Experiment No. 2

Title: Transposition Cipher

Batch: A3 Roll No.: 16010421059 Experiment No.:02

Aim: To implement transposition cipher – Row transposition and column transposition cipher.

Resources needed: Windows/Linux

Theory

Pre Lab/ Prior Concepts:

Symmetric-key algorithms are a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of cipher text. The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption. Symmetric-key encryption can use either stream ciphers or block ciphers. Transposition Cipher is block cipher. Ancient cryptographic systems are classified as: Substitution and Permutation/Transposition Ciphers.

Transposition Cipher/Permutation Cipher

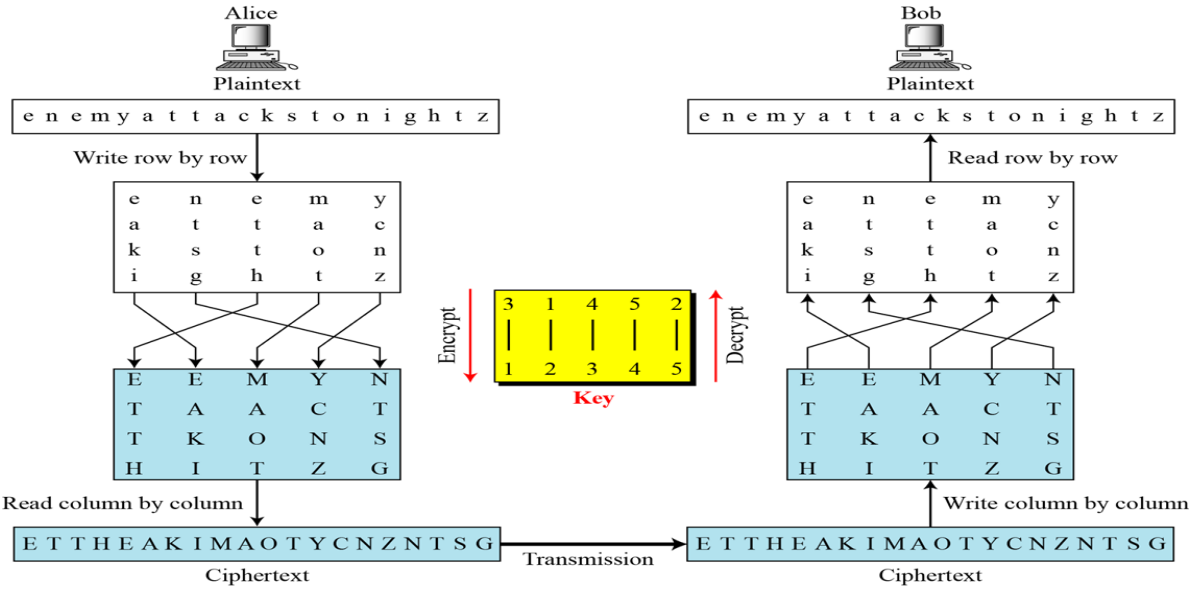
A transposition cipher rearranges (permutes) symbols in a block without altering actual values. It has the same frequency distribution as the original text .So it is easily recognizable.

EXAMPLE :

Plaintext: HELLO MY DEAR
Cipher text: ELHLMDOYAER

There are varieties of transposition ciphers like: keyless and keyed transposition ciphers.

Following figure shows the combination of both keyed and keyless. To encrypt with a transposition cipher, we first write the plaintext into a matrix of a given size and then permute the rows or columns according to specified permutations.



For the transposition, the key consists of the size of the matrix and the row or column permutations. The recipient who knows the key can simply put the cipher text into the appropriate sized matrix and undo the permutations to recover the plaintext.
Unlike a simple substitution, the transposition does nothing to disguise the letters that appear in the message But it does appear to thwart an attack that relies on the statistical information contained in the plaintext, since the plaintext statistics are disbursed throughout the cipher text. The double transposition is not a trivial cipher to break.

Activity :
Step 1 – Go through the theory explained and the instructions given by the instructor.

Step 2 – Derive/find encryption/decryption formula for each of the following transposition ciphers. Assume P as a plaintext square matrix, K as a row/column key and C as a ciphertext square matrix.

1) Row transposition cipher –

C[i][j] =

P[i][j] =

2) Column transposition cipher –

C[i][j] =

P[i][j] =

3) Double transposition cipher (row followed by column)

C[i][j] =

P[i][j] =

Step 3 – Implement

1) Row transposition cipher
2) Column transposition cipher
3) Double transposition cipher (row followed by column)

Implementation:
The program should have encryption function and decryption function for each cipher. Function should take message and a key as input from the user and display the expected output.

