CLASSICAL ENCRYPTION TECHNIQUES : There are two basic building blocks of all encryption techniques: substitution and transposition.

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

<u>Caesar cipher (or) shift cipher:</u>

The earliest known use of a substitution cipher and the simplest was by Julius Caesar.

The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.

e.g., plain text : pay more money

Cipher text: SDB PRUH PRQHB

Note that the alphabet is wrapped around, so that letter following z is a.

For each plaintext letter p, substitute the cipher text letter c such that

C = E(p) = (p+3) mod 26

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

C = E (p) = (p+k) mod26

Where k takes on a value in the range 1 to 25.

The decryption algorithm is simply

P = D(C) = (C-k) mod26

Egs: plaintext **PROTOCOL**.
Where a key is needed I'll use **NETWORK** (you gave this earlier). For Caesar I'll use a standard **shift = 3**.

1) CAESAR CIPHER (shift = 3)

Encryption

Plaintext: PROTOCOL
Alphabet positions (A=0..Z=25):
P=15, R=17, O=14, T=19, O=14, C=2, O=14, L=11

Apply Cipher = (Plain + 3) mod 26:

- P (15) → 18 → S

- R (17) → 20 → U
- O (14) → 17 → R
- T (19) → 22 → W
- O (14) → 17 → R
- C (2) → 5 → F
- O (14) → 17 → R
- L (11) → 14 → O

Ciphertext: **SURWRFR O** → remove spacing → **SURWRFRO**

Decryption

Ciphertext: SURWRFRO
Apply Plain = (Cipher − 3) mod 26:

- S (18) → 15 → P
- U (20) → 17 → R
- R (17) → 14 → O
- W (22) → 19 → T
- R (17) → 14 → O
- F (5) → 2 → C
- R (17) → 14 → O
- O (14) → 11 → L

Recovered plaintext: **PROTOCOL**

Characteristics

- **Very easy to implement**
- **Only 25 possible keys → Not secure**

2. MONOALPHABETIC CIPHER

Also called Simple Substitution Cipher.

Here, each letter of plaintext is mapped to a unique letter of ciphertext, but the mapping can be any permutation of the alphabet (not a fixed shift).

Example Key Mapping:

Plain : ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: QWERTYUIOPASDFGHJKLZXCVBNM

Encryption

Replace each plaintext letter using the key mapping.

HELLO → ITSSG

Decryption

Use reverse mapping (Cipher → Plain).

Characteristics

- Stronger than Caesar Cipher
- 26! possible keys
- Still vulnerable to **frequency analysis**

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

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

Encryption (apply mapping letterwise)

Plaintext: P R O T O C O L

- P → H
- R → K
- O → G
- T → Z
- O → G
- C → E
- O → G
- L → S

Ciphertext: **HKGZG EGS** → **HKGZG EGS** (no spaces) → **HKGZG EGS** → final **HKGZG EGS** → cleaned **HKGZG EGS** — better shown as **HKGZGEGS**.

Decryption:

Take ciphertext HKGZGEGS. Use reverse mapping (Cipher→Plain):

- H → P
- K → R
- G → O
- Z → T
- G → O
- E → C
- G → O
- S → L

Recovered plaintext: **PROTOCOL**

Note: With monoalphabetic substitution you must keep the mapping secure. Frequency analysis can break it.

3. PLAYFAIR CIPHER

A **digraph substitution cipher** where plaintext is encrypted **two letters at a time** using a **5×5 key matrix** formed with a keyword.

Steps

1. Choose a keyword (e.g., "NETWORK")
2. Prepare 5x5 matrix (I/J combined)
3. Break plaintext into pairs
4. Apply Playfair rules:
    - Same row → take letters to the **right**
    - Same column → take letters **down**
    - Rectangle → take letters in the **same row, opposite corners**
    - Add **X** between repeated letters or at end if needed

Example

Plaintext:

BALLOON → BA LX LO ON

Characteristics

- Encrypts **pairs**, not single letters

- Better security than Caesar/monoalphabetic
- Still vulnerable to digraph frequency attack

Eg: PLAYFAIR CIPHER (keyword = NETWORK, I/J combined)

Build 5×5 key square (keyword NETWORK, remove duplicates, fill remaining A–Z except J)

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

Prepare plaintext into digraphs

Plaintext: PROTOCOL

Split into pairs (two letters each): PR | OT | OC | OL

(Length is even, no padding required.)

Encryption — apply Playfair rules to each digraph

Rules reminder:

- If both letters in same row → replace each by letter to its **right** (wrap around).
- If both letters in same column → replace each by letter **below** (wrap).
- Otherwise (rectangle) → each letter replaced by the letter in the same row but in the **column of the other** letter (i.e., opposite corners of rectangle).

