discovered save yourself" is encrypted as

```
key:          deceptivedeceptivedeceptive
plaintext:    wearediscoveredsaveyourself
ciphertext:   ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Expressed numerically, we have the following result.

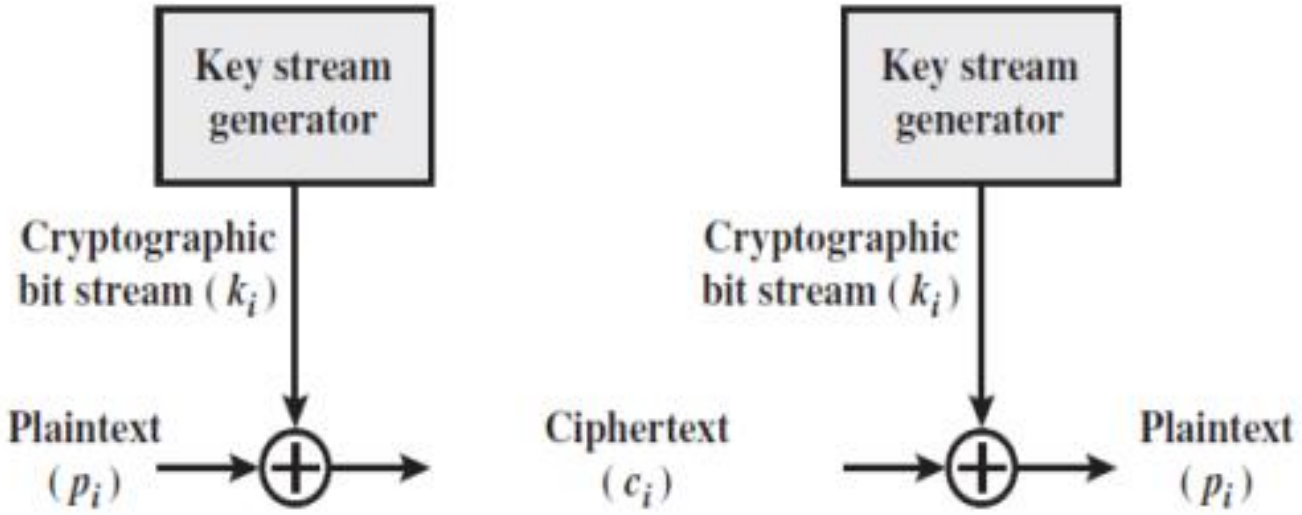
key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9

Suppose we wish to encrypt the plaintext message `THE SUN AND THE MAN IN THE MOON`, using the keyword `KING`.

Vernam Cipher

The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918.



The system can be expressed succinctly as follows

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation

Encryption Algorithm

•Assign a number to each character of the plain text and the key according to alphabetical order.Convert the number into binary form.

•Bitwise XOR both the number (Corresponding plain-text character number and Key character number).

•Subtract the number from 26 if the resulting number is greater than or equal to 26, if it isn't then leave it.

For the Decryption apply the just reverse process of encryption

Ex:

Plain-Text: O A K

Key: S O N

O ==> 14 = 0 1 1 1 0

S ==> 18 = 1 0 0 1 0

Bitwise XOR Result: 1 1 1 0 0 = 28

the resulting number is greater than 26, subtract 26 from it.

28 - 26 = 2 ==> C

CIPHER-TEXT: **C**

FInal CIpher text : COH

- Ex:

Plain-Text: RAMSWARUPK

R	A	M	S	W	A	R	U	P	K
17	0	12	18	22	0	17	20	15	10
R	A	N	C	H	O	B	A	B	A
17	0	13	2	7	14	1	0	1	0

0	0	1	16	17	14	16	20	14	10

CIPHER-TEXT: A A B Q R O Q U O K

Ex:

Plain text : Hello ,key- NCBTA

Cipher text: UGMEO

One time pad

- The key is to be used to encrypt and decrypt a single message, and then is discarded.
- Each new message requires a new key of the same length as the new message. Such a scheme, known as a one-time pad, is unbreakable.

Advantages of the One-Time Pad:

1. Perfect Secrecy : When used correctly, the one-time pad offers complete security, meaning that the ciphertext reveals no information about the plaintext.

2. Randomness : If the key is truly random and kept secret, the ciphertext will also be random, making it immune to frequency analysis and other forms of cryptanalysis.

3. Simplicity : The concept is straightforward; it simply requires a random key of equal length to the message.

Disadvantages of the One-Time Pad:

1**. Key Distribution Problem :** Each sender and receiver must have access to the same key, which can be a significant logistical challenge, especially for large quantities of data.

2. Key Management : Generating and securely managing large amounts of random keys can be impractical. For heavily used systems, millions of random characters may be needed regularly.

3. Limited Utility : Due to the difficulties in key distribution and management, the one-time pad is primarily useful for low-bandwidth channels that require very high security.

For everymessage to be sent, a key of equal length is needed by both sender and receiver.

Ex 1:

Input: Message = HELLO, Key = MONEY

Plain text — H E L L O ?

7 4 11 11 14

Key — M O N E Y ?

12 14 13 4 24

Plain text + key ?

19 18 24 15 38 ?

19 18 24 15 12 (= 38 – 26)

Cipher Text : **T S Y P M**

- Ciphertext to Message

Cipher Text — T S Y P M ?

19 18 24 15 12

Key — M O N E Y?

12 14 13 4 24

Decryption: Cipher text - key ?

7 4 11 11 -12 ?

7 4 11 11 14

Message ? **H E L L O**

Ex2:

Input: Message = SAVE, Key = LIFE

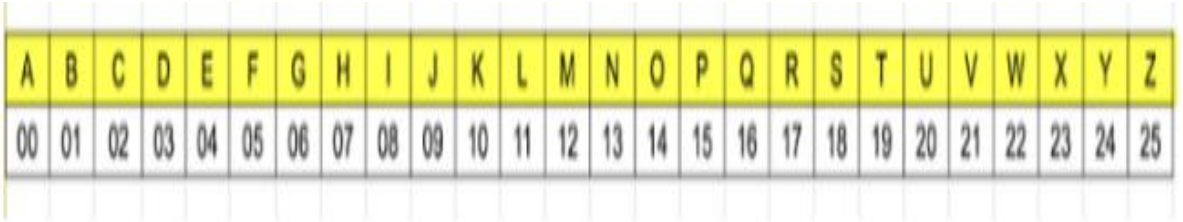
Cipher Text :- DIAI

Ex3: In the One-time pad version of a Vignere Cipher, Key stream is 9,0,1,7,23, 15, 21, 14, 11, 11, 2,8, 9. In this scheme, encryption is done by shifting with number mentioned in the key. Encrypt the plain text sendmoremoney and using the Cipher text obtained, find a key such that Cipher text decrypts to c a s h n o t n e e d e d . (06 Marks)

a) Encrypt the plaintext "sendmoremoney" with the key stream 9, 0, 1, 7, 23, 15, 21, 14, 11, 11, 2, 8, 9.

