

## UNIT-I Computer and Network Security Concepts:

Computer Security Concepts, The OSI Security Architecture, Security Attacks, Security Services, Security Mechanisms, A Model for Network Security, Classical Encryption Techniques: Symmetric Cipher Model, Substitution Techniques, Transposition Techniques, Steganography, Block Ciphers: Traditional Block Cipher Structure, The Data Encryption Standard, Advanced Encryption Standard: AES Structure, AES Transformation Functions.

### UNIT-1

#### Computer and Network Security Concepts

##### INTRODUCTION

Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”.One is confidentiality which basically means that we need to be sure that nobody will see our information as it travels across a network. Authentication and access control is also another capability provided by cryptography. Some other capabilities provided by cryptography are non-repudiation and integrity.

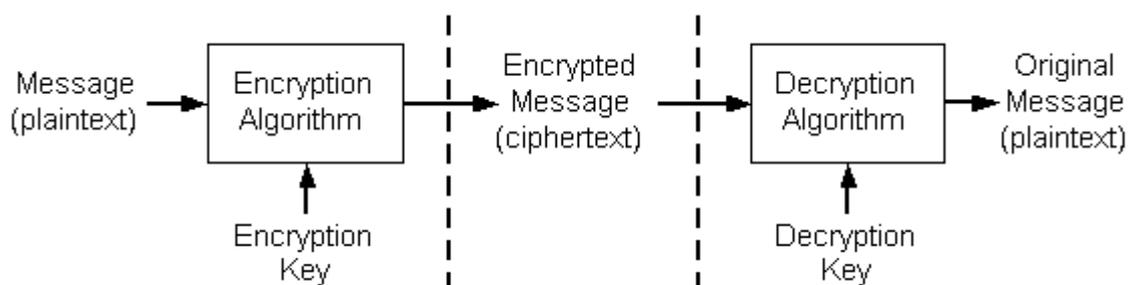
In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

##### Basic Concepts

**Cryptography** The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

**Plaintext** can refer to anything which humans can understand and/or relate to. This may be as simple as English sentences, a script, or Java code. If you can make sense of what is written, then it is in plaintext.

**Ciphertext**, or encrypted text, is a series of randomized letters and numbers which humans cannot make any sense of. An encryption algorithm takes in a plaintext message, runs the algorithm on the plaintext, and produces a ciphertext. The ciphertext can be reversed through the process of decryption, to produce the original plaintext.



**Key** Some critical information used by the cipher, known only to the sender& receiver.

## **The Basic Principles**

### **1. Encryption**

In a simplest form, encryption is to convert the data in some unreadable form. This helps in protecting the privacy while sending the data from sender to receiver. On the receiver side, the data can be decrypted and can be brought back to its original form. The reverse of encryption is called as decryption. The concept of encryption and decryption requires some extra information for encrypting and decrypting the data. This information is known as key. There may be cases when same key can be used for both encryption and decryption while in certain cases, encryption and decryption may require different keys.

### **2. Authentication**

This is another important principle of cryptography. In a layman's term, authentication ensures that the message was originated from the originator claimed in the message. Suppose, Alice sends a message to Bob and now Bob wants proof that the message has been indeed sent by Alice. This can be made possible if Alice performs some action on message that Bob knows only Alice can do. Well, this forms the basic fundamental of Authentication.

### **3. Integrity**

Now, one problem that a communication system can face is the loss of integrity of messages being sent from sender to receiver. This means that Cryptography should ensure that the messages that are received by the receiver are not altered anywhere on the communication path. This can be achieved by using the concept of cryptographic hash.

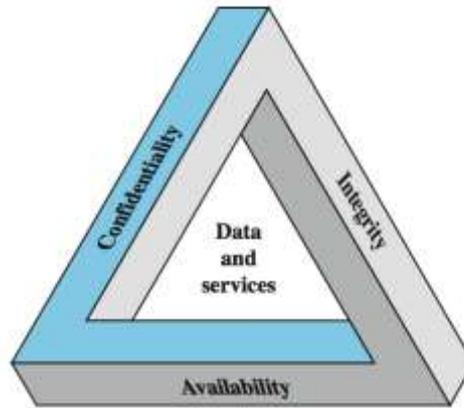
### **4. Non Repudiation**

What happens if Alice sends a message to Bob but denies that she has actually sent the message? Cases like these may happen and cryptography should prevent the originator or sender to act this way. One popular way to achieve this is through the use of digital signatures.

## **1.1 COMPUTER SECURITY CONCEPTS**

### **A Definition Of Computer Security**

These three concepts form what is often referred to as the CIA triad (Figure 1.1). The three concepts embody the fundamental security objectives for both data and for information and computing services. FIPS PUB 199 provides a useful characterization of these three objectives in terms of requirements and the definition of a loss of security in each category:



- **Confidentiality** (covers both data confidentiality and privacy): preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.
- **Integrity** (covers both data and system integrity): Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.
- **Availability**: Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system. Although the use of the CIA triad to define security objectives is well established, some in the security field feel that additional concepts are needed to present a complete picture. Two of the most commonly mentioned are:
  - **Authenticity**: The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator.
  - **Accountability**: The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity.

### 1.1.1 The challenges of computer security

1. The major requirements for security services can be given self-explanatory, one-word labels: confidentiality, authentication, nonrepudiation, or integrity.
2. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features.
3. When the various aspects of the threat are considered that elaborate security mechanisms make sense.
4. It is necessary to decide where to use the security mechanisms, both in terms of physical placement(at what points in a network) and in a logical sense(at what layer or layers of a architecture such as TCP/IP)
5. Security mechanism typically involve more than a particular algorithm or protocol.

6. Computer and network security is essentially a battle of wits between a attackers and the designer/ administrator.
7. Security requires regular, even constant, monitoring.
8. Security is still too often an afterthought to be incorporated into a system after the design is complete rather than being an integral part of the design process.
9. Many users and even security administrator view strong security as an impediment to efficient and user-friendly operation of an information system

## 2. The OSI Security Architecture

The OSI security architecture focuses on security attack, security mechanism , security service

- **security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A process that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization.

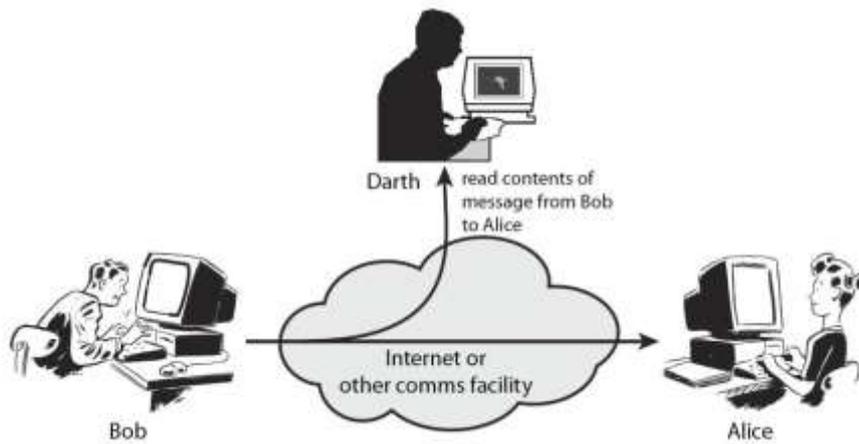
### 1.3 Security Attacks

A useful means of classifying security attacks, used both in X.800 and RFC 2828, is in terms of *passive attacks* and *active attacks*. A passive attack attempts to learn or make use of information from the system but does not affect system resources.

*Passive attacks* are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are:

- release of message contents - as shown above in Stallings Figure 1.2a here
- traffic analysis - monitor traffic flow to determine location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.

These attacks are difficult to detect because they do not involve any alteration of the data.

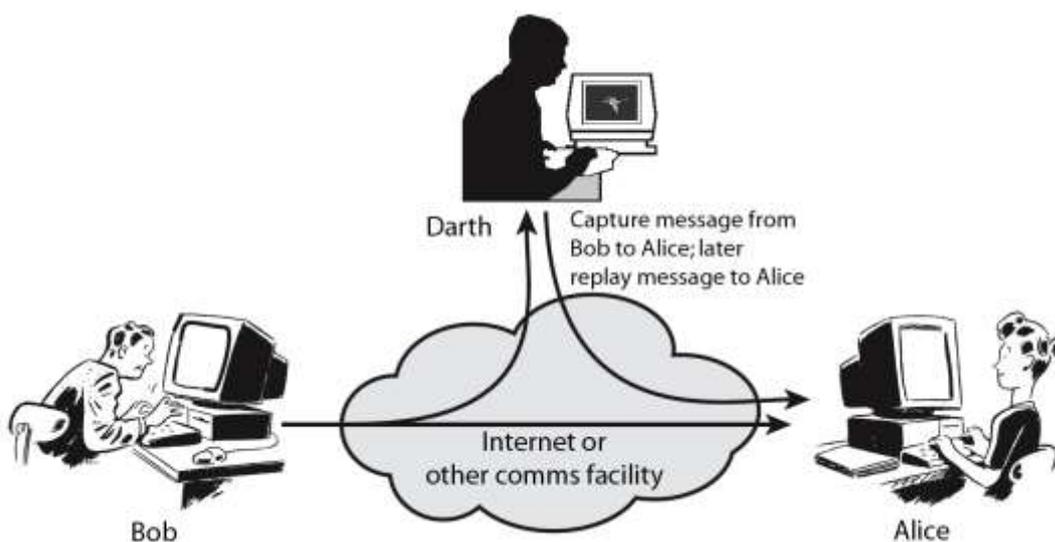


## ACTIVE ATTACKS

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service:

- masquerade of one entity as some other
- replay previous messages (as shown above in Stallings Figure 1.3b)
- modify/alter (part of) messages in transit to produce an unauthorized effect
- denial of service - prevents or inhibits the normal use or management of communications facilities

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them.



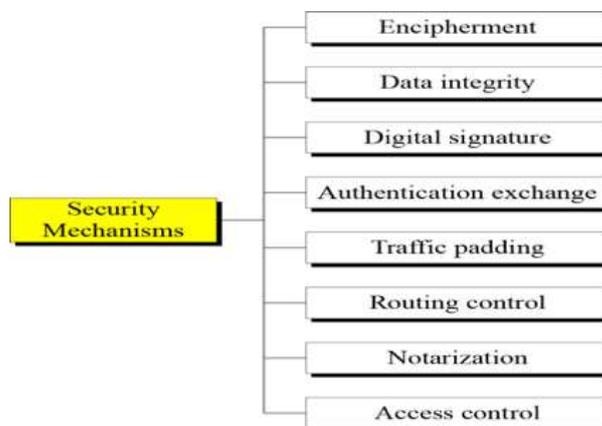
## SECURITY SERVICES



- **Authentication:** assures recipient that the **message is from the source** that it **claims** to be from.
- **Access Control:** controls who can have **access to resource** under what **condition**
- **Availability:** available to authorized entities for 24/7.
- **Confidentiality:** information is not made available to unauthorized individual
- **Integrity:** assurance that the message is unaltered
- **Non-Repudiation:** protection against denial of sending or receiving in the communication

## SECURITY MECHANISMS

Network Security is field in computer technology that deals with ensuring security of computer network infrastructure. As the network is very necessary for sharing of information whether it is at hardware level such as printer, scanner, or at software level.



### 1. Encipherment :

This security mechanism deals with hiding and covering of data which helps data to become confidential. It is achieved by applying mathematical calculations or algorithms which reconstruct information into not readable form. It is achieved by two famous techniques named Cryptography and Encipherment. Level of data encryption is dependent on the algorithm used for encipherment.

### 2. Access Control :

This mechanism is used to stop unattended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using firewall, or just by adding PIN to data.

### 3. **Notarization :**

This security mechanism involves use of trusted third party in communication. It acts as mediator between sender and receiver so that if any chance of conflict is reduced. This mediator keeps record of requests made by sender to receiver for later denied.

### 4. **Data Integrity :**

This security mechanism is used by appending value to data to which is created by data itself. It is similar to sending packet of information known to both sending and receiving parties and checked before and after data is received. When this packet or data which is appended is checked and is the same while sending and receiving data integrity is maintained.

### 5. **Authentication exchange :**

This security mechanism deals with identity to be known in communication. This is achieved at the TCP/IP layer where two-way handshaking mechanism is used to ensure data is sent or not

### 6. **Bit stuffing :**

This security mechanism is used to add some extra bits into data which is being transmitted. It helps data to be checked at the receiving end and is achieved by Even parity or Odd Parity.

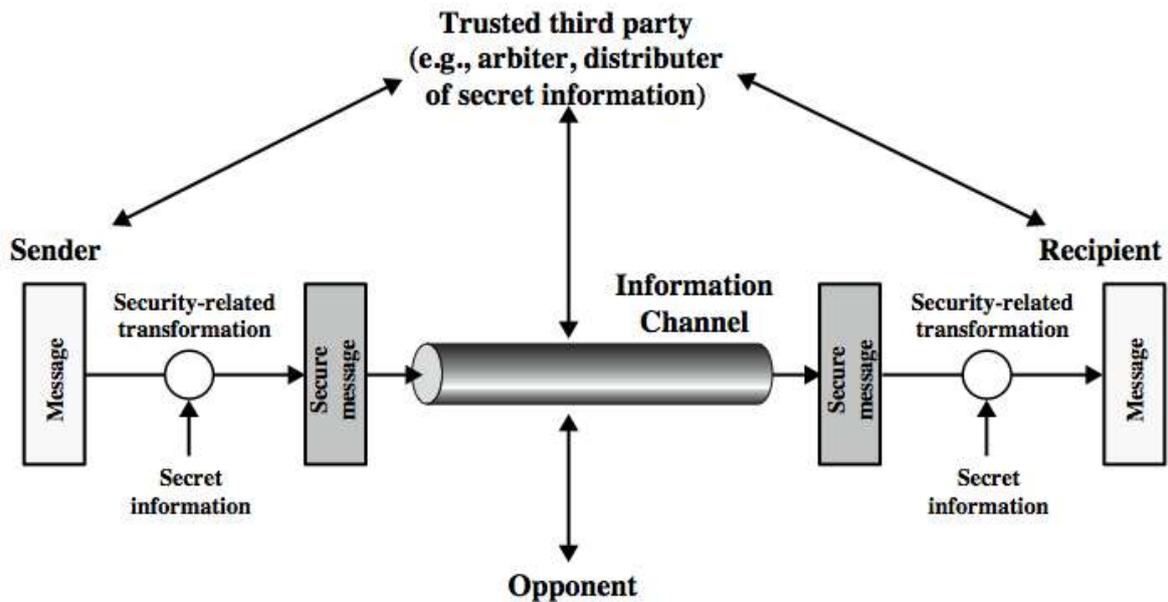
### 7. **Digital Signature :**

This security mechanism is achieved by adding digital data that is not visible to eyes. It is form of electronic signature which is added by sender which is checked by receiver electronically. This mechanism is used to preserve data which is not more confidential but sender's identity is to be notified.

## **A Model for Network Security**

In considering the place of encryption, its useful to use the following two models from Stallings section 1.6.

The first, illustrated in Figure models information being transferred from one party to another over an insecure communications channel, in the presence of possible opponents. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. They can use an appropriate security transform (encryption algorithm), with suitable keys, possibly negotiated using the presence of a trusted third party. Parts One through Four of this book concentrates on the types of security mechanisms and services that fit into the model shown here.

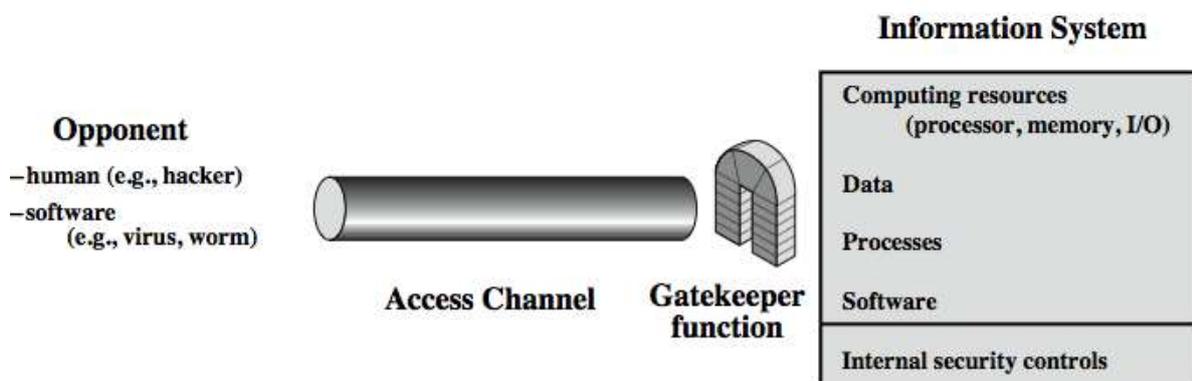


The second, illustrated in Figure 1.5, model is concerned with controlled access to information or resources on a computer system, in the presence of possible opponents. Here appropriate controls are needed on the access to and within the system, to provide suitable security.

The security mechanisms needed to cope with unwanted access fall into two broad categories (as shown in this figure).

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 CIPHER MODEL

A symmetric encryption scheme has five ingredients

- **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 ciphertext and the secret key and produces the original plaintext.

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.

We assume that it is impractical to decrypt a message on the basis of the ciphertext *plus* knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption is what makes it feasible for widespread use. The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms. These chips are widely available and incorporated into a number of products. With the use of symmetric encryption, the principal security problem is maintaining the secrecy of the key.

Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure 2.2. A source produces a message in plaintext,  $X = [X_1, X_2, \dots, X_M]$ . The  $M$  elements of  $X$  are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet  $\{0, 1\}$  is typically used. For encryption, a key of the form  $K = [K_1, K_2, \dots, K_J]$  is generated. If the key is generated at the message source, then

it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

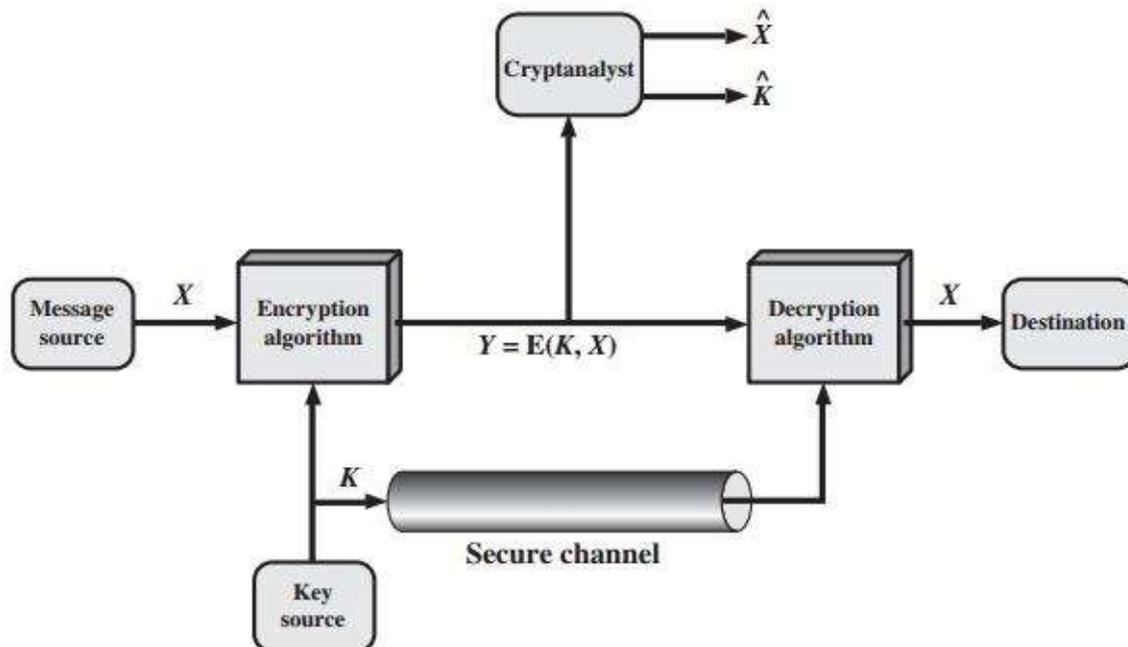


Figure 2.2 Model of Symmetric Cryptosystem

With the message  $X$  and the encryption key  $K$  as input, the encryption algorithm forms the ciphertext  $Y = [Y_1, Y_2, \dots, Y_N]$ . We can write this as

$$Y = E(K, X).$$

This notation indicates that  $Y$  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  $K$ .

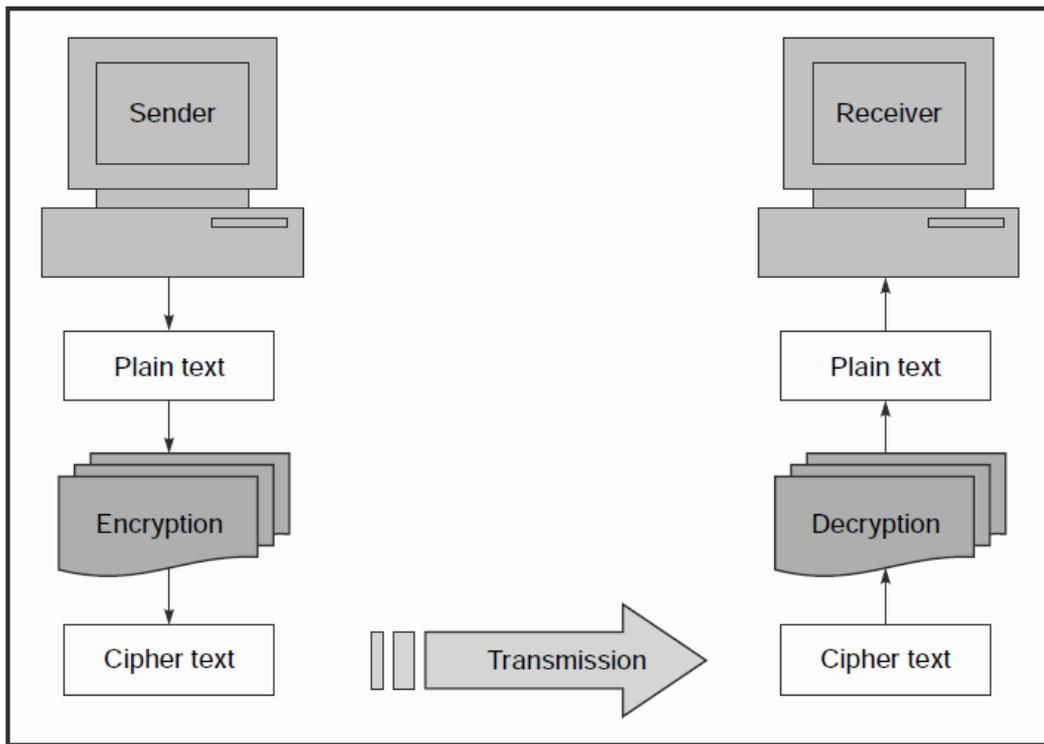
The intended receiver, in possession of the key, is able to invert the transformation:  $X = D(K, Y)$

## SUBSTITUTION TECHNIQUES

### 1. CAESAR CIPHER

This was first proposed by **Julius Caesar** and is termed as **Caesar Cipher**.

Caesar Cipher is a special case of substitution techniques wherein each alphabet in a message is replaced by an alphabet three places down the line. For instance, using the Caesar Cipher, the plain text ATUL will become cipher text DWXO.



Elements of cryptographic operations

In the **substitution cipher technique**, the characters of a plain text message are replaced by other *characters, numbers or symbols*.

An attack on a cipher text message, wherein the attacker attempts to use all possible permutations and combinations, is called as a **Bruteforce attack**. The process of trying to break any cipher text message to obtain the original plain text message itself is called as **Cryptanalysis** and the person attempting a cryptanalysis is called as a **cryptanalyst**.

### MONO-ALPHABETIC CIPHER

Mono-alphabetic ciphers pose a difficult problem for a cryptanalyst because it can be very difficult to crack thanks to the high number of possible permutations and combinations.

Use random substitution. This means that in a given plain text message, **each A can be replaced by any other alphabet (B through Z), each B can also be replaced by any other random alphabet (A or C through Z) and so on**. The crucial difference being, there is no relation between the replacement of B and replacement of A. That is, if we have decided to replace each A with D, we need not necessarily replace each B with E – we can replace each B with any other character!

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
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

samba

the Cipher text is : HOSKO

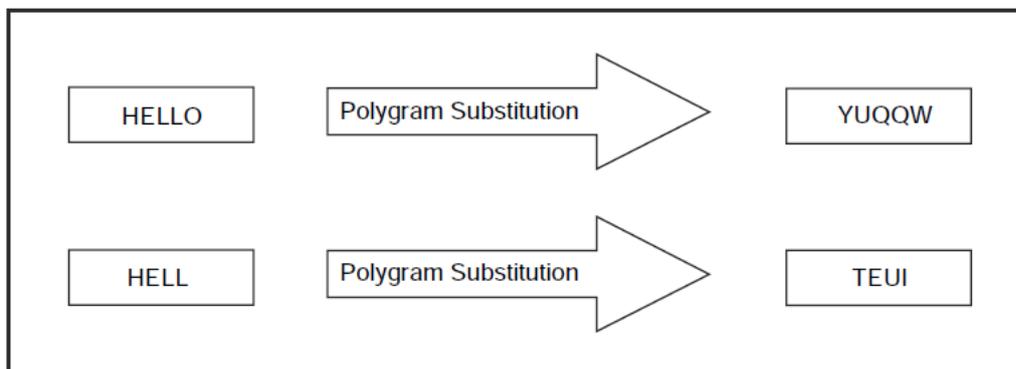
### Homophonic Substitution Cipher

Homophonic Substitution Cipher also involves substitution of one plain text character with a cipher text character at a time, however the cipher text character can be any one of the chosen set.

The Homophonic Substitution Cipher is very similar to Mono-alphabetic Cipher. Like a plain substitution cipher technique, we replace one alphabet with another in this scheme. However, the difference between the two techniques is that whereas the replacement alphabet set in case of the simple substitution techniques is fixed (e.g. replace A with D, B with E, etc.), in the case of Homophonic Substitution Cipher, **one plain text alphabet can map to more than one cipher text alphabet**. For instance, A can be replaced by D, H, P, R; B can be replaced by E, I, Q, S, etc.

### **Polygram Substitution Cipher**

In Polygram Substitution Cipher technique, **rather than replacing one plain text alphabet with one cipher text alphabet at a time, a block of alphabets** is replaced with another block. For instance, HELLO could be replaced by YUQQW, but HELL could be replaced by a totally different cipher text block TEUI, as shown in Fig.



### **Polyalphabetic Substitution Cipher**

A poly-alphabetic cipher is any cipher based on substitution, using several substitution alphabets. In polyalphabetic substitution ciphers, the plaintext letters are enciphered differently based upon their installation in the text. Rather than being a one-to-one correspondence, there is a one-to-many relationship between each letter and its substitutes.

For example, 'a' can be enciphered as 'd' in the starting of the text, but as 'n' at the middle. The polyalphabetic ciphers have the benefit of hiding the letter frequency of the basic language. Therefore attacker cannot use individual letter frequency static to divide the ciphertext.

As the name polyalphabetic recommend this is achieved by **using multiple keys rather than only one key**. This implies that the key should be a stream of subkeys, in which each subkey depends somehow on the position of the plaintext character that needs subkey for encipherment.

**Vigenere cipher** is one of the simplest and popular algorithms in polyalphabetic cipher. In this approach, the alphabetic text is encrypted using a sequence of **multiple Caesar ciphers** based on the letters of a keyword.

The Vigenère cipher includes several simple substitution ciphers in sequence with several shift values. In this cipher, the keyword is repeated just before it connects with the duration of the plaintext.

**Encryption Process:**

$$C_i = (P_i + K_i) \text{ mod } 26$$

In this process sum of  $i^{\text{th}}$  position of plain text and  $i^{\text{th}}$  position of key will be added and applied modulus 26 on the result, the generated positional value will be considered as Cipher text.

**Decryption Process**

$$P_i = (C_i - K_i) \text{ mod } 26$$

In this process sum of  $i^{\text{th}}$  position of Cipher text and  $i^{\text{th}}$  position of key will be subtracted and applied modulus 26 on the result, the generated positional value will be considered as Plain text.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
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

Key: samba

Plain text: hello students how are you

Cipher text:

Key	s	a	m	B	a	s	a	m	b	a	s	a	m	b	a	s	a	m	b	a	s	A
PT	h	e	l	L	o	s	t	u	d	e	n	t	s	h	o	w	a	r	e	y	o	U
CT	z	e	x	M	o	k	t	g	E	e	f	t	e	i	o	o	a	c	f	y	g	U

Apply Encryption process to generate cipher text

That is 's' position is 18 and 'h' position is 7 so now

$$C_1 = (p_1 + k_1) \text{ mod } 26$$

$$= (18 + 7) \text{ mod } 26$$

$$= (25) \text{ mod } 26$$

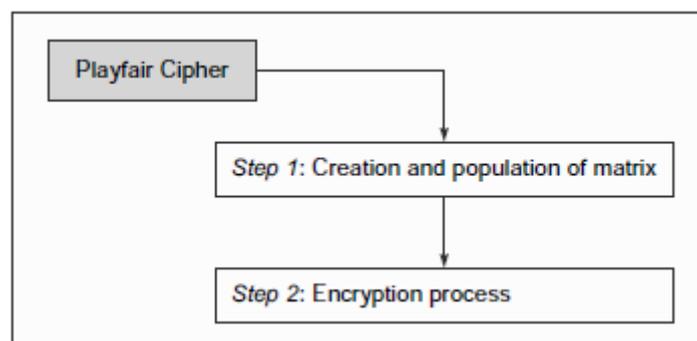
$$= 25 \text{ (which is equivalent to 'z')}$$

Like wise generate the table accordingly.

**Playfair Cipher:**

The Playfair Cipher, also called as **Playfair Square**, is a cryptographic technique that is used for manual encryption of data.

The Playfair encryption scheme uses two main processes, as shown in Fig



Playfair cipher steps

**Step 1: Creation Population of Matrix** and The Playfair Cipher makes use of a 5 x 5 matrix (table), which is used to store a keyword or phrase that becomes the key for encryption and decryption.

The way this is entered into the 5 x 5 matrix is based on some simple rules, as shown below

1. Enter the keyword in the matrix row-wise: left-to-right, and then top-to-bottom.
2. Drop duplicate letters.
3. Fill the remaining spaces in the matrix with the rest of the English alphabets (A-Z) that were not a part of our keyword. While doing so, combine I and J in the same cell of the table. In other words, if I or J is a part of the keyword, disregard both I and J while filling the remaining slots.

**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.

**For example:**

**PlainText:** "instruments"

**After Split:** 'in' 'st' 'ru' 'me' 'nt' 'sz'

1. Pair cannot be made with same letter. Break the letter in single and add a bogus letter to the previous letter.

**Plain Text:** "hello"

**After Split:** 'he' 'lx' 'lo'

Here 'x' is the bogus letter.

2. If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter

**Plain Text:** "helloe"

**AfterSplit:** 'he' 'lx' 'lo' 'ez'

Here 'z' is the bogus letter.

**Rules for Encryption:**

- **If both the letters are in the same column:** Take the letter below each one (going back to the top if at the bottom).

**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

If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).

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

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:	<table border="1"><tr><td>M</td><td>O</td><td>N</td><td>A</td><td>R</td></tr><tr><td>C</td><td>H</td><td>Y</td><td>B</td><td>D</td></tr><tr><td>E</td><td>F</td><td>G</td><td>I</td><td>K</td></tr><tr><td>L</td><td>P</td><td>Q</td><td>S</td><td>T</td></tr><tr><td>U</td><td>V</td><td>W</td><td>X</td><td>Z</td></tr></table>	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:	<table border="1"><tr><td>M</td><td>O</td><td>N</td><td>A</td><td>R</td></tr><tr><td>C</td><td>H</td><td>Y</td><td>B</td><td>D</td></tr><tr><td>E</td><td>F</td><td>G</td><td>I</td><td>K</td></tr><tr><td>L</td><td>P</td><td>Q</td><td>S</td><td>T</td></tr><tr><td>U</td><td>V</td><td>W</td><td>X</td><td>Z</td></tr></table>	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:	<table border="1"><tr><td>M</td><td>O</td><td>N</td><td>A</td><td>R</td></tr><tr><td>C</td><td>H</td><td>Y</td><td>B</td><td>D</td></tr><tr><td>E</td><td>F</td><td>G</td><td>I</td><td>K</td></tr><tr><td>L</td><td>P</td><td>Q</td><td>S</td><td>T</td></tr><tr><td>U</td><td>V</td><td>W</td><td>X</td><td>Z</td></tr></table>	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
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																																																																												
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																																																																												
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:	<table border="1"><tr><td>M</td><td>O</td><td>N</td><td>A</td><td>R</td></tr><tr><td>C</td><td>H</td><td>Y</td><td>B</td><td>D</td></tr><tr><td>E</td><td>F</td><td>G</td><td>I</td><td>K</td></tr><tr><td>L</td><td>P</td><td>Q</td><td>S</td><td>T</td></tr><tr><td>U</td><td>V</td><td>W</td><td>X</td><td>Z</td></tr></table>	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:	<table border="1"><tr><td>M</td><td>O</td><td>N</td><td>A</td><td>R</td></tr><tr><td>C</td><td>H</td><td>Y</td><td>B</td><td>D</td></tr><tr><td>E</td><td>F</td><td>G</td><td>I</td><td>K</td></tr><tr><td>L</td><td>P</td><td>Q</td><td>S</td><td>T</td></tr><tr><td>U</td><td>V</td><td>W</td><td>X</td><td>Z</td></tr></table>	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:	<table border="1"><tr><td>M</td><td>O</td><td>N</td><td>A</td><td>R</td></tr><tr><td>C</td><td>H</td><td>Y</td><td>B</td><td>D</td></tr><tr><td>E</td><td>F</td><td>G</td><td>I</td><td>K</td></tr><tr><td>L</td><td>P</td><td>Q</td><td>S</td><td>T</td></tr><tr><td>U</td><td>V</td><td>W</td><td>X</td><td>Z</td></tr></table>	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
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																																																																												
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																																																																												
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																																																																												

### Hill Cipher

Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, ..., Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible  $n \times n$  matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible  $n \times n$  matrices (modulo 26).

Hill Cipher (Encryption)

### The Hill Algorithm

This can be expressed as

$$C = E(K,P) = P \times K \pmod{26}$$

$$P = D(K,C) = C K^{-1} \pmod{26} = P \times K \times K^{-1} \pmod{26}$$

$$(C_1 \ C_2 \ C_3) = (P_1 \ P_2 \ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \pmod{26} \leftarrow \text{Encryption}$$

$$C_1 = (P_1 K_{11} + P_2 K_{21} + P_3 K_{31}) \pmod{26}$$

$$C_2 = (P_1 K_{12} + P_2 K_{22} + P_3 K_{32}) \pmod{26}$$

$$C_3 = (P_1 K_{13} + P_2 K_{23} + P_3 K_{33}) \pmod{26}$$

## Hill Cipher Example

**Question:** Encrypt "pay more money" using Hill cipher with key

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

**Solution:**

p	a	y	m	o	r	e	m	o	n	e	y
15	0	24	12	14	17	4	12	14	13	4	24

**Key** = 3 x 3 matrix.

**PT** = pay mor emo ney

## Hill Cipher Example

**Encrypting:** pay

$$(C_1 C_2 C_3) = (P_1 P_2 P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \text{ mod } 26$$

$$\begin{aligned} (C_1 C_2 C_3) &= (15 \ 0 \ 24) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \text{ mod } 26 \\ &= (15 \times 17 + 0 \times 21 + 24 \times 2 \quad 15 \times 17 + 0 \times 18 + 24 \times 2 \quad 15 \times 5 + 0 \times 21 + 24 \times 19) \text{ mod } 26 \\ &= (303 \ 303 \ 531) \text{ mod } 26 \\ &= (17 \ 17 \ 11) \\ &= (R \ R \ L) \end{aligned}$$

## Hill Cipher Example

**Encrypting:** mor

$$(C_1 C_2 C_3) = (P_1 P_2 P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \text{ mod } 26$$

$$\begin{aligned} (C_1 C_2 C_3) &= (12 \ 14 \ 17) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \text{ mod } 26 \\ &= (12 \times 17 + 14 \times 21 + 17 \times 2 \quad 12 \times 17 + 14 \times 18 + 17 \times 2 \quad 12 \times 5 + 14 \times 21 + 17 \times 19) \text{ mod } 26 \\ &= (532 \ 490 \ 677) \text{ mod } 26 \\ &= (12 \ 22 \ 1) \\ &= (M \ W \ B) \end{aligned}$$

## Hill Cipher Example

Encrypting: emo

$$(C_1 C_2 C_3) = (P_1 P_2 P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \pmod{26}$$

$$\begin{aligned} (C_1 C_2 C_3) &= (4 12 14) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \pmod{26} \\ &= (4 \times 17 + 12 \times 21 + 14 \times 2 \quad 4 \times 17 + 12 \times 18 + 14 \times 2 \quad 4 \times 5 + 12 \times 21 + 14 \times 19) \pmod{26} \\ &= (348 \ 312 \ 538) \pmod{26} \\ &= (10 \ 0 \ 18) \\ &= (K \ A \ S) \end{aligned}$$

NESO ACADEMY

## Hill Cipher Example

Encrypting: ney

$$(C_1 C_2 C_3) = (P_1 P_2 P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \pmod{26}$$

$$\begin{aligned} (C_1 C_2 C_3) &= (13 \ 4 \ 24) \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \pmod{26} \\ &= (13 \times 17 + 4 \times 21 + 24 \times 2 \quad 13 \times 17 + 4 \times 18 + 24 \times 2 \quad 13 \times 5 + 4 \times 21 + 24 \times 19) \pmod{26} \\ &= (348 \ 312 \ 538) \pmod{26} \\ &= (15 \ 3 \ 7) \\ &= (P \ D \ H) \end{aligned}$$

NESO ACADEMY

## Hill Cipher Example

PT	p	a	y	m	o	r	e	m	o	n	e	y
CT	R	R	L	M	W	B	K	A	S	P	D	H

**Question:** Encrypt "pay more money" using Hill cipher with key

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

**Plaintext** : pay more money

**Ciphertext** : RRLMWBKASPDH

## TRANSPOSITION TECHNIQUES

Transposition techniques differ from substitution techniques in the way that they do not simply replace one alphabet with another: they also perform some permutation over the plain text alphabets.

