Cryptography and Network Security

- Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings.
- Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit.
- Cryptography can reformat and transform our data, making it safer on its trip between computers.
- The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

- **Computer Security** – generic name for the collection of tools designed to protect data and to thwart hackers
- **Network Security** – measures to protect data during their transmission
- **Internet Security** – measures to protect data during their transmission over a collection of interconnected networks

- **Security Attacks, Services and Mechanisms**
- To assess the security needs of an organization effectively, the manager responsible for security needs some systematic way of defining the requirements for security and characterization of approaches to satisfy those requirements. One approach is to consider three aspects of information security:
- **Security attack** – Any action that compromises the security of information owned by an organization.
- **Security mechanism** – A mechanism that is designed to detect, prevent or recover from a security attack.
- **Security service** – A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and

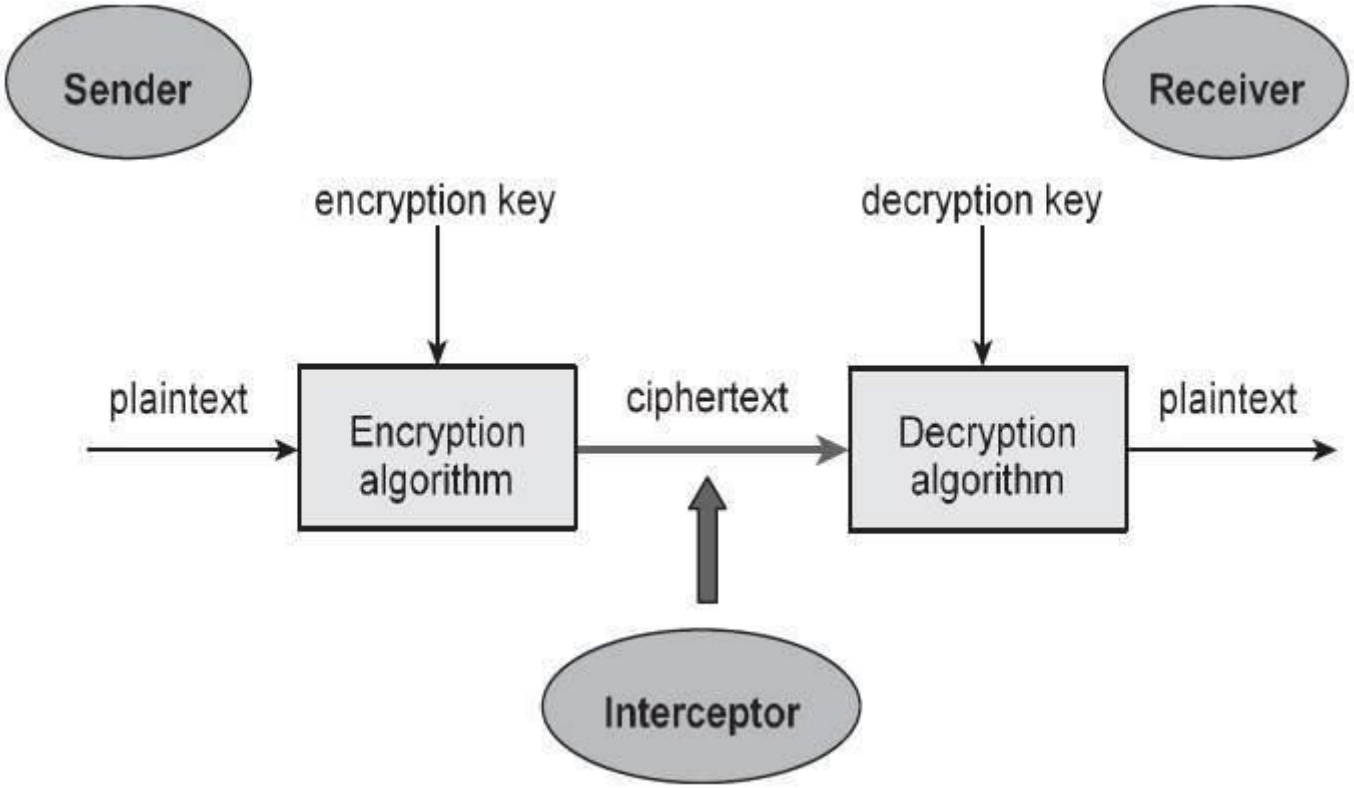
  they make use of one or more security mechanisms to provide the service.

- **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** The original intelligible message
- **Cipher text** The transformed message
- **Cipher** An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods
- **Key** Some critical information used by the cipher, known only to the sender& receiver

- **Encipher** (encode) The process of converting plaintext to cipher text using a cipher and a key
- **Decipher** (decode) the process of converting cipher text back into plaintext using a cipher and a key
- **Cryptanalysis** The study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called **code breaking**
- **Cryptology** Both cryptography and cryptanalysis
- **Code** An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Cryptography

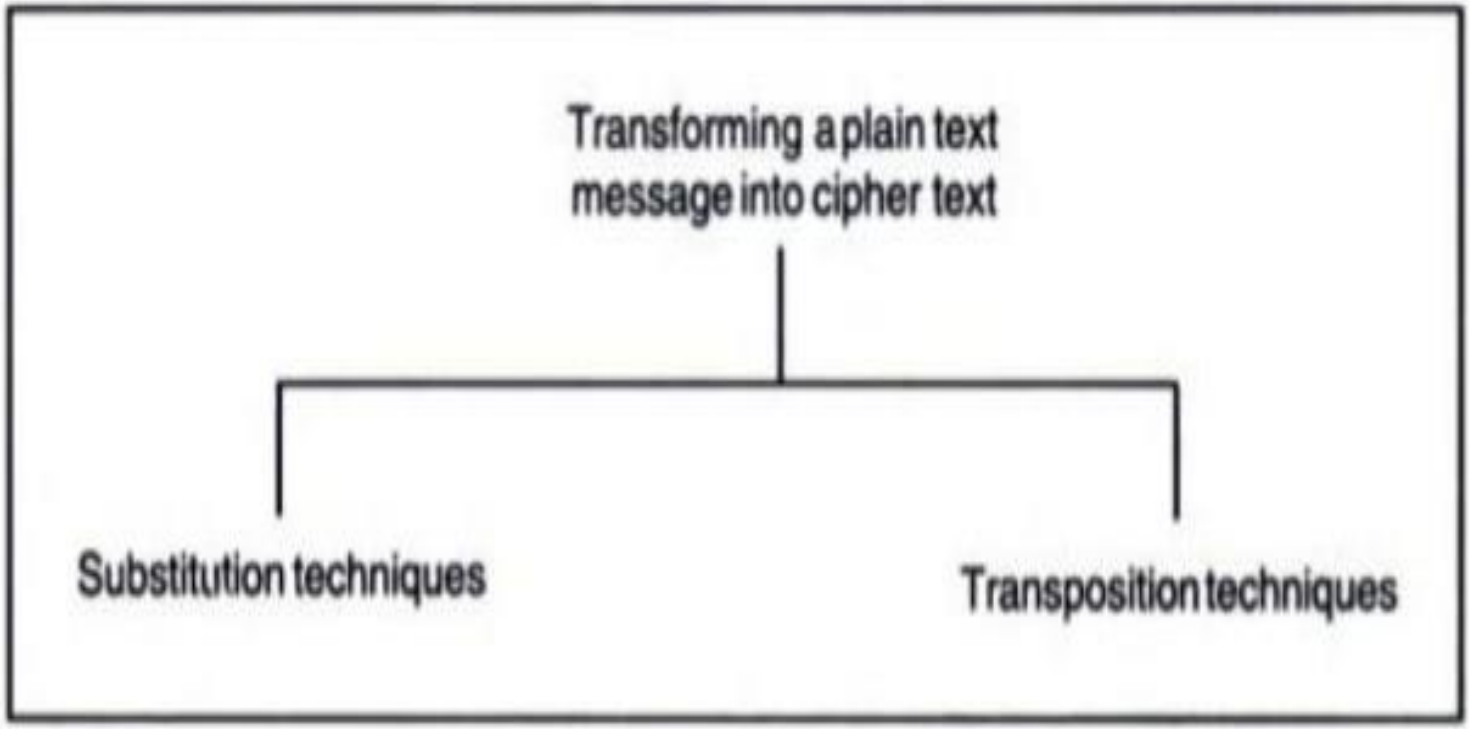
- Cryptographic systems are generally classified along 3 independent dimensions:
- **Type of operations used for transforming plain text to cipher text**
- All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.
- **The number of keys used**
- If the sender and receiver uses same key then it is said to be **symmetric key (or) single key (or) conventional encryption.**
- If the sender and receiver use different keys then it is said to be **public key encryption.**
- **The way in which the plain text is processed**
- A **block cipher** processes the input and block of elements at a time, producing output block for each input block.


Sender
Receiver
encryption key
decryption key
plaintext
Encryption algorithm
ciphertext
Decryption algorithm
plaintext
Interceptor

Components of a Cryptosystem

- The various components of a basic cryptosystem are as follows −
- **Plaintext.**
- **Encryption Algorithm.** It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
- **Ciphertext.**
- **Decryption Algorithm,** It is a mathematical process, that produces a unique plaintext for any given ciphertext and decryption key.

- It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext.
- **Encryption Key.** It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
- **Decryption Key.** It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.


Transforming a plain text
message into cipher text
Substitution techniques
Transposition techniques

Substitution technique

- Caesar Cipher
- It is a mono-alphabetic cipher wherein each letter of the plaintext is substituted by another letter to form the ciphertext. It is a simplest form of substitution cipher scheme.
- This cryptosystem is generally referred to as the **Shift Cipher**. The concept is to replace each alphabet by another alphabet which is 'shifted' by some fixed number between 0 and 25.
  Ex. Plain text: Hello
- Here is the ciphertext alphabet for a Shift of 3
- Ciphertext: khoor

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

- Process of Shift Cipher
- In order to encrypt a plaintext letter, the sender positions the sliding ruler underneath the first set of plaintext letters and slides it to LEFT by the number of positions of the secret shift.
- On receiving the ciphertext, the receiver who also knows the secret shift, positions his sliding ruler underneath the ciphertext alphabet and slides it to RIGHT by the agreed shift number, 3 in this case.

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

- # Monoalphabetic and Polyalphabetic Cipher
- Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. For example, if ‘A’ is encrypted as ‘D’, for any number of occurrence in that plaintext, ‘A’ will always get encrypted to ‘D’.
- All of the substitution ciphers we have discussed earlier in this chapter are monoalphabetic; these ciphers are highly susceptible to cryptanalysis.
- Polyalphabetic Cipher is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process. The next two examples, **playfair and Vigenere Cipher are polyalphabetic ciphers.**

Playfair Cipher

- In this scheme, pairs of letters are encrypted, instead of single letters as in the case of simple substitution cipher.
- In playfair cipher, initially a key table is created. The key table is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table as we need only 25 alphabets instead of 26. If the plaintext contains J, then it is replaced by I.
- The sender and the receiver deicide on a particular key, say ‘tutorials’. In a key table, the first characters (going left to right) in the table is the phrase, excluding the duplicate letters. −

- The rest of the table will be filled with the remaining letters of the alphabet, in natural order. The key table works out to be

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

- Process of Playfair Cipher
- First, a plaintext message is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Let us say we want to encrypt the message “hide money”. It will be written as −
- HI DE MO NE YZ
- The rules of encryption are −
  - If both the letters are in the same column, take the letter below each one (going back to the top if at the bottom)

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

'H' and 'I' are in same column, hence take letter below them to replace. HI → QC

- If both letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

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

'D' and 'E' are in same row, hence take letter to the right of them to replace. DE → EF

- If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

T	U	O	R	I
A	L	S	B	C
D	E	F	G	H

'M' and 'O' nor on same column or same row, hence form rectangle as shown, and replace letter by picking up opposite corner letter on same row MO -> NU

- Using these rules, the result of the encryption of ‘hide money’ with the key of ‘tutorials’ would be −
- QC EF NU MF ZV
- Decrypting the Playfair cipher is as simple as doing the same process in reverse. Receiver has the same key and can create the same key table, and then decrypt any messages made using that key.