We are given the plaintext "sendmoremoney" and the key stream "9 0 1 7 23 15 21 14 11 11 2 8 9". To encrypt, we shift each letter of the plaintext by the corresponding number in the key stream:

- s + 9 = b
- e + 0 = e
- n + 1 = o
- d + 7 = k
- m + 23 = j
- o + 15 = d
- r + 21 = m
- e + 14 = s
- m + 11 = x
- o + 11 = z
- n + 2 = p
- e + 8 = m
- y + 9 = h



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

The ciphertext is: **beokjdmsxzpmh.**

- **Finding a Key for New Plaintext**
- We need to find a key that will decrypt "beokjdmsxzpmh" to "cashnotneeded". We use the same shifting principle:

  - b - c = 25 (z)
  - e - a = 4 (e)
  - o - s = -4 (w)
  - k - h = 3 (d)
  - j - n = -4 (w)
  - d - o = -11 (p)
  - m - t = -7 (s) t
  - s - n = 5 (e) f
  - x - e = 19 (t)
  - z - d = 22 (w v
  - p - d = 12 (m
  - m - e = 8 (i)
  - h - d = 4 (e)

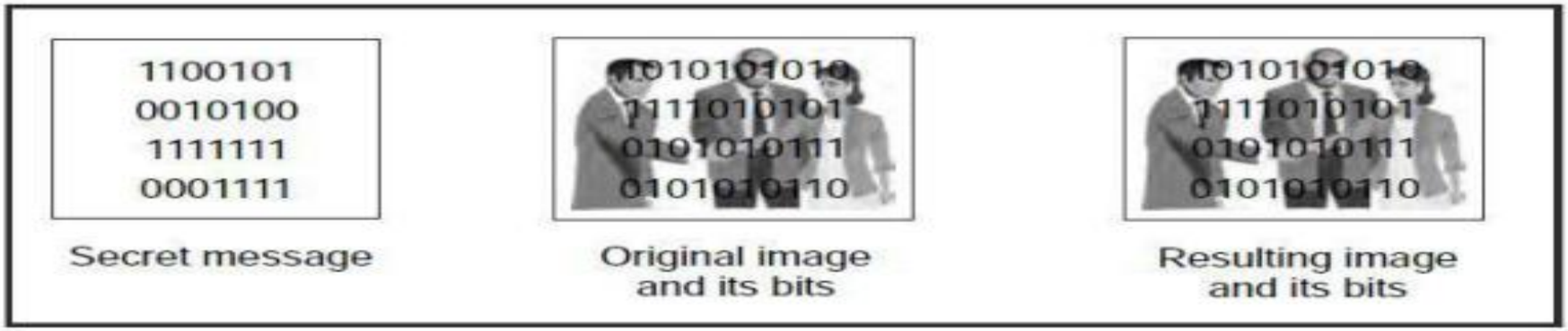
K=(C–P)

Key :

zewdwptftvmie

Stegnography

- Steganography is a technique that facilitates hiding of a message that is to be kept secret inside other messages.
- The sender used methods such as invisible ink, tiny pin punctures on specific characters, minute variations betweenhandwritten characters, pencil marks on handwritten characters, etc.



The objective of this chapter is to illustrate the principles of modern symmetric ciphers. For this purpose, we focus on the most widely used symmetric cipher: the Data Encryption Standard (DES). Although numerous symmetric ciphers have been developed since the introduction of DES, and although it is destined to be replaced by the Advanced Encryption Standard (AES), DES remains the most important such algorithm. Furthermore, a detailed study of DES provides an understanding of the principles used in other symmetric ciphers.

Traditional Block cipher Structure

- A block cipher is a **symmetric encryption algorithm** that encrypts data in fixed-size chunks (called blocks) using a secret key.
- 'symmetric', we mean that the size of input text and output text (ciphertext) is same bits.
- Block size: Common sizes are 64 bits, 128 bits, or 256 bits.
- Symmetric key: The same key is used for both encryption and decryption.
- **Block cipher techniques are fundamental encryption methods used in data security.**
- **These are widely applied in secure communications, file encryption, and network security (e.g., SSL/TLS, VPNs).**

Popular Block Cipher Algorithms

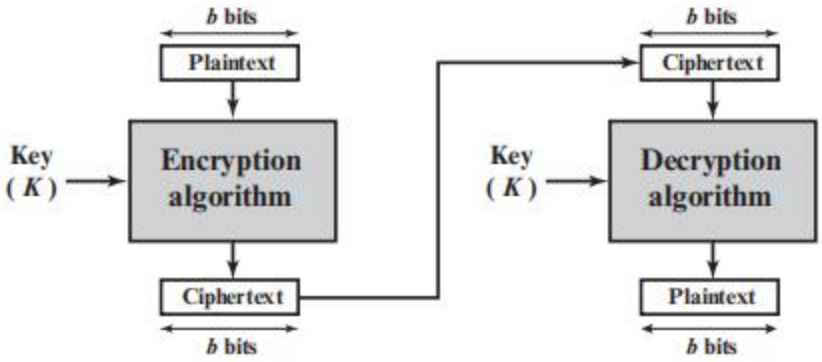
- DES (Data Encryption Standard)
- 3DES (Triple DES)
- AES (Advanced Encryption Standard)
- Blowfish
- Twofish
- IDEA (International Data Encryption Algorithm)

Several important symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher [FEIS73]. For that reason, it is important to examine the design principles of the Feistel cipher. We begin with a comparison of stream ciphers and block ciphers. Then we discuss the motivation for the Feistel block cipher structure. Finally, we discuss some of its implications.

- A block cipher is an **encryption/decryption** scheme in which a **block of plaintext is treated as a whole** and used to produce a ciphertext **block of equal length**.
- Many blockciphers have a **Feistel structure**. Such a structure consists of a **number of identical rounds of processing**.
- In each round, a **substitution** is performed on one half of the data being processed, followed by a **permutation** that interchanges the two halves.
- The **original key is expanded so that a different key is used for each round**.
- Many **symmetricblock encryption algorithms** in current use are based on a structure referred to as aFeistel 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.

Block Ciphers

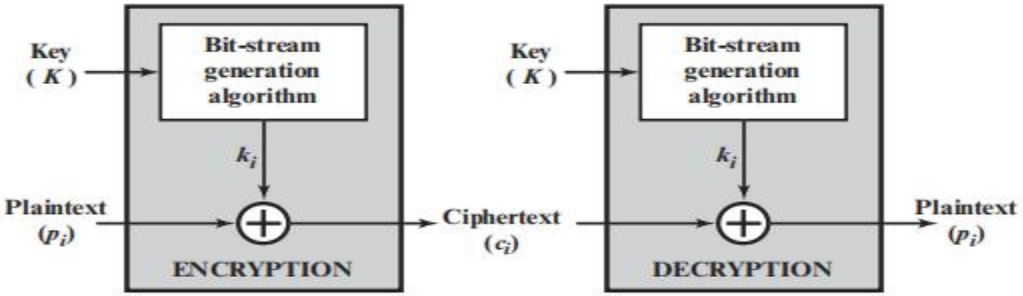
- With block ciphers, the plaintext is split into fixed size chunks called blocks, and each block is encrypted separately.
