Encryption Techniques

Encryption is a method of securing information by converting plaintext into an unreadable format (ciphertext). The two primary types of encryption techniques are **Substitution Ciphers** and **Transposition Ciphers**.

Encryption Process

Let's encrypt the word **HELLO** using a shift of **3**.

Step 1: Assign numerical values to letters

Each letter is assigned a numerical value based on its position in the English alphabet.

Letter	A	B	C	D	E	F	G	H	I	J
Number	0	1	2	3	4	5	6	7	8	9

Letter	N	O	P	Q	R	S	T	U	V	W
Number	13	14	15	16	17	18	19	20	21	22

The plaintext **HELLO** corresponds to:

- **H** → 7
- **E** → 4
- **L** → 11
- **L** → 11
- **O** → 14

Step 2: Apply the shift

Each letter's numerical value is shifted forward by **3** (modulo 26 to ensure it stays within the alphabet).

$$\text{Shifted Value} = (\text{Original Value} + \text{Shift}) \mod 26$$

Letter	Original Value	Shift (+3)	New Value
H	7	7 + 3 = 10	K
E	4	4 + 3 = 7	H
L	11	11 + 3 = 14	O
L	11	11 + 3 = 14	O
O	14	14 + 3 = 17	R

So, **HELLO** becomes **KHOOR**.

Decryption Process

To decrypt, we **reverse the shift** by subtracting **3**.

$$\text{Original Value} = (\text{Cipher Value} - \text{Shift}) \mod 26$$

Letter	Cipher Value	Shift (-3)	Original Value
K	10	10 - 3 = 7	H
H	7	7 - 3 = 4	E
O	14	14 - 3 = 11	L
O	14	14 - 3 = 11	L
R	17	17 - 3 = 14	O

So, **KHOOR** decrypts back to **HELLO**.

General Formula for Encryption & Decryption

For any letter **X**:

Encryption

$$C = (P + S) \mod 26$$

where:

- (C) = Ciphertext letter
- (P) = Plaintext letter (numerical value)
- (S) = Shift value

Decryption

$$P = (C - S) \mod 26$$

This process ensures that the encryption and decryption are reversible.

Example with a Different Shift

Let's encrypt **WORLD** with a shift of **5**.

1. **Convert to numbers**
   - W → 22
   - O → 14
   - R → 17
   - L → 11
   - D → 3
2. **Apply Shift (+5)**
   - 22 → 27 → **1** (B)
   - 14 → 19 → **T**
   - 17 → 22 → **W**
   - 11 → 16 → **Q**
   - 3 → 8 → **I**
3. **Ciphertext: BTWQI**

To decrypt, shift back by **5** and retrieve **WORLD**.

This step-by-step approach ensures clarity in applying the **Caesar Cipher**! 🚀

Monoalphabetic Cipher – Step-by-Step Explanation

Encryption Steps

1. **Choose a Random Substitution Alphabet**
   - Example Mapping:

   ```
   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
   ```

2. **Replace Each Letter in the Plaintext with Its Corresponding Cipher Letter**
   - **Plaintext:** HELLO
   - Mapping each letter using the table:
     - H → I
     - E → T
     - L → S
     - L → S
     - O → G
   - **Ciphertext: IWTTS**

Decryption Steps

1. **Use the Reverse Mapping to Retrieve Original Letters**
   - **Ciphertext:** IWTTS
   - Reverse lookup from the mapping:
     - I → H
     - T → E
     - S → L
     - S → L
     - G → O
   - **Plaintext: HELLO**

Security Considerations

- **Stronger than Caesar Cipher** since it doesn't follow a simple shift pattern.
- **Vulnerable to frequency analysis** since letter distributions in a language remain unchanged.