Hill cipher

- Each letter is represented by a number modulo 26. Though this is not an essential feature of the cipher, this simple scheme is often used:

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Number	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	25

- 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). The cipher can, of course, be adapted to an alphabet with any number of letters; all arithmetic just needs to be done modulo the number of letters instead of modulo 26.
- Consider the message 'ACT', and the key below (or GYB/NQK/URP in letters):

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

- Since 'A' is 0, 'C' is 2 and 'T' is 19, the message is the vector:

$$\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$$

- Thus the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26}$$

- which corresponds to a ciphertext of 'POH'. Now, suppose that our message is instead 'CAT', or:

$$\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$$

- This time, the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \equiv \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$$

- which corresponds to a ciphertext of 'FIN'.

Transposition Cipher

It is another type of cipher where the order of the alphabets in the plaintext is rearranged to create the ciphertext. The actual plaintext alphabets are not replaced.

1. Rail Fence cipher

- The Rail Fence cipher works by writing your message on alternate lines across the page, and then reading off each line in turn.
- For example, the plaintext "defend the east wall" is written as shown below, with all spaces removed.

D		F		N		T		E		A		T		A		L
	E		E		D		H		E		S		W		L	

- The ciphertext is then read off by writing the top row first, followed by the bottom row, to get "DFNTEATALEEDHESWL".
- **Encryption**
  To encrypt a message using the Rail Fence Cipher, write your message in zigzag lines across the page, and then read off each row. Firstly, you need to have a key, which for this cipher is the number of rows you are going to have.
- Then start writing the letters of the plaintext diagonally down to the right until you reach the number of rows specified by the key.
- Then bounce back up diagonally until you hit the first row again. This continues until the end of the plaintext.
- For the plaintext we used above, "defend the east wall", with a key of 3, we get the encryption process shown below.

D				N				E				T				L		
	E		E		D		H		E		S		W		L		X	
		F				T				A				A				X

Decryption

The decryption process for the Rail Fence Cipher involves reconstructing the diagonal gr
used to encrypt the message. We start writing the message, but leaving a dash in place
spaces yet to be occupied. Gradually, you can replace all the dashes with the correspon
letters, and read off the plaintext from the table.

We start by making a grid with as many rows as the key is, and as many columns as the
length of the ciphertext. We then place the first letter in the top left square, and dashes
diagonally downwards where the letters will be. When we get back to the top row, we pla
the next letter in the ciphertext. Continue like this across the row, and start the next row w
you reach the end.

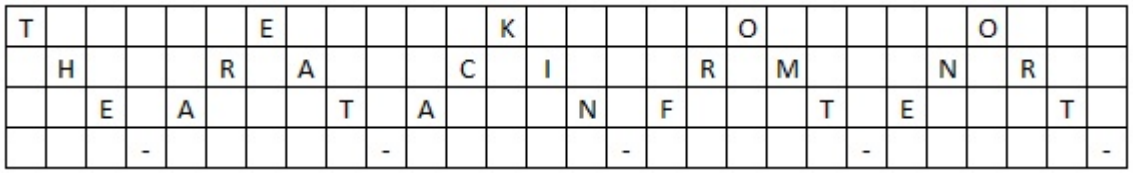
For example, if you receive the ciphertext "TEKOOHRACIRMNREATANFTETYTGHH",
encrypted with a key of 4, you start by placing the "T" in the first square. You then dash t
diagonal down spaces until you get back to the top row, and place the "E" here. Continui
fill the top row you get the pattern below.

T						E						K						O						O			
	-				-		-				-		-				-		-				-			-		
		-		-				-		-				-		-				-		-					-	
			-						-						-								-						-

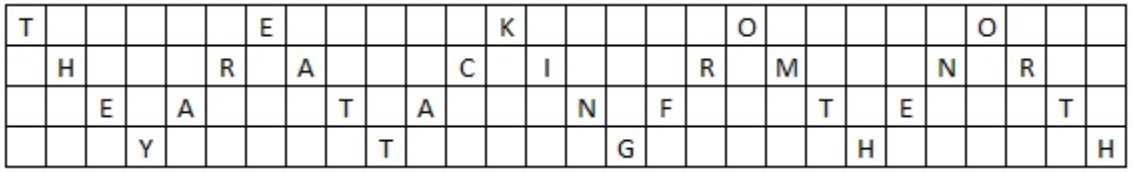
The first row of the decryption process for the Rail Fence Cipher. We have a table with 4 rows because the key is 4, and 28 columns as the ciphertext has length 28.
Continuing this row-by-row, we get the successive stages shown below.

T						E						K						O						O			
	H				R		A				C		I				R		M				N		R			
		-		-				-		-				-		-				-		-				-		
			-						-						-						-						-	

The second stage in the decryption process.

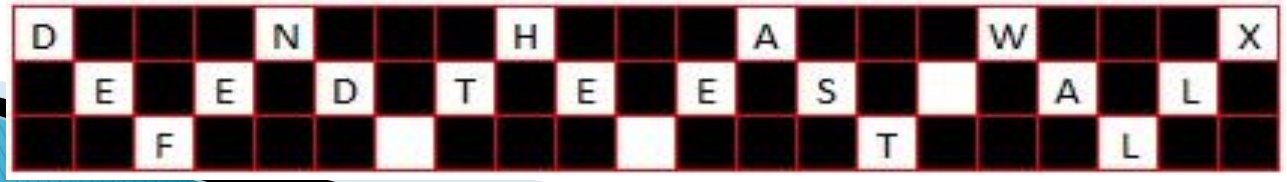


The third stage in the decryption process.



The fourth and final stage in the decryption process.

From this we can now read the plaintext off following the diagonals to get "they are attacking from the north".



Columnar Transition Technique

- **A. Basic Technique**
- It is a slight variation to the Rail-fence technique, let's see its algorithm:
- In a rectangle of pre-defined size, write the plain-text message row by row.
- Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
- Thus Cipher-text is obtained.
- **example:**
- Original message: **"INCLUDEHELP IS AWESOME".**
- Now we apply the above algorithm and create the rectangle of 4 columns (we decide to make a rectangle with four column it can be any number.)

Column 1	Column 2	Column 3	Column 4
I	N	C	L
U	D	E	H
E	L	P	I
S	A	W	E
S	O	M	E

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.
Cipher-text: **LHIEEIUESSCEPWMNDLAO**

2) Columnar Transition Technique

A. Basic Technique

It is a slight variation to the Rail-fence technique, let's see its algorithm:

1. In a rectangle of pre-defined size, write the plain-text message row by row.
2. Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
3. Thus Cipher-text is obtained.

Let's see an example:

Original message: **"INCLUDEHELP IS AWESOME"**.

Now we apply the above algorithm and create the rectangle of 4 columns (we decide to make a rectangle with four column it can be any number.)

Column 1	Column 2	Column 3	Column 4
I	N	C	L
U	D	E	H
E	L	P	I
S	A	W	E
S	O	M	E

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.

Cipher-text: **LHIEEIUESSCEPWMNDLAO**

B. Columnar Technique with multiple rounds

In this method, we again change the chipper text we received from a Basic technique that is in round 1 and again follows the same procedure for the cipher-text from round 1.

Algorithm:

1. In a rectangle of pre-defined size, write the plain-text message row by row.
2. Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
3. Thus, Cipher-text of round 1 is obtained.
4. Repeat from step 1 to 3.

Example:

Original message: **"INCLUDEHELP IS AWESOME"**.

Now we apply the above algorithm and create the rectangle of 4 column (we decide to make a rectangle with four column it can be any number.)

Column 1	Column 2	Column 3	Column 4
I	N	C	L
U	D	E	H
E	L	P	I
S	A	W	E
S	O	M	E

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.

Cipher-text of round 1: **LHIEEIUESSCEPWMNDLAO**

Round 2:

Column 1	Column 2	Column 3	Column 4
L	H	I	E
E	I	U	E
S	S	C	E
P	W	M	N
D	L	A	O

Now, we decide to go with a previous order that is 4,1,3,2.

Cipher-text: **EEENLESPICUMHISW**

These multi-round columnar techniques are harder to crack as compared to methods seen earlier.

3) Vernam Cipher (One-Time Pad)

The **Vernam Cipher** has a specific subset **one-time pad**, which uses input ciphertext as a random set of non-repeating character. The thing to notice here is that, once an input cipher text gets used it will never be used again hence one-time pad and length of cipher-text is the size that of message text.

Algorithm:

1. Plain text character will be represented by the numbers as **A=0, B=1, C=2,... Z=25**.
2. Add each corresponding number of a plain text message to the input cipher text alphabet numbers.
3. If the sum is greater than or equal to 26, subtract 26 from it.
4. Translate each number back to corresponding letters and we got our cipher text.

Example:

Our message is **"INCLUDEHELP"** and input cipher text is **"ATQXRZWOBYV"**

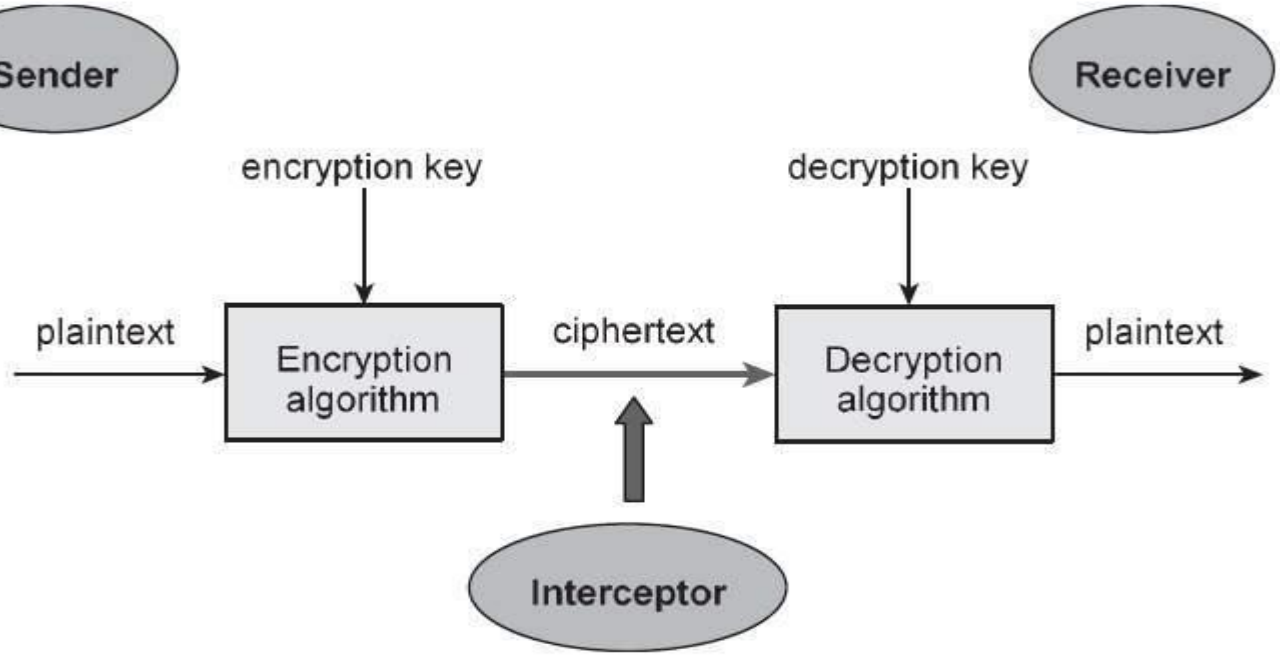
	I	N	C	L	U	D	E	H	E	L	P
Plain text:	8	13	2	11	20	3	4	7	4	11	15
One-time pad:	0	19	16	23	17	25	22	14	1	24	21
	A	T	Q	X	R	Z	W	O	B	Y	V
Initial Total:	8	32	18	34	37	28	26	21	5	35	36
Subtract 26, if >25:	8	6	18	8	11	2	0	21	5	9	10
Cipher Text:	I	G	S	I	L	C	A	V	F	J	K

Example of Vernam Cipher

One time pad should be discarded after every single use and this technique is proved highly secure and suitable for small messages but illogical if used for long messages.