- Typically all blocks in the plaintext are encrypted using the same key.
- Block ciphers include DES, AES, RSA, and ECC.
- Block sizes used in secret key cryptography are usually smaller — 64 bits in DES and 128 bits in AES.
- The block size in RSA is much larger — 768 or more bits, while the block size in ECC is about 200 bits.
- If two blocks of plaintext within a message are identical, their corresponding ciphertexts are identical. This statement, however, is only partially true.



(b) Block cipher

Stream cipher

- Stream ciphers typically operate on bits.
- The one-time pad is an example of a stream cipher.
- Practical stream ciphers typically generate a pseudo-random keystream which is a function of a fixed length key and a per-message bit string.
- The key is known to both the sender and the receiver.
- The per-message string could be a message sequence number.
- Alternatively, it could be a random number generated by the sender and transmitted to the receiver along with the encrypted message.
- The ciphertext is itself obtained by performing an ⊕ operation between the plaintext and the keystream.
- An example of a stream cipher is RC4 used in the wireless LAN protocol, IEEE 802.11.
- Stream ciphers are usually faster than block ciphers and use less complicated circuits. However, RC4 and some other stream ciphers have been shown to be vulnerable to attack.



(a) Stream cipher using algorithmic bit-stream generator

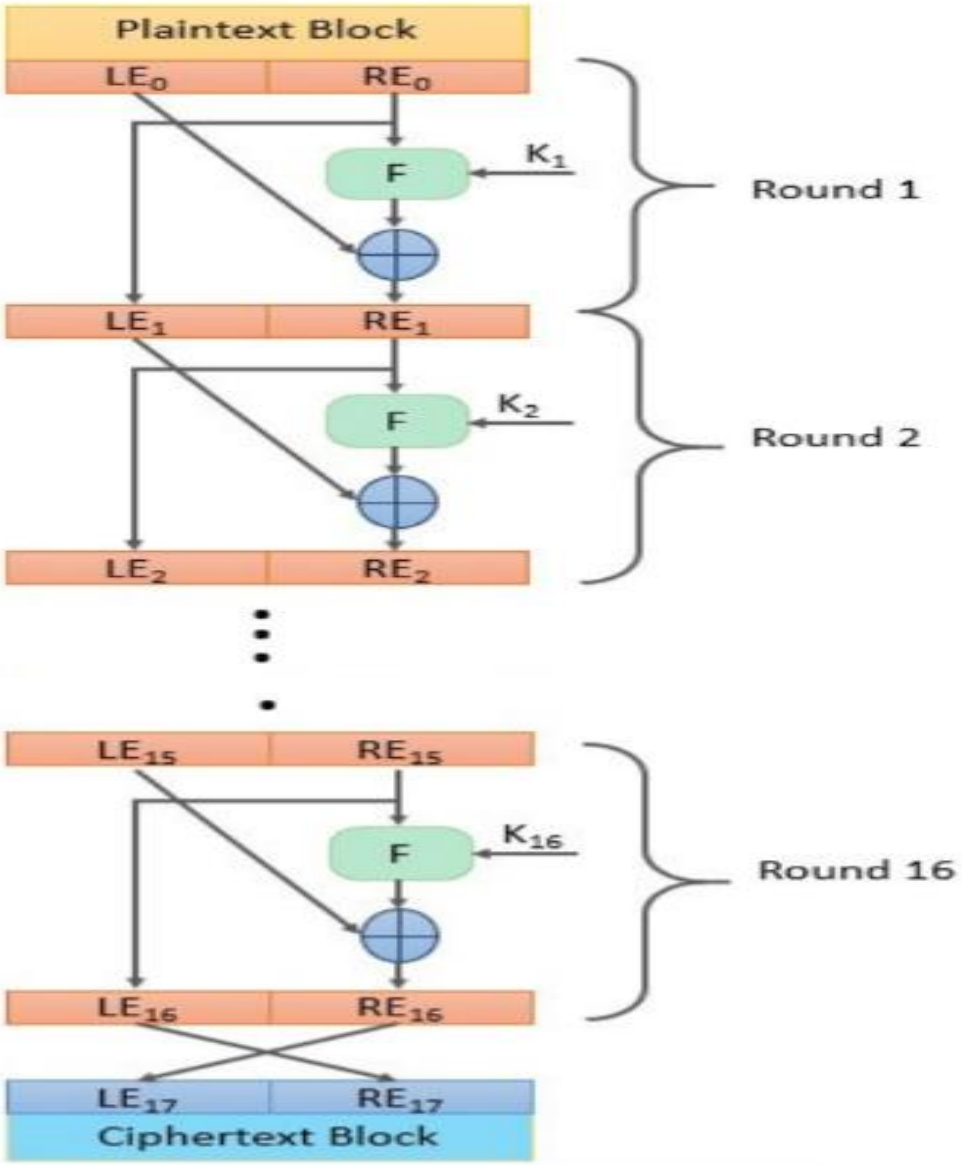
Fiestal Structure

- A Feistel structure is a design model used to build block ciphers.
- Instead of encrypting a whole block in one go, **it splits the block into two halves and repeatedly processes them through multiple rounds of substitutions and permutations**.
- This structure was introduced by **Horst Feistel** and is used in many famous ciphers (e.g.**, DES, Blowfish, Twofish).**
- Feistel cipher proposed a structure which implements;**substitution and permutation;**alternately to obtain cipher text from the pain text and vice-versa. and each block has to undergo **many rounds where eachround has the same function.**
- Block cipher is a type of encryption algorithm that processes fixed-size blocks of data, usually 64 or 128 bits, to produce ciphertext.

Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:

- **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 in which the elements appear in the sequence is changed.

Fiestal Structure



Feistel Cipher Encryption Structure

Let's say we have a plaintext block (e.g., 64 bits):
Split it into **Left half (LE_o) and Right half (RE_o).**

For each **round i** (out of N total rounds):
1.Apply a **round function F** to one half (usually the right half).
2.Combine the result with the other half using **XOR**.
3.Swap the halves.
Equations:
For round **i:**
$LE_i = RE_{i-1}$
$RE_i = LE_{i-1}$ XOR $F(RE_{i-1}, K_i)$
LE_i , RE_i = left & right halves after round i
K_i= round key derived from main key
F = round function (can involve substitution, permutation, expansion, etc.)
After the last round, the two halves are combined → ciphertext.