Vigenère Cipher – Step-by-Step Explanation

Encryption Steps

1. **Choose a Key Word**
   - Example: **LEMON**
   - If the key is shorter than the plaintext, repeat it until it matches the length of the plaintext.
2. **Write the Plaintext and Repeating Key**
   - **Plaintext:** A T T A C K
   - **Key:** L E M O N L
3. **Convert Letters to Numbers (A=0, B=1, ..., Z=25)**

Plaintext Letter	A	T	T	A	C	K
Key Letter	L	E	M	O	N	L
Plaintext Value	0	19	19	0	2	10
Key Value	11	4	12	14	13	11
New Value	11	23	5	14	15	21
Cipher Letter	L	X	F	O	P	V

1. **Final Ciphertext**: **LXFOPV**

Decryption Steps

1. **Use the Same Key to Reverse the Shift**
   - **Ciphertext:** L X F O P V
   - **Key:** L E M O N L
2. **Convert Back to Numbers and Subtract Key Values**

Ciphertext Letter	L	X	F	O	P	V
Key Letter	L	E	M	O	N	L
Cipher Value	11	23	5	14	15	21
Key Value	11	4	12	14	13	11
New Value	0	19	19	0	2	10

Ciphertext Letter	L	X	F	O	P	V
Plaintext	A	T	T	A	C	K

1. **Final Plaintext**: **ATTACK**

Security Considerations

✅ **More secure** than Caesar and Monoalphabetic ciphers since it uses multiple shifting values.
❌ **Still vulnerable** to frequency analysis if the key is short or reused frequently (Kasiski Examination).

Vernam Cipher (One-Time Pad) – Step-by-Step Explanation

Encryption Steps

1. **Convert Plaintext and Key to Numbers**
    - A = 0, B = 1, ..., Z = 25
    - **Plaintext:** HELLO
    - **Key:** XMCKL

Letter	H	E	L	L	O
Value	7	4	11	11	14
Key	X	M	C	K	L
Key Val	23	12	2	10	11

1. **Apply XOR (⊕) Modulo 26**
    - Formula:

$$C_i = (P_i + K_i) \mod 26$$

Plaintext	7	4	11	11	14
Key	23	12	2	10	11

Plaintext	7	4	11	11	14
Sum	30	16	13	21	25
Mod 26	4	16	13	21	25
Cipher	Z	E	B	B	W

1. **Final Ciphertext**: **ZEBBW**

Decryption Steps

1. **Use the Same Key to Reverse**
    - Formula:

$$P_i = (C_i - K_i + 26) \mod 26$$

Ciphertext	Z	E	B	B	W
Value	25	4	1	1	22
Key	X	M	C	K	L
Key Value	23	12	2	10	11
Diff	2	-8	-1	-9	11
Mod 26	7	4	11	11	14
Plaintext	H	E	L	L	O

1. **Final Plaintext**: **HELLO**

Security Considerations

✅ **Perfect security** if the key is:

- Truly random
- At least as long as the message
- Used only once

❌ **Vulnerabilities**:

- If the key is reused, the cipher can be broken using statistical analysis.
- Generating a truly random key and securely sharing it is challenging.

Playfair Cipher – Step-by-Step Explanation

Step 1: Construct the 5×5 Key Square

1. Write the **keyword** (without repeating letters).
   - Example Keyword: **MONARCHY**
   - Remove duplicates: **MONARCHY**
2. Fill in the remaining **letters of the alphabet** (excluding 'J', as it's usually merged with 'I').

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

Step 2: Divide Plaintext into Digraphs (Pairs of Two Letters)

- **Plaintext: HELLO**
- Split into **pairs:** HE LL OX (*X is added as padding for odd-length words*).

Step 3: Apply Playfair Rules for Encryption

Each digraph (pair) is encrypted using the following rules:

1. **Same Row:** Replace each letter with the **next** letter in the row.
2. **Same Column:** Replace each letter with the **below** letter in the column.
3. **Different Row and Column:** Form a **rectangle**, replacing letters with those at opposite corners.

Digraph	Rule Applied	Encrypted Pair
HE	Rectangle Rule → (H, E) → (G, C)	GC
LL	Same Row Rule → (L, L) → (P, P)	CN
OX	Rectangle Rule → (O, X) → (Z, B)	ZB

- **Ciphertext: GC CN ZB**

Decryption Process

1. Use the **same key square**.
2. Apply **reverse rules**:
   - **Same row:** Shift **left**.
   - **Same column:** Shift **up**.
   - **Rectangle rule:** Use the **opposite corners**.
3. Remove **padding (X)** if it was added.