Cryptosystems

- A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a **cipher system**.
- This basic model is depicted in the illustration below



Components of a Cryptosystem

- The various components of a basic cryptosystem are as follows −
- **Plaintext.** It is the data to be protected during transmission.
- **Encryption Algorithm.** It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
- **Ciphertext.** It is the scrambled version of the plaintext produced by the encryption algorithm using a specific the encryption key. The ciphertext is not guarded. It flows on public channel. It can be intercepted or compromised by anyone who has access to the communication channel.
- **Decryption Algorithm,** It is a mathematical process, that produces a unique plaintext for any given ciphertext and decryption key. It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.

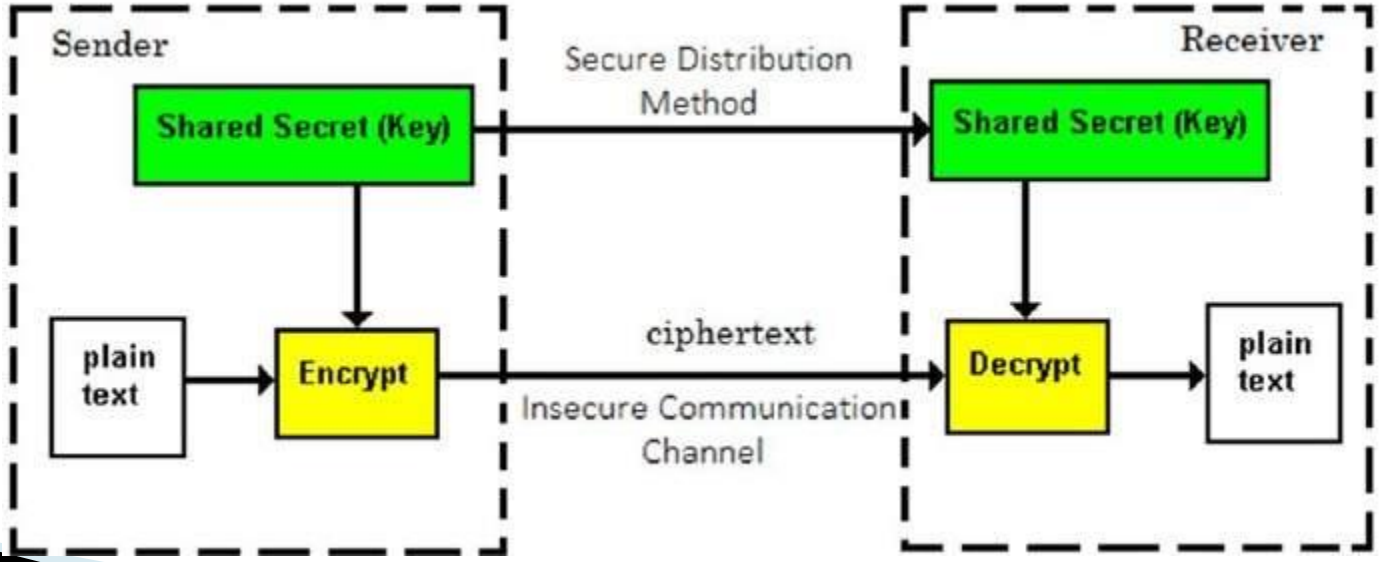
- **Encryption Key.** It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
- **Decryption Key.** It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.

Types of Cryptosystems

- Fundamentally, there are two types of cryptosystems based on the manner in which encryption-decryption is carried out in the system −
- Symmetric Key Encryption
- Asymmetric Key Encryption

Symmetric Key Encryption

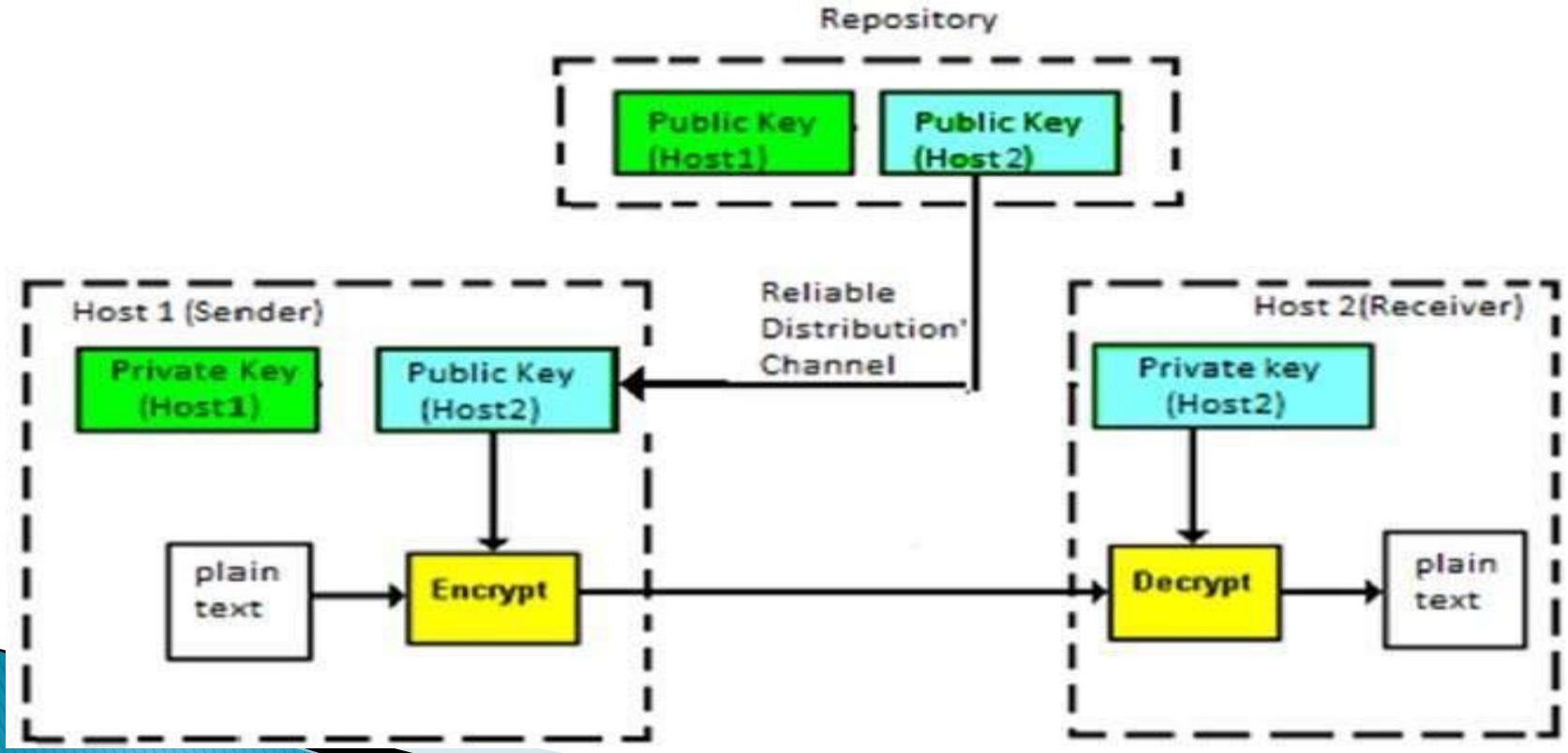
- The encryption process where **same keys are used for encrypting and decrypting** the information is known as Symmetric Key Encryption.
- The study of symmetric cryptosystems is referred to as **symmetric cryptography**. Symmetric cryptosystems are also sometimes referred to as **secret key cryptosystems**.



- **The salient features of cryptosystem based on symmetric key encryption are −**
- Persons using symmetric key encryption must share a common key prior to exchange of information.
- Keys are recommended to be changed regularly to prevent any attack on the system.
- A robust mechanism needs to exist to exchange the key between the communicating parties. As keys are required to be changed regularly, this mechanism becomes expensive.
- In a group of **n** people, to enable two-party communication between any two persons, the number of keys required for group is **n × (n – 1)/2**.
- Length of Key (number of bits) in this encryption is smaller and hence, process of encryption-decryption is faster than asymmetric key encryption.

Asymmetric Key Encryption

- The encryption process where **different keys are used for encrypting and decrypting the information** is known as Asymmetric Key Encryption. Though the keys are different, they are mathematically related and hence, retrieving the plaintext by decrypting ciphertext is feasible. The process is depicted in the following illustration −