Step 1: The plain text is divided into the blocks of a fixed size and only one block is processed at a time. So, the input to encryption algorithm is a plain text block and a key K.

Step 2: The plain text block is divided into two equal halves which we will denote as a **LE_0** as the left half of the plain text block and **RE_0** as the right half of the plain text block. Now, both these halves of the plain text block (LE_0 & RE_0) undergoes multiple rounds to produce ciphertext block.

In each round, the **encryption function** is applied on the right half **RE_i** of the plaintext block along with the key K_i. The result of this encryption function is then **XORed** with the left half **LE_i**. The result of XOR function becomes the new right half for next round **RE_{i+1}**. Whereas, the old right half RE_i becomes the new left half **LE_{i+1}** for the next round as you can see in the figure above.

Each round executes the same function. In each round initially, encryption function on the right half of the plain text block. The result of this round function is XORed by the with the left half of the block. After this substitution function, a **permutation** function is implemented by swapping the two halves and the result of this permutation is provided to the next round.

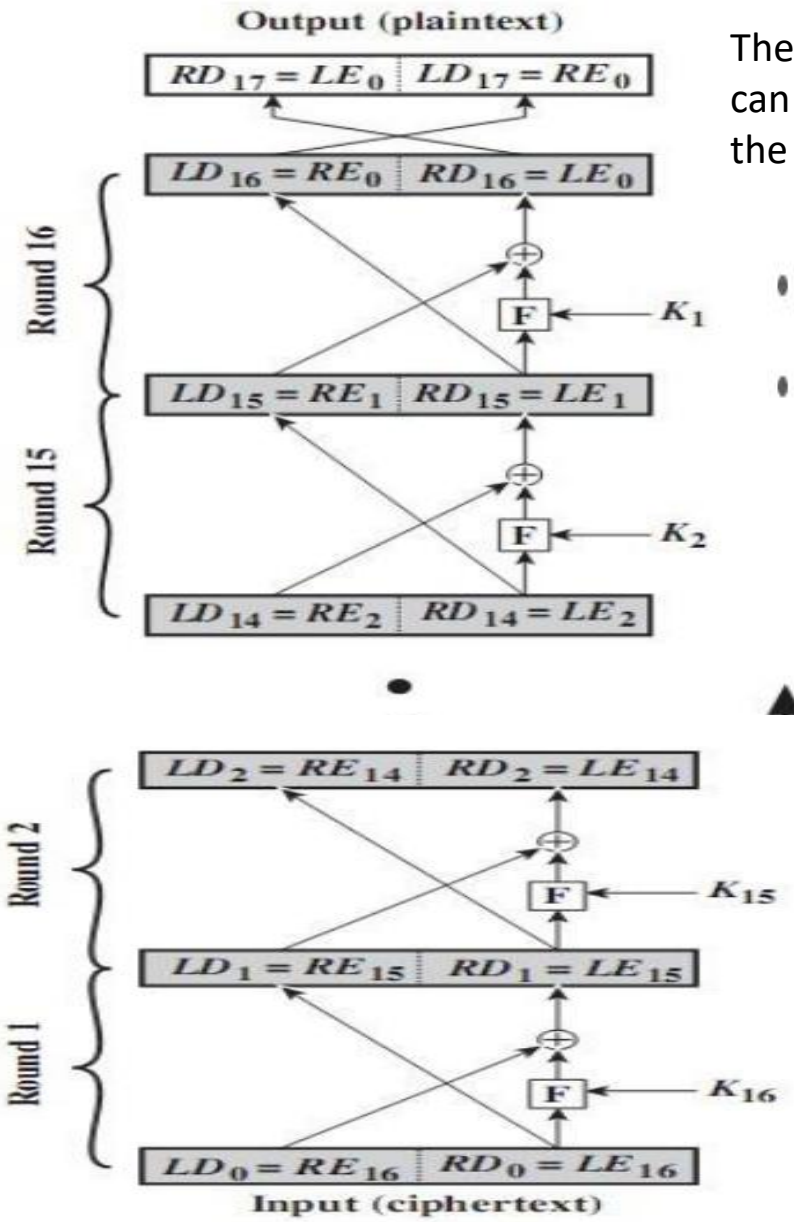
The design features of Feistel cipher

- **Block Size:**The block cipher is considered more secure if the **block size is larger**. But the larger block size can reduce the execution speed of encryption and decryption. Generally, the block size of a block cipher is of 64-bit. But, the modern-day block cipher such as AES has 128-bit block size.

- **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. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.

- **Diffusion and Confusion :** These two concepts are fundamental to the design of block ciphers. **Diffusion ensures that a change in a single bit of plaintext results in a significant change in the ciphertext**, spreading the influence of that bit across the output. Confusion aims to make the **relationship between the ciphertext and the key as complex as possib**le, thwarting attempts to deduce the key from the ciphertext. Effective use of complex substitution algorithms enhances both diffusion and confusion.

Fiestal Decryption Algorithm



The main strength of a Feistel cipher is that the same structure can be used for encryption and decryption — only the order of the keys changes.

- The **same round function F** is used in both encryption and decryption.
- The only difference: **keys are applied in reverse order.**
  - If encryption uses $K_1, K_2, ..., K_n$,
  - Decryption uses $K_n, K_{n-1}, ..., K_1$.

▲ the **decryption equations are**:

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

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

LD(Left decryption)
RD(Right decryption)

- The process of decryption with a Feistel cipher is essentially the same as the encryption process.

The rule is as follows: Use the ciphertext as input to the algorithm,but use the subkeys **K_i in reverse order**. That is, use Kn in the first round, K_{n-1} in the second round, and so on until K_1 is used in the last round.

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

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 XOR has the following properties:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$
$$D \oplus D = 0$$
$$E \oplus 0 = E$$

The Data Encryption Standard

- The most widely used symmetric key block cipher encryption scheme is based on the **Data Encryption Standard (DES)** adopted in 1977 by the National Institute of Standards and Technology (NIST).

The algorithm itself is referred to as the **Data Encryption Algorithm (DEA)**.