Security Considerations

✅ **More secure than monoalphabetic ciphers** due to digraph encryption.
✅ **Resistant to frequency analysis** because letter pairs are encoded together.
❌ **Vulnerable to known plaintext attacks**.
❌ **Pattern recognition is still possible** with large ciphertexts.

Hill Cipher – Step-by-Step Explanation

Step 1: Convert Plaintext to Numbers

Each letter corresponds to a number (A = 0, B = 1, ..., Z = 25).

- **Plaintext: HI**
  - H → **7**, I → **8**
  - Represented as a column matrix:

$$P = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Step 2: Choose a Key Matrix

The **key matrix** must be invertible modulo 26. Example:

$$K = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$$

Step 3: Perform Matrix Multiplication

$$C = K \times P \mod 26$$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} (2 \times 7 + 3 \times 8) \\ (1 \times 7 + 4 \times 8) \end{bmatrix}$$

$$= \begin{bmatrix} 38 \\ 39 \end{bmatrix}$$

Applying modulo 26:

$$\begin{bmatrix} 38 \mod 26 \\ 39 \mod 26 \end{bmatrix} = \begin{bmatrix} 12 \\ 13 \end{bmatrix}$$

- **Mapped to Letters:**
  - 12 → **M**, 13 → **N**
- **Ciphertext: MN**

Decryption Process

1. Compute the **inverse of the key matrix** modulo 26.
2. Multiply the ciphertext vector by the inverse matrix.

3. Apply modulo 26.
4. Convert numbers back to letters.

Security Considerations

✅ **Stronger than substitution ciphers** since multiple letters are encrypted at once.
✅ **Harder to break using frequency analysis**.
❌ **Key matrix must be carefully chosen** (must have an inverse mod 26).
❌ **Vulnerable to known plaintext attacks** if the key matrix is small.

---

2) Transposition Techniques (Ciphers)

- Unlike substitution ciphers, transposition ciphers **rearrange the positions** of letters instead of replacing them.

Types of Transposition Ciphers

2.1) Columnar Transposition Cipher – Step-by-Step Explanation

Step 1: Write Plaintext in Rows

- **Plaintext:** `ATTACK AT DAWN`
- Remove spaces: `ATTACKATDAWN`
- **Key:** `3 1 4 5 2` (determines column order)
- Arrange in a grid (pad with `X` if needed):

3	1	4	5	2
A	T	T	A	C
K	A	T	D	A
W	N	X	X	X

Step 2: Read Column-wise by Key Order

- **Key Order: 3 → 1 → 4 → 5 → 2**
- Read the **third column first, then first, etc.**
    - Column **3**: `T T X`
    - Column **1**: `A K W`
    - Column **4**: `A D X`
    - Column **5**: `C A X`
    - Column **2**: `T A N`
- **Ciphertext:** `TTX AKW ADX CAX TAN`

Decryption Process

1. **Reconstruct the grid** using the key order.
2. **Read row-wise** to retrieve the original plaintext.

Security Considerations

✅ **Stronger than substitution ciphers**.
✅ **Easier to implement** with different key lengths.
❌ **Vulnerable to frequency analysis** if too short.
❌ **Can be broken with anagramming techniques**.

2.2) Double Columnar Transposition Cipher – Step-by-Step Explanation

Step 1: First Columnar Transposition

- **Plaintext:** `DEFEND THE EAST WALL`
- Remove spaces: `DEFENDTHEEASTWALL`
- **Key 1:** `3 1 4 5 2`
- Arrange in a grid:

3	1	4	5	2
D	E	F	E	N
D	T	H	E	E

3	1	4	5	2
A	S	T	W	A
L	L	X	X	X

- **Read Column-wise (Key Order: 3 → 1 → 4 → 5 → 2)**
  - Column **3**: `F H T X`
  - Column **1**: `D D A L`
  - Column **4**: `E E W X`
  - Column **5**: `N E A X`
  - Column **2**: `E T S L`
- **Intermediate Ciphertext:** `FHTX DDAL EEWX NEAX ETSL`