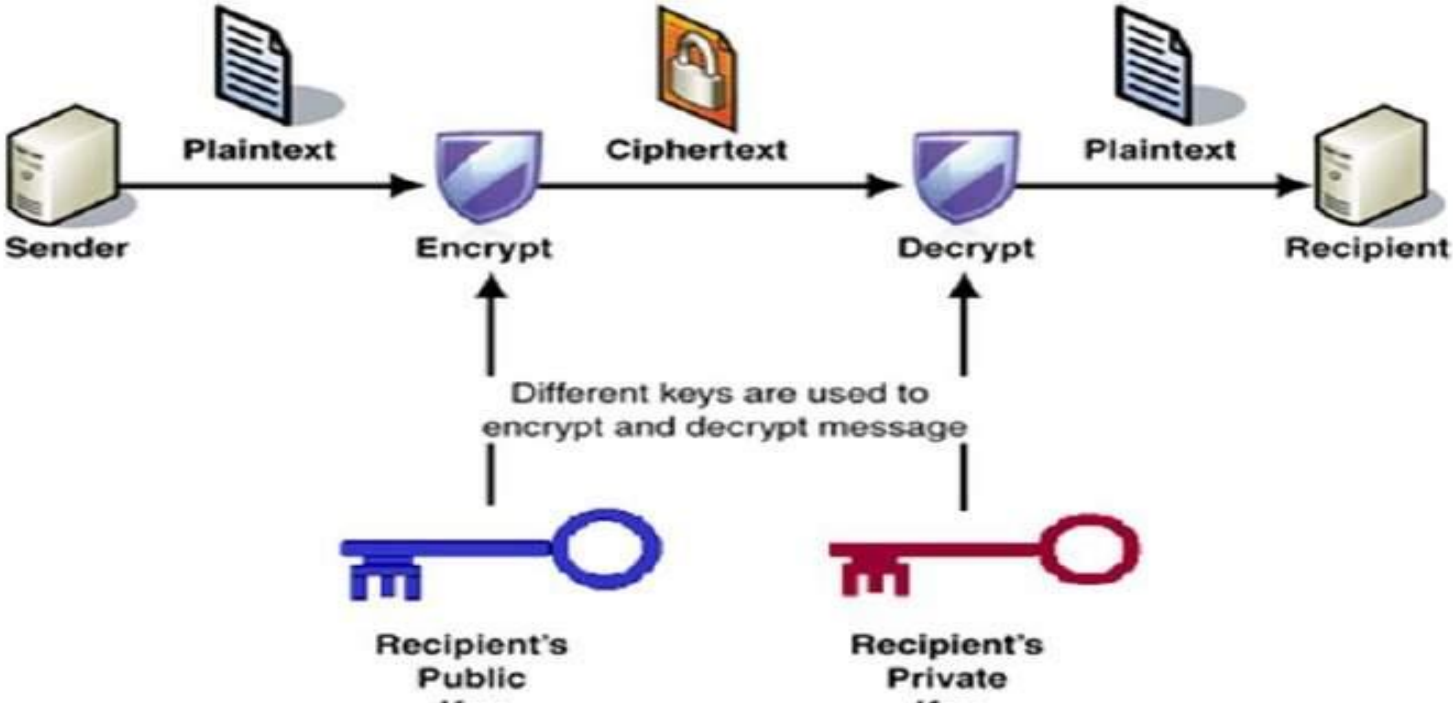


The salient features of this encryption scheme are as follows

- Every user in this system needs to have a pair of dissimilar keys, **private key** and **public key**. These keys are mathematically related − when one key is used for encryption, the other can decrypt the ciphertext back to the original plaintext.
- It requires to put the public key in public repository and the private key as a well-guarded secret. Hence, this scheme of encryption is also called **Public Key Encryption**.
- Though public and private keys of the user are related, it is computationally not feasible to find one from another. This is a strength of this scheme.
- When *Host1* needs to send data to *Host2,* he obtains the public key of *Host2* from repository, encrypts the data, and transmits.
- *Host2* uses his private key to extract the plaintext.
- Length of Keys (number of bits) in this encryption is large and hence, the process of encryption-decryption is slower than symmetric key encryption.

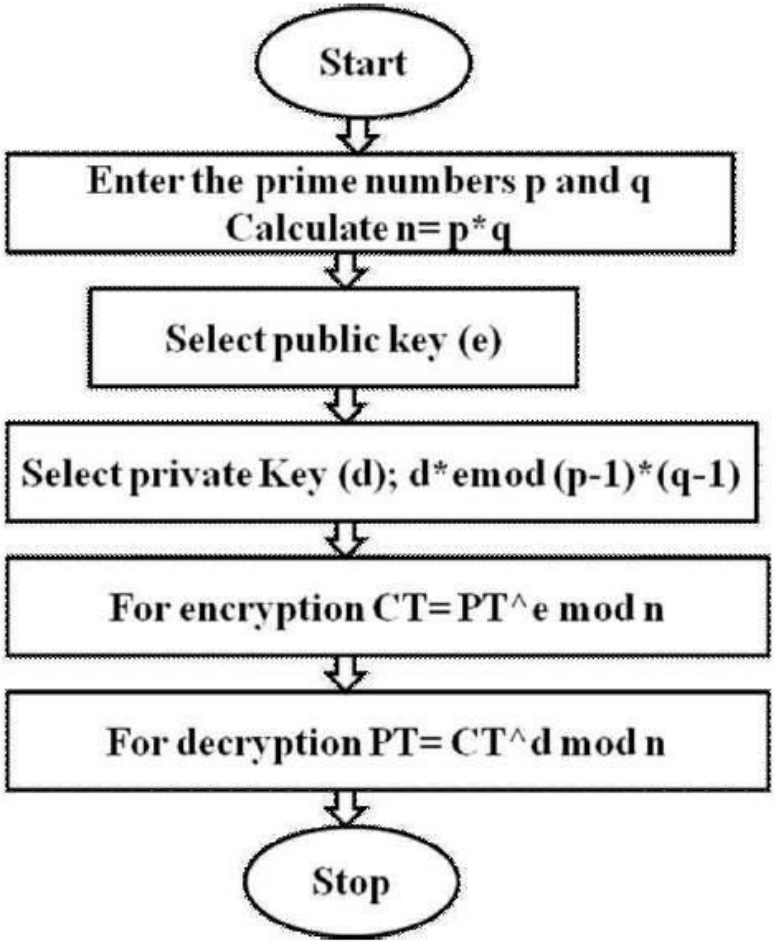
Public Key Cryptography

- Symmetric cryptography was well suited for organizations such as governments, military, and big financial corporations were involved in the classified communication.
- With the spread of more unsecure computer networks in last few decades, a genuine need was felt to use cryptography at larger scale. The symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to the public key cryptosystems.
- The process of encryption and decryption is depicted in the following illustration



- The most important properties of public key encryption scheme are −
- Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
- Each receiver possesses a unique decryption key, generally referred to as his private key.
- Receiver needs to publish an encryption key, referred to as his public key.
- Encryption algorithm is complex enough to prohibit attacker from deducing the plaintext from the ciphertext and the encryption (public) key.

RSA


Start
Enter the prime numbers p and q
Calculate n= p*q
Select public key (e)
Select private Key (d); d*emod (p-1)*(q-1)
For encryption CT= PT^e mod n
For decryption PT= CT^d mod n
Stop

RSA Cryptosystem

- This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars **Ron Rivest, Adi Shamir,** and **Len Adleman** and hence, it is termed as RSA cryptosystem.
- Generation of RSA Key Pair
- Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below −
- **Generate the RSA modulus (n)**
  - Select two large primes, p and q.
  - Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.

- **Find Derived Number (e)**
  - Number **e** must be greater than 1 and less than $(p - 1)(q - 1)$.
  - There must be no common factor for e and $(p - 1)(q - 1)$ except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.
- **Form the public key**
  - The pair of numbers (n, e) form the RSA public key and is made public.
  - Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.
- **Generate the private key**
  - Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
  - Number d is the inverse of e modulo (p – 1)(q – 1). This means that d is the number less than (p – 1)(q – 1) such that when multiplied by e, it is equal to 1 modulo (p – 1)(q – 1).
  - This relationship is written mathematically as follows −
  - $ed = 1 \bmod (p - 1)(q - 1)$

Example

- An example of generating RSA Key pair is given below. (For ease of understanding, the primes p & q taken here are small values. Practically, these values are very high).
- Let two primes be p = 7 and q = 13. Thus, modulus n = pq = 7 x 13 = 91.
- Select e = 5, which is a valid choice since there is no number that is common factor of 5 and (p − 1)(q − 1) = 6 × 12 = 72, except for 1.
- The pair of numbers (n, e) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input p = 7, q = 13, and e = 5 to the Extended Euclidean Algorithm. The output will be d = 29.
- Check that the d calculated is correct by computing −
- de = 29 × 5 = 145 = 1 mod 72 Hence, public key is (91, 5) and private keys is (91, 29).

Encryption and Decryption

- RSA Encryption
- Suppose the sender wish to send some text message to someone whose public key is (n, e).
- The sender then represents the plaintext as a series of numbers less than n.
- To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as −
- $C = P^e \bmod n$
- In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n.
- Returning to our Key Generation example with plaintext P = 10, we get ciphertext C −
- $C = 10^5 \bmod 91$

- RSA Decryption
- The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a ciphertext C.
- Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.
- Plaintext = C^d mod n Returning again to our numerical example, the ciphertext C = 82 would get decrypted to number 10 using private key 29 −
- Plaintext = 82^{29} mod 91 = 10

diffie hellman key exchange 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:

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

   $B = g^y \bmod n$
5. Bob sends the number B to Alice.
6. A now computes the secret key K1 as follows:

   $K1 = B^x \bmod n$
7. B now computes the secret key K2 as follows:

   $K2 = A^y \bmod n$

Let us try to understand what this actually means, in simple terms.

(a) Firstly, take a look at what Alice does in step 6. Here, Alice computes:
$K1 = B^x \bmod n$.
What is B? From step 4, we have:
$B = g^y \bmod n$.
Therefore, if we substitute this value of B in step 6, we will have the following eq
$K1 = (g^y)^x \bmod n = g^{yx} \bmod n$.

(b) Now, take a look at what Bob does in step 7. Here, Bob computes:
$K2 = A^y \bmod n$.

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

1. Alice wants to communicate with Bob securely, and therefore, she first wants to do a Diffie-Hellman key exchange with him. For this purpose, she sends the values of n and g to Bob, as usual. Let n = 11 and g = 7. (As usual, these values will form the basis of Alice's A and Bob's B, which will be used to calculate the symmetric key K1 = K2 = K.)
2. Alice does not realize that the attacker Tom is listening quietly to the conversation between her and Bob. Tom simply picks up the values of n and g, and also forwards them to Bob as they originally were (i.e. n = 11 and g = 7). This is shown in Fig. 2.27.

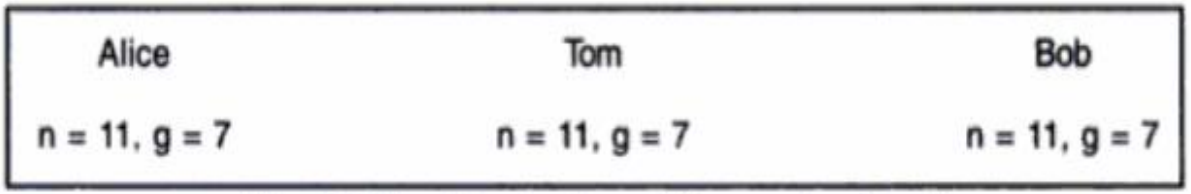


Fig. 2.27 *Man-in-the-middle attack—Part I*

3. Now, let us assume that Alice, Tom and Bob select random numbers x and y as shown in Fig. 2.28.

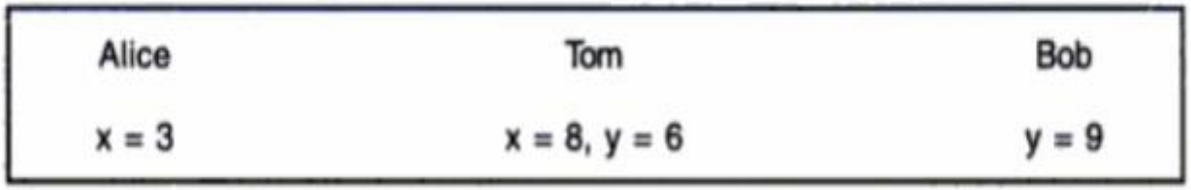


Fig. 2.28 *Man-in-the-middle attack—Part II*

4. One question at this stage could be: why does Tom selects both x and y? We shall answer that shortly. Now, based on these values, all the three persons calculate the values of A and B as shown in Fig. 2.29. Note that Alice and Bob calculate only A and B, respectively. However, Tom calculates both A and B. We shall revisit this shortly.

Alice	Tom	Bob
$A = g^x \bmod n$ $= 7^3 \bmod 11$ $= 343 \bmod 11$ $= 2$	$A = g^x \bmod n$ $= 7^8 \bmod 11$ $= 5764801 \bmod 11$ $= 9$ $B = g^y \bmod n$ $= 7^6 \bmod 11$ $= 117649 \bmod 11$ $= 4$	$B = g^y \bmod n$ $= 7^9 \bmod 11$ $= 40353607 \bmod 11$ $= 8$

5. Now, the real drama begins, as shown in Fig. 2.30.

   As shown in the figure, the following things happen:

   (a) Alice sends her A (i.e. 2) to Bob. Tom intercepts it, and instead, sends his A (i.e. 9) to Bob. Bob has no idea that Tom had hijacked Alice's A and has instead given his A to Bob.