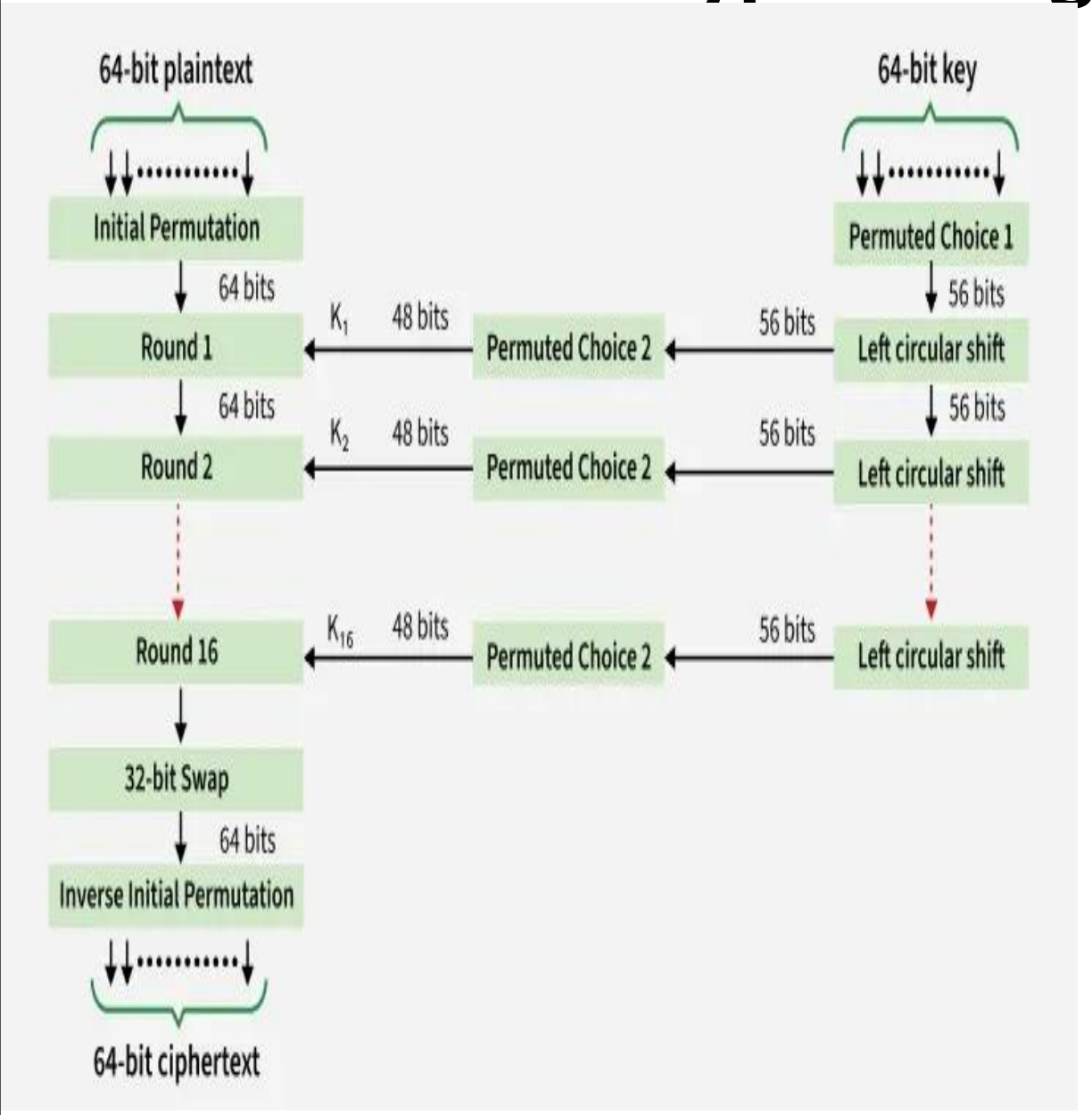
For DES, data are encrypted in **64-bit block size** using a **56-bit key size (64 bits but 8 bits used for parity). and 16 rounds of fiestal structure.**

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 is based on the two attributes of **Feistel cipher i.e. 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 along with other operations.**

DES Encryption Algorithm



The encryption starts with a **64-bit plaintext** that needs to be encrypted using a **64-bit key.** Plaintext is passed to **Initial Permutation function and key is permuted using Permuted Choice 1 (PC-1).**

The 64-bit plaintext block is input into an **Initial Permutation (IP)** function that **rearranges the order of bits.**The initial permutation (IP) happens **only once** and it happens **before the first round.**

The 64-bit initial key is converted **into 56-bit effective key. This 56-bit** key further generates **48-bit subkeys for each of the 16 Feistel rounds.**

32-bit block after permutation is the output of mangler function.

32-bit Swap and Inverse Initial Permutation:

- There are two inputs to the encryption function: **the plaintext to be encrypted and the key.**the **plaintext must be 64 bits in length and the key is 56 bits in length**.

Looking at the left-hand side of the figure, we can see that the **processing of the plaintext proceeds in three phases.**

First, the 64-bit plaintext passes through an **initial permutation (IP)** that rearranges the bits to produce the permuted input.The order of bits is changed using **predefined table.** The IP table is a **8×8 matrix (64 entries)** where each entry specifies the new position of a bit from the original plaintext.

Initial Permutation Table							
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

- The first bit of the permuted block is taken from the 58th bit of the original plaintext.
- The second bit comes from the 50th bit and so on.
- The last (64th) bit comes from the 7th bit of the original plaintext.

The initial permutation (IP) happens **only once** and it happens before the first round. The permutation this function do is fixed and does not depend on the plaintext. This rearranged 64-bit plaintext then go through **16 rounds.** Each of this **round uses a different 48-bit subkey** from the previous round subkey. These subkeys are generated from 64-bit key.

This is followed by a **phase consisting of sixteen rounds of the same function**, which involves **both permutation and substitution functions**. The output of the last (sixteenth) 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 a permutation [IP-1] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

Key Transformation/Key Schedule

- The right-hand portion of Figure shows the way in which the 56-bit key is used.
- The **64-bit initial key** is converted into **56-bit effective key**. This 56-bit key further generates **48-bit subkeys** for each of the **16 Feistel rounds.**
- **Conversion of 64-bit Key into 56-bit Key**

Initial key first go through **Permuted Choice 1 (PC-1)/compression P-Box** which reduces the key to **56 bits**. **In PC-1 every eighth bit in key is discarded**. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are

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

These discarded bits are called **parity bits** which are used for **error checking**. Remaining **56 bits are split into two 28-bit halves:**

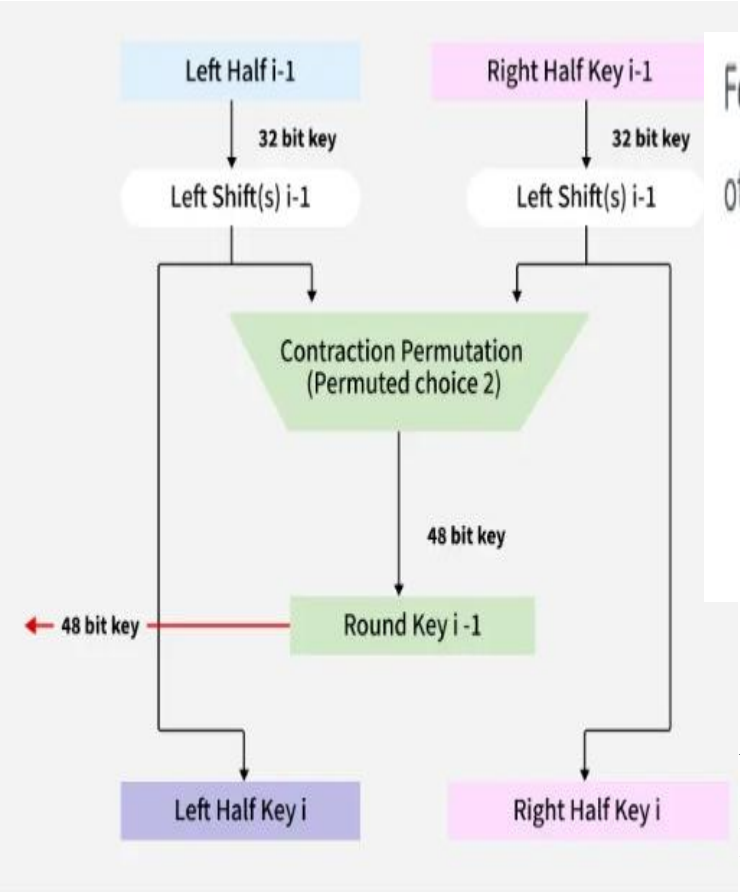
Left Half (Ci): First 28 bits.