Step 2: Apply Second Columnar Transposition

- **Key 2:** `4 2 5 1 3`
- Arrange intermediate ciphertext into a new grid:

4	2	5	1	3
F	H	T	X	D
D	A	L	E	E
W	X	N	E	A
X	E	T	S	L

- **Read Column-wise (Key Order: 4 → 2 → 5 → 1 → 3)**
  - Column **4**: `X E S`
  - Column **2**: `H A X`
  - Column **5**: `T L A`
  - Column **1**: `F D W`
  - Column **3**: `D E N`
- **Final Ciphertext:** `XES HAX TLA FDW DEN`

Decryption Process

1. **Reverse the second transposition** (rearrange using Key 2).

2. **Reverse the first transposition** (rearrange using Key 1).
3. **Read row-wise** to reconstruct the original plaintext.

Security Considerations

✅ **Stronger than single columnar transposition**
✅ **Difficult to break using frequency analysis**
❌ **Still vulnerable to modern cryptanalysis if not used with other encryption**

2.3) Route Cipher

- The text is written in a **grid** and read in a **specific pattern** (e.g., spiral, zig-zag, diagonal, etc.).
- Commonly used for military-grade encryption in older times.

Example

- **Plaintext:** `DEFEND THE EAST WALL`
- Remove spaces: `DEFENDTHEEASTWALL`
- **Grid Size:** `4 × 5` (as close to square as possible)

Step 1: Arrange Text in Grid

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

Step 2: Read in a Spiral Order

1️⃣ Start from **top-left** and move **right** → `D E F E N`
2️⃣ Move **down** the last column → `E A X`
3️⃣ Move **left** across the bottom row → `X X L`
4️⃣ Move **up** the first column → `L A D`
5️⃣ Continue inward → `T S T H E W E`

Ciphertext: `D E F E N E A X X X L L A D T S T H E W E`

Decryption Process

1. Create the **same grid dimensions**.
2. Fill in characters following the **reverse spiral order**.
3. Read **row-wise** to recover the original message.

Variations of Route Cipher

- **Clockwise Spiral**
- **Zig-Zag Pattern**
- **Diagonal Reading**
- **Randomized Routes for Extra Security**

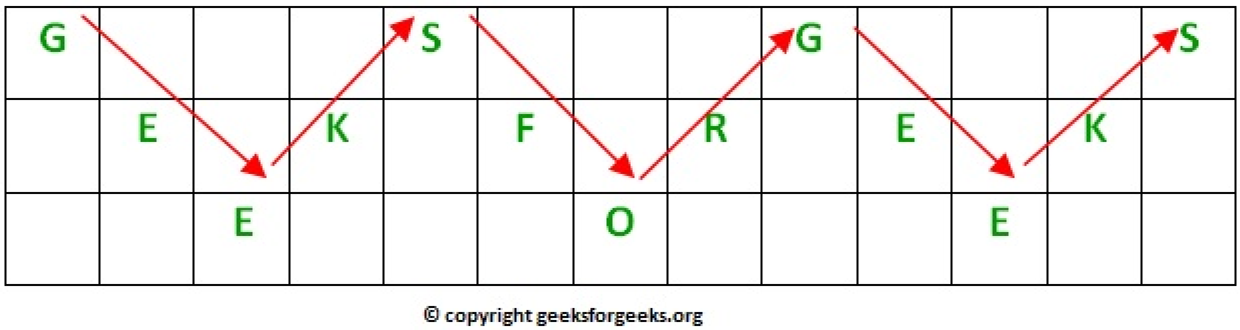
✅ **More secure than simple columnar ciphers**
❌ **If pattern is known, easy to break**

2.4) Rail Fence Cipher

- A **transposition cipher** where plaintext is written in a **zigzag** pattern across multiple rails (rows).
- The ciphertext is obtained by reading row-wise.

Example

- **Plaintext:** `GeeksforGeeks`
- **Rails (Depth):** `3`
- **Step 1: Arrange text in a zigzag pattern**



Padding with X because a letter is less

- **Step 2: Read row-wise**
  - **Row 1:** `GSGS`
  - **Row 2:** `EKFREK`
  - **Row 3:** `EOE`

Ciphertext: `GsGsekfrekeoe`

Decryption Process

1. Determine the **rail depth** and form an empty zigzag pattern.
2. Fill in the ciphertext **row-wise**.
3. Read **column-wise** to reconstruct the original message.

Given Information

- **Ciphertext:** `GsGsekfrekeoe`
- **Key (Rails):** 3

Length of Ciphertext: 13

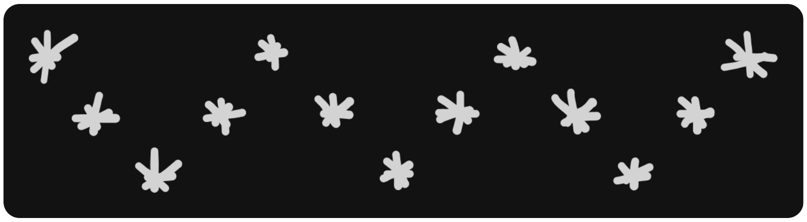
Step 1: Determine the matrix dimensions

We will create a **3 × 13 matrix** since:

- The number of **rows (rails)** is equal to the key, i.e., 3.
- The number of **columns** is equal to the length of the ciphertext, i.e., 13.

Step 2: Construct the empty zigzag matrix

The matrix will be filled diagonally, alternating between downward and upward directions. We will mark the places where characters will be placed using an asterisk (`*`).



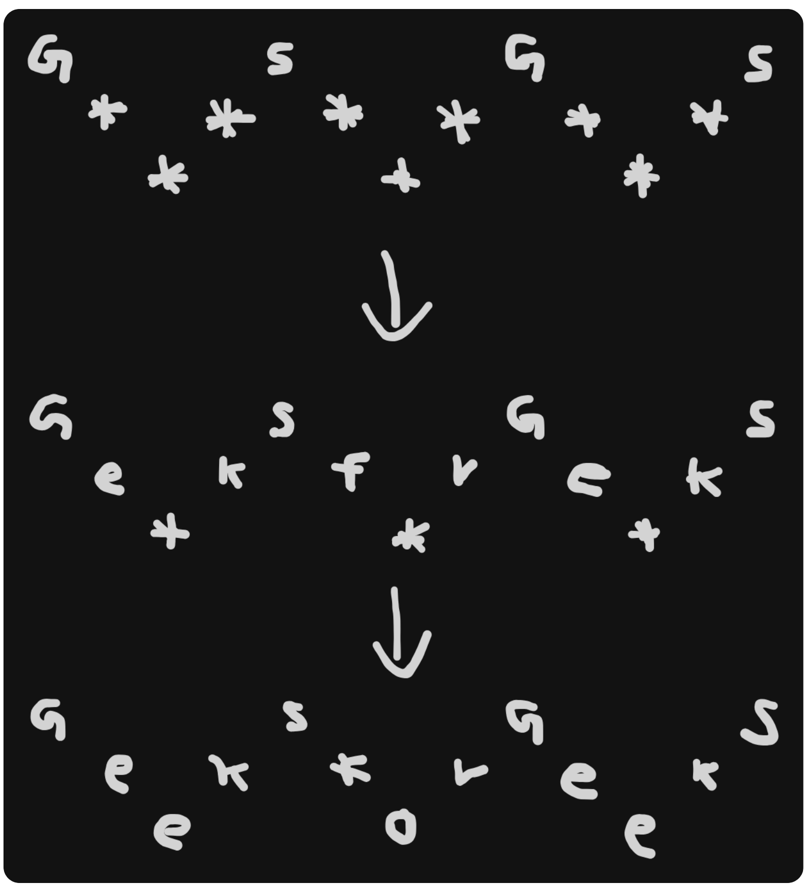
Initial empty matrix (using `*` for positions):

Step 3: Fill in the ciphertext

Now, we fill the matrix with the ciphertext `GsGsekfrekeoe`, starting from the row 1 and moving row by row, filling the positions marked by `*`.