### Rail Fence Technique

Rail fence technique involves writing plain text as sequence of diagonals and then reading it row-by-row to produce cipher text.

Suppose that we have a plain text message Come home tomorrow. How would we transform that into a cipher text message using the Rail Fence Technique? This is shown in Fig.

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in Step 1 as a sequence of rows.
3. here depth=2

Original plain text message: **Come home tomorrow**

1. After we arrange the plain text message as a sequence of diagonals, it would look as follows (write the first character on the first line i.e. C, then second character on the second line, i.e. o, then the third character on the first line, i.e. m, then the fourth character on the second line, i.e. e, and so on). This creates a zigzag sequence, as shown below.

2. Now read the text row-by-row, and write it sequentially. Thus, we have: **Cmhmtmrooeoorw** as the cipher text.

Example of rail technique

### Simple Columnar Transposition Technique

Basic Technique Variations of the basic transposition technique such as Rail Fence Technique exist. Such a scheme call as Simple Columnar Transposition Technique.

- Write the plain text message row-by-row in a rectangle of a pre-defined size.

- Read the message column-by-column. However, it need not be in the order of columns 1,2, 3 etc. It can be any random order such as 2, 3, 1, etc.
- The message thus obtained is the cipher text message.

The Simple Columnar Transposition Technique simply arranges the plain text as a sequence of rows of a rectangle that are read in columns randomly.

Original plain text message: **Come home tomorrow**

- Let us consider a rectangle with six columns. Therefore, when we write the message in the rectangle row-by-row (suppressing spaces), it would look as follows:

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
C	o	m	e	h	o
m	e	t	o	m	o
r	r	o	w		

- Now, let us decide the order of columns as some random order, say 4, 6, 1, 2, 5 and 3. Then read the text in the order of these columns.
- The cipher text thus obtained would be **eowoocmroerhmmto**.

Example of simple columnar technique

## STEGANOGRAPHY

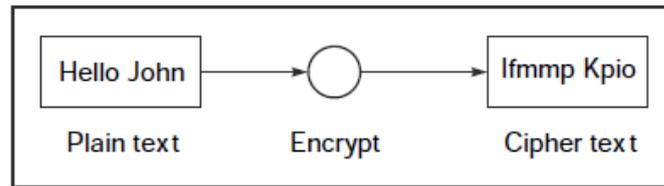
Steganography is a technique that facilitates hiding of a message that is to be kept secret inside other messages.

The sender used methods such as invisible ink, tiny pin punctures on specific characters, minute variations between handwritten characters, pencil marks on handwritten characters, etc.

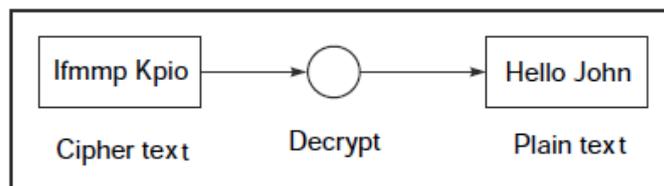
<pre>1100101 0010100 1111111 0001111</pre>		
Secret message	Original image and its bits	Resulting image and its bits

## ENCRYPTION AND DECRYPTION

The process of encoding plain text messages into cipher text messages is called as **encryption**. The process of transforming cipher text messages back to plain text messages is called as **decryption**.



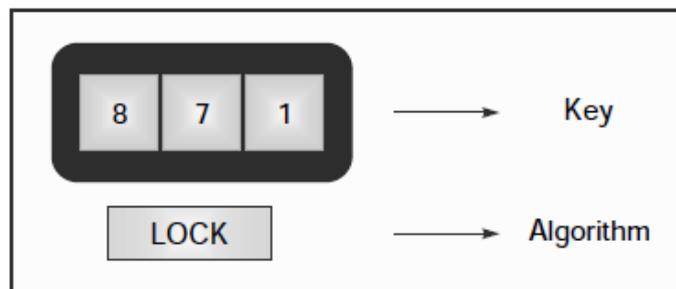
┆ Fig. 2.40 *Encryption*



┆ Fig. 2.41 *Decryption*

Every encryption and decryption process has two aspects: the **algorithm** and the **key** used for encryption and decryption.

let us take the example of a combination lock, which we use in real life. We need to remember the combination (which is a number, such as 871) needed to open up the lock. The facts that it is a combination lock and how to open it (algorithm) are pieces of public knowledge. However, the actual value of the key required for opening a specific lock (key), which is 871 in this case, is kept secret. The idea is illustrated in Fig



Broadly, there are two cryptographic mechanisms, depending on **what keys are used**. If the same key is used for encryption and decryption, we call the mechanism as **Symmetric Key Cryptography**. However, if two different keys are used in a cryptographic mechanism, wherein one key is used for encryption and another, different key is used for decryption; we call the mechanism as **Asymmetric Key Cryptography**.

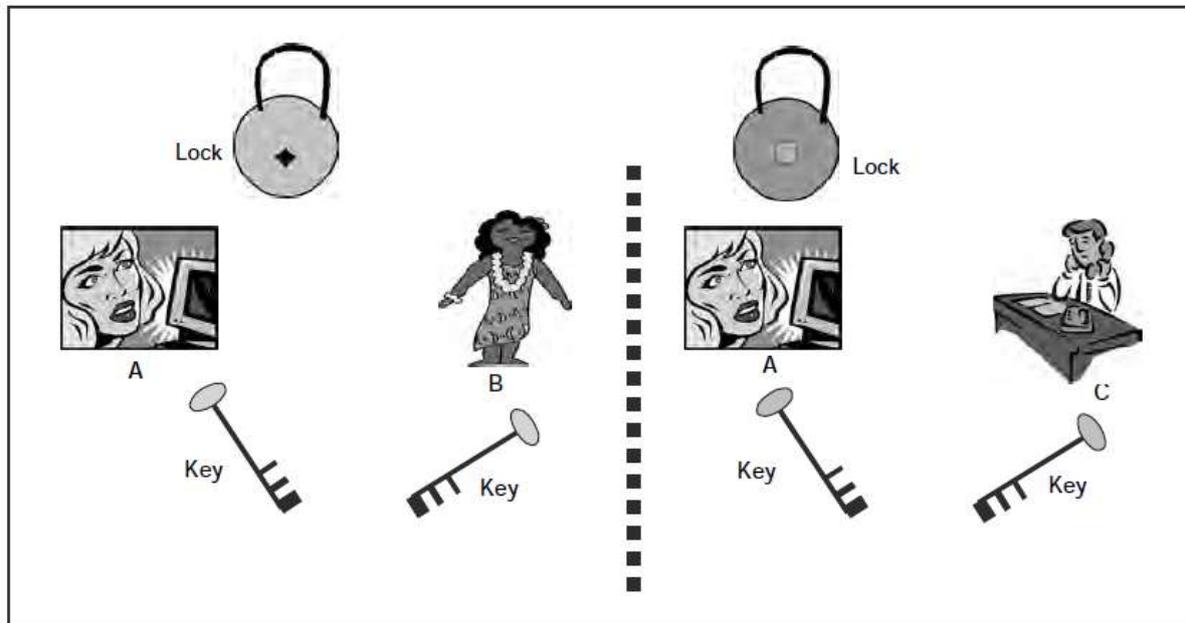
### Symmetric and Asymmetric Key Cryptography

The sender and the receiver will use the same key to lock and unlock, this is called as symmetric key operation (when used in the context of cryptography, this operation is called as **symmetric key cryptography**).

Person A wants to send a highly confidential letter to another person B. A and B both reside in the same city, but are separated by a few miles and for some reason, cannot meet each other.

With the symmetric key cryptography A can send securely to the B.

Let us now imagine that not only A and B but also thousands of people want to send such confidential letters securely to each other. What would happen if they decide to go for symmetric key operation? If we examine this approach more closely, we can see that it has one big drawback if the number of people that want to avail of its services is very large.



Use of separate locks and keys per communication pair

we have the following situation:

- When A wanted to communicate only with B, we needed one lock-and-key pair (A-B).
- When A wants to communicate with B and C, we need two lock-and-key pairs (A-B and A-C).

Thus, we need one lock-and-key pair per person with whom A wants to communicate. If B also wants to communicate with C, we have B-C as the third communicating pair, requiring its own lock-and-key pair. Thus, we would need three lock-and-key pairs to serve the needs of three communicating pairs.

Therefore, can we see that, in general, for n persons, the number of lock-and-key pairs is

$$\frac{n*(n-1)}{2}$$

Parties involved	Number of lock-and-key pairs required
2 (A, B)	1 (A-B)
3 (A, B, C)	3 (A-B, A-C, B-C)
4 (A, B, C, D)	6 (A-B, A-C, A-D, B-C, B-D, C-D)
5 (A, B, C, D, E)	10 (A-B, A-C, A-D, A-E, B-C, B-D, B-E, C-D, C-E, D-E)

### Diffie–Hellman Key Exchange/Agreement Algorithm

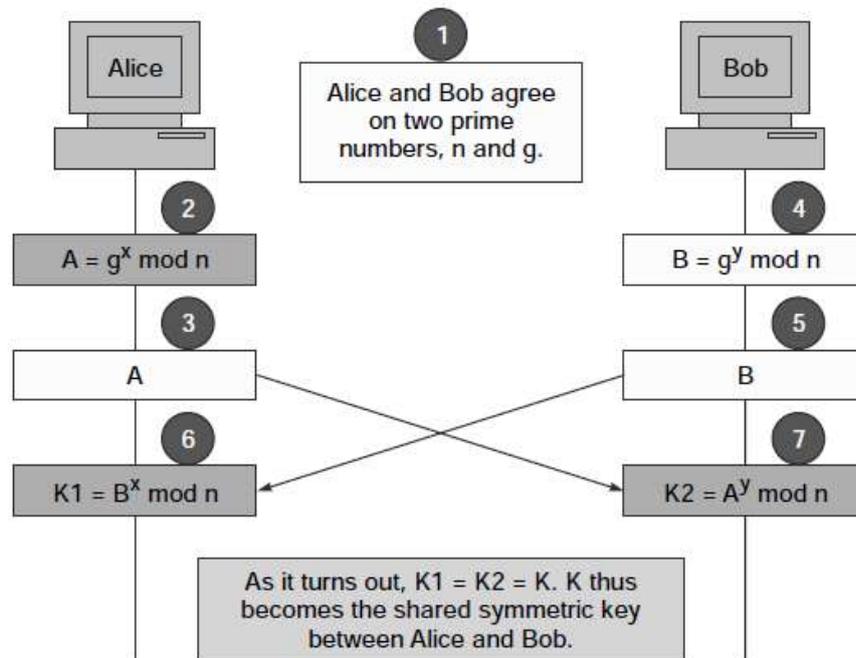
In this scheme the two parties, who want to communicate securely, can agree on a **symmetric key** using this technique. This key can then be used for encryption/ decryption. However, we must note that Diffie–Hellman key exchange algorithm can be used only for key agreement, but not for encryption or decryption of messages. Once both the parties agree on the key to be

used, they need to use other symmetric key encryption algorithms for actual encryption or decryption of messages.

### Description of the Algorithm

1. Firstly, Alice and Bob agree on two large prime numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number  $x$ , and calculates  $A$  such that:  
 $A = g^x \text{ mod } n$
3. Alice sends the number  $A$  to Bob.
4. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that:  
 $B = g^y \text{ mod } n$
5. Bob sends the number  $B$  to Alice.
6. A now computes the secret key  $K1$  as follows:  
 $K1 = B^x \text{ mod } n$
7. B now computes the secret key  $K2$  as follows:  
 $K2 = A^y \text{ mod } n$

### *Diffie–Hellman key exchange algorithm*



## Example of the Algorithm

1. Firstly, Alice and Bob agree on two large prime numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.  

Let  $n = 11, g = 7$ .
2. Alice chooses another large random number  $x$ , and calculates  $A$  such that:  
 $A = g^x \bmod n$   

Let  $x = 3$ . Then, we have,  $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$ .
3. Alice sends the number  $A$  to Bob.  

Alice sends 2 to Bob.
4. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that:  
 $B = g^y \bmod n$   

Let  $y = 6$ . Then, we have,  $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$ .
5. Bob sends the number  $B$  to Alice.  

Bob sends 4 to Alice.
6. A now computes the secret key  $K1$  as follows:  
 $K1 = B^x \bmod n$   

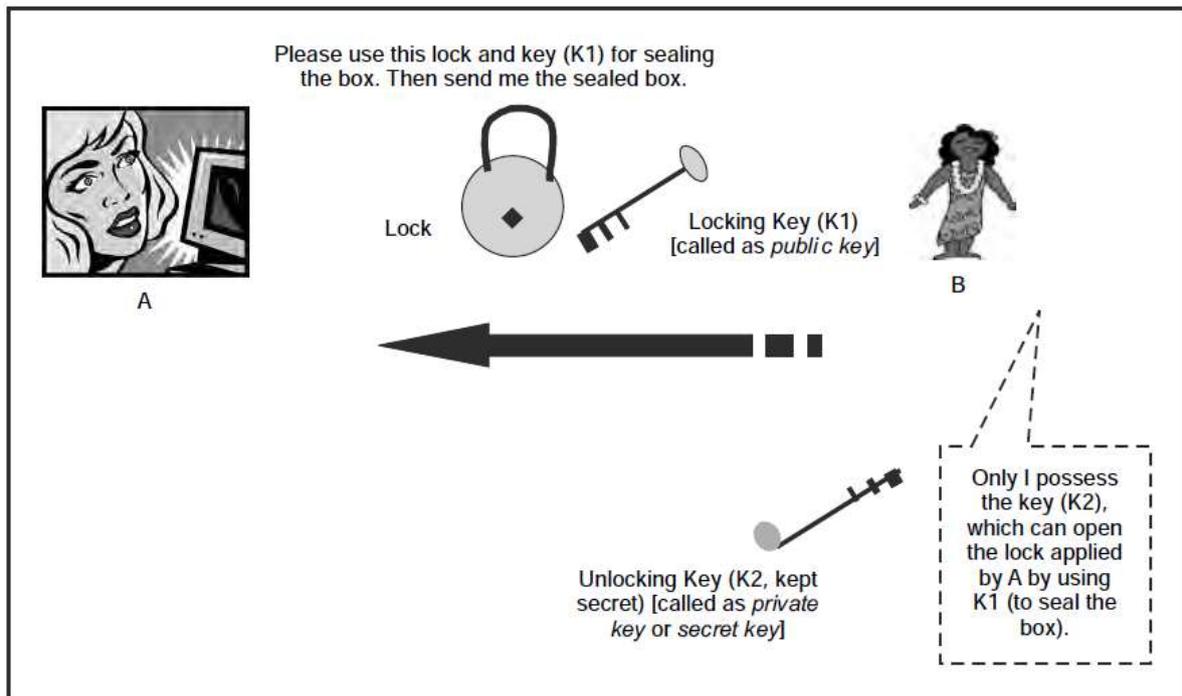
We have,  $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$ .
7. B now computes the secret key  $K2$  as follows:  
 $K2 = A^y \bmod n$   

We have,  $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$ .

## Asymmetric Key Operation

In this scheme, (Alice)  $A$  and (Bob)  $B$  do not have to jointly approach (Tom)  $T$  for a lock-and-key pair. Instead,  $B$  alone approaches  $T$ , obtains a lock and a key ( $K1$ ) that can seal the lock and sends the lock and key  $K1$  to  $A$ .  $B$  tells  $A$  that  $A$  can use that lock and key to seal the box before sending the sealed box to  $B$ .

An interesting property of this scheme is that  $B$  possesses a different but related key ( $K2$ ), which is obtained by  $B$  from  $T$  along with the lock and key  $K1$ , only which can open the lock. It is guaranteed that no other key and of course, including the one used by  $A$  (i.e.  $K1$ ) for locking, can open the lock. Since one key ( $K1$ ) is used for locking and another, different key ( $K2$ ) is used for unlocking; we will call this scheme as asymmetric key operation. Also,  $T$  is clearly defined here as a **trusted third party**.  $T$  is certified as a highly trustworthy and efficient agency by the government.

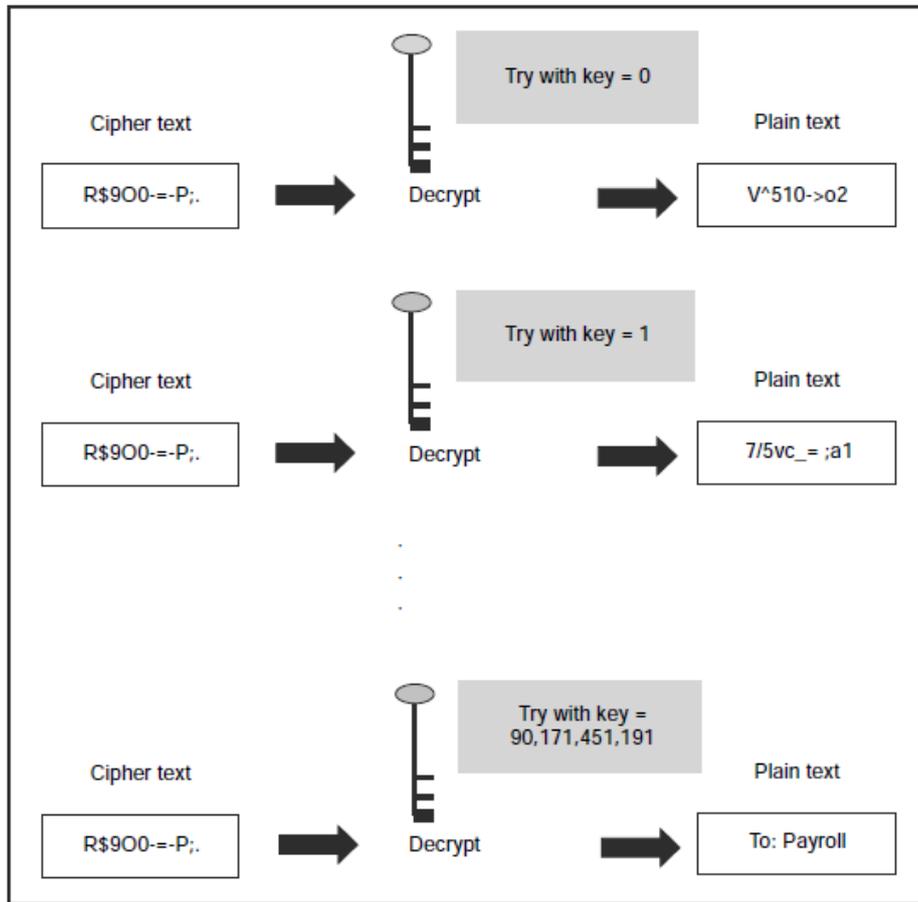


## KEY RANGE AND KEY SIZE

The cryptanalyst is armed with the following information:

- The encryption/decryption algorithm
- The encrypted message
- Knowledge about the key size (e.g. the value of the key is a number between 0 and 100 billion).