Right Half (Di): Last 28 bits

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

Generating 48-bit Round Subkeys

For each of the 16 rounds, right half (Ci) and left half (Di) undergo **circular left shift operation**.



Key Transformation in DES

For Feistel round 1, 2, 9, and 16 both halves (left and right) undergo 1-bit left shift operation. For others rounds (3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15) the halves undergo 2-bit left shift operation.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
#key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Circular Left Shift Operation on Feistel Rounds

After circular shift operation is performed, Ci and Di are again combined into **56-bit block.** This block then go through **Permutation Choice 2 (PC-2). The PC-2 selects and arrange 48 bits out of the 56 to form the round subkey (Ki).**

Permutation Choice 2 Table							
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

According to this table 14th bit is placed to first position, 17th bit to second position, 11th bit to 3rd position and so on. The output 48-bit subkey of this table is used to cipher the plaintext in the Feistel round.

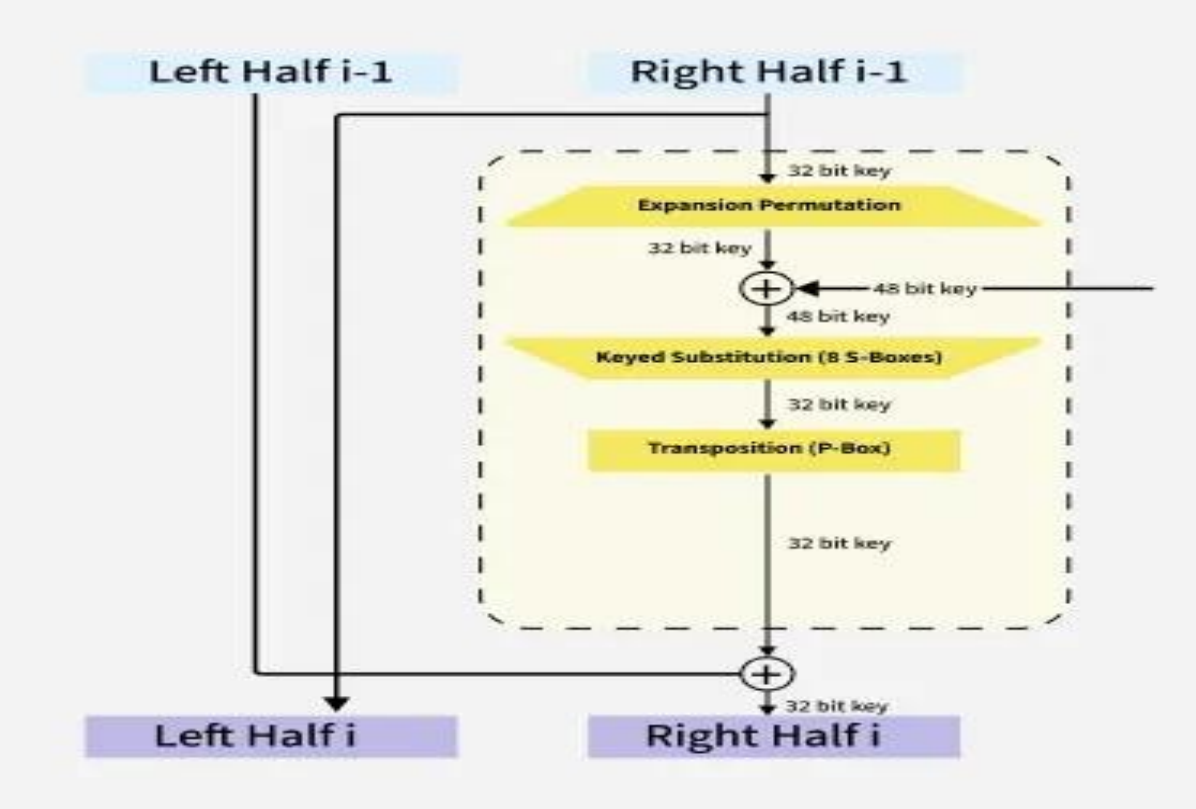
For next round we use already left shifted C_i and D_i as left and right half. We again perform the circular left shift operation on both halves. We again combine the result into 56-bit block and use permutation choice 2 to contract this block into 48-bit subkey for next round.

The process of Circular Left Shift and Permutation Choice 2 is followed for 16 rounds and different round subkey (K_i) is generated for each Feistel Round.

Each 48-bit subkey (Ki) is XORed with the expanded right half in the **Feistel Round**. FOllows what happens in every single Feistel round.(in next slide)

Feistel Rounds (1 - 16)

- Every round receives **64-bits permuted plaintext** from the **Initial Permutation function** and **48-bit transformed subkey (Ki).**
- The permuted 64-bit plaintext is divided into two halves called as **Left Plaintext (LPT) and Right Plaintext (RPT).**
- Both of these halves are **32 bit in size**. The right half or Right Plaintext (RPT) is processed using **Mangler (F) function.** Mangler (F) function involves expansion, key mixing, substitution (S-boxes), and permutation (P-box) of RPT.



- In this permutation 32-bit Right Plaintext (RPT) is expanded into 48 bits using **expansion box or E-box table.**

E-Box Expansion 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

The **48-bit expanded block is generated by arranging the bits as in E-Box table.**
This expanded block is XORed (⊕) with the 48-bit round subkey that is generated during key transformation process. The **XOR or Exclusive OR** operation returns '0' as output if both inputs are same, else the out will be '1'. After XOR is performed, the resulting 48-bit block is split into eight chunks of 6-bit size each. Each of the chunk is then fed into a different **S-box Substitution** (S1 to S8).

The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input
and produces 4 bits as output.