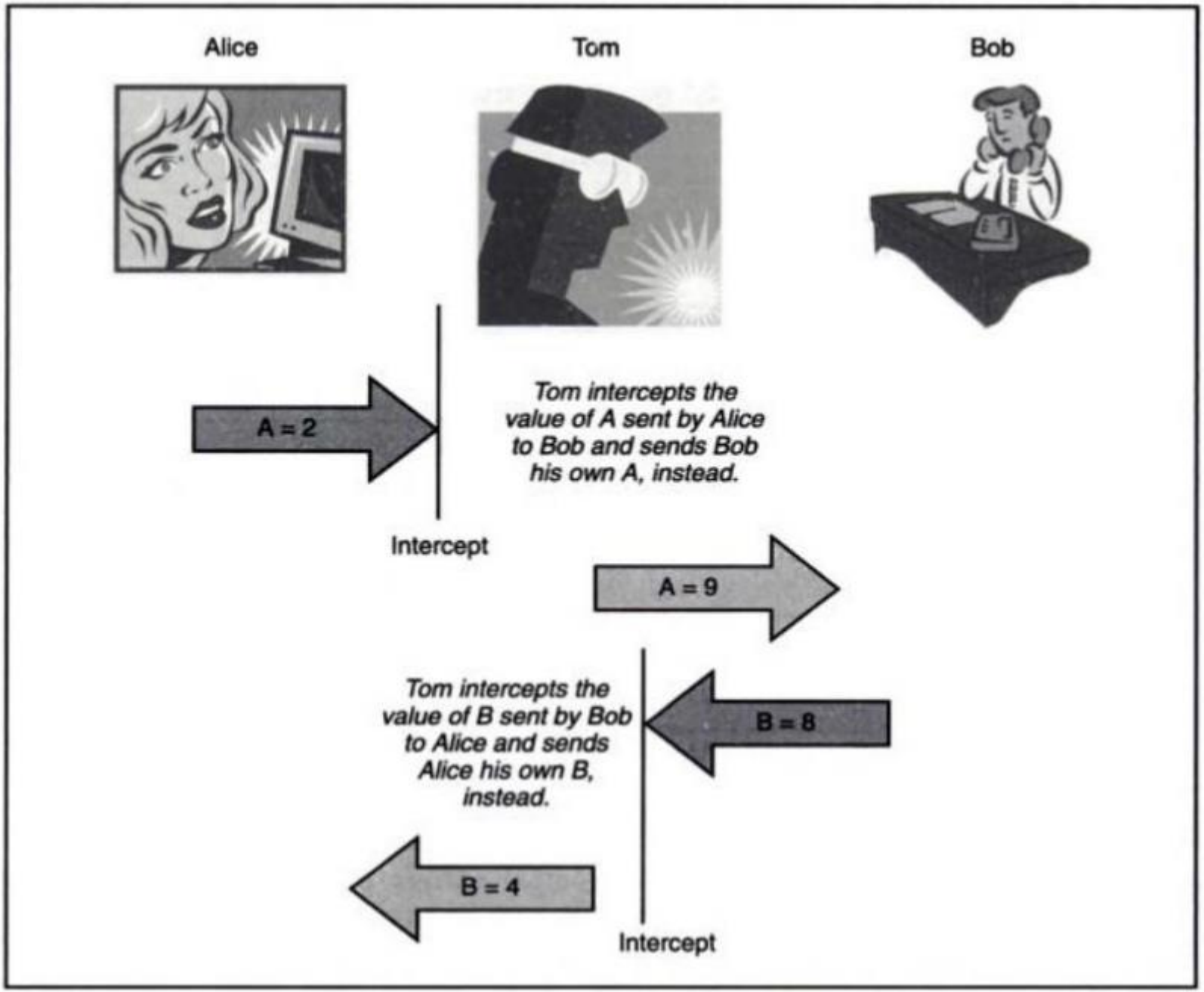


Fig. 2.30 *Man-in-the-middle attack—Part IV*

(b) In return, Bob sends his B (i.e. 8) to Alice. As before, Tom intercepts it, and instead, sends his B (i.e. 4) to Alice. Alice thinks that this B came from Bob. She has no idea that Tom had intercepted the transmission from Bob, and changed B.

(c) Therefore, at this juncture, Alice, Tom and Bob have the values of A and B as shown in Fig. 2.31.

Alice	Tom	Bob
A = 2, B = 4*	A = 2, B = 8	A = 9*, B = 8

(Note: * indicates that these are the values after Tom hijacked and changed them.)

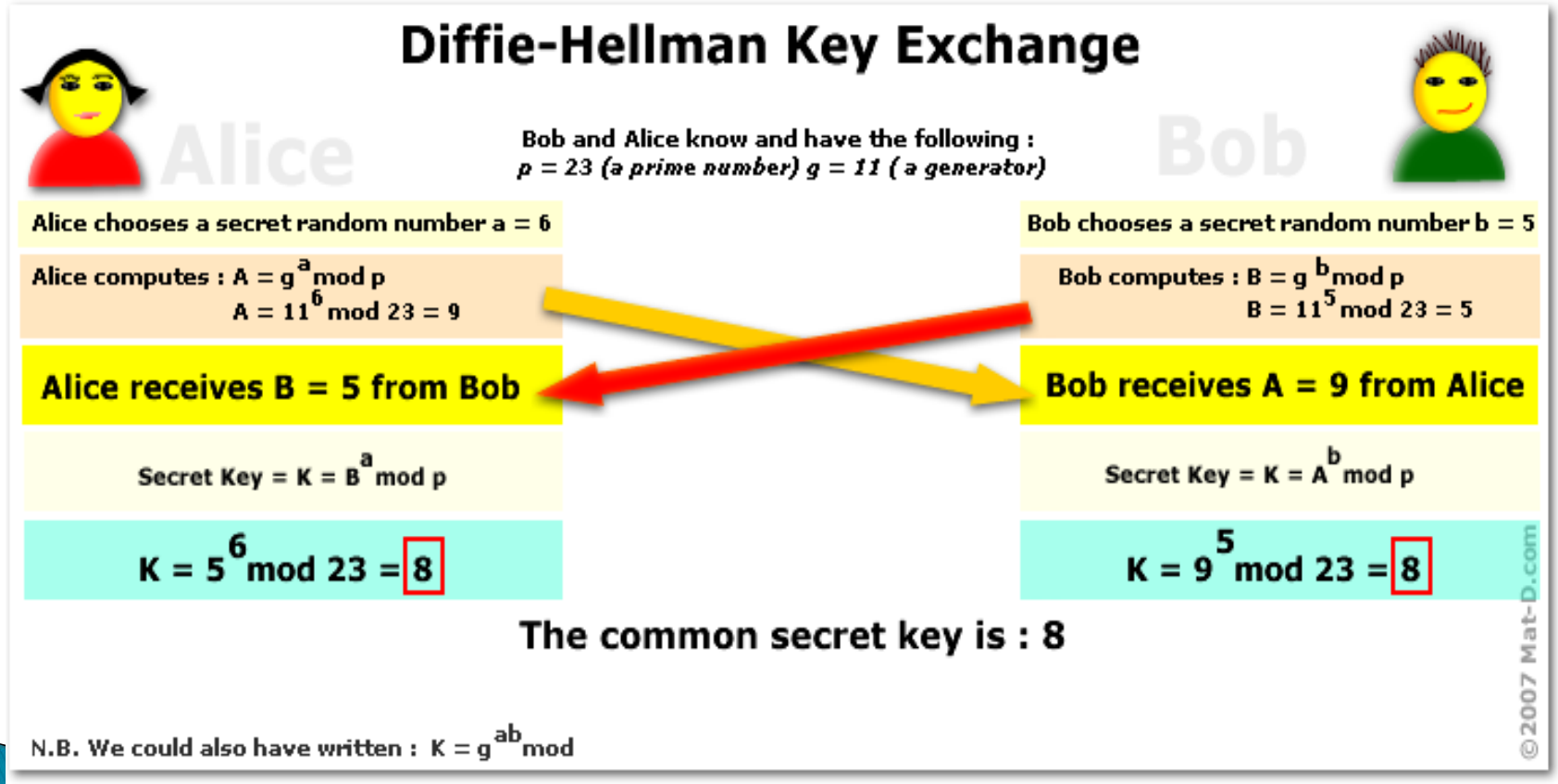
Fig. 2.31 *Man-in-the-middle attack—Part V*

6. Based on these values, all the three persons now calculate their keys as shown in Fig. 2.32. We will notice that Alice calculates only K1, Bob calculates only K2, whereas Tom calculates both K1 and K2. Why does Tom need to do this? We shall discuss that soon.

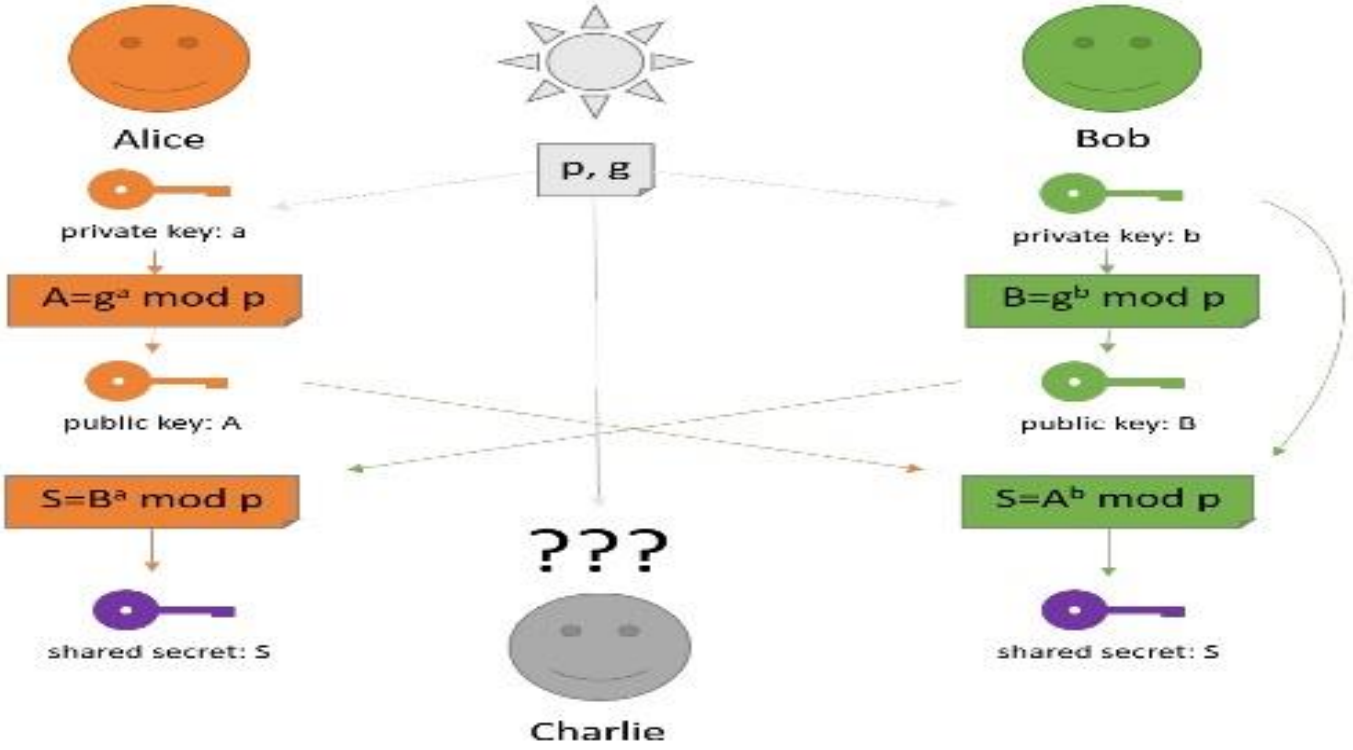
Alice	Tom	Bob
$K1 = B^x \bmod n$	$K1 = B^x \bmod n$	$K2 = A^y \bmod n$
$= 4^3 \bmod 11$	$= 8^8 \bmod 11$	$= 9^9 \bmod 11$
$= 64 \bmod 11$	$= 16777216 \bmod 11$	$= 387420489 \bmod 11$
$= 9$	$= 5$	$= 5$
	$K2 = A^y \bmod n$	
	$= 2^6 \bmod 11$	
	$= 64 \bmod 11$	
	$= 9$	