For example consider the **brute force attack** here, which works on the principle of trying every possible key in the key range, until you get the right key.



Brute force attack

A 2-bit binary number has four possible states:

- 00
- 01
- 10
- 11

If we have one more bit to make it a 3-bit binary number, the number of possible states also doubles to eight, as follows:

- 000
- 001
- 010
- 011
- 100
- 101
- 110
- 111

In general, if an  $n$  bit binary number has  $k$  possible states, an  $n+1$  bit binary number will have  $2k$  possible states.

Understanding key range

With every incremental bit, the attacker has to perform double the number of operations as compared to the previous key size. It is found that for a 56-bit key, it takes 1 second to search 1 percent of the key range. Taking this argument further, it takes about 1 minute to search about half of the key range (which is what is required, on an average, to crack a key). Using this as the basis, let us have a look at the similar values (time required for a search of 1 percent and 50 percent of the key space) for various key sizes. This is shown in Table

<i>Key size on bits</i>	<i>Time required to search 1 percent of the key space</i>	<i>Time required to search 50 percent of the key space</i>
56	1 second	1 minute
57	2 seconds	2 minutes
58	4 seconds	4 minutes
64	4.2 minutes	4.2 hours
72	17.9 hours	44.8 days
80	190.9 days	31.4 years
90	535 years	321 centuries
128	146 billion millennia	8 trillion millennia

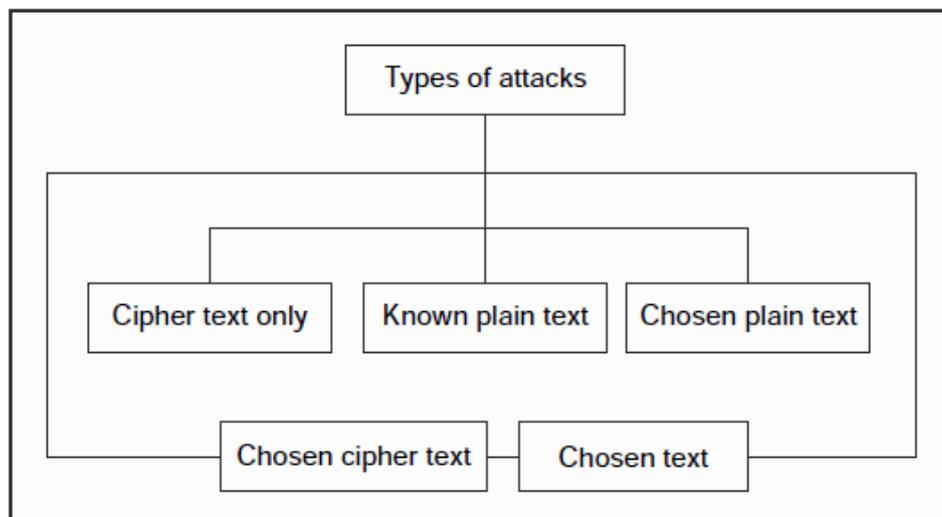
We can represent the possible values in the key range using hexadecimal notation and see visually how an increase in the key size increases the key range and therefore, the complexity for an attacker.



Key sizes and ranges

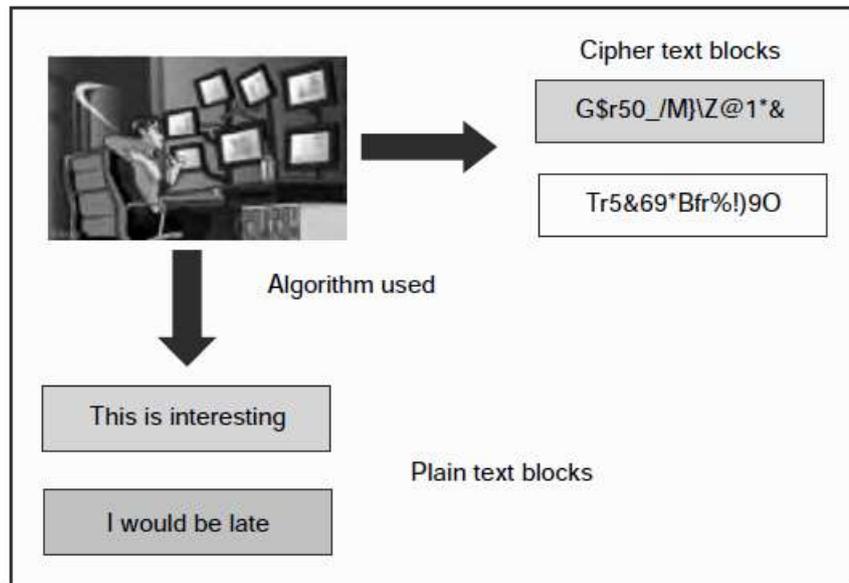
## POSSIBLE TYPES OF ATTACKS

When the sender of a message encrypts a plain text message into its corresponding cipher text, there are five possibilities for an attack on this message.



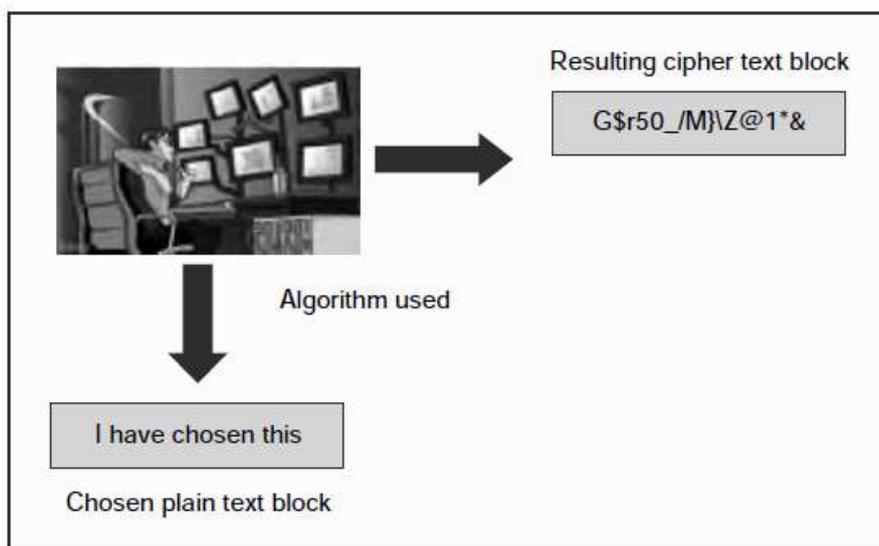
**Cipher text only attack:** In this type of attack, the attacker does not have any clue about the plain text and has some or all of the cipher text. The attacker analyzes the cipher text at leisure to try and figure out the original plain text.

**Known plain text attack:** In this case, the attacker knows about some pairs of plain text and corresponding cipher text for those pairs. Using this information, the attacker tries to find other pairs and therefore, know more and more of the plain text. Examples of such known plain texts are company banners, file headers, etc. which are found commonly in all the documents of a particular company.



Known plain text attack

**Chosen plain text attack:** Here, the attacker selects a plain text block and tries to look for the encryption of the same in the cipher text. Here, the attacker is able to choose the messages to encrypt. Based on this, the attacker intentionally picks patterns of cipher text that result in obtaining more information about the key.



Chosen plain text attack

**Chosen cipher text attack:** In the chosen cipher text attack, the attacker knows the cipher text to be decrypted, the encryption algorithm that was used to produce this cipher text and the corresponding plain text block. The attacker's job is to discover the key used for encryption.

**Chosen text attack:** The chosen text attack is essentially a combination of chosen plain text attack and chosen cipher text attack.

## BLOCK CIPHER PRINCIPLES

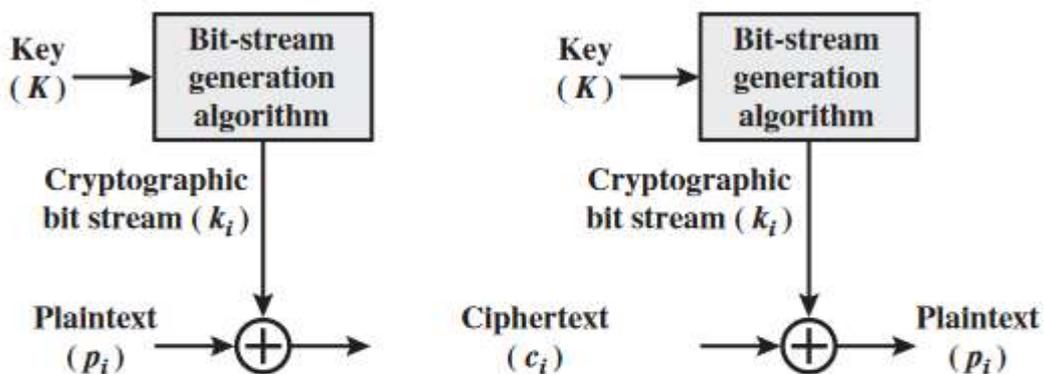
### Stream Ciphers and Block Ciphers

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 the autokeyed Vigenère cipher and the Vernam cipher.

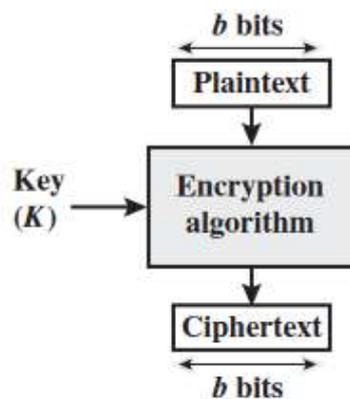
In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the keystream ( $K_i$ ) is as long as the plaintext bit stream ( $P_i$ ). If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream.

The bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong.



(a) Stream cipher using algorithmic bit-stream generator

A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key.



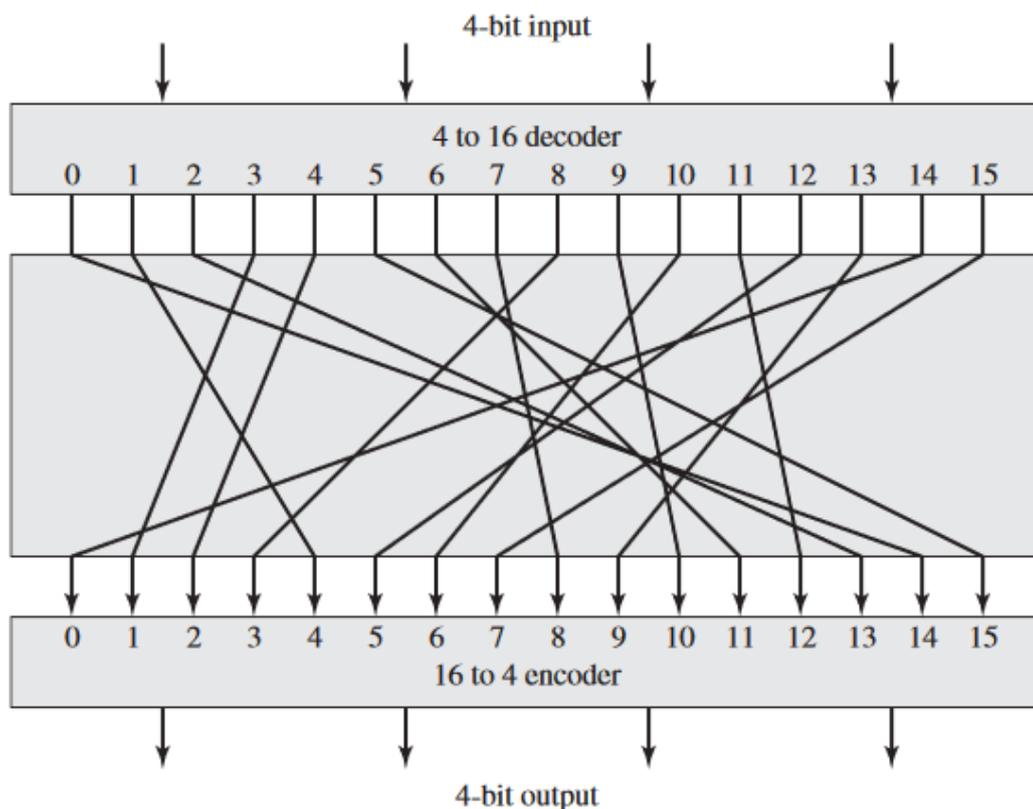
(b) Block cipher

### 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 and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block.

Reversible Mapping		Irreversible Mapping	
Plaintext	Ciphertext	Plaintext	Ciphertext
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

The logic of a general substitution cipher for 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.



**Figure 3.2** General  $n$ -bit- $n$ -bit Block Substitution (shown with  $n = 4$ ).

**Table 3.1 Encryption and Decryption Tables for Substitution Cipher of Figure 3.2**

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

### Block Cipher Principles

A block cipher is designed by considering its three critical aspects which are listed as below:

1. Number of Rounds
2. Design of Function F
3. Key Schedule Algorithm

#### 1. Number of Rounds

The number of rounds judges the strength of the block cipher algorithm. It is considered that more is the number of rounds, difficult is for cryptanalysis to break the algorithm.

It is considered that even if the function F is relatively weak, the number of rounds would make the algorithm tough to break.

#### 2. Design of Function F

The function F of the block cipher must be designed such that it must be impossible for any cryptanalysis to unscramble the substitution. The criterion that strengthens the function F is its non-linearity.

More the function F is nonlinear, more it would be difficult to crack it. Well, while designing the function F it should be confirmed that it has a good avalanche property which states that a change in one-bit of input must reflect the change in many bits of output.

The Function F should be designed such that it possesses a bit independence criterion which states that the output bits must change independently if there is any change in the input bit.

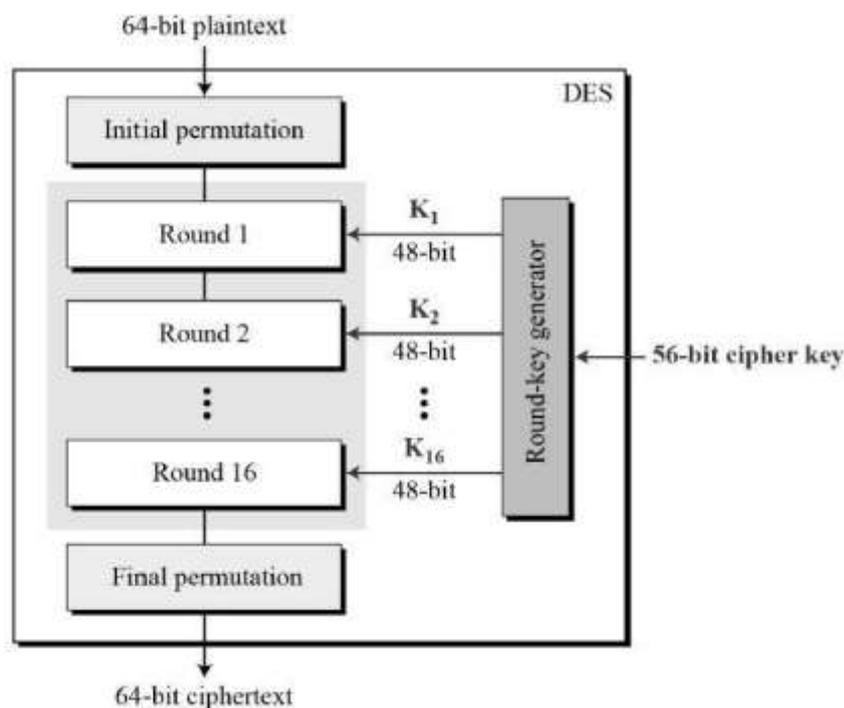
### 3. Key Schedule Algorithm

It is suggested that the key schedule should confirm the strict avalanche effect and bit independence criterion.

