PROJECT REPORT

ON

Demonstration of Rail Fence Cipher, Encryption and Decryption

Submitted to

NMAM INSTITUTE OF TECHNOLOGY, NITTE
(Off-Campus Centre, Nitte Deemed to be University, Nitte - 574 110, Karnataka, India)

In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

INFORMATION SCIENCE AND Engineering

By

Avyay Nayak Sujir	**NNM22IS029**
Anusha Sharma	**NNM22IS022**

Under the guidance of

Dr. Sumathi Pawar

Associate Professor Department of ISE

2024-2025

NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to certify that Mr. Avyay Nayak Sujir bearing USN NNM22IS029 and Ms. Anusha Sharma bearing USN NNM22IS022 of III-year B.Tech., a bonafide student of NMAM Institute of Technology, Nitte, has carried out project on "Demonstration of Rail Fence Cipher, Encryption and Decryption" as part of the **Information and Network Security (IS4001-1)** *course during 2024-25, fulfilling the partial requirements for the award of degree of Bachelor of Technology in Information Science and Engineering at NMAM Institute of Technology, Nitte.*

……………………………………..........
Signature of Course Instructor

Dr. Sumathi Pawar ,
Assistant Professor,
Department of ISE,
NMAMIT, NITTE (DU)

ABSTRACT

In the evolving landscape of information security, classical encryption techniques continue to serve as foundational concepts. The Rail Fence Cipher is a simple transposition cipher that rearranges the characters of a plaintext message using a zig-zag pattern based on a specified number of rails. This project explores the implementation of the Rail Fence Cipher for both encryption and decryption. The process demonstrates how text can be transformed to ensure a basic level of confidentiality and then reverted back to its original form. While the cipher lacks robustness against modern cryptographic attacks, it remains an effective tool for understanding the principles of transposition ciphers. The project includes a step-by-step explanation of the algorithm, Python-based implementation, and analysis of its effectiveness and limitations.

INTRODUCTION

Cryptography is the science of securing communication. One of the simplest and earliest techniques used is the **Transposition Cipher**, where characters are rearranged without changing them. A popular type of transposition cipher is the **Rail Fence Cipher**. This project demonstrates the encryption and decryption process using the Rail Fence Cipher technique.

OBJECTIVE

- To understand the working of Rail Fence Cipher.
- To implement encryption and decryption using the Rail Fence Cipher in a programming language.
- To analyze the advantages and limitations of this method.

METHODOLOGY:

The methodology involves implementing the Rail Fence Cipher for both encryption and decryption using a systematic, step-by-step approach. The key steps are outlined below:

1. Define Input

- Take the **plaintext** (original message) and the number of **rails** (levels of zig-zag).

2. Encryption Process

- Create a matrix with rows equal to the number of rails and columns equal to the length of the plaintext.
- Traverse the matrix in a zig-zag manner (downward and then upward) placing characters of the plaintext accordingly.
- After filling the matrix, read the characters row by row to generate the **ciphertext**.

3. Decryption Process

- Create an empty matrix of the same size used during encryption.
- Use a zig-zag pattern to **mark positions** where characters would be placed.
- Fill the matrix row by row using the ciphertext characters.
- Traverse the matrix again in a zig-zag pattern to retrieve the **original plaintext**.

4. Implementation

- The encryption and decryption algorithms are implemented using Python.
- The program accepts user input for plaintext and the number of rails.
- The output displays the encrypted and decrypted messages.

VISUALIZATION OF THE RAIL MATRIX WITH EXAMPLE:

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Rail 1	**H**		**L**		**O**		**O**		**L**	
Rail 2		**E**		**L**		**W**		**R**		**D**

Code Snippets:

Encryption:

```
def encrypt_rail_fence(text, key):
    if key <= 1:
        return text

    # Create the rail matrix
    rail = [['\n' for _ in range(len(text))] for _ in range(key)]

    direction_down = False
    row = 0

    # Populate the rail matrix in zig-zag manner
    for i in range(len(text)):
        if row == 0 or row == key - 1:
            direction_down = not direction_down

        rail[row][i] = text[i]

        row += 1 if direction_down else -1

    # Construct the result by reading the rail matrix
    result = []
    for i in range(key):
        for j in range(len(text)):
            if rail[i][j] != '\n':
                result.append(rail[i][j])

    return ''.join(result)
```

Decryption:

```
def decrypt_rail_fence(cipher, key):
    if key <= 1:
        return cipher

    # Create the rail matrix
    rail = [['\n' for _ in range(len(cipher))] for _ in range(key)]

    # Mark the places to be filled
    direction_down = None
    row, col = 0, 0

    for i in range(len(cipher)):
        if row == 0:
            direction_down = True
        if row == key - 1:
            direction_down = False

        rail[row][col] = '*'
        col += 1

        row += 1 if direction_down else -1

    # Fill the rail matrix with the cipher characters
    index = 0
    for i in range(key):
        for j in range(len(cipher)):
            if rail[i][j] == '*' and index < len(cipher):
                rail[i][j] = cipher[index]
                index += 1

    # Now read the matrix in zig-zag manner to construct the original message
    result = []
    row, col = 0, 0
    for i in range(len(cipher)):
        if row == 0:
            direction_down = True
        if row == key - 1:
            direction_down = False

        if rail[row][col] != '\n':
            result.append(rail[row][col])
            col += 1
        row += 1 if direction_down else -1

    return ''.join(result)
```

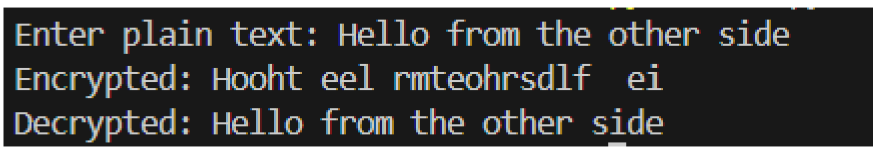
Main Function:

```
if __name__ == "__main__":
    text = input("Enter plain text: ")
    key = int(input("Enter key :" ))

    encrypted = encrypt_rail_fence(text, key)
    print("Encrypted:", encrypted)

    decrypted = decrypt_rail_fence(encrypted, key)
    print("Decrypted:", decrypted)
```

OUTPUT:



```
Enter plain text: Hello from the other side
Encrypted: Hooht eel rmteohrsdlf  ei
Decrypted: Hello from the other side
```

APPLICATIONS

- Teaching tool for transposition ciphers and classical cryptography
- Used in basic message obfuscation
- Common in puzzles, games, and CTF challenges
- Suitable for encoding secret or playful messages

CONCLUSION

The Rail Fence Cipher is a classic example of a transposition cipher that offers a simple yet effective way to understand the basics of encryption and decryption. Through the zig-zag arrangement of characters, it demonstrates how the position of characters can be manipulated to obscure a message. Although it lacks the complexity and security of modern cryptographic algorithms, its simