Rashtriya Raksha University

School of Information Technology, Artificial Intelligence & Cyber Security (SITAICS)

At- Lavad, Dahegam, Gandhinagar, Gujarat-382305

Practical File

(Introduction to Cryptography)

Name: Sarthak Sanay
Enrollment No: 230031101611051
Subject Name: Introduction to Cryptography
Subject Code: G4A19ITC
Program: B.Tech CSE (with specialization in Cyber Security)
Year: 2nd year (Semester-IV)

This is certifying that Mr. Sarthak Sanay has satisfactorily completed all experiments in the practical work prescribed by SITAICS in the ITC laboratory.

Dr. Ashish Revar
SUBJECT INCHARGE

PRACTICAL - 6

AIM: TO IMPLEMENT RAILFENCE TRANSPOSITION CIPHER

BRIEF :-

The Rail Fence cipher is a simple transposition cipher that writes plaintext letters in a zig-zag pattern across a fixed number of "rails" (rows) and then reads them off row by row to form the ciphertext. By choosing a numeric depth (the key), the sender and receiver agree on how many rails to use; the plaintext is written down and up through the rails in sequence, creating a diagonal stripe. Once all letters are placed, the rows are concatenated in order to produce the encrypted message.

Decryption reverses this process by reconstructing the zig-zag matrix - marking the positions to be filled, and then refilling each rail with the appropriate ciphertext segment before reading the letters in their original zig-zag order. Though trivial to implement and useful for teaching the basics of transposition, the Rail Fence cipher offers minimal security by modern standards and can be broken easily by analyzing rail patterns.

ALGORITHM / PSEUDOCODE :-

```
repeat
print menu
read ch

if ch == 1 then
    read plain_text
    read depth
    text = remove spaces from plain_text
    // build rails
    rails = array of depth empty strings
    row = 0; dir = 1
    for each c in text do
        rails[row] += c
        if row == 0 then
            dir = 1
        else if row == depth - 1 then dir = -1
            row += dir
    end for
    cipher_text = join rails with spaces
    print cipher_text

else if ch == 2 then
    read cipher_text
    read depth
    rails_str = split cipher_text by spaces
    // reconstruct zig-zag
    length = total characters in rails_str
    mark zigzag positions in matrix[depth][length]
    fill matrix row by row from rails_str
    plain_text = read matrix in zigzag order
    print plain_text

else if ch == 0 then
    exit loop

else
    print "Invalid choice"
end if
until ch == 0
```

CODE :-

```
print("\nRail Fence Cipher Encryption & Decryption Tool:-")
ch = 1

while ch != 0:
    ch = int(input(
        "\nEnter 1 to Encrypt. \n"
        "Enter 2 to Decrypt. \n"
        "Enter 0 to Exit. \n"
        "Enter choice: "
    ))

    match ch:
        case 1:
            print("\nEncrypting Rail Fence Cipher!\n")
            plain_text = input("Enter plain text: ")
            depth = int(input("Enter depth: "))

            # remove all spaces
            text = plain_text.replace(" ", "")
            rails = [[] for _ in range(depth)]
            row, direction = 0, 1

            for char in text:
                rails[row].append(char)
                if row == 0:
                    direction = 1
                elif row == depth - 1:
                    direction = -1
                row += direction

            cipher_text = ""
            for rail in rails:
                for char in rail:
                    cipher_text += char
                cipher_text += " "

            print("Plain Text:  ", plain_text)
            print("Cipher Text: ", cipher_text, "\n")

        case 2:
            print("\nDecrypting Rail Fence Cipher!\n")
            cipher_text = input("Enter cipher text: ")
            depth = int(input("Enter depth: "))

            rails_str = cipher_text.split()
            length = sum(len(r) for r in rails_str)

            # build rail matrix
```

```
            matrix = [[''] * length for _ in range(depth)]
            row, direction = 0, 1
            for col in range(length):
                matrix[row][col] = '*'
                if row == 0:
                    direction = 1
                elif row == depth - 1:
                    direction = -1
                row += direction

            # fill characters
            index = 0
            for i in range(depth):
                for col in range(length):
                    if matrix[i][col] == '*' and index <
len(rails_str[i]):
                        matrix[i][col] = rails_str[i][index]
                        index += 1
                index = 0

            plain_text = ""
            row, direction = 0, 1
            for col in range(length):
                plain_text += matrix[row][col]
                if row == 0:
                    direction = 1
                elif row == depth - 1:
                    direction = -1
                row += direction

            print("Cipher Text: ", cipher_text)
            print("Plain Text:  ", plain_text, "\n")

        case 0:
            print("\nProgram exited successfully!")

        case _:
            print("\nEnter correct choice!\n")
```

OUTPUT :-

```
@sanaysarthak ➜/workspaces/crypto-lab/Rail Fence Algorithm (main) $ python rail-fence-cipher-full.py

Rail Fence Cipher Encryption & Decryption Tool:-

Enter 1 to Encrypt.
Enter 2 to Decrypt.
Enter 0 to Exit.
Enter choice: 1

Encrypting Rail Fence Cipher!

Enter plain text: BIG THINGS HAVE SMALL BEGINNINGS
Enter depth: 3
Plain Text:   BIG THINGS HAVE SMALL BEGINNINGS
Cipher Text:  BHSELGI ITIGHVSALEINNS GNAMBNG


Enter 1 to Encrypt.
Enter 2 to Decrypt.
Enter 0 to Exit.
Enter choice: 2

Decrypting Rail Fence Cipher!

Enter cipher text: BHSELGI ITIGHVSALEINNS GNAMBNG
Enter depth: 3
Cipher Text:  BHSELGI ITIGHVSALEINNS GNAMBNG
Plain Text:   BIGTHINGSHAVESMALLBEGINNINGS


Enter 1 to Encrypt.
Enter 2 to Decrypt.
Enter 0 to Exit.
Enter choice: 0

Program exited successfully!
```