Fig. 2.32 *Man-in-the-middle attack—Part VI*

diffie hellman key exchange algorithm

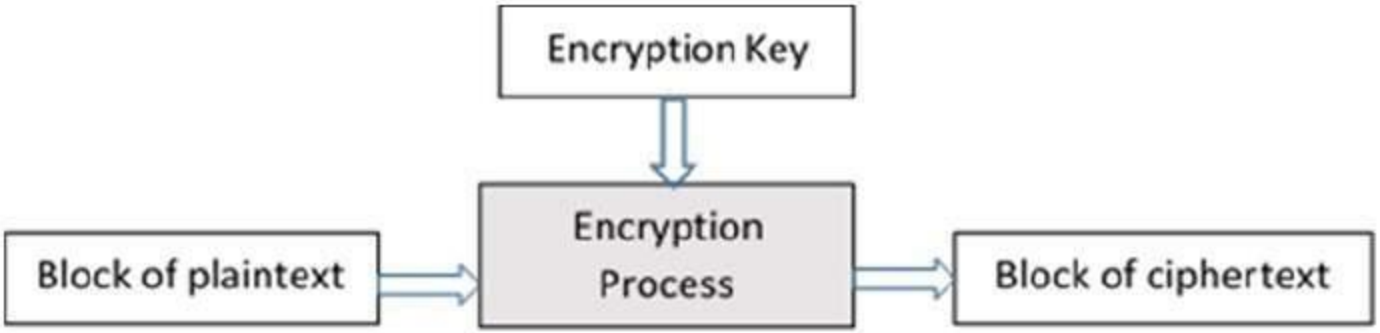


Working of Diffie-Hellman



Block Ciphers

- In this scheme, the plain binary text is processed in blocks (groups) of bits at a time; i.e. a block of plaintext bits is selected, a series of operations is performed on this block to generate a block of ciphertext bits. The number of bits in a block is fixed. For example, the schemes DES and AES have block sizes of 64 and 128, respectively.
- The basic scheme of a block cipher is depicted as follows −



- **Block Cipher Schemes**
- There is a vast number of block ciphers schemes that are in use. Many of them are publically known. Most popular and prominent block ciphers are listed below.
- **Digital Encryption Standard (DES)** − The popular block cipher of the 1990s. It is now considered as a 'broken' block cipher, due primarily to its small key size.
- **Triple DES** − It is a variant scheme based on repeated DES applications. It is still a respected block ciphers but inefficient compared to the new faster block ciphers available.
- **Advanced Encryption Standard (AES)** − It is a relatively new block cipher based on the encryption algorithm **Rijndael** that won the AES design competition.

DES

3.4.2 How DES Works

1. *Basic principles*

DES is a block cipher. It encrypts data in blocks of size 64 bits each. That is, 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is shown in Fig. 3.16.

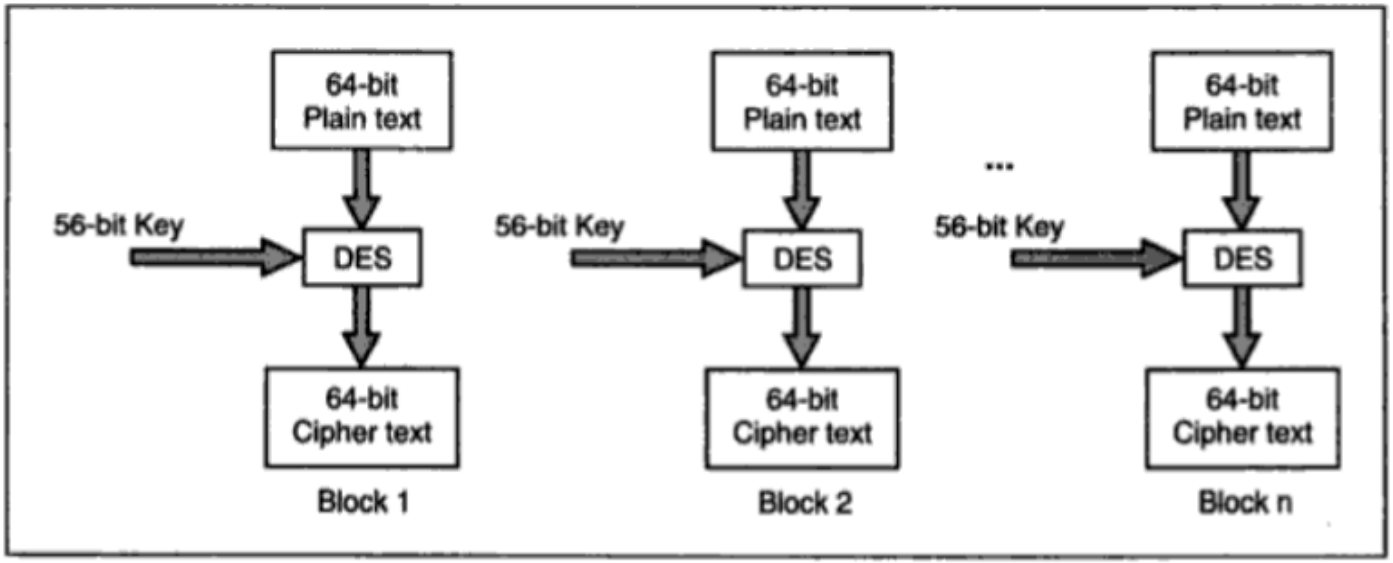


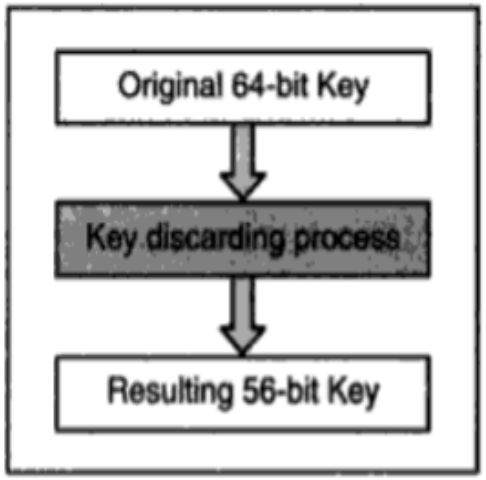
Fig. 3.16 *The conceptual working of DES*

We have mentioned that DES uses a 56-bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every eighth bit of the key is discarded to produce a 56-bit key. That is, bit positions 8, 16, 24, 32, 40, 48, 56 and 64 are discarded. This is shown in Fig. 3.17 with shaded bit positions indicating discarded bits. (Before discarding, these bits can be used for parity checking to ensure that the key does not contain any errors.)

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

Fig. 3.17 *Discarding of every 8th bit of the original key (Shaded bit positions are discarded)*

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key, as shown in Fig. 3.18.



Simplistically, DES is based on the two fundamental attributes of cryptography: substitution (also called as confusion) and transposition (also called as diffusion). DES consists of 16 steps, each of which is called as a **round**. Each *round* performs the steps of substitution and transposition. Let us now discuss the broad-level *steps* in DES.

1. In the first step, the 64-bit plain text block is handed over to an **Initial Permutation (IP)** function.
2. The Initial Permutation is performed on plain text.
3. Next, the Initial Permutation (IP) produces two halves of the permuted block; say Left Plain Text (LPT) and Right Plain Text (RPT).
4. Now, each of LPT and RPT goes through 16 *rounds* of encryption process, each with its own key.
5. In the end, LPT and RPT are rejoined, and a **Final Permutation (FP)** is performed on the combined block.
6. The result of this process produces 64-bit cipher text.

Fig. 3.18 *Key discarding process*

This process is shown diagrammatically in Fig. 3.19.

Let us now understand each of these processes in detail.

Initial Permutation

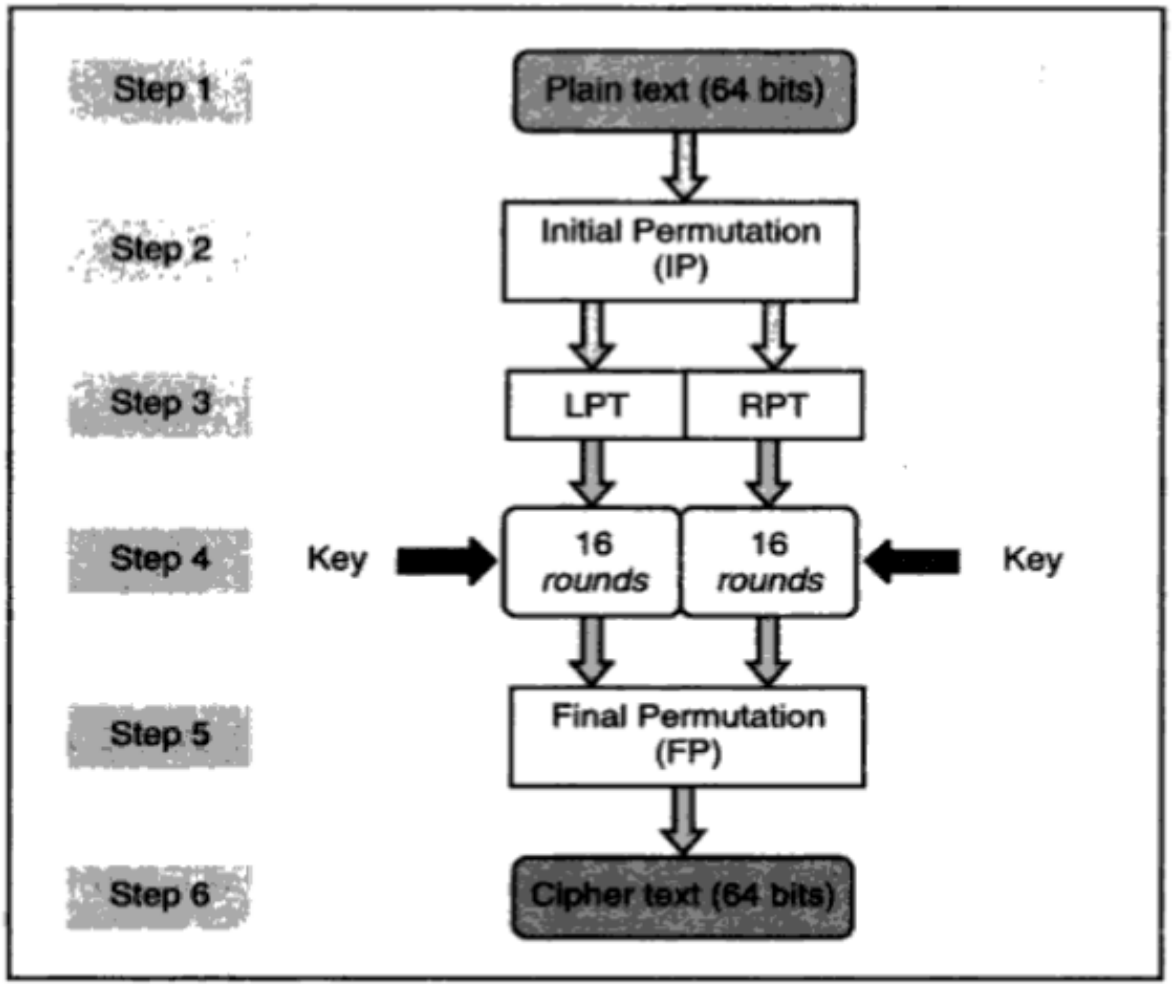


Table 3.1 *Idea of IP*

Bit position in the plain text block	*To be overwritten with the contents of this bit position*
1	58
2	50
3	42
..	..
64	7

The complete transposition table used by IP is shown in Fig. 3.20. This table (and all others in this chapter) should be read from left to right, top to bottom. For instance, we have noted that 58 in the first position indicates that the contents of the 58th bit in the original plain text block will overwrite the contents of the 1st bit position, during IP. Similarly, 1 is shown at the 40th position in the table, which means that the first bit will overwrite the 40th bit in the original plain text block. The same rule applies for all other bit positions.