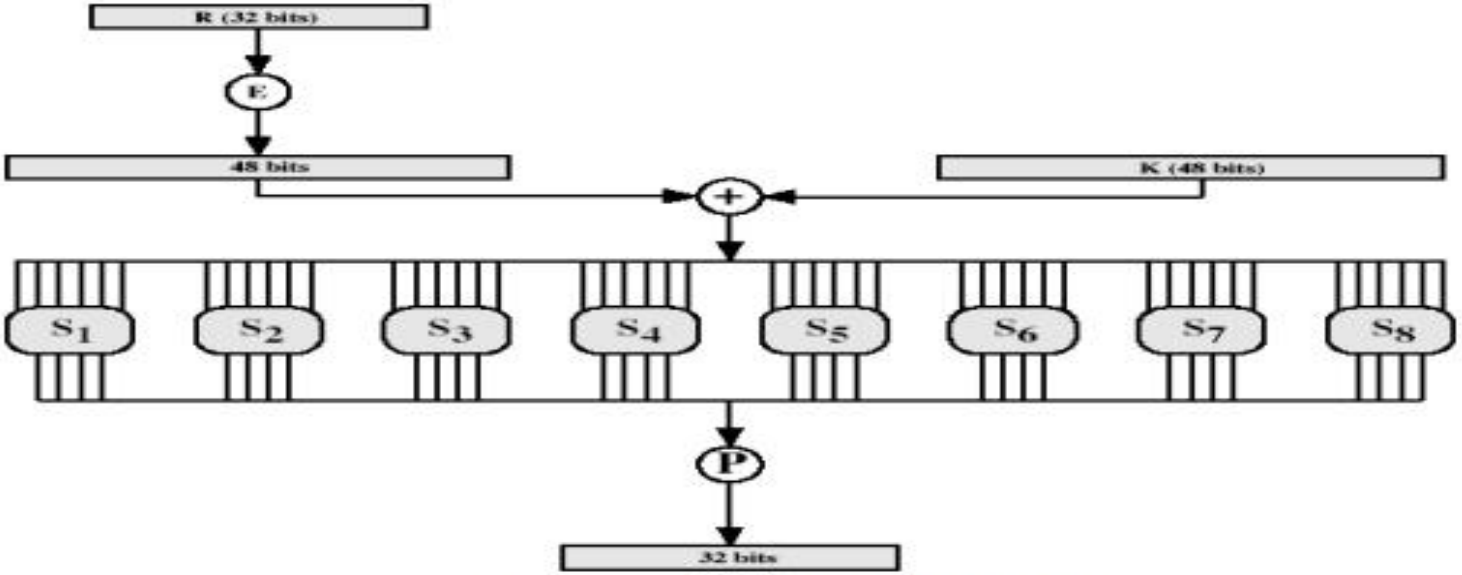
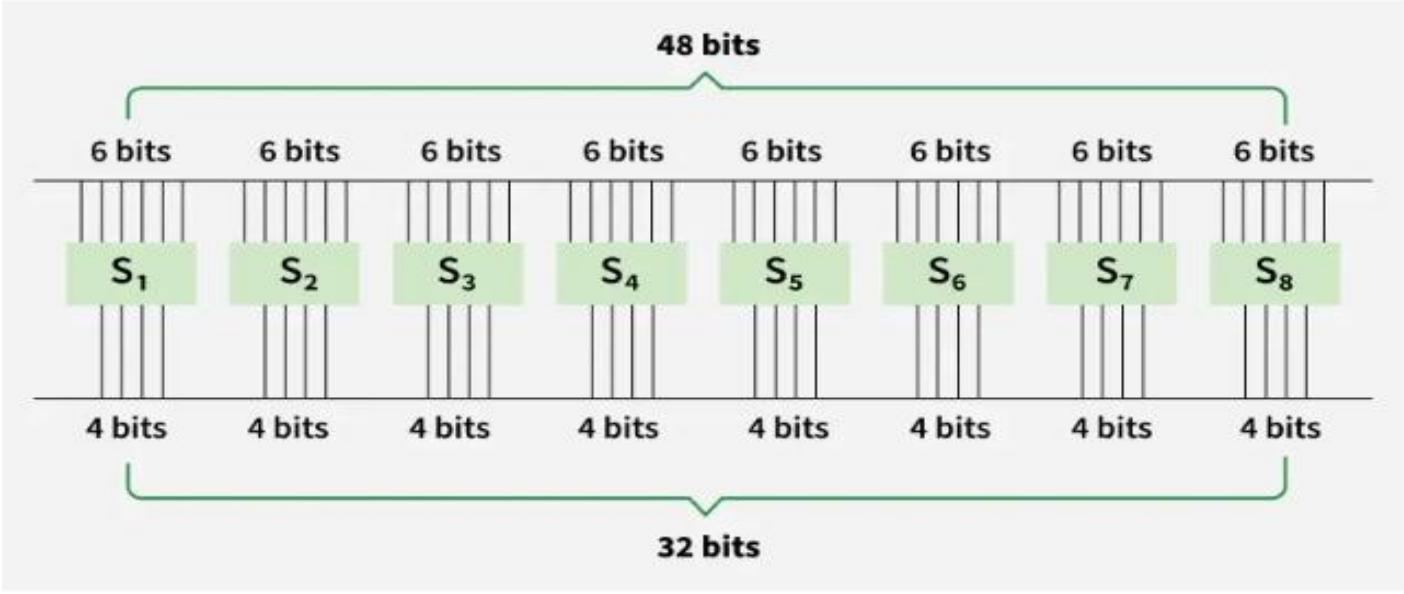


Fig 2.10 Calculation of F(R, K)

These 6 bits chunks will be converted into 4 bits using S-Boxes.



S-Box

S-Boxes are predefined lookup tables which reduces 6 bits chunk into 4 bits. Below is the list of these S-Boxes.

- After that we combine all of these 4-bit chunks to get 32-bit block as output using **P-box permutation table.** This permutation is called Transposition. The mangler function finishes here.
- 32-bit block after permutation is the output of mangler function.

32-bit Swap and Inverse Initial Permutation

- After these 16 rounds we get two blocks (Left and Right) of 32-bit each.
- The two 32-bit halves are again swapped back, resulting in a 64-bit block. This step is called **32-bit Swap in DES encryption algorithm.**
- Finally, the block undergoes an **Inverse Initial Permutation (IP^{-1}).** This is essentially the inverse of the initial permutation applied at the beginning.

-Take 64-bit preoutput (after the swap)

-Reorder bits according to IP^{-1} table → get 64-bit ciphertext

- **The IP^{-1} ensures that the final ciphertext is in proper bit order.**

Preoutput Bit Position	IP^{-1} Output Position
1	40
2	8
3	48
…	…
64	25

The 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**.
- If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.

DES Decryption

- Decryption in DES follows the same process as encryption but **in reverse order**. Since DES is a symmetric-key algorithm, the same key is used for both encryption and decryption, but the subkeys (round keys) are applied in reverse order.

- **Reverse Subkey Application:** The 16 round keys generated during key scheduling are used in reverse order (from K16 to K1) during decryption.
- **Inverse Feistel Function:** The Feistel network structure ensures that decryption mirrors encryption. Each round performs the same operations (expansion, S-box substitution, permutation), but with reversed subkeys.