Ciphertext: `GsGsekfrekeoe`

We fill the matrix in a row - wise



Step 4: Reconstruct the original message

Now that we've filled the matrix, we read the **columns** of the matrix to reconstruct the original text

---

Summary of Horizontal Decryption Process:

1. **Create an empty matrix** with **number of rows = key** and **number of columns = length of ciphertext**.
2. **Fill the matrix horizontally** with the ciphertext characters, starting from the top row and moving left to right.
3. **Read the matrix column-wise** to reconstruct the original message.

✅ **More secure than simple columnar transposition**
❌ **Easily breakable with rail-depth guessing**

2.5) Simple Transposition Cipher (Permutation Cipher)

A **Simple Transposition Cipher**, also known as a **Permutation Cipher**, is a type of cipher that rearranges the letters of the plaintext according to a fixed, predefined permutation pattern.

Key Concept:

- The basic idea is to permute or shuffle the positions of the letters in the plaintext based on a specified **key** or **permutation**.
- The **key** is a sequence of numbers that indicates the order in which the letters of the plaintext should be rearranged.

How It Works:

1. **Write the Plaintext:**
    - The first step is to write down the plaintext message that you want to encrypt.
2. **Apply the Permutation:**
    - Use a permutation (a predefined set of positions) to rearrange the letters.
    - Each number in the permutation tells you the new position for the corresponding letter in the plaintext.
3. **Generate the Ciphertext:**

    - The ciphertext is created by taking the letters from the plaintext and rearranging them according to the permutation.

Example:

Plaintext: `HELLO`

- This is the message we want to encrypt.

Permutation: `(4, 1, 3, 5, 2)`

- This is the key. It tells us how to rearrange the letters of the plaintext.
    - **4** means the 4th letter of the plaintext goes to the 1st position of the ciphertext.
    - **1** means the 1st letter of the plaintext goes to the 2nd position of the ciphertext.
    - **3** means the 3rd letter of the plaintext goes to the 3rd position of the ciphertext.
    - **5** means the 5th letter of the plaintext goes to the 4th position of the ciphertext.
    - **2** means the 2nd letter of the plaintext goes to the 5th position of the ciphertext.

Step 1: Write the Plaintext:

Plaintext: H E L L O

Step 2: Apply the Permutation:

- According to the permutation `(4, 1, 3, 5, 2)`, we rearrange the letters as follows:
    - The 4th letter of `HELLO` (which is `L`) moves to the 1st position.
    - The 1st letter of `HELLO` (which is `H`) moves to the 2nd position.
    - The 3rd letter of `HELLO` (which is `L`) stays in the 3rd position.
    - The 5th letter of `HELLO` (which is `O`) moves to the 4th position.
    - The 2nd letter of `HELLO` (which is `E`) moves to the 5th position.

Step 3: Generate the Ciphertext:

After applying the permutation, we get the ciphertext:
L E H L O

So, the ciphertext for the plaintext **"HELLO"** with the permutation **(4, 1, 3, 5, 2)** is **"LEHLO"**.

Advantages:

- Simple to implement.

- Doesn't require complex algorithms or computations.

Disadvantages:

- Not very secure by modern standards since the cipher is vulnerable to frequency analysis and other attacks.
- Security can be improved by using more complex permutations or combining with other ciphers.

Summary:

- A **Simple Transposition Cipher** rearranges the characters of a plaintext based on a fixed permutation.
- The key (permutation) tells you the new order of the letters.
- **Example:** With the permutation `(4, 1, 3, 5, 2)`, the plaintext "HELLO" becomes "LEHLO".

Comparison of Substitution and Transposition Ciphers

Feature	Substitution Cipher	Transposition Cipher
Method	Replaces letters	Rearranges letters
Security Level	Less secure	More secure
Example	Caesar Cipher	Rail Fence Cipher

Made By Yashank