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

Fig. 3.20 *Initial Permutation (IP) table*

As we have noted, after IP is done, the resulting 64-bit permuted text block is divided into two half blocks. Each half block consists of 32 bits. We have called the left block as LPT and the right block as RPT. Now, 16 *rounds* are performed on these two blocks. We shall discuss this process now.

3. *Rounds*

Each of the 16 rounds, in turn, consists of the broad level steps outlined in Fig. 3.21.

Let us discuss these details step-by-step.

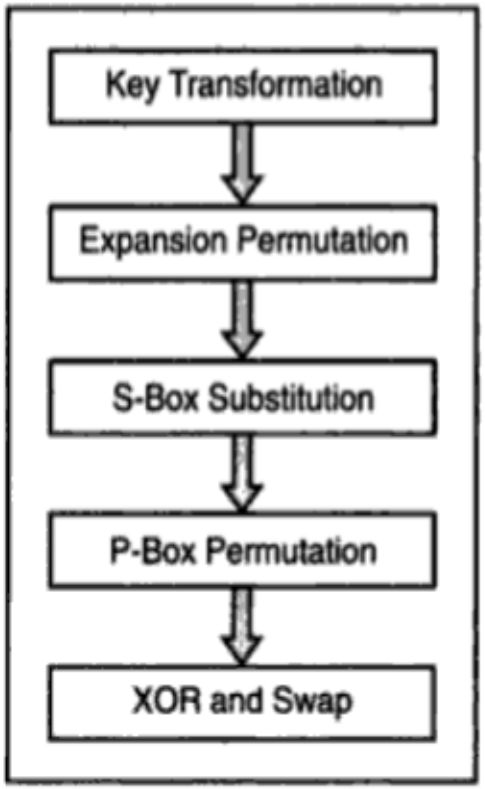


Fig. 3.21 *Details of one round in DES*

Step 1: Key Transformation

We have noted that the initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each round, a 56-bit key is available. From this 56-bit key, a different 48-bit **sub-key** is generated during each *round* using a process called as **key transformation**. For this, the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round. For example, if the *round* number is 1, 2, 9 or 16, the shift is done by only one position. For other rounds, the circular shift is done by two positions. The number of key bits shifted per round is shown in Fig. 3.22.

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

Fig. 3.22 *Number of key bits shifted per round*

After an appropriate shift, 48 of the 56 bits are selected. For selecting 48 of the 56 bits, the table shown in Fig. 3.23 is used. For instance, after the shift, bit number 14 moves into the first position, bit number 17 moves into the second position, and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce the 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub-set of the original 56-bit key, it is called as **compression permutation**.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Fig. 3.23 *Compression permutation*

Because of this compression permutation technique, a different subset of key bits gets used in each round. That makes DES not so easy to crack.

Step 2: Expansion Permutation

Recall that after Initial Permutation, we had two 32-bit plain text areas, called as Left Plain Text (LPT) and Right Plain Text (RPT). During **expansion permutation**, the RPT is expanded from 32 bits to 48 bits. Besides increasing the bit size from 32 to 48, the bits are permuted as well, hence the name *expansion permutation*. This happens as follows:

1. The 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits. This is shown in Fig. 3.24.

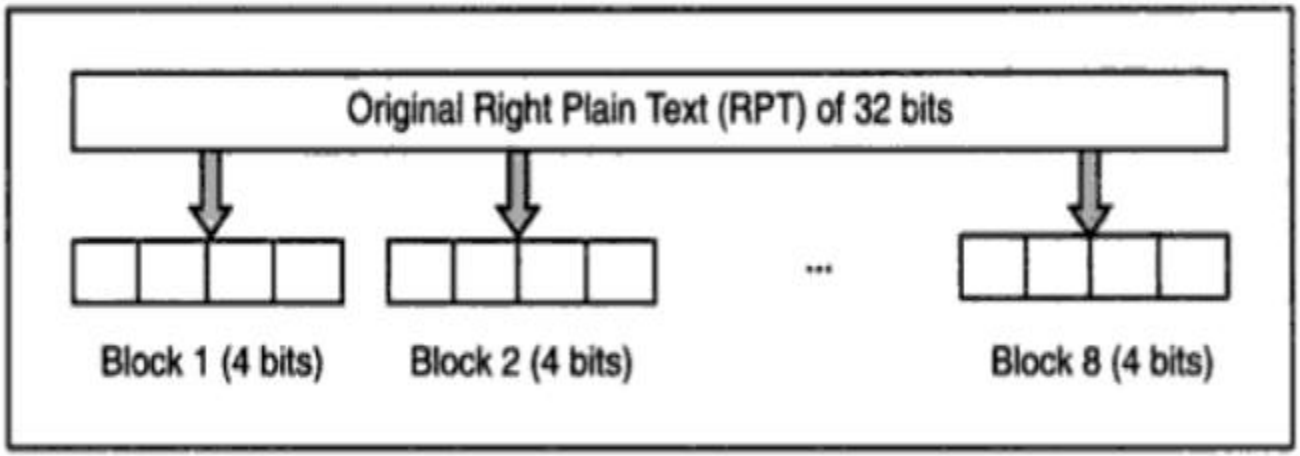


Fig. 3.24 *Division of 32-bit RPT into eight 4-bit blocks*

2. Next, each 4-bit block of the above step is then expanded to a corresponding 6-bit block. That is, per 4-bit block, 2 more bits are added. What are these two bits? They are actually the repeated first and the fourth bits of the 4-bit block. The second and the third bits are written down as they were in the input. This is shown in Fig. 3.25. Note that the first input bit is outputted to the second output position, and also repeats in output position 48. Similarly, the 32nd input bit is found in the 47th output position as well as in the first output position.

Clearly, this process results into expansion as well as permutation of the input bits while creating the output.

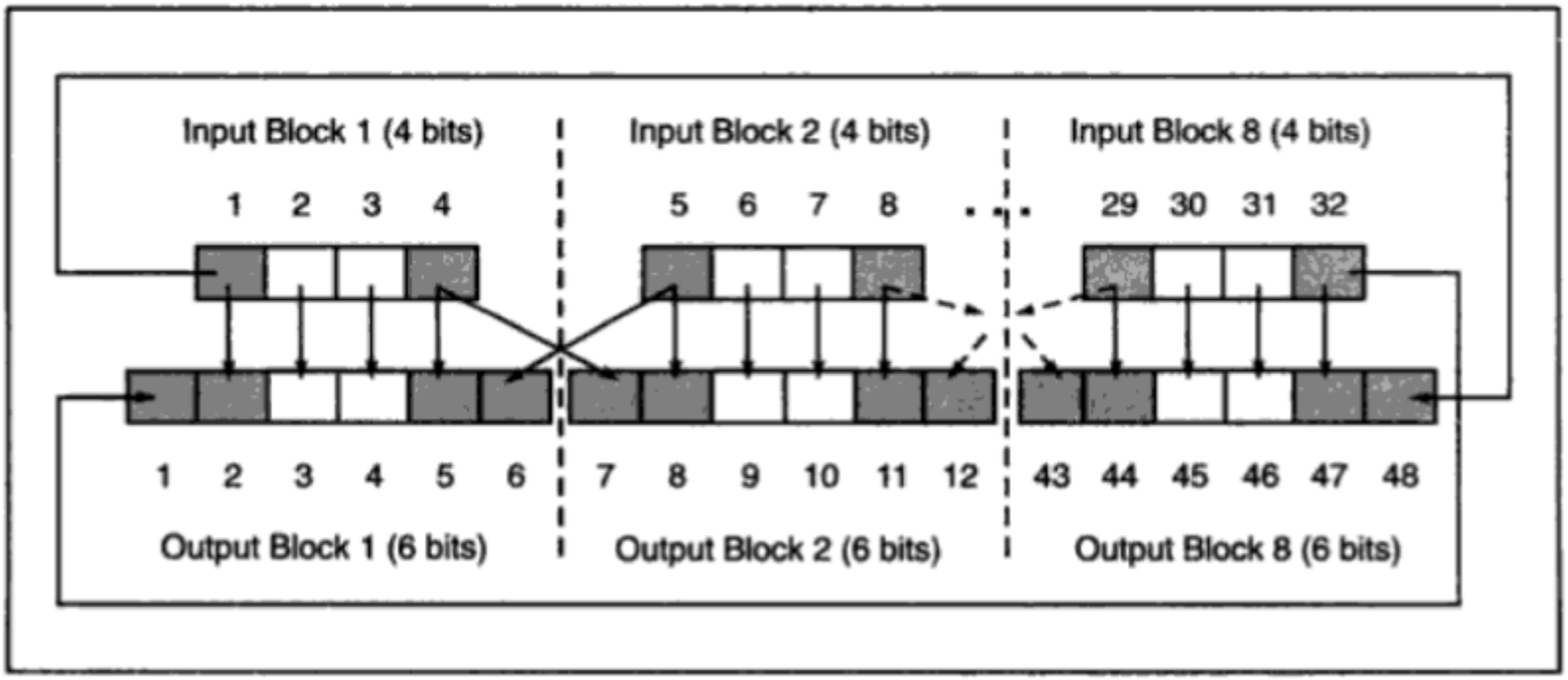


Fig. 3.25 *RPT expansion permutation process*

As we can see, the first input bit goes into the second and the 48th output positions. The second input bit goes into the third output position, and so on. Consequently, we will observe that the expansion permutation has actually used the table shown in Fig. 3.26.

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Fig. 3.26 *RPT expansion permutation table*

As we have seen, firstly, the *Key transformation* process compresses the 56-bit key to 48 bits. Then, the *Expansion permutation* process expands the 32-bit RPT to 48 bits. Now, the 48-bit key is XORed with the 48-bit RPT, and the resulting output is given to the next step, which is the **S-box Substitution** (which we shall discuss in the next section) as shown in Fig. 3.27.