- **Final Permutation (FP):** After 16 rounds, the output undergoes the Inverse Initial Permutation (IP), reversing the initial shuffling.

DES Example

Let M be the plain text message M = 0123456789ABCDEF, where M is in hexadecimal (base 16) format.Let **K be the hexadecimal key K = 133457799BBCDFF1**

Rewriting M and K in binary format, we get the 64-bit block of text:

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

L = 0000 0001 0010 0011 0100 0101 0110 0111

R = 1000 1001 1010 1011 1100 1101 1110 1111

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

- **Step 1:** Create 16 subkeys, each of which is 48-bits long.

The 64-bit key is permuted according to the following table, **PC-1**. Since the first entry in the table is "57", this means that the 57th bit of the original key **K** becomes the first bit of the permuted key **K+**. The 49th bit of the original key becomes the second bit of the permuted key. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

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

Example: From the original 64-bit 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

Next, split this key into left and right halves, C_0 and D_0, where each half has 28 bits.

- **From the permuted key K+, we get**

C0 = 1111000 0110011 0010101 0101111

D0 = 0101010 1011001 1001111 0001111

With C_0 and D_0 defined, we now create sixteen blocks C_n and D_n, 1<=n<=16. Each pair of blocks C_n and D_n is formed from the previous pair C_{n-1} and D_{n-1}, respectively, for n = 1, 2, ..., 16, using the following schedule of "**left shifts**" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

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

Example: From original pair pair C_0 and D_0 we obtain:

C_0 = 1111000011001100101010101111
D_0 = 0101010101100110011111001111

C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110

C_2 = 1100001100110010101010111111
D_2 = 0101010110011001111000111101

C_3 = 0000110011001010101011111111
D_3 = 0101011001100111100011110101

C_4 = 0011001100101010101111111100
D_4 = 0101100110011110001111010101

C_5 = 1100110010101010111111110000
D_5 = 0110011001111000111101010101

C_6 = 0011001010101011111111000011
D_6 = 1001100111100011110101010101

C_7 = 1100101010101111111100001100
D_7 = 0110011110001111010101010110

$C_8 = 001010101011111111100001100111$
$D_8 = 100111100011110101010101011001$

$C_9 = 010101010111111111000011001100$
$D_9 = 001111000111101010101010110011$

$C_{10} = 010101011111111100001100110011001$
$D_{10} = 111100011110101010101011001100$

$C_{11} = 010101111111110000110011001100101$
$D_{11} = 110001111010101010110011001100111$

$C_{12} = 010111111111000011001100110010101$
$D_{12} = 000111101010101011001100111001111$

$C_{13} = 011111111100001100110011001010101$
$D_{13} = 011110101010101100110011100111100$

$C_{14} = 111111110000110011001010101010101$
$D_{14} = 111010101010110011001111100010001$

$C_{15} = 111110000110011001010101010111$
$D_{15} = 101010101011001100111110001111$

$C_{16} = 111100001100110010101010101111$
$D_{16} = 010101010110011001111100011111$

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

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

Example: For the first key we have C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

which, after we apply the permutation **PC-2**, becomes

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

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

Step 2: Encode each 64-bit block of data.

There is an *initial permutation* **IP** of the 64 bits of the message data **M**. This rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order. The 58th bit of **M** becomes the first bit of **IP**. The 50th bit of **M** becomes the second bit of **IP**. The 7th bit of **M** is the last bit of **IP**.

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

Example: Applying the initial permutation to the block of text **M**, given previously, we get
M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Here the 58th bit of **M** is "1", which becomes the first bit of **IP**. The 50th bit of **M** is "1", which becomes the second bit of **IP**. The 7th bit of **M** is "0", which becomes the last bit of **IP**.
Next divide the permuted block **IP** into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

Example: From **IP**, we get L_0 and R_0
L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

We now proceed through 16 iterations, for $1<=n<=16$, using a function f which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, (bit-by-bit addition modulo 2).** Then for **n** going from 1 to 16 we calculate

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

This results in a final block, for $n = 16$, of $L_{16}R_{16}$. That is, in each iteration, we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation f.

Example: 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

Let **E** be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

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

Thus the first three bits of $\mathbf{E}(R_{n-1})$ are the bits in positions 32, 1 and 2 of R_{n-1} while the last 2 bits of $\mathbf{E}(R_{n-1})$ are the bits in positions 32 and 1.

DES Example

For this example, the plaintext is a hexadecimal palindrome. The plaintext, key, and resulting ciphertext are as follows:

Plaintext:	`02468aceeca86420`
Key:	`0f1571c947d9e859`
Ciphertext:	`da02ce3a89ecac3b`

Table 4.2 DES Example

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Note: DES subkeys are shown as eight 6-bit values in hex format

The strength of DES

- **Use of 56 bit keys**

1. With a key length of 56 bits, there are 2^{56} possible keys, which is approximatly.A brute force attack on such number of keys is impossible.
2. A machine implementing one DES encryption per microsecond would take more than thousands of years to divide the cipher.
3. It is necessary that there is more to key-search attack than easily running through all possible keys. If the message is only plaintext in English, thus the result pops out simply, although the task of identifying English would have to be automated.
4. If the text message has been compressed before encryption, then identification is more complex.

The strength of DES-Contd

- **The Nature of algorithm**

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm.

- **Timing Attacks**

1. A timing attack is a security exploit that allows an attacker to **spot vulnerabilities in a local or a remote system** to extract potentially responsive or secret data by acquiring the concerned system's response time to several inputs. A timing attack is a type of a broader class of attacks known as Sidechannel attacks.
2. A 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. This is a long way from knowing the actual key, but it is an intriguing first step.

Block cipher design principles

Block ciphers are built in the **Feistel cipher structure.**

Block cipher has a s**pecific number of rounds and keys** for generating ciphertext.

Some of these principles are:

- The number of rounds,
- Design of the function F,
- Key scheduling

1)The number of rounds

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- it just reflects the number of rounds to be suitable for an algorithm to make it more complex,
- in DES we have 16 rounds ensuring it to be more secure while in AES we have 10 rounds which makes it more secure.

2) Design of the function F

- The function F provides the element of confusion in a Feistel cipher, want it to be difficult to “unscramble” the substitution performed by F
- One obvious criterion is that F be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be.
- One of the most intense areas of research in the field of symmetric block ciphers is that of S-box design. Would like any change to the input vector to an S-box to result in random-looking changes to the output. The relationship should be nonlinear and difficult to approximate with linear functions.

3) Key scheduling

• A final area of block cipher design, and one that has received less attention than S-box design, is the key schedule algorithm. With any Feistel block cipher, the key schedule is used to generate a subkey for each round.

• Would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key. The key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.