## DATA ENCRYPTION STANDARD

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration –

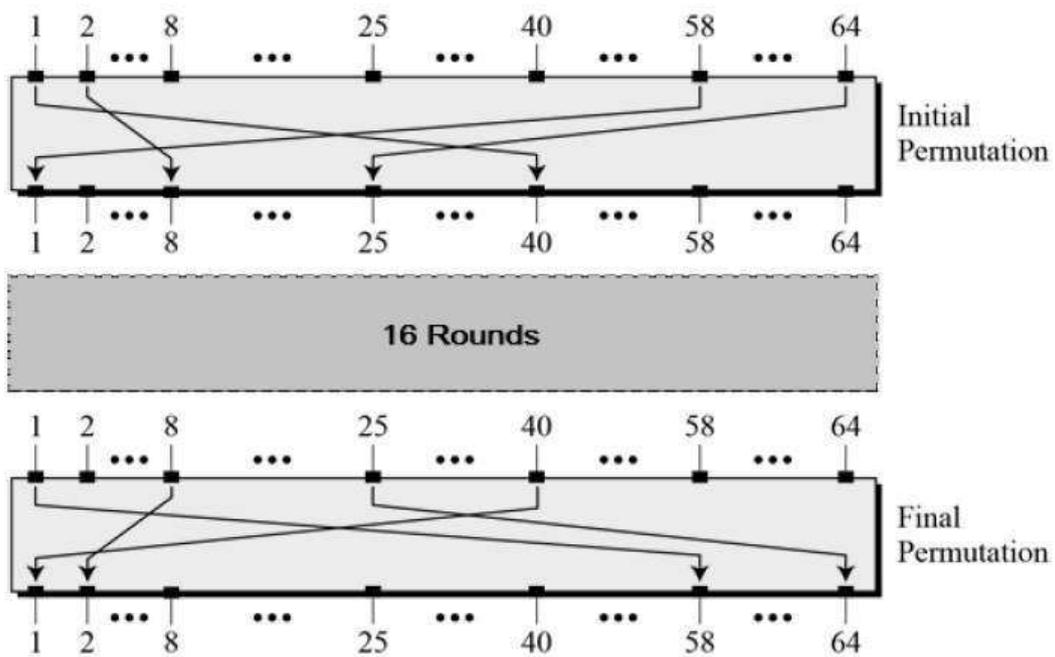


Since DES is based on the Feistel Cipher, all that is required to specify DES is –

- Round function
- Key schedule
- Any additional processing – Initial and final permutation

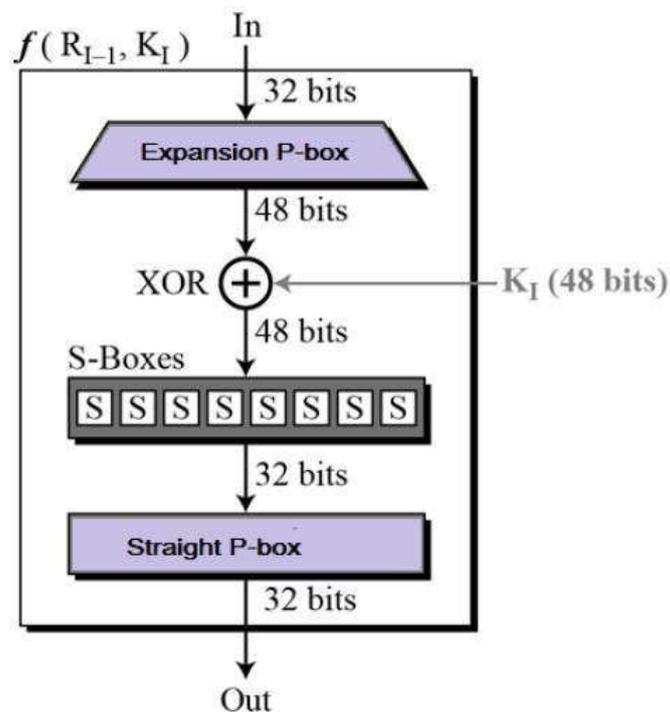
### Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows –

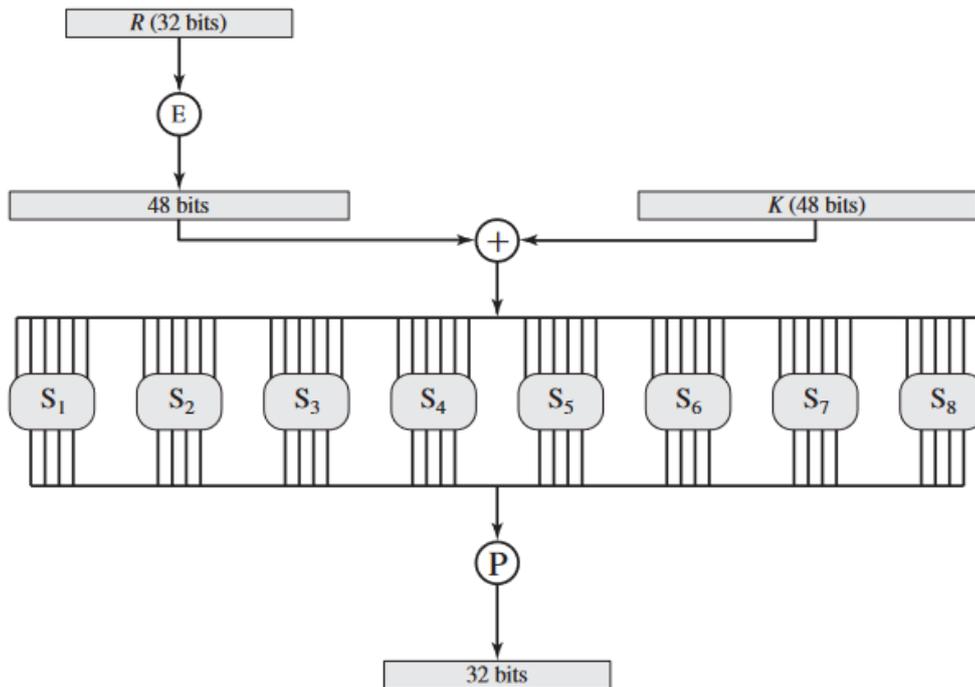


### Round Function

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



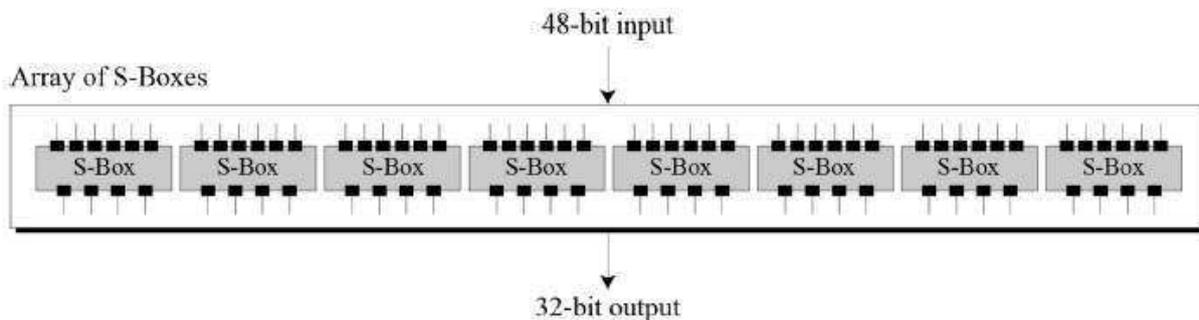
**Expansion Permutation Box** – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration



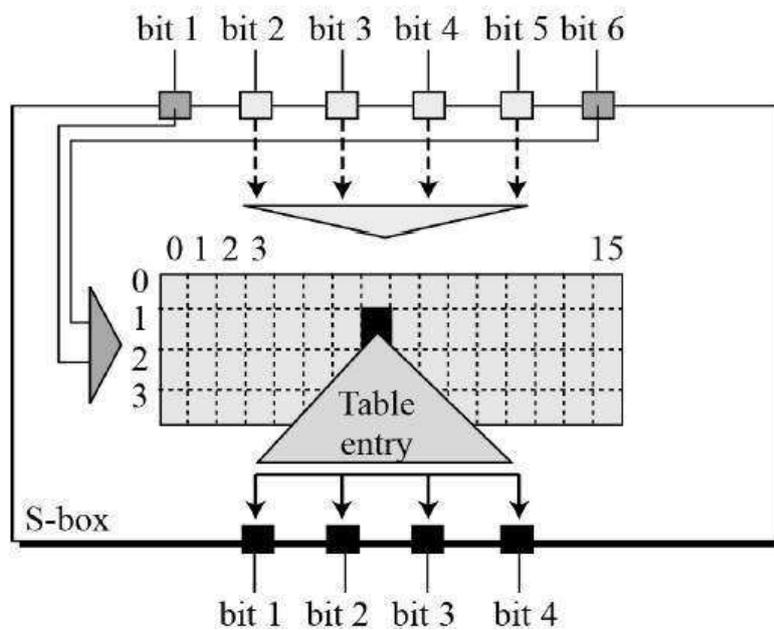
The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown

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

- **XOR (Whitener).** – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** – The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration –



The S-box rule is illustrated below

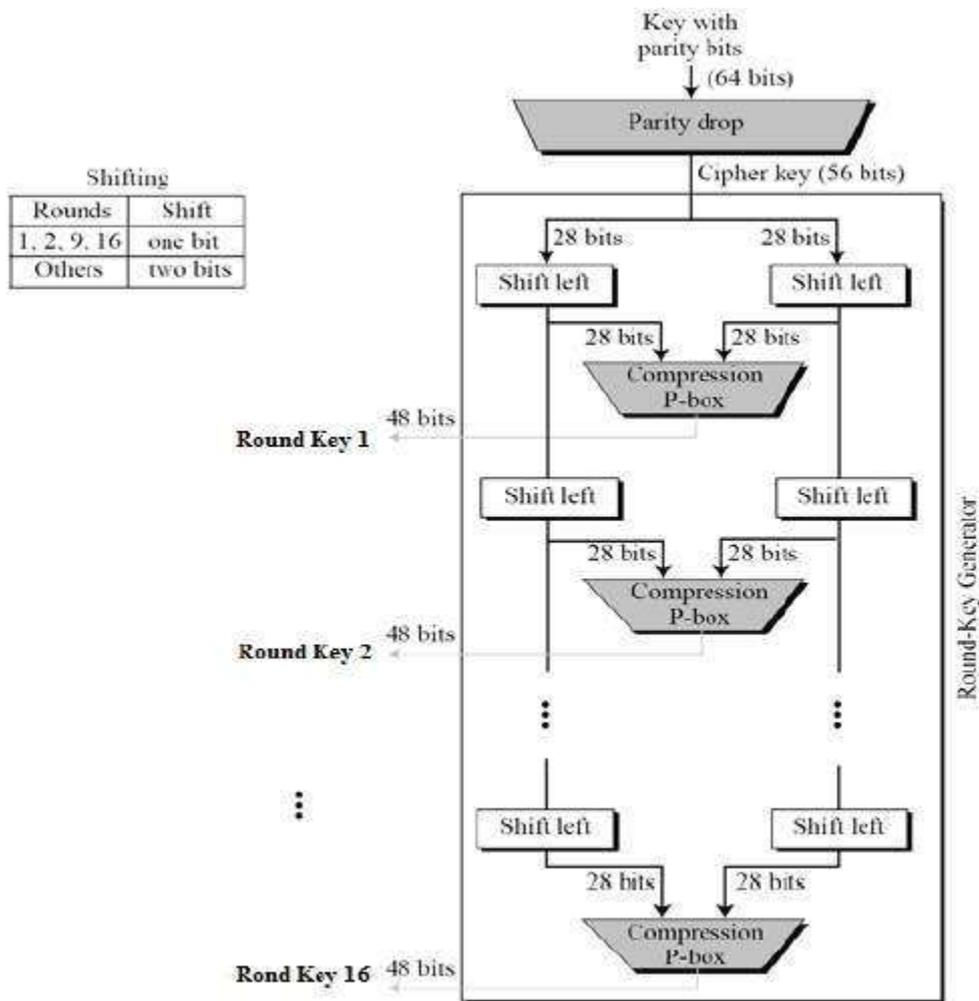


□ There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.

□ **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

### Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration



## ADVANCED ENCRYPTION STANDARD

The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001. AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications.

AES is a block cipher intended to replace DES for commercial applications. It uses a 128-bit block size and a key size of 128, 192, or 256 bits.

AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key.

### General Structure

Figure shows the overall structure of the AES encryption process. The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.

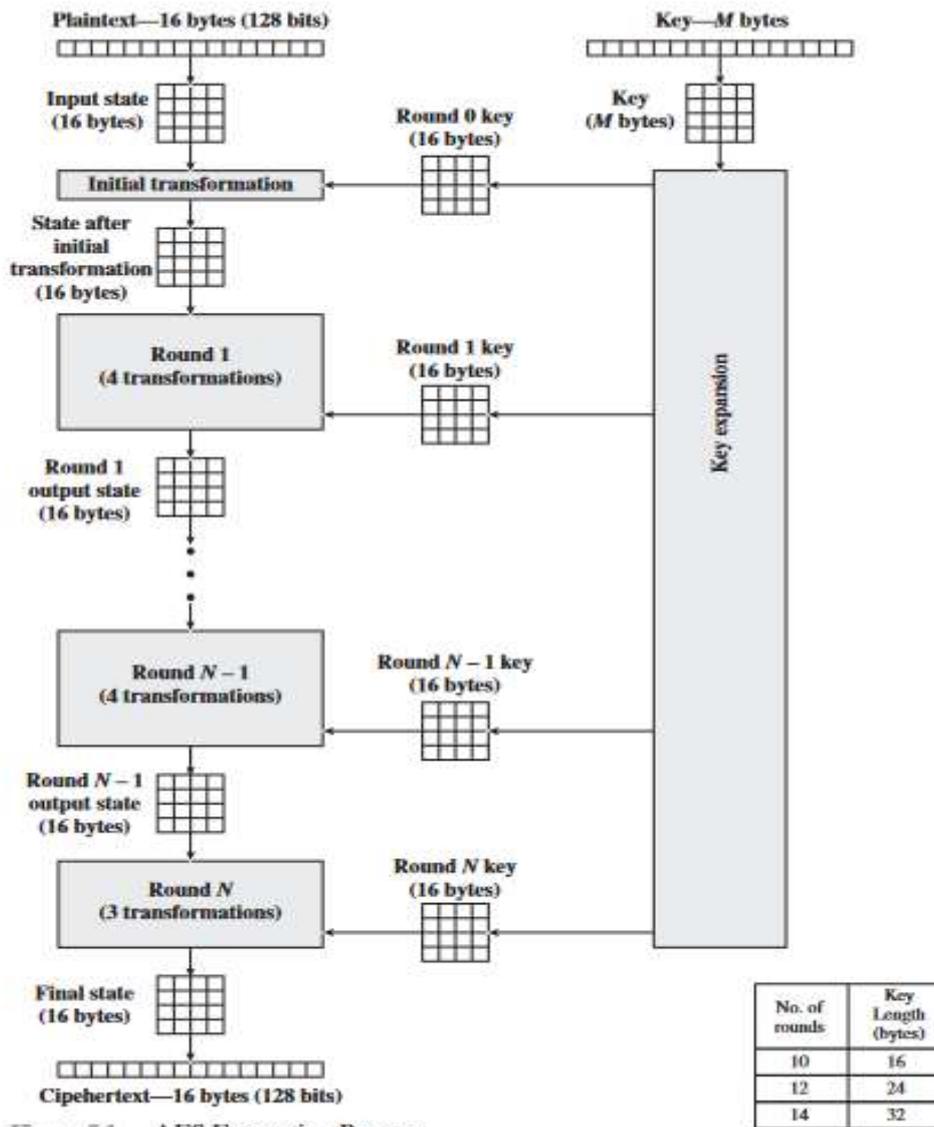


Figure 5.1 AES Encryption Process

Advanced Encryption Standard is found at least six times faster than triple DES. A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows –

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

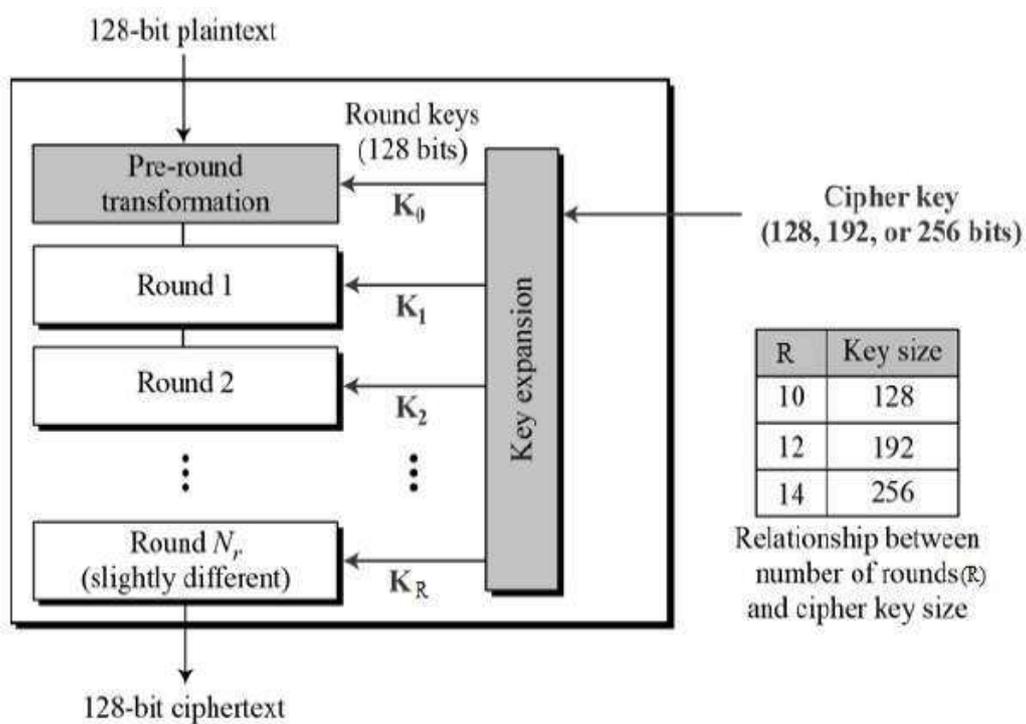
## Operation of AES

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix –

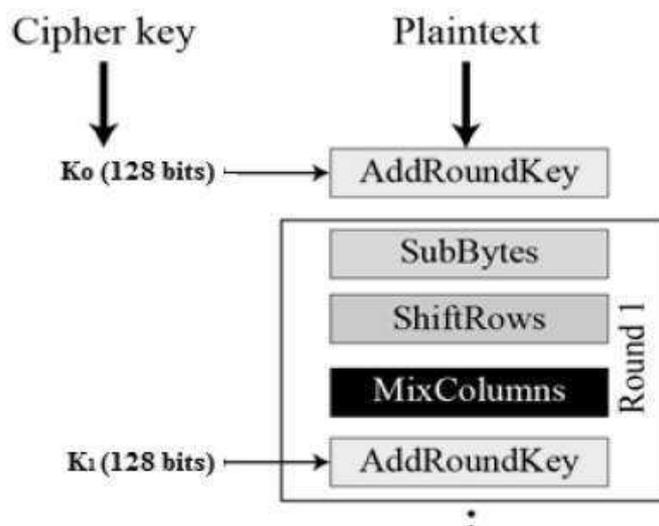
Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration



### Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below –



### Byte Substitution (SubBytes)

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

### Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row. Shift is carried out as follows –

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

$$\begin{array}{cc}
 [ b_0 | b_1 | b_2 | b_3 ] & [ b_0 | b_1 | b_2 | b_3 ] \\
 | b_4 | b_5 | b_6 | b_7 | & \rightarrow | b_5 | b_6 | b_7 | b_4 | \\
 | b_8 | b_9 | b_{10} | b_{11} | & | b_{10} | b_{11} | b_8 | b_9 | \\
 [ b_{12} | b_{13} | b_{14} | b_{15} ] & [ b_{15} | b_{12} | b_{13} | b_{14} ]
 \end{array}$$

### MixColumns

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

$$\begin{array}{l}
 [ c_0 ] \\
 | c_1 | \\
 | c_2 | \\
 [ c_3 ]
 \end{array}
 =
 \begin{array}{l}
 [ 2 \ 3 \ 1 \ 1 ] \\
 | 1 \ 2 \ 3 \ 1 | \\
 | 1 \ 1 \ 2 \ 3 | \\
 [ 3 \ 1 \ 1 \ 2 ]
 \end{array}
 \begin{array}{l}
 [ b_0 ] \\
 || b_1 | \\
 || b_2 | \\
 [ b_3 ]
 \end{array}$$

### Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

### **Decryption Process**

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order –

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

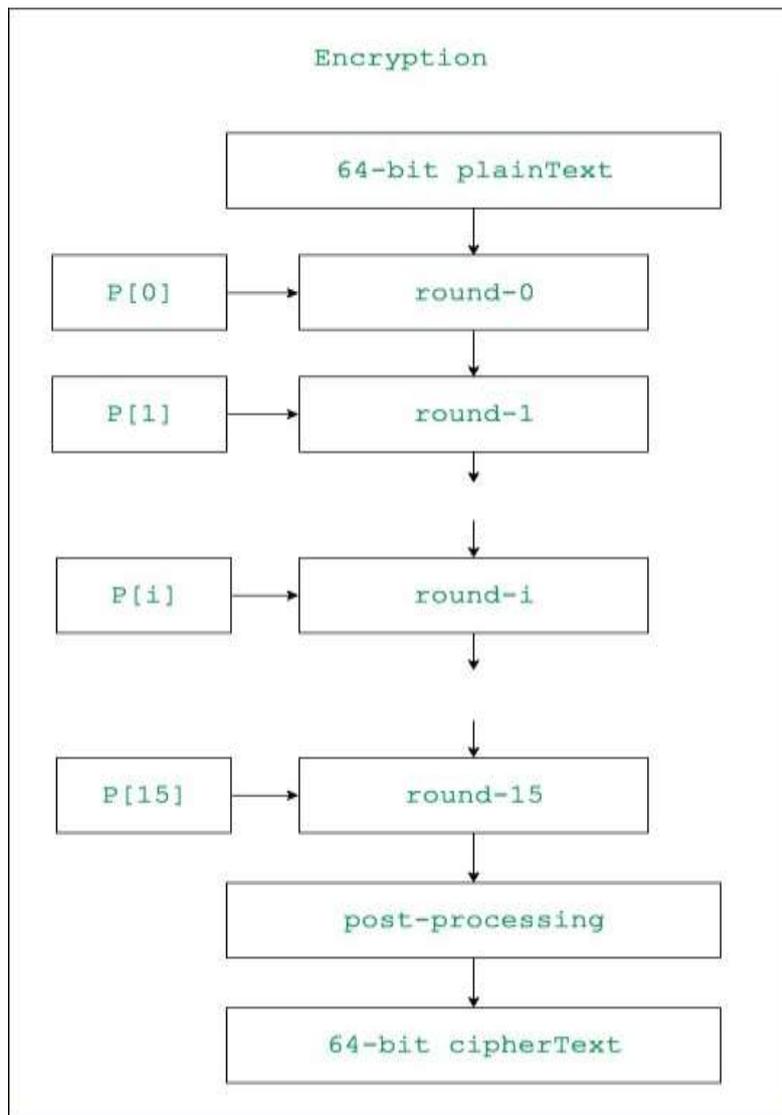
### **BLOWFISH ALGORITHM**

Blowfish is an encryption technique designed by Bruce Schneier in 1993 as an alternative to DES Encryption Technique. It is significantly faster than DES and provides a good encryption rate with no effective cryptanalysis technique found to date. It is one of the first, secure block cyphers not subject to any patents and hence freely available for anyone to use.

1. blockSize: 64-bits
2. keySize: 32-bits to 448-bits variable size
3. number of subkeys: 18 [P-array]
4. number of rounds: 16
5. number of substitution boxes: 4 [each having 512 entries of 32-bits each]

### **Blowfish Encryption Algorithm**

The entire encryption process can be elaborated as:



Lets see each step one by one:

**Step1: Generation of subkeys:**

- 18 subkeys  $\{P[0] \dots P[17]\}$  are needed in both encryption as well as decryption process and the same subkeys are used for both the processes.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

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

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

Now each of the subkey is changed with respect to the input key as:

$P[0] = P[0] \text{ xor } 1\text{st } 32\text{-bits of input key}$   
 $P[1] = P[1] \text{ xor } 2\text{nd } 32\text{-bits of input key}$

.

.

.

$P[i] = P[i] \text{ xor } (i+1)\text{th } 32\text{-bits of input key}$   
(roll over to 1st 32-bits depending on the key length)

.

.

.

$P[17] = P[17] \text{ xor } 18\text{th } 32\text{-bits of input key}$   
(roll over to 1st 32-bits depending on key length)

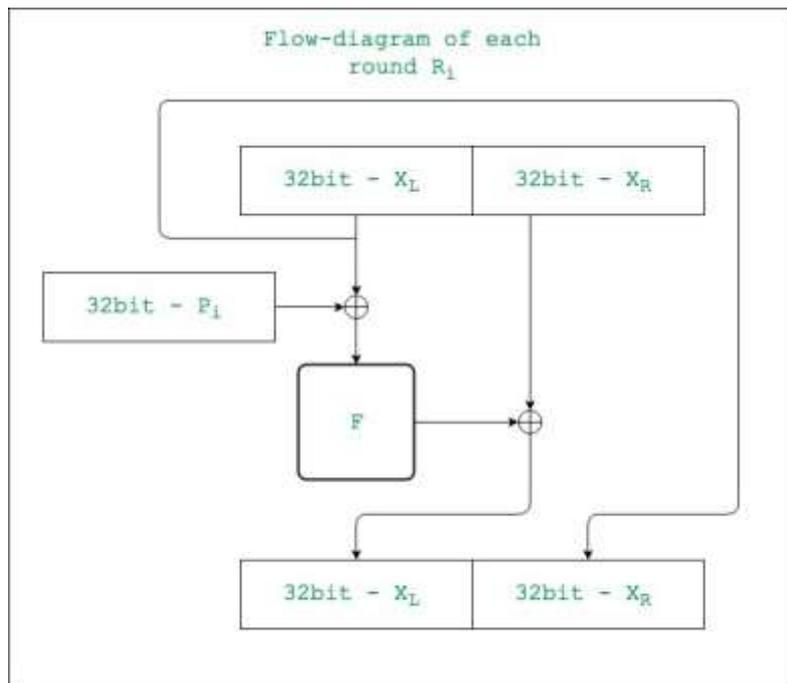
**The resultant P-array holds 18 subkeys that is used during the entire encryption process**

**Step2: initialise Substitution Boxes:**

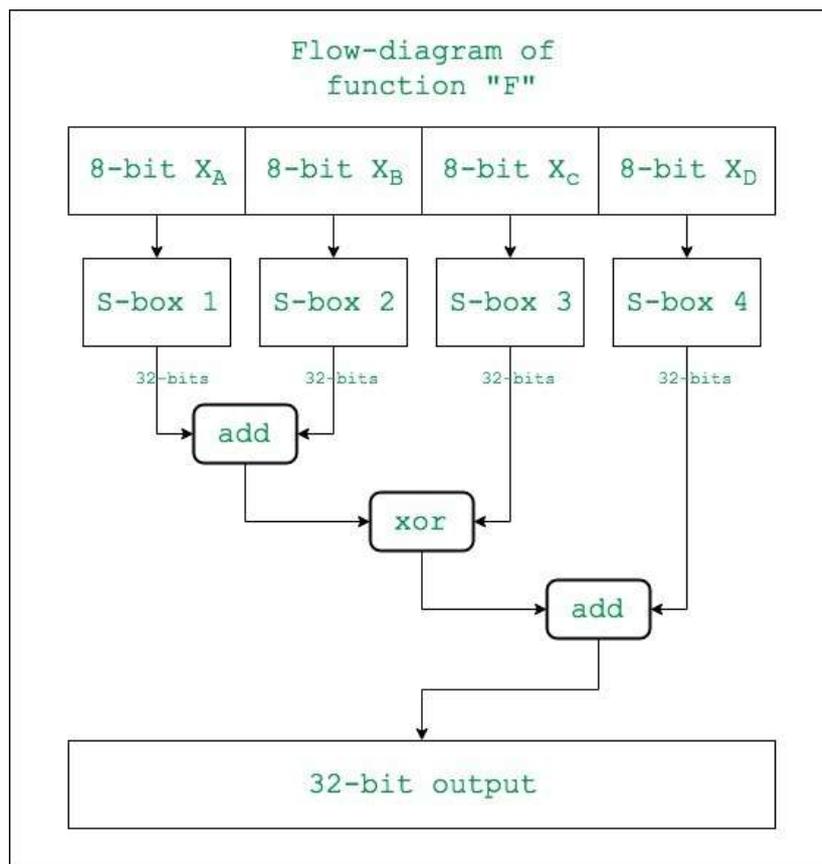
- 4 Substitution boxes(S-boxes) are needed  $\{S[0] \dots S[4]\}$  in both encryption aswell as decryption process with each S-box having 256 entries  $\{S[i][0] \dots S[i][255], 0 \leq i \leq 4\}$  where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array.

**Step3: Encryption:**

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

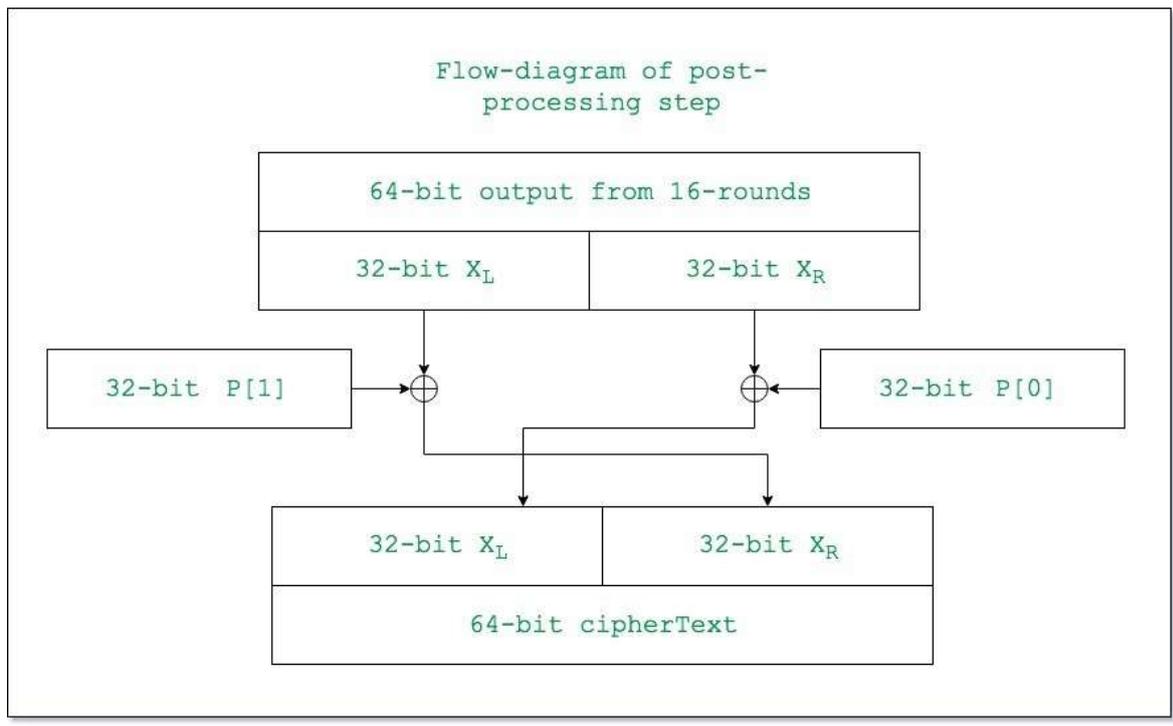


The description of the function "F" is as follows:



Here the function "add" is addition modulo  $2^{32}$ .

**b. Post-processing:** The output after the 16 rounds is processed as follows:



## DIFFERENTIAL AND LINEAR CRYPTANALYSIS

One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis.

**DIFFERENTIAL CRYPTANALYSIS ATTACK** The differential cryptanalysis attack is complex, provides a complete description. The rationale behind differential cryptanalysis is to observe the behaviour of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block.

We begin with a change in notation for DES. Consider the original plaintext block  $m$  to consist of two halves  $m_0, m_1$ . Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round. So, at each round, only one new 32-bit block is created. If we label each new block  $m_i (2 \leq i \leq 17)$ , then the intermediate message halves are related as follows:

$$m_{i+1} = m_{i-1} \oplus f(m_i, K_i), \quad i = 1, 2, \dots, 16$$

In differential cryptanalysis, we start with two messages,  $m$  and  $m'$ , with a known XOR difference  $\Delta m = m \oplus m'$ , and consider the difference between the intermediate message halves:  $\Delta m_i = m_i \oplus m'_i$ . Then we have

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

### Linear Cryptanalysis

This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given  $2^{43}$  known plaintexts, as compared to  $2^{47}$  chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may

be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES.

We now give a brief summary of the principle on which linear cryptanalysis is based. For a cipher with  $n$ -bit plaintext and ciphertext blocks and an  $m$ -bit key, let the plaintext block be labeled  $P[1], \dots, P[n]$ , the cipher text block  $C[1], \dots, C[n]$ , and the key  $K[1], \dots, K[m]$ . Then define

$$A[i, j, \dots, k] = A[i] \oplus A[j] \oplus \dots \oplus A[k]$$

The objective of linear cryptanalysis is to find an effective *linear* equation of the form:

$$P[\alpha_1, \alpha_2, \dots, \alpha_a] \oplus C[\beta_1, \beta_2, \dots, \beta_b] = K[\gamma_1, \gamma_2, \dots, \gamma_c]$$

(where  $x = 0$  or  $1$ ;  $1 \leq a$ ;  $b \leq n$ ;  $c \leq m$ ; and where the  $\alpha$ ,  $\beta$ , and  $\gamma$  terms represent fixed, unique bit locations) that holds with probability  $p \neq 0.5$ . The further  $p$  is from  $0.5$ , the more effective the equation. Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext–ciphertext pairs. If the result is  $0$  more than half the time, assume  $K[\gamma_1, \gamma_2, \dots, \gamma_c] = 0$ . If it is  $1$  most of the time, assume  $K[\gamma_1, \gamma_2, \dots, \gamma_c] = 1$ .

## BLOCK CIPHER MODES OF OPERATION

A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

### Electronic Code Book (ECB) Mode

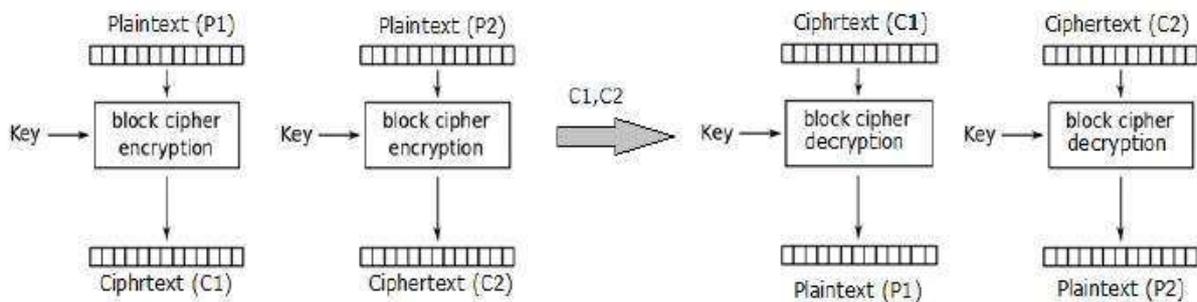
This mode is a most straightforward way of processing a series of sequentially listed message blocks.

#### Operation

- The user takes the first block of plaintext and encrypts it with the key to produce the first block of ciphertext.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

The ECB mode is **deterministic**, that is, if plaintext block  $P_1, P_2, \dots, P_m$  are encrypted twice under the same key, the output ciphertext blocks will be the same.

In fact, for a given key technically we can create a codebook of ciphertexts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding ciphertext. Thus, the operation is analogous to the assignment of code words in a codebook, and hence gets an official name – Electronic Codebook mode of operation (ECB). It is illustrated as follows



### Analysis of ECB Mode

In reality, any application data usually have partial information which can be guessed. For example, the range of salary can be guessed. A ciphertext from ECB can allow an attacker to guess the plaintext by trial-and-error if the plaintext message is within predictable.

For example, if a ciphertext from the ECB mode is known to encrypt a salary figure, then a small number of trials will allow an attacker to recover the figure. In general, we do not wish to use a deterministic cipher, and hence the ECB mode should not be used in most applications.

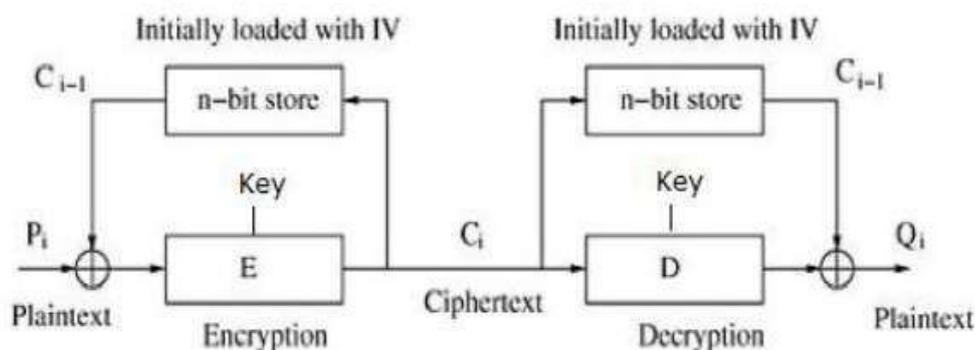
### Cipher Block Chaining (CBC) Mode

CBC mode of operation provides message dependence for generating ciphertext and makes the system non-deterministic.

#### Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows

- Load the n-bit Initialization Vector (IV) in the top register.
- XOR the n-bit plaintext block with data value in top register.
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.
- For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.



### Analysis of CBC Mode

In CBC mode, the current plaintext block is added to the previous ciphertext block, and then the result is encrypted with the key. Decryption is thus the reverse process, which involves decrypting the current ciphertext and then adding the previous ciphertext block to the result.

Advantage of CBC over ECB is that changing IV results in different ciphertext for identical message. On the drawback side, the error in transmission gets propagated to few further block during decryption due to chaining effect.

It is worth mentioning that CBC mode forms the basis for a well-known data origin authentication mechanism. Thus, it has an advantage for those applications that require both symmetric encryption and data origin authentication.