Now each pair:

Pair 1: PR

- P at (3,2) — row3,col2
- R at (1,0) — row1,col0
  Rectangle → take opposite corners:
- P → (3,0) = **L**

- R → (1,2) = **A**
  PR → **LA**

Pair 2: OT

- O at (0,4) — row0,col4
- T at (0,2) — row0,col2
  Same row (row0) → replace by letters to the **right**:
- O (col4) → wrap to col0 = **N**
- T (col2) → right col3 = **W**
  OT → **NW**

Pair 3: OC

- O at (0,4) (row0,col4)
- C at (1,4) (row1,col4)
  Same column (col4) → replace by letters **below**:
- O (row0) → down row1 = **C**
- C (row1) → down row2 = **I**
  OC → **CI**

Pair 4: OL

- O at (0,4) (row0,col4)
- L at (3,0) (row3,col0)
  Rectangle → opposite corners:
- O → (0,0) = **N**
- L → (3,4) = **S**
  OL → **NS**

Ciphertext

Concatenate pairs: **LA NW CI NS** → **LANWCINS**

Decryption (reverse Playfair rules)

Break ciphertext LANWCINS into digraphs: LA | NW | CI | NS

Reverse the rules:

- same row → shift **left**
- same column → shift **up**

- rectangle → swap columns (opposite corners again)

LA

- L (3,0), A (1,2) → rectangle → P (3,2), R (1,0) → **PR**

NW

- N (0,0), W (0,3) → same row → shift left:
  - N ← O (wrap) → **O**
  - W ← T → **T**
    → **OT**

CI

- C (1,4), I (2,4) → same column → shift up:
  - C ← O (0,4) → **O**
  - I ← C (1,4) → **C**
    → **OC**

NS

- N (0,0), S (3,4) → rectangle → O (0,4), L (3,0) → **OL**

Concatenate: **PR OT OC OL** → **PROTOCOL**

PLAINTEXT: PROTOCOL KEY: NETWORK.

Encryption:

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

PR OT ~~O~~C OL

PR:
Take opposite corners
P : L
R : A
PR : LA.

OT:
Same row: Replace ~~by~~ letters to the right.

O : N
T : W

OT : NW

OC:
Same column: replace by letters below.

