**IMPLEMENTATION OF TRANSPOSITION TECHNIQUE**

**Objective**

To implement the transposition technique for encryption and decryption using a simple columnar transposition cipher.

**THEORY**

The transposition cipher is a classical cryptographic technique that rearranges the characters in the plaintext according to a specific rule or pattern. Unlike substitution ciphers (e.g., Caesar Cipher), which replace characters, transposition ciphers only change the order of characters.

**Columnar Transposition Cipher**

A common transposition method is the **columnar transposition cipher**, where the plaintext is written into a grid row-wise and read column-wise based on a predefined key.

**Steps for Columnar Transposition Cipher**

1. **Encryption**:
   * Write the plaintext into a grid row-wise based on a fixed column size (determined by a key).
   * Rearrange the columns according to the numerical order of the key.
   * Read the grid column-wise to obtain the ciphertext.
2. **Decryption**:
   * Construct an empty grid with the correct column order.
   * Fill in the ciphertext column-wise according to the key order.
   * Read row-wise to obtain the original plaintext.

**ALGORITHM**

**Encryption Process**

1. Convert the plaintext into uppercase (if required).
2. Determine the number of columns using the length of the key.
3. Arrange the text into a grid with the specified number of columns.
4. Rearrange the columns based on the key order.
5. Read column-wise to get the ciphertext.

**Decryption Process**

1. Calculate the grid size based on the ciphertext length and number of columns.
2. Fill the columns in the correct order according to the key.
3. Read row-wise to retrieve the original plaintext.

**INTERACTION WITH PROGRAM**

Encrypted Text: LOHOSNRLTEPTANISIO

Decrypted Text: HELLOTRANSPOSITION

**EXPLANATION OF OUTPUT**

1. **Encryption**:
   * The plaintext "HELLOTRANSPOSITION" is arranged in a grid with 4 columns.
   * The columns are rearranged as per the key "3214".
   * The encrypted text is read column-wise, resulting in "LOHOSNRLTEPTANISIO".
2. **Decryption**:
   * The ciphertext is placed back into the matrix using the column order.
   * The plaintext is reconstructed by reading row-wise, retrieving "HELLOTRANSPOSITION".

**CONCLUSION**

In this experiment, we implemented the transposition cipher technique using a columnar transposition method. We successfully encrypted and decrypted a given plaintext using a key-based column rearrangement approach. This technique ensures that the message remains concealed by altering the order of characters without changing them.

**REFERENCES**

1. William Stallings, *Cryptography and Network Security*, Pearson
2. Nina Godbole, *Information Systems Security*, Wiley

**CODE FOR COLUMNAR TRANSPOSITION CIPHER**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// Function to generate numeric key order

vector<int> getKeyOrder(string key) {

vector<pair<char, int>> keyOrder;

vector<int> order(key.length());

for (int i = 0; i < key.length(); i++) {

keyOrder.push\_back({key[i], i});

}

sort(keyOrder.begin(), keyOrder.end());

for (int i = 0; i < key.length(); i++) {

order[keyOrder[i].second] = i;

}

return order;

}

// Encryption function

string encrypt(string text, string key) {

int col = key.length();

int row = (text.length() + col - 1) / col; // Calculate required rows

vector<int> order = getKeyOrder(key);

// Fill matrix row-wise

vector<vector<char>> matrix(row, vector<char>(col, ' '));

int index = 0;

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

if (index < text.length()) {

matrix[i][j] = text[index++];

}

}

}

// Read column-wise based on key order

string ciphertext = "";

for (int i = 0; i < col; i++) {

int colIdx = find(order.begin(), order.end(), i) - order.begin();

for (int j = 0; j < row; j++) {

if (matrix[j][colIdx] != ' ')

ciphertext += matrix[j][colIdx];

}

}

return ciphertext;

}

// Decryption function

string decrypt(string ciphertext, string key) {

int col = key.length();

int row = (ciphertext.length() + col - 1) / col;

vector<int> order = getKeyOrder(key);

// Fill matrix column-wise

vector<vector<char>> matrix(row, vector<char>(col, ' '));

int index = 0;

for (int i = 0; i < col; i++) {

int colIdx = find(order.begin(), order.end(), i) - order.begin();

for (int j = 0; j < row; j++) {

if (index < ciphertext.length()) {

matrix[j][colIdx] = ciphertext[index++];

}

}

}

// Read row-wise to get plaintext

string plaintext = "";

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

if (matrix[i][j] != ' ')

plaintext += matrix[i][j];

}

}

return plaintext;

}

// Driver function

int main() {

string plaintext = "HELLOTRANSPOSITION";

string key = "3214"; // Key used for transposition

string encryptedText = encrypt(plaintext, key);

cout << "Encrypted Text: " << encryptedText << endl;

string decryptedText = decrypt(encryptedText, key);

cout << "Decrypted Text: " << decryptedText << endl;

return 0;

}**OUTPUT**

Encrypted Text: LOHOSNRLTEPTANISIO

Decrypted Text: HELLOTRANSPOSITION