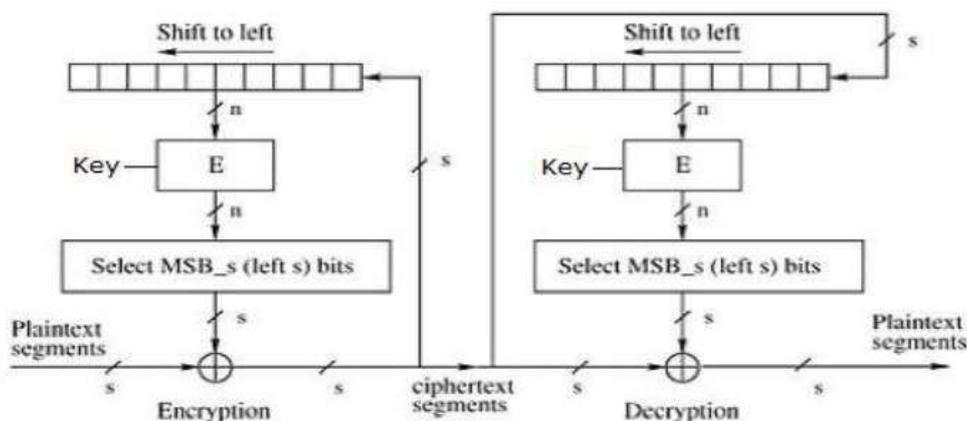
### Cipher Feedback (CFB) Mode

In this mode, each ciphertext block gets ‘fed back’ into the encryption process in order to encrypt the next plaintext block.

#### Operation

The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size ‘s’ bits where  $1 < s < n$ . The CFB mode requires an initialization vector (IV) as the initial random n-bit input block. The IV need not be secret. Steps of operation are –

- Load the IV in the top register.
- Encrypt the data value in top register with underlying block cipher with key K.
- Take only ‘s’ number of most significant bits (left bits) of output of encryption process and XOR them with ‘s’ bit plaintext message block to generate ciphertext block.
- Feed ciphertext block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed.
- Essentially, the previous ciphertext block is encrypted with the key, and then the result is XORed to the current plaintext block.
- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption.



#### Analysis of CFB Mode

CFB mode differs significantly from ECB mode, the ciphertext corresponding to a given plaintext block depends not just on that plaintext block and the key, but also on the previous ciphertext block. In other words, the ciphertext block is dependent of message.

CFB has a very strange feature. In this mode, user decrypts the ciphertext using only the encryption process of the block cipher. The decryption algorithm of the underlying block cipher is never used.

Apparently, CFB mode is converting a block cipher into a type of stream cipher. The encryption algorithm is used as a key-stream generator to produce key-stream that is placed in the bottom register. This key stream is then XORed with the plaintext as in case of stream cipher.

By converting a block cipher into a stream cipher, CFB mode provides some of the advantageous properties of a stream cipher while retaining the advantageous properties of a block cipher.

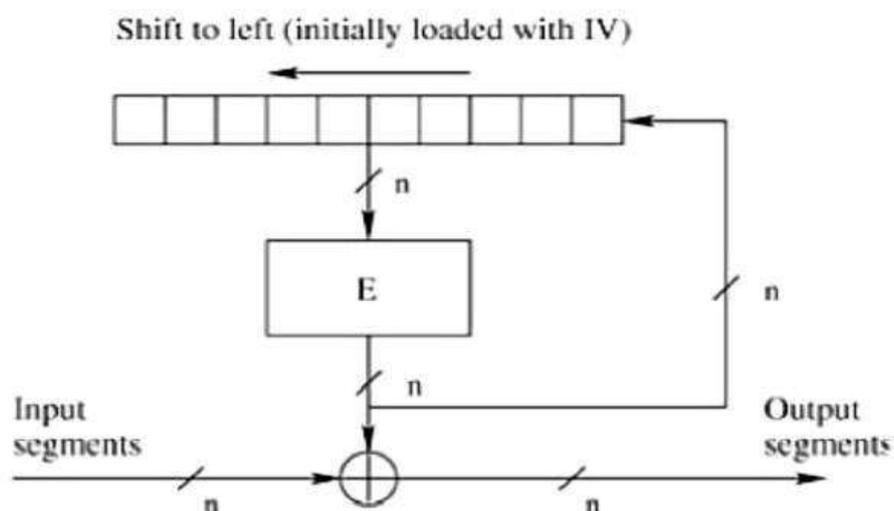
On the flip side, the error of transmission gets propagated due to changing of blocks.

### Output Feedback (OFB) Mode

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

The operation is depicted in the following illustration –



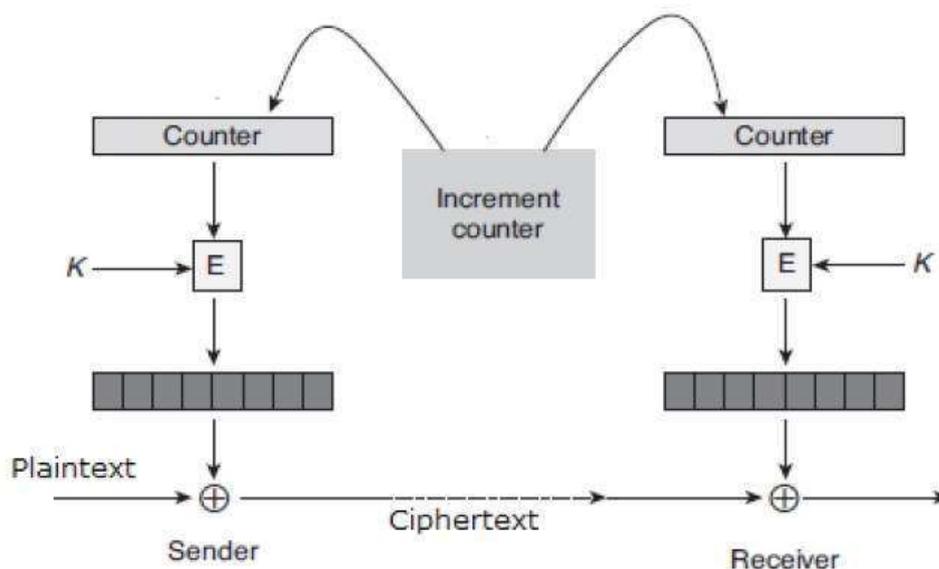
### Counter (CTR) Mode

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

### Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are –

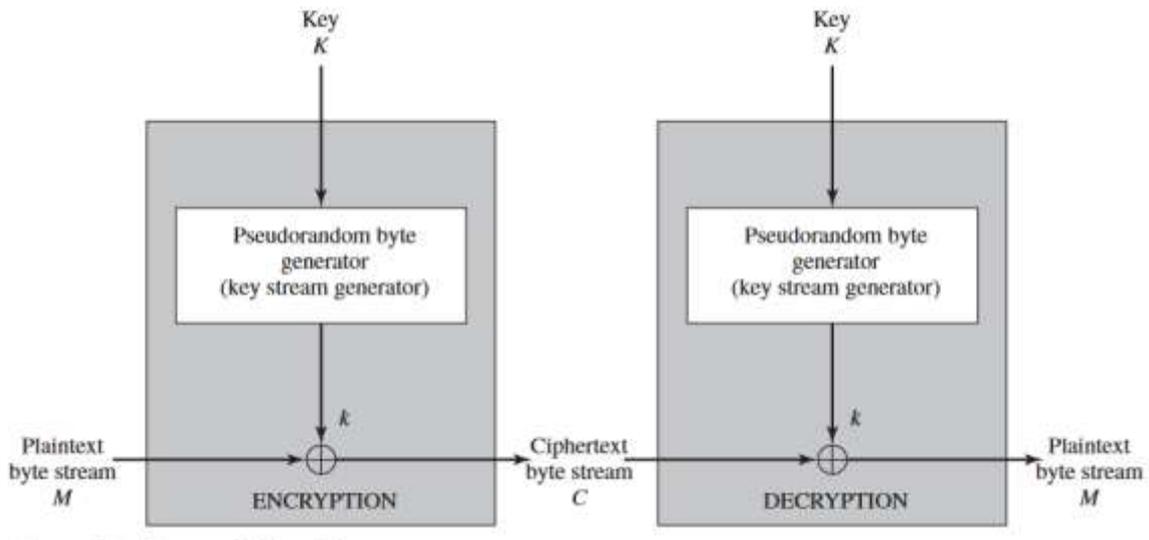
- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.
- Encrypt the contents of the counter with the key and place the result in the bottom register.
- Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode.
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.



## STREAM CIPHERS

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. A key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bit-wise exclusive-OR (XOR) operation. For example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is

$$\begin{array}{rcl}
 & 11001100 & \text{plaintext} \\
 \oplus & \underline{01101100} & \text{key stream} \\
 & 10100000 & \text{ciphertext}
 \end{array}$$



## STREAM CIPHERS

Decryption requires the use of the same pseudorandom sequence

$$\begin{array}{r}
 10100000 \text{ ciphertext} \\
 \oplus 01101100 \text{ key stream} \\
 \hline
 11001100 \text{ plaintext}
 \end{array}$$

### RC4

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. Analysis shows that the period of the cipher is overwhelmingly likely to be greater than  $10^{100}$ . Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software. RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers.

The RC4 algorithm is remarkably simple and quite easy to explain. A variable length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector  $S$ , with elements  $S[0], S[1], S[2], \dots, S[255]$ .

#### Initialization of $S$

To begin, the entries are set equal to the values from 0 through 255 in ascending order; that is,  $S[0], S[1], S[2], \dots, S[255] = 255$ .

A temporary vector,  $T$ , is also created. If the length of the key  $K$  is 256 bytes, then  $T$  is transferred to  $T$ . Otherwise, for a key of length  $keylen$  bytes, the first  $keylen$  elements of  $T$  are copied from  $K$ , and then  $K$  is repeated as many times as necessary to fill out  $T$ . These preliminary operations can be summarized as

```

/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];

```

Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]:

```

/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);

```

### Stream Generation

Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S. After S[255] is reached, the process continues, starting over again at S[0].

```

/* Stream Generation */
i, j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];

```

### LOCATION AND PLACEMENT OF ENCRYPTION FUNCTION

If encryption is to be used to counter attacks on confidentiality, we need to decide what to encrypt and where the encryption function should be located. To begin, this section examines the potential locations of security attacks and then looks at the two major approaches to encryption placement: link and end to end.

#### Potential Locations for Confidentiality Attacks

As an example, consider a user workstation in a typical business organization. Figure 7.1 suggests the types of communications facilities that might be employed by such a workstation and therefore gives an indication of the points of vulnerability.

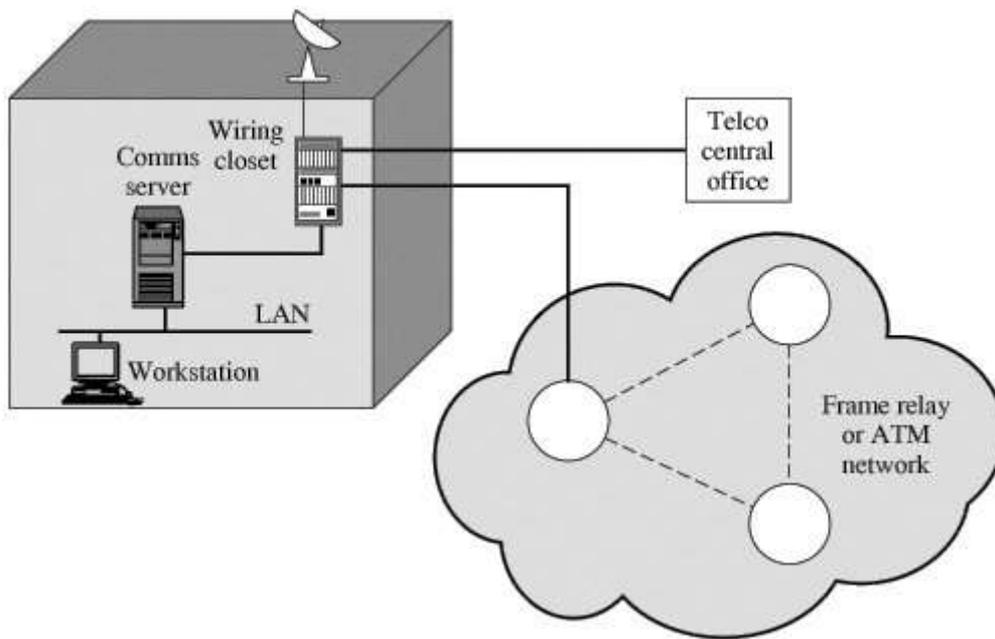
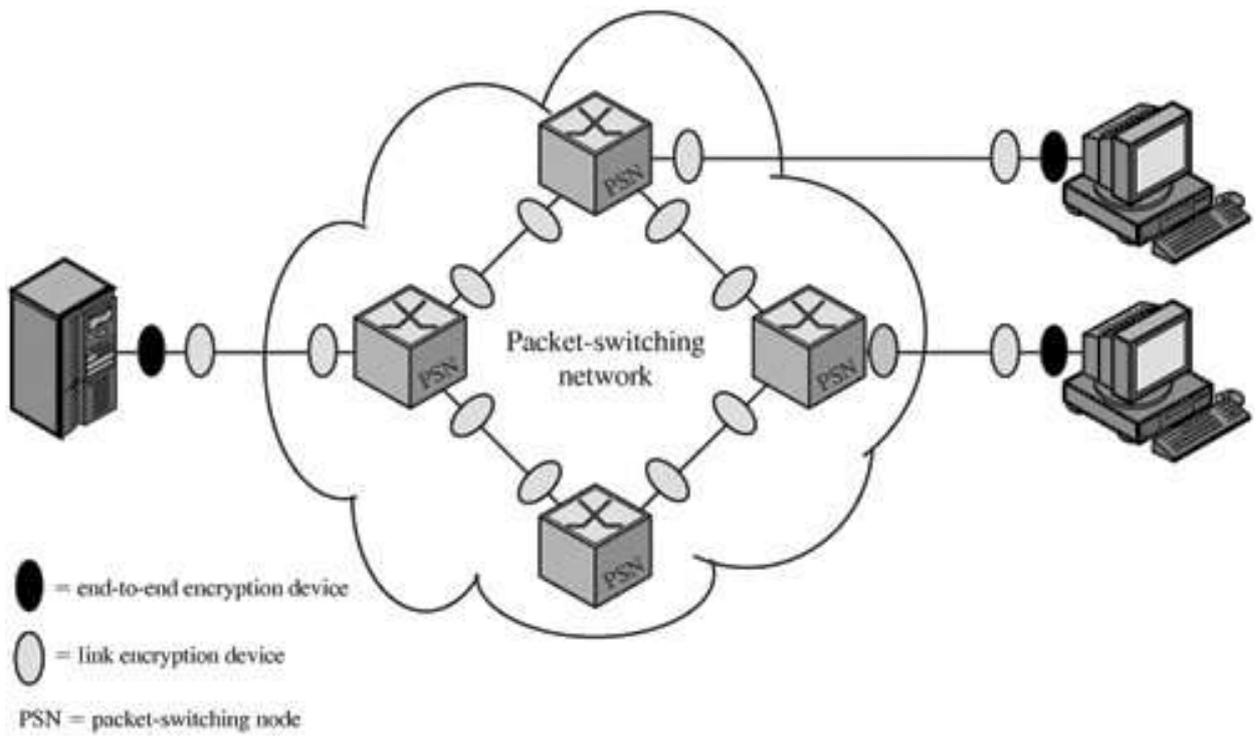


Figure 7.1. Points of Vulnerability

In most organizations, workstations are attached to local area networks (LANs). Typically, the user can reach other workstations, hosts, and servers directly on the LAN or on other LANs in the same building that are interconnected with bridges and routers. Here, then, is the first point of vulnerability. In this case, the main concern is eavesdropping by another employee. Typically, a LAN is a broadcast network: Transmission from any station to any other station is visible on the LAN medium to all stations. Data are transmitted in the form of frames, with each frame containing the source and destination address. An eavesdropper can monitor the traffic on the LAN and capture any traffic desired on the basis of source and destination addresses. If part or all of the LAN is wireless, then the potential for eavesdropping is greater.

### **Link versus End-to-End Encryption**

The most powerful and most common approach to securing the points of vulnerability highlighted in the preceding section is encryption. If encryption is to be used to counter these attacks, then we need to decide what to encrypt and where the encryption gear should be located. As Figure indicates, there are two fundamental alternatives: link encryption and end-to-end encryption.



Encryption Across a Packet-Switching Network

Table 7.1. Characteristics of Link and End-to-End Encryption [PFLE02]

Link Encryption	End-to-End Encryption
Security within End Systems and Intermediate Systems	
Message exposed in sending host	Message encrypted in sending host
Message exposed in intermediate nodes	Message encrypted in intermediate nodes
Role of User	
Applied by sending host	Applied by sending process
Transparent to user	User applies encryption
Host maintains encryption facility	User must determine algorithm
One facility for all users	Users selects encryption scheme
Can be done in hardware	Software implementation
All or no messages encrypted	User chooses to encrypt, or not, for each message
Implementation Concerns	
Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair	Requires one key per user pair
Provides host authentication	Provides user authentication

## PRINCIPLES OF PUBLIC KEY CRYPTOSYSTEMS

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of key distribution.

The second problem that Diffie pondered, and one that was apparently unrelated to the first, was that of digital signatures.

**Public key Cryptosystem** – Asymmetric algorithms depends on one key for encryption and a distinct but related key for decryption. These algorithms have the following characteristics which are as follows –

- It is computationally infeasible to decide the decryption key given only information of the cryptographic algorithm and the encryption key.