Results: (Program with output as per the format)
Input:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

string rowTranspositionEncrypt(const string &plaintext, const string &key) {
   int keyLength = key.length();
   int rows = (plaintext.length() + keyLength - 1) / keyLength;

   string ciphertext;
   for (char c : key) {
      int col = key.find(c);
      for (int row = 0; row < rows; ++row) {
         int index = row * keyLength + col;
         if (index < plaintext.length())
            ciphertext += plaintext[index];
         else
            ciphertext += ' ';
      }
   }

   return ciphertext;
}
```

```
string rowTranspositionDecrypt(const string &ciphertext, const string &key) {
    int keyLength = key.length();
    int rows = (ciphertext.length() + keyLength - 1) / keyLength;

    string plaintext(ciphertext.length(), ' ');
    int index = 0;

    for (char c : key) {
        int col = key.find(c);
        for (int row = 0; row < rows; ++row) {
            int matrixIndex = row * keyLength + col;
            if (index < ciphertext.length())
                plaintext[matrixIndex] = ciphertext[index++];
        }
    }

    return plaintext;
}

string columnTranspositionEncrypt(const string &plaintext, const string &key) {
    int keyLength = key.length();
    int columns = keyLength;
    int rows = (plaintext.length() + columns - 1) / columns;

    string ciphertext(plaintext.length(), ' ');

    int index = 0;
    for (int i = 0; i < columns; ++i) {
        int col = key.find(key[i]);
        for (int j = 0; j < rows; ++j) {
            int matrixIndex = j * columns + col;
            if (matrixIndex < plaintext.length())
                ciphertext[index++] = plaintext[matrixIndex];
        }
    }
```

```
    return ciphertext;
}

string columnTranspositionDecrypt(const string &ciphertext, const string &key) {
    int keyLength = key.length();
    int columns = keyLength;
    int rows = (ciphertext.length() + columns - 1) / columns;

    string plaintext(ciphertext.length(), ' ');

    int index = 0;
    for (int i = 0; i < columns; ++i) {
        int col = key.find(key[i]);
        for (int j = 0; j < rows; ++j) {
            int matrixIndex = j * columns + col;
            if (matrixIndex < ciphertext.length())
                plaintext[matrixIndex] = ciphertext[index++];
        }
    }

    return plaintext;
}

void displayMatrix(const string &text, const string &key) {
    int rows = (text.length() + key.length() - 1) / key.length();
    int cols = key.length();

    cout << "Matrix:" << endl;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            int index = i + j * rows;
            if (index < text.length())
                cout << text[index] << " ";
            else
                cout << "  ";
```

```
        }
        cout << endl;
    }
}

int main() {
    int c;
    string plaintext, key, encrypted, decrypted;

    cout << "Enter plaintext: ";
    getline(cin, plaintext);

    cout << "Enter key: ";
    cin >> key;

    cout << "1. Row Transposition Encryption\n2. Row Transposition Decryption\n3. Column
Transposition Encryption\n4. Column Transposition Decryption\n" << endl;
    cout << "Enter your choice (1/2/3/4): ";
    cin >> c;

    switch (c) {
        case 1:
            encrypted = rowTranspositionEncrypt(plaintext, key);
            displayMatrix(encrypted, key);
            cout << "\nEncrypted: " << encrypted << endl;
            break;
        case 2:
            decrypted = rowTranspositionDecrypt(plaintext, key);
            displayMatrix(decrypted, key);
            cout << "Decrypted: " << decrypted << endl;
            break;
        case 3:
            encrypted = columnTranspositionEncrypt(plaintext, key);
            displayMatrix(encrypted, key);
            cout << "\nEncrypted: " << encrypted << endl;
            break;
```

```
        case 4:
            decrypted = columnTranspositionDecrypt(plaintext, key);
            displayMatrix(decrypted, key);
            cout << "Decrypted: " << decrypted << endl;
            break;
        default:
            cout << "Invalid input" << endl;
            break;
    }

    return 0;
}
```

Output:

Output

```
/tmp/wljZOnN6R5.o
Enter plaintext: chinmay
Enter key: key
\1. Row Transposition Encryption
2. Row Transposition Decryption
3. Column Transposition Encryption
4. Column Transposition Decryption

Enter your choice (1/2/3/4): 1
Matrix:
c h i
n m a
y

Encrypted: cnyhm ia
```

Output

```
/tmp/wljZOnN6R5.o
Enter plaintext: cnyhm ia
Enter key: key
1. Row Transposition Encryption
2. Row Transposition Decryption
3. Column Transposition Encryption
4. Column Transposition Decryption

