Experiment No. 2

Title: Transposition Cipher

Batch: B-3 **Roll No.: 16010422234** **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 a block cipher. Ancient cryptographic systems are classified as: Substitution and Permutation/Transposition Ciphers.

Transposition Cipher/Permutation Cipher

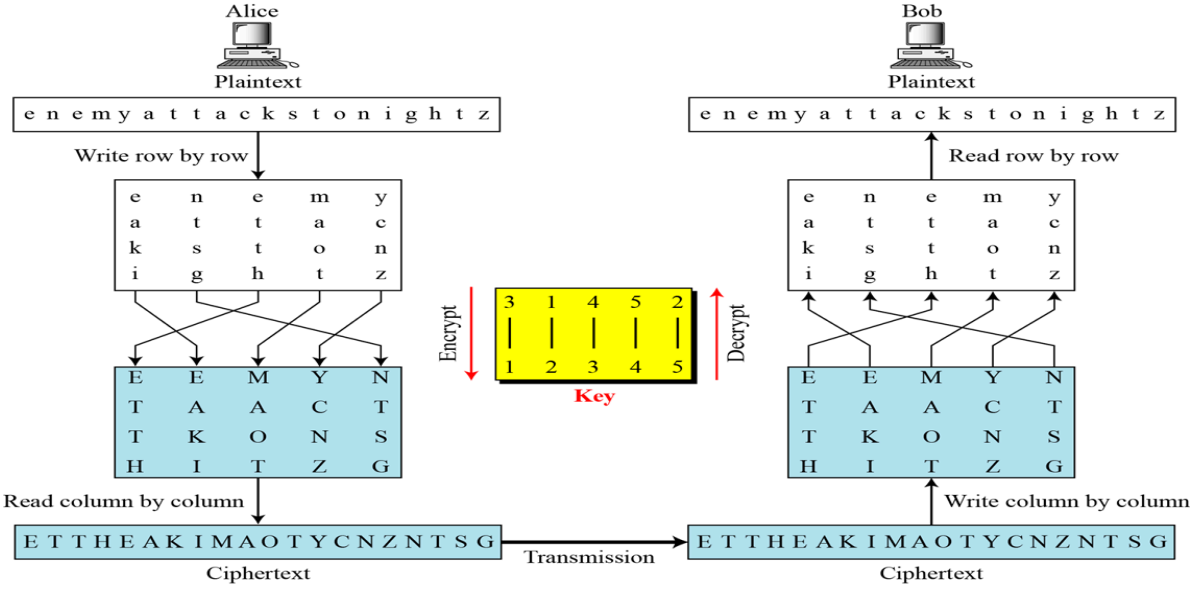
A transposition cipher rearranges (permutes) symbols in a block without altering actual values. It has the same frequency distribution as the original text .So it is easily recognizable.

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 ciphertext. 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 cipher. 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

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

Results: (Program with output as per the format)