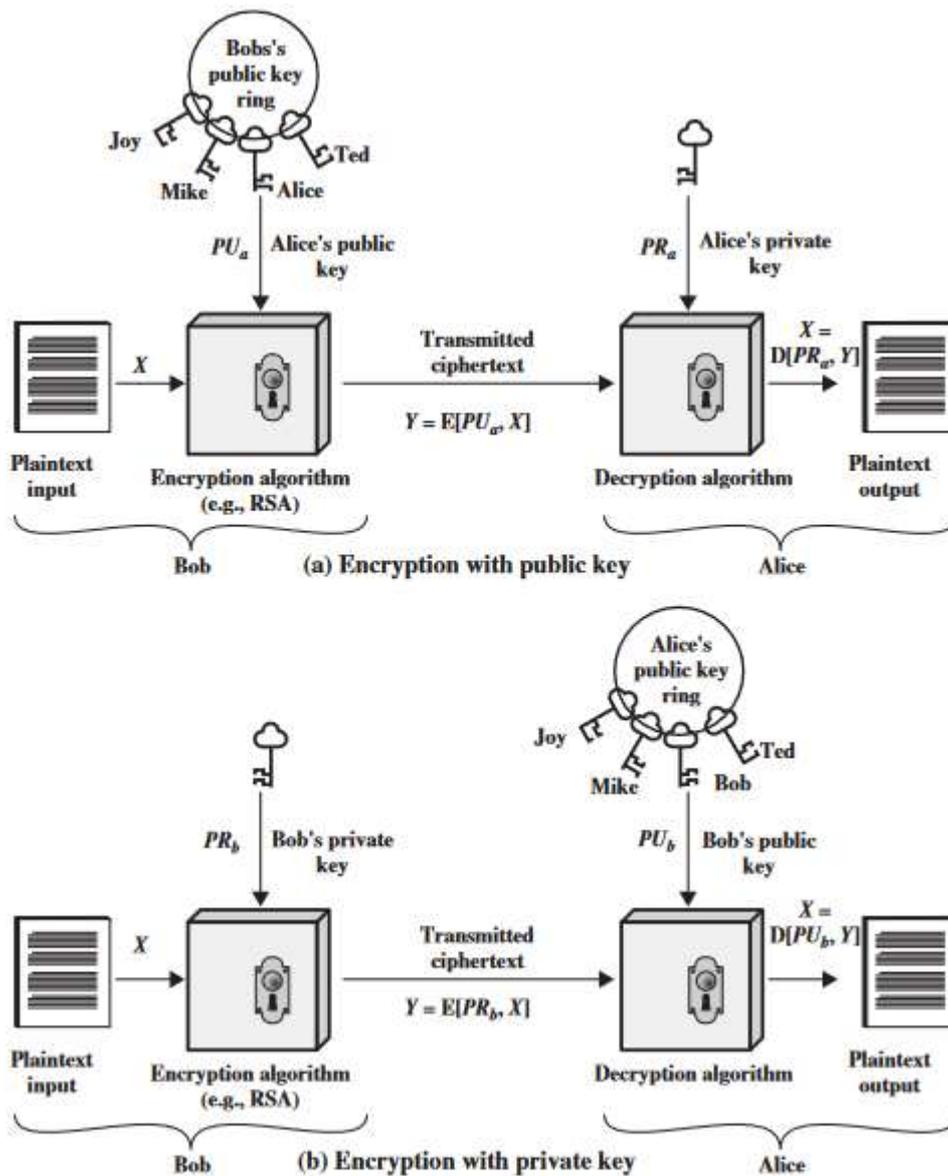
- There are two related keys such as one can be used for encryption, with the other used for decryption.

A public key encryption scheme has the following ingredients which are as follows

- **Plaintext** – This is the readable message or information that is input into the algorithm as input.
- **Encryption algorithm** – The encryption algorithm performs several conversions on the plaintext.
- **Public and Private keys** – This is a set of keys that have been selected so that if one can be used for encryption, and the other can be used for decryption.
- **Ciphertext** – This is a scrambled message generated as output. It is based on the plaintext and the key. For a given message, there are two specific keys that will create two different ciphertexts.
- **Decryption Algorithm** – This algorithm gets the ciphertext and the matching key and creates the original plaintext.

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As the figure suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.



## Public Key Cryptography Requirements

To accomplish the public key cryptography there are following requirements as discussed below.

- The computation of the pair of keys i.e. private key and the public key must be easy.
- Knowing the encryption algorithm and public key of the intended receiver, computation of cipher text must be easy.
- For a receiver of the message, it should be computationally easy to decrypt the obtained cipher text using his private key.
- It is also required that any opponent in the network knowing the public key should be unable to determine its corresponding private key.
- Having the cipher text and public key an opponent should be unable to determine the original message.
- The two keys i.e. public and private key can be implemented in both orders  $D[PU, E(PR, M)] = D[PR, E(PU, M)]$

## RSA ALGORITHM

In this algorithm two keys were used. One is private key and another one is public key.

## RSA Algorithm

- Ron Rivest, Adi Shamir and Len Adleman have developed this algorithm (Rivest-Shamir-Adleman). It is a block cipher which converts plain text into cipher text and vice versa at receiver side.
- **The algorithm works as follow:**
  1. Select two prime numbers  $p$  and  $q$  where  $p \neq q$ .
  2. Calculate  $n = p * q$ .
  3. Calculate  $\Phi(n) = (p-1) * (q-1)$ .
  4. Select  $e$  such that,  $e$  is relatively prime to  $\Phi(n)$   
i.e.  $(e, \Phi(n)) = 1$  and  $1 < e < \Phi(n)$
  5. Calculate  $d = e^{-1} \text{ mod } \Phi(n)$  or  $ed = 1 \text{ mod } \Phi(n)$ .
  6. Public key =  $\{e, n\}$ , private key =  $\{d, n\}$ .
  7. Find out cipher text using the formula,  
 $C = P^e \text{ mod } n$  where,  $P < n$  and  
 $C = \text{Cipher text}$ ,  $P = \text{Plain text}$ ,  $e = \text{Encryption key}$  and  $n = \text{block size}$ .
  8.  $P = C^d \text{ mod } n$ . Plain text  $P$  can be obtain using the given formula.  
where,  $d = \text{decryption key}$ .



## RSA Algorithm step by step explanation

- **Step – 1:** Select two prime numbers  $p$  and  $q$  where  $p \neq q$ .
- **Step – 2:** Calculate  $n = p * q$ .
- **Step – 3:** Calculate  $\Phi(n) = (p-1) * (q-1)$ .
- **Step – 4:** Select  $e$  such that,  $e$  is relatively prime to  $\Phi(n)$   
i.e.  $(e, \Phi(n)) = 1$  and  $1 < e < \Phi(n)$

### ❖ Explanation with example:

1. Two prime numbers  $p = 13$ ,  $q = 11$ .
2.  $n = p * q = 13 * 11 = 143$ .
3.  $\Phi(n) = (13 - 1) * (12 - 1) = 12 * 10 = 120$ .
4. Select  $e = 13$ ,  $\text{gcd}(13, 120) = 1$ .

➤ **Step – 5:** Calculate  $d = e^{-1} \bmod \Phi(n)$  or  $ed = 1 \bmod \Phi(n)$ .

❖ Explanation with example:

5. Finding d:

$$\rightarrow e * d \bmod \Phi(n) = 1$$

$$\rightarrow 13 * d \bmod 120 = 1$$

(How to find:  $d * e = 1 \bmod \Phi(n) \rightarrow d = ((\Phi(n) * i) + 1) / e$

$$d = (120 + 1) / 13 = 9.30 (\because i = 1)$$

$$d = (240 + 1) / 13 = 18.53 (\because i = 2)$$

$$d = (360 + 1) / 13 = 27.76 (\because i = 3)$$

$$d = (480 + 1) / 13 = 37 (\because i = 4)$$

➤ **Step – 6:** Public key = {e, n}, private key = {d, n}.

➤ **Step – 7:** Find out cipher text using the formula,

$$C = P^e \bmod n \text{ where, } P < n$$

*C = Cipher text, P = Plain text, e = Encryption key and n=block size.*

➤ **Step – 8:**  $P = C^d \bmod n$ . Plain text P can be obtain using the given formula.

*where, d = decryption key.*

❖ Explanation with example:

6. Public key = {13, 143} and private key = {37, 143}.

7. **Encryption :** Plain text  $P = 13$ . (where,  $P < n$ )

$$C = P^e \bmod n = 13^{13} \bmod 143 = 52. \quad \boxed{C = 52}$$

8. **Decryption:**

$$P = C^d \bmod n = 52^{37} \bmod 143 = 13. \quad \boxed{P = 13}$$

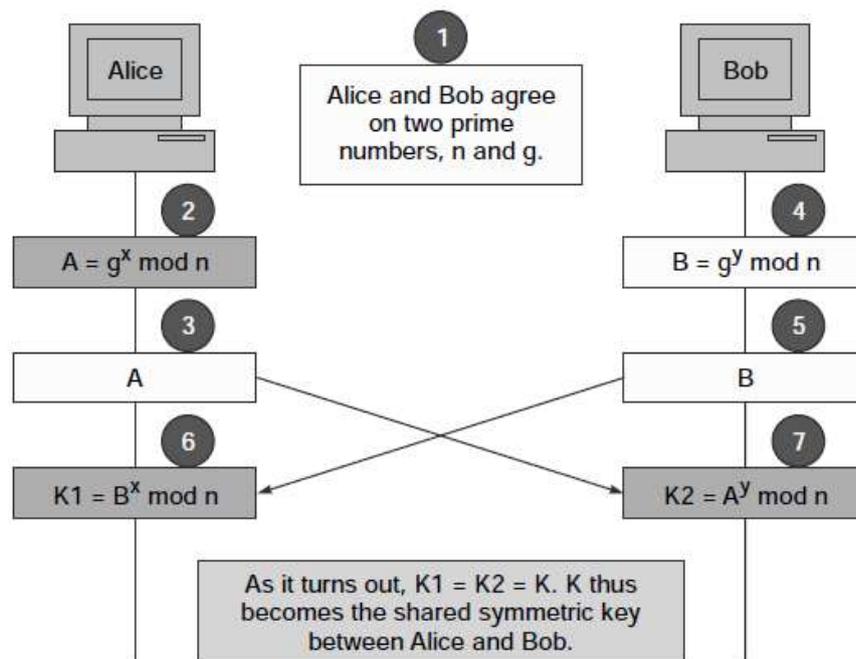
## Diffie–Hellman Key Exchange/Agreement Algorithm

In this scheme the two parties, who want to communicate securely, can agree on a **symmetric key** using this technique. This key can then be used for encryption/ decryption. However, we must note that Diffie–Hellman key exchange algorithm can be used only for key agreement, but not for encryption or decryption of messages. Once both the parties agree on the key to be used, they need to use other symmetric key encryption algorithms for actual encryption or decryption of messages.

## Description of the Algorithm

1. Firstly, Alice and Bob agree on two large prime numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number  $x$ , and calculates  $A$  such that:  
 $A = g^x \text{ mod } n$
3. Alice sends the number  $A$  to Bob.
4. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that:  
 $B = g^y \text{ mod } n$
5. Bob sends the number  $B$  to Alice.
6. A now computes the secret key  $K1$  as follows:  
 $K1 = B^x \text{ mod } n$
7. B now computes the secret key  $K2$  as follows:  
 $K2 = A^y \text{ mod } n$

*Diffie–Hellman key exchange algorithm*



Example of the Algorithm

1. Firstly, Alice and Bob agree on two large prime numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.  

Let  $n = 11, g = 7$ .
2. Alice chooses another large random number  $x$ , and calculates  $A$  such that:  
 $A = g^x \text{ mod } n$   

Let  $x = 3$ . Then, we have,  $A = 7^3 \text{ mod } 11 = 343 \text{ mod } 11 = 2$ .
3. Alice sends the number  $A$  to Bob.  

Alice sends 2 to Bob.
4. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that:  
 $B = g^y \text{ mod } n$   

Let  $y = 6$ . Then, we have,  $B = 7^6 \text{ mod } 11 = 117649 \text{ mod } 11 = 4$ .
5. Bob sends the number  $B$  to Alice.  

Bob sends 4 to Alice.
6. A now computes the secret key  $K1$  as follows:  
 $K1 = B^x \text{ mod } n$   

We have,  $K1 = 4^3 \text{ mod } 11 = 64 \text{ mod } 11 = 9$ .
7. B now computes the secret key  $K2$  as follows:  
 $K2 = A^y \text{ mod } n$   

We have,  $K2 = 2^6 \text{ mod } 11 = 64 \text{ mod } 11 = 9$ .

**ELAGAMAL CRYPTOGRAPHY (ECC)**

In this ECC we have three phases

1. Key generation
2. Encryption
3. Decryption

1. Generate a random integer  $X_A$ , such that  $1 < X_A < q - 1$ .
2. Compute  $Y^A = \alpha^{X_A} \bmod q$ .
3. A's private key is  $X_A$ ; A's public key is  $\{q, \alpha, Y_A\}$ .

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer  $M$  in the range  $0 \leq M \leq q - 1$ . Longer messages are sent as a sequence of blocks, with each block being an integer less than  $q$ .
2. Choose a random integer  $k$  such that  $1 \leq k \leq q - 1$ .
3. Compute a one-time key  $K = (Y_A)^k \bmod q$ .
4. Encrypt  $M$  as the pair of integers  $(C_1, C_2)$  where

$$C_1 = \alpha^k \bmod q; C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing  $K = (C_1)^{X_A} \bmod q$ .
2. Compute  $M = (C_2 K^{-1}) \bmod q$ .

ElGamal process as follows,

Global Public Elements	
$q$	prime number
$\alpha$	$\alpha < q$ and $\alpha$ a primitive root of $q$

Key Generation by Alice	
Select private $X_A$	$X_A < q - 1$
Calculate $Y_A$	$Y_A = \alpha^{X_A} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	$X_A$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer $k$	$k < q$
Calculate $K$	$K = (Y_A)^k \bmod q$
Calculate $C_1$	$C_1 = \alpha^k \bmod q$
Calculate $C_2$	$C_2 = KM \bmod q$
Ciphertext:	$(C_1, C_2)$

### Decryption by Alice with Alice's Private Key

Ciphertext:	$(C_1, C_2)$
Calculate $K$	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Thus,  $K$  functions as a one-time key, used to encrypt and decrypt the message.

For example, let us start with the prime field GF(19); that is,  $q = 19$ . It has primitive roots  $\{2, 3, 10, 13, 14, 15\}$ , as shown in Table 8.3. We choose  $\alpha = 10$ .

Alice generates a key pair as follows:

1. Alice chooses  $X_A = 5$ .
2. Then  $Y_A = \alpha^{X_A} \bmod q = 10^5 \bmod 19 = 3$  (see Table 8.3).
3. Alice's private key is 5; Alice's public key is  $\{q, \alpha, Y_A\} = \{19, 10, 3\}$ .

Suppose Bob wants to send the message with the value  $M = 17$ . Then,

1. Bob chooses  $k = 6$ .
2. Then  $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$ .
3. So

$$C_1 = \alpha^k \bmod q = 10^6 \bmod 19 = 11$$

$$C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$$

4. Bob sends the ciphertext (11, 5).

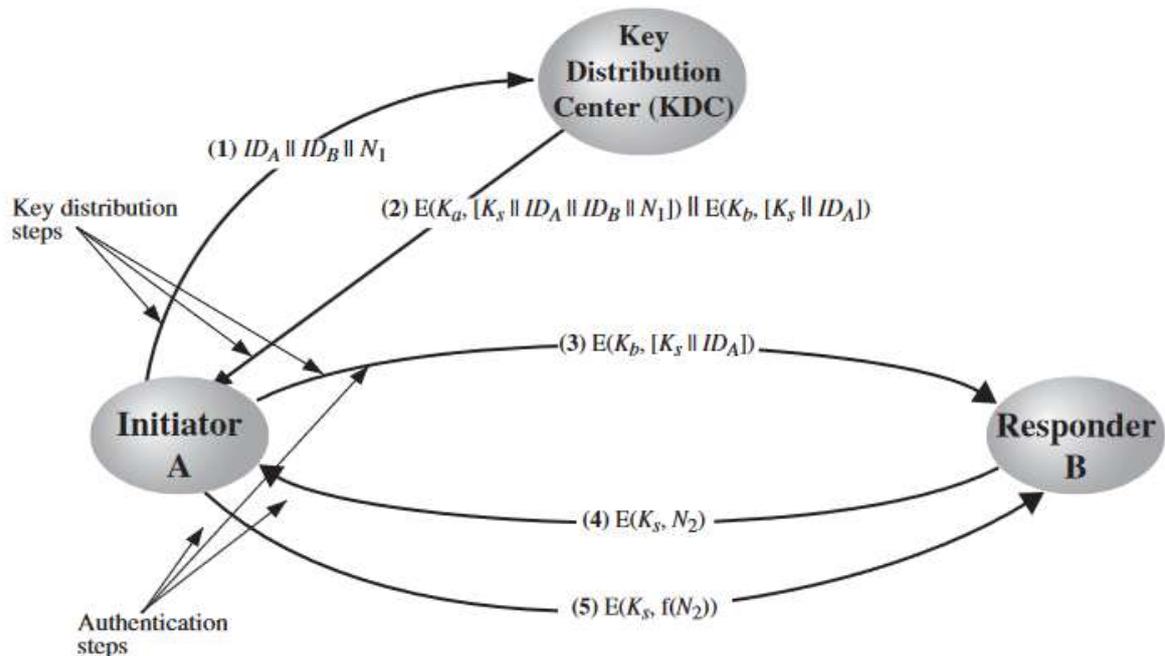
For decryption:

1. Alice calculates  $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$ .
2. Then  $K^{-1}$  in GF(19) is  $7^{-1} \bmod 19 = 11$ .
3. Finally,  $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$ .

### KEY DISTRIBUTION

- Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data. Some sort of mechanism or protocol is needed to provide for the secure distribution of keys.
- Key distribution often involves the use of master keys, which are infrequently used and are long lasting, and session keys, which are generated and distributed for temporary use between two parties.
- Public-key encryption schemes are secure only if the authenticity of the public key is assured. A public-key certificate scheme provides the necessary security.
- X.509 defines the format for public-key certificates. This format is widely used in a variety of applications.
- A public-key infrastructure (PKI) is defined as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
- Typically, PKI implementations make use of X.509 certificates

### A Key Distribution Scenario



User A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key,  $K_a$ , known only to itself and the KDC; similarly, B shares the master key  $K_b$  with the KDC.

### Hierarchical Key Control

It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so. As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.

A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities. Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

### Session Key Lifetime

The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

### Decentralized Key Control

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution. Thus, there may need to be as many as  $n(n-1)/2$  master keys for a configuration with  $n$  end systems.

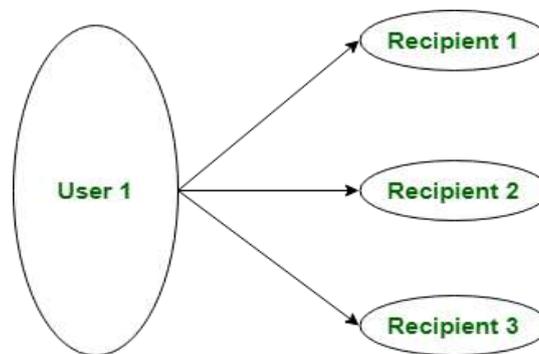
### Distribution of Public Key:

The public key can be distributed in four ways:

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates.

These are explained as following below:

**1. Public Announcement:** Here the public key is broadcasted to everyone. The major weakness of this method is a forgery. Anyone can create a key claiming to be someone else and broadcast it. Until forgery is discovered can masquerade as claimed user.



Public Key Announcement

**2. Publicly Available Directory:** In this type, the public key is stored in a public directory. Directories are trusted here, with properties like Participant Registration, access and allow to modify values at any time, contains entries like {name, public-key}. Directories can be accessed electronically still vulnerable to forgery or tampering.

**3. Public Key Authority:** It is similar to the directory but, improves security by tightening control over the distribution of keys from the directory. It requires users to know the public key for the directory. Whenever the keys are needed, real-time access to the directory is made by the user to obtain any desired public key securely.

**4. Public Certification:** This time authority provides a certificate (which binds an identity to the public key) to allow key exchange without real-time access to the public authority each time. The certificate is accompanied by some other info such as period of validity, rights of use, etc. All of this content is signed by the private key of the certificate authority and it can be verified by anyone possessing the authority's public key.