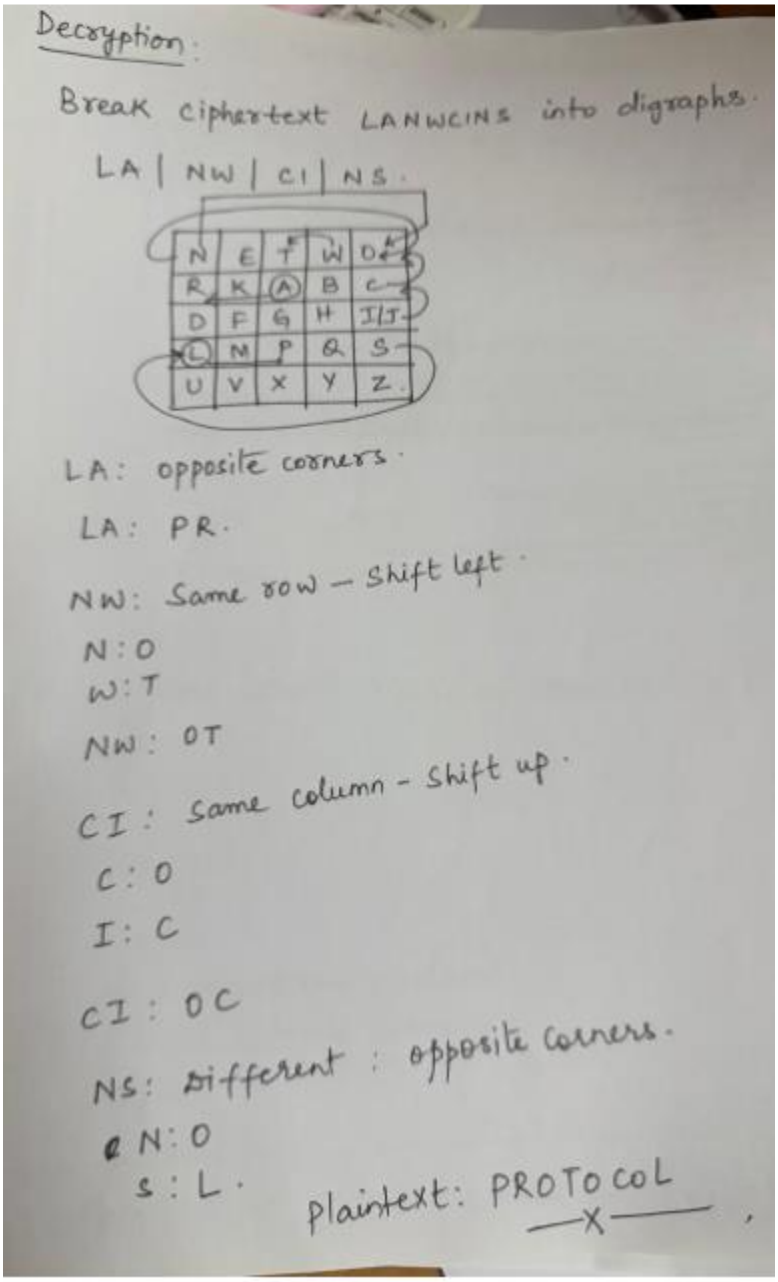
O : C
C : I

OC : CI

OL:
~~So~~ Different row & column.
Take opposite corners.

O : N
L : S.

Plain Text: PROTOCOL : CIPHERTEXT: LANWCINS



Decryption:

Break ciphertext LANWCINS into digraphs.

LA | NW | CI | NS.

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

LA: opposite corners.

LA: PR.

NW: Same row – Shift left.

N: O
W: T

NW: OT

CI: Same column - Shift up.

C: O

I: C

CI: OC

NS: Different : opposite corners.

N: O
S: L.

Plaintext: PROTOCOL

—X—

TRANSPOSITION TECHNIQUES: All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol.

A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

<u>Rail fence</u> : is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Plaintext= meet at the school house

To encipher this message with a rail fence of depth 2, we write the message as follows:

M		E		A		T		E		C		O		L		O		S	
	E		T		T		H		S		H		O		H		U		E

The encrypted message is MEATECOLOSETTHSHOHUE

Eg:2 (Example with **key = 3 rails**)

Encryption Steps

Plaintext: **PROTOCOL**

Write letters in a zig-zag pattern with 3 rails:

P				O				
	R		T		C		L	
		O				O		

CIPHERTEXT:

Join rows:

- Row 1 → **P O**
- Row 2 → **RTCL**
- Row 3 → **OO**

PORTCLOO

Decryption Steps:

HOW DO WE KNOW THE POSITIONS DURING DECRYPTION?

When decrypting Rail Fence Cipher, we recreate the zig-zag pattern using:

- **Ciphertext length (N)**
- **Key (number of rails)**

This zig-zag pattern tells us exactly where each letter belongs before reading the plaintext.

1. Count letters = 8
2. Determine the zig-zag pattern positions
3. Fill ciphertext row-wise

4. Read in zig-zag

STEP 1 — Make an empty zig-zag pattern

For the ciphertext **PORTCLOO**,
Length = **8**,
Key = **3 rails**

We first draw rails:

1 2 3 4 5 6 7 8

X				X			
	X		X		X		X
		X				X	

This is how we KNOW the positions.

We have not used any letters yet — only the key and length.

Count how many letters go in each rail

From the pattern:

- Rail 1 has **2 X's**
- Rail 2 has **4 X's**
- Rail 3 has **2 X's**

So ciphertext is split:

- Rail 1: **PO**
- Rail 2: **RTCL**
- Rail 3: **OO**

P				O				
	R		T		C		L	
		O				O		

Read in zig-zag order:

That gives:

PROTOCOL (plaintext)

2. COLUMNAR TRANSPOSITION CIPHER

We use the key: **NETWORK**

Step 1: Assign key order

Key:
N E T W O R K

Sort alphabetically:
E K N O R T W
1 2 3 4 5 6 7 ← numbers assigned

Mapping to original:

Letter	**N**	**E**	**T**	**W**	**O**	**R**	**K**
Order	3	1	6	7	4	5	2

Step 2: Write plaintext row-wise

Plaintext: PROTOCOL (8 letters)

We have 7 columns → need 2 rows
Pad with X for missing cells:

N	**E**	**T**	**W**	**O**	**R**	**K**
P	R	O	T	O	C	O
L	X	X	X	X	X	X

Step 3: Read columns in key order

Order: 1 → 7
i.e., E, K, N, O, R, T, W

- Column E → R X
- Column K → O X
- Column N → P L
- Column O → O X
- Column R → C X
- Column T → O X
- Column W → T X

Ciphertext

Combine all:

RXOXPLOXCXOXT

Decryption Steps

1. Write the key with column numbers
2. Create an empty table with 2 rows × 7 columns
3. Fill columns **in the order of key numbers**
4. Read row-wise

Reconstructed table:

N	**E**	**T**	**W**	**O**	**R**	**K**
P	R	O	T	O	C	O
L	X	X	X	X	X	X

Reading row-wise → **PROTOCOL**

Decrypted plaintext

PROTOCOL