Program:

```
def pad_text(text, size):
    return text + 'X' * (size - len(text) % size)

def create_matrix(text, rows, cols):
    return [text[i:i + cols] for i in range(0, len(text), cols)]

def permute(matrix, key):
    return [matrix[key.index(i + 1)] for i in range(len(key))]

def inverse_permute(matrix, key):
    inverse_key = sorted(range(len(key)), key=lambda k: key[k])
```

```
    return [matrix[inverse_key.index(i)] for i in range(len(key))]

def encrypt_row_transposition(plain_text, rows, cols, row_key):
    plain_text = pad_text(plain_text, rows * cols)
    matrix = create_matrix(plain_text, rows, cols)
    row_key = [int(k) for k in row_key]
    permuted_matrix = permute(matrix, row_key)
    transposed = ''.join([''.join(row) for row in permuted_matrix])
    return transposed

def decrypt_row_transposition(cipher_text, rows, cols, row_key):
    matrix = create_matrix(cipher_text, rows, cols)
    row_key = [int(k) for k in row_key]
    permuted_matrix = inverse_permute(matrix, row_key)
    transposed = ''.join([''.join(row) for row in permuted_matrix])
    return transposed[:rows * cols]

def encrypt_column_transposition(plain_text, rows, cols, col_key):
    if len(col_key) != cols:
            raise ValueError("Key length must match the number of
columns.")
    plain_text = pad_text(plain_text, rows * cols)
    matrix = create_matrix(plain_text, rows, cols)
    col_key = [int(k) for k in col_key]
       sorted_col_key = sorted(range(len(col_key)), key=lambda k:
col_key[k])
     transposed = ''.join(''.join(row[i] for i in sorted_col_key) for
row in matrix)
    return transposed

def decrypt_column_transposition(cipher_text, rows, cols, col_key):
    if len(col_key) != cols:
            raise ValueError("Key length must match the number of
columns.")
    num_of_rows = len(cipher_text) // cols
    matrix = create_matrix(cipher_text, num_of_rows, cols)
    col_key = [int(k) for k in col_key]
       sorted_col_key = sorted(range(len(col_key)), key=lambda k:
col_key[k])
     transposed = ''.join(''.join(row[sorted_col_key.index(i)] for i
in range(len(col_key))) for row in matrix)
    return transposed[:rows * cols]

def  double_transposition_encrypt(plain_text, rows, cols, row_key,
col_key):
```

```
    row_transposed = encrypt_row_transposition(plain_text, rows,
cols, row_key)
    double_transposed = encrypt_column_transposition(row_transposed,
rows, cols, col_key)
    return double_transposed

def double_transposition_decrypt(cipher_text, rows, cols, row_key,
col_key):
    col_transposed = decrypt_column_transposition(cipher_text, rows,
cols, col_key)
    double_transposed = decrypt_row_transposition(col_transposed,
rows, cols, row_key)
    return double_transposed

def menu():
    while True:
        print("\nMenu:")
        print("1. Encrypt Row Transposition Cipher")
        print("2. Decrypt Row Transposition Cipher")
        print("3. Encrypt Column Transposition Cipher")
        print("4. Decrypt Column Transposition Cipher")
        print("5. Encrypt Double Transposition Cipher")
        print("6. Decrypt Double Transposition Cipher")
        print("7. Exit")
        choice = int(input("Enter your choice: "))

        if choice == 7:
            break

        if choice in [1, 3, 5]:
            plain_text = input("Enter the plaintext: ").replace(" ",
"")
            rows = int(input("Enter the number of rows: "))
            cols = int(input("Enter the number of columns: "))

        if choice in [2, 4, 6]:
            cipher_text = input("Enter the cipher text: ")
            rows = int(input("Enter the number of rows: "))
            cols = int(input("Enter the number of columns: "))

        if choice == 1:
            row_key = input("Enter the row key: ")
                cipher_text = encrypt_row_transposition(plain_text,
rows, cols, row_key)
            print(f"Cipher Text: {cipher_text}")
```

```
        elif choice == 2:
            row_key = input("Enter the row key: ")
             decrypted_text = decrypt_row_transposition(cipher_text,
rows, cols, row_key)
            print(f"Decrypted Text: {decrypted_text}")
        elif choice == 3:
            col_key = input("Enter the column key: ")
              cipher_text = encrypt_column_transposition(plain_text,
rows, cols, col_key)
            print(f"Cipher Text: {cipher_text}")
        elif choice == 4:
            col_key = input("Enter the column key: ")
                                                  decrypted_text    =
decrypt_column_transposition(cipher_text, rows, cols, col_key)
            print(f"Decrypted Text: {decrypted_text}")
        elif choice == 5:
            row_key = input("Enter the row key: ")
            col_key = input("Enter the column key: ")
              cipher_text = double_transposition_encrypt(plain_text,
rows, cols, row_key, col_key)
            print(f"Cipher Text: {cipher_text}")
        elif choice == 6:
            row_key = input("Enter the row key: ")
            col_key = input("Enter the column key: ")
                                                  decrypted_text    =
double_transposition_decrypt(cipher_text,   rows,   cols,   row_key,
col_key)
            print(f"Decrypted Text: {decrypted_text}")
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    menu()
```

Output:

```
PS C:\Users\chand\Downloads\V SEM\INS\EXP2>

Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 1
Enter the plaintext: HELLO WORLD!
Enter the number of rows: 4
Enter the number of columns: 3
Enter the row key: 3142
Cipher Text: LOWD!XHELORL

Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 2
Enter the cipher text: LOWD!XHELORL
Enter the number of rows: 4
Enter the number of columns: 3
Enter the row key: 3142
Decrypted Text: HELLOWORLD!X
```

```
Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 3
Enter the plaintext: HELLO WORLD!
Enter the number of rows: 4
Enter the number of columns: 3
Enter the column key: 213
Cipher Text: EHLOLWROL!DX

Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 4
Enter the cipher text: EHLOLWROL!DX
Enter the number of rows: 4
Enter the number of columns: 3
Enter the column key: 213
Decrypted Text: HELLOWORLD!X
```

```
Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 5
Enter the plaintext: HELLO WORLD!
Enter the number of rows: 3
Enter the number of columns: 4
Enter the row key: 213
Enter the column key: 3142
Cipher Text: WROOELHLDXL!XXXXXXXXXXX

Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 6
Enter the cipher text: WROOELHLDXL!XXXXXXXXXXX
Enter the number of rows: 3
Enter the number of columns: 4
Enter the row key: 213
Enter the column key: 3142
Decrypted Text: HELLOWORLD!X
```

```
Menu:
1. Encrypt Row Transposition Cipher
2. Decrypt Row Transposition Cipher
3. Encrypt Column Transposition Cipher
4. Decrypt Column Transposition Cipher
5. Encrypt Double Transposition Cipher
6. Decrypt Double Transposition Cipher
7. Exit
Enter your choice: 7
PS C:\Users\chand\Downloads\V SEM\INS\EXP2>
```

Questions:

1. Compare substitution ciphers and transposition/permutation ciphers.

Ans: Substitution ciphers replace each letter of the plaintext with another letter or symbol, whereas transposition ciphers rearrange the positions of the letters without altering the actual letters.

2. Define confusion and diffusion properties. Comment on of both substitution and transposition ciphers w.r.t. confusion and diffusion properties.

Ans: Confusion: The relationship between the ciphertext and the key should be as complex as possible. Substitution ciphers provide confusion by substituting plaintext characters with other characters based on a key.

Diffusion: The influence of a plaintext character should be spread over many characters in the ciphertext. Transposition ciphers provide diffusion by permuting the characters of the plaintext based on a key.

Substitution ciphers focus more on confusion, while transposition ciphers emphasize diffusion. Combining both types of ciphers can enhance security.

Outcomes: Illustrate different cryptographic algorithms for security.

Conclusion:

In this experiment, the implementation of row and column transposition ciphers was successfully completed. The encryption and decryption processes for these ciphers were demonstrated, highlighting their respective roles in ensuring the security of communication.

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