Enter your choice (1/2/3/4): 2
Matrix:
c n y
h m
i a
Decrypted: chinmay
```

Output

```
/tmp/wljZOnN6R5.o
Enter plaintext: chinmay
Enter key: key
1. Row Transposition Encryption
2. Row Transposition Decryption
3. Column Transposition Encryption
4. Column Transposition Decryption

Enter your choice (1/2/3/4): 3
Matrix:
c h a
n m
y i

Encrypted: cnyhmia
```

Output

```
/tmp/wljZOnN6R5.o
Enter plaintext: cnyhmia
Enter key: key
1. Row Transposition Encryption
2. Row Transposition Decryption
3. Column Transposition Encryption
4. Column Transposition Decryption

Enter your choice (1/2/3/4): 4
Matrix:
c n y
h m
i a
Decrypted: chinmay
```

Questions:

1) Substitution Ciphers vs. Transposition/Permutation Ciphers:

Substitution Ciphers:

- Substitution ciphers involve replacing each letter or character in the plaintext with another letter or character. The replacement is typically based on a predetermined rule or key.
- Examples of substitution ciphers include the Caesar cipher and the Atbash cipher.
- Substitution ciphers do not change the order of the characters; they only change the characters themselves.
- These ciphers provide confusion by making it difficult to discern the original message, but they do not offer diffusion as the original character positions remain the same.
- Substitution ciphers are relatively easy to break with modern cryptanalysis techniques, especially if the key space is small.

Transposition/Permutation Ciphers:
- Transposition ciphers involve rearranging the characters or blocks of characters in the plaintext without altering the characters themselves.
- Examples of transposition ciphers include the Rail Fence cipher and the Columnar Transposition cipher.
- Transposition ciphers provide diffusion by changing the order of characters, making the relationship between the original and encrypted text less obvious.
- However, they may not provide strong confusion, as the character identity remains the same, and patterns in the original message may still be discernible.
- Transposition ciphers can be more secure if used in conjunction with other encryption techniques.

2) Confusion and Diffusion Properties:

Confusion and diffusion are two fundamental principles in cryptography:

- Confusion: Confusion aims to obscure the relationship between the plaintext and the ciphertext. It is achieved by making it difficult to predict the ciphertext character(s) based on knowledge of the key or other ciphertext characters. In other words, confusion ensures that a change in the key or plaintext results in a significant change in the ciphertext. Substitution ciphers typically provide confusion because they replace characters in a non-trivial manner, making it hard to deduce the relationship between the original and encrypted text.

- Diffusion: Diffusion aims to spread the influence of each character in the plaintext across the entire ciphertext. This ensures that a change in one character of the plaintext affects many characters in the ciphertext. Transposition ciphers provide diffusion because they change the order of characters, causing a change in one part of the plaintext to affect a large portion of the ciphertext. This makes it challenging to detect patterns or redundancies in the ciphertext.

Both substitution and transposition ciphers provide some level of confusion and diffusion, but they excel in different aspects:

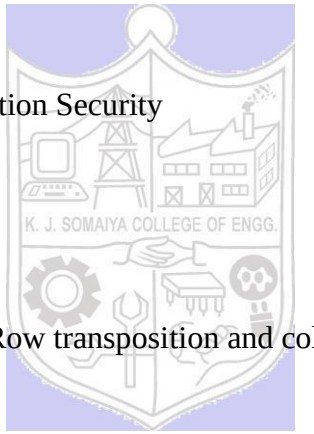
- Substitution ciphers primarily focus on confusion by replacing characters, but they may not provide strong diffusion as the character positions remain the same. This can make them vulnerable to frequency analysis and other cryptanalysis techniques.

- Transposition ciphers, on the other hand, excel at diffusion because they change the order of characters, but they may not provide strong confusion as the character identity remains the same. This means that patterns in the original message might still be recognizable, making them susceptible to certain attacks.

In practice, modern cryptographic systems use a combination of both substitution and transposition techniques, along with other cryptographic principles, to achieve a higher level of security, ensuring strong confusion and diffusion properties.

Outcomes:

CO1: Describe the basics of Information Security



Conclusion:

Implemented transposition cipher – Row transposition and column transposition cipher succesfully.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References: Books/ Journals/ Websites:

1. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
2. Mark Stamp, “Information Security Principles and Practice”, Wiley.
3. William Stalling, “Cryptography and Network Security”, Prentice Hall