S-Box Substitution

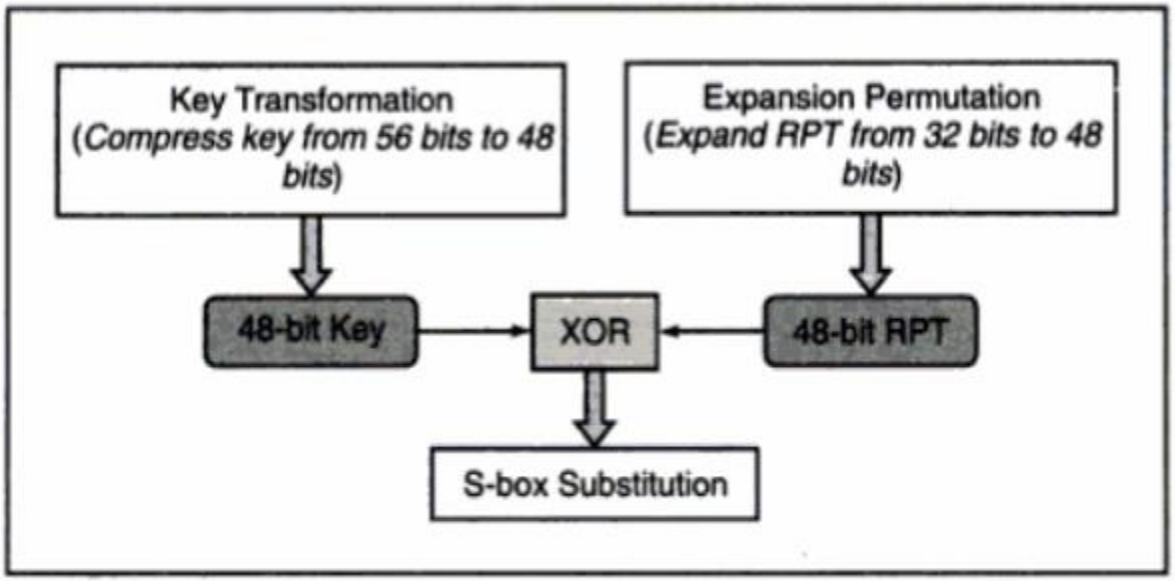


Fig. 3.27 *Way to S-box Substitution*

The substitution is performed by eight **substitution boxes** (also called as **S-boxes**). Each of the eight S-boxes has a 6-bit input and a 4-bit output. The 48-bit input block is divided into 8 sub-blocks (each containing 6 bits), and each such sub-block is given to a S-box. The S-box transforms the 6-bit input into a 4-bit output, as shown in Fig. 3.28.

What is the logic used by S-box substitution for selecting only four of the six bits? We can conceptually think of every S-box as a table that has 4 rows (numbered 0 to 3) and 16 columns (numbered 0 to 15). Thus, we have 8 such tables, one for each S-box. At the intersection of every row and column, a 4-bit number (which will be the 4-bit output for that S-box) is present. This is shown in Fig. 3.29 [(a)–(h)]

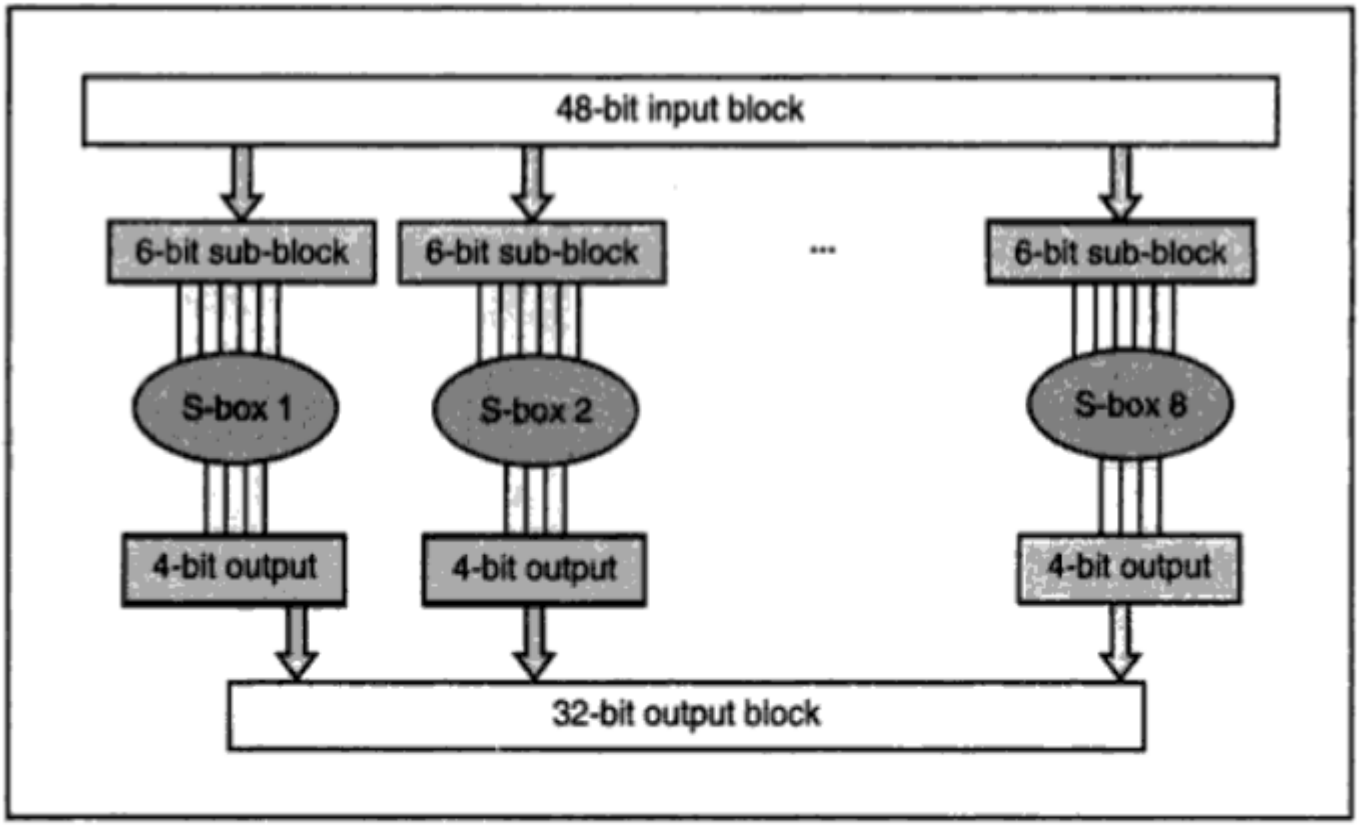


Fig. 3.28 *S-box substitution*

Step 4: P-box Permutation

The output of S-box consists of 32 bits. These 32 bits are permuted using a P-box. This straightforward permutation mechanism involves simple permutation (i.e. replacement of each bit with another bit, as specified in the P-box table, without any expansion or compression). This is called as **P-box Permutation**. The P-box is shown in Fig. 3.32. For example, a 16 in the first block indicates that the bit at position 16 of the original input moves to bit at position 1 in the output, and a 10 in the block number 16 indicates that the bit at the position 10 of the original input moves to bit at the position 16 in the output.

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Fig. 3.32 *P-box permutation*

Step 5: XOR and Swap

Note that we have been performing all these operations only on the 32-bit right half portion of the 64-bit original plain text (i.e. on the RPT). The left half portion (i.e. LPT) was untouched so far. At this juncture, the left half portion of the initial 64-bit plain text block (i.e. LPT) is XORed with the output produced by P-box permutation. The result of this XOR operation becomes the new right half (i.e. RPT). The old right half (i.e. RPT) becomes the new left half, in a process of swapping. This is shown in Fig. 3.33.

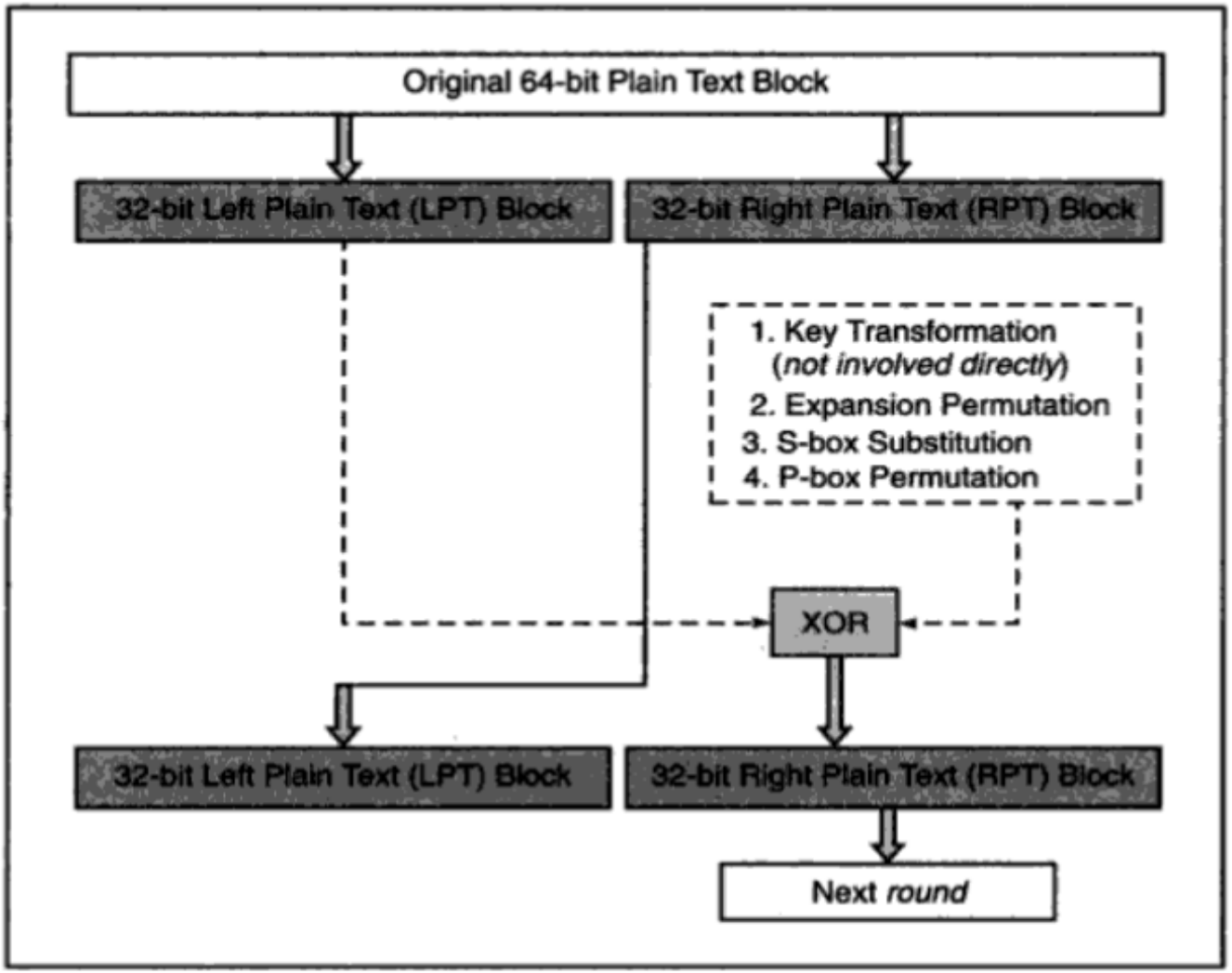


Fig. 3.33 *XOR and Swap*

4. *Final Permutation*

At the end of the 16 rounds, the **Final Permutation** is performed (only once). This is a simple transposition, based on Fig. 3.34. For instance, the 40th input bit takes the position of the 1st output bit, and so on.

The output of the Final Permutation is the 64-bit encrypted block.

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

Fig. 3.34 *Final Permutation*

5. *DES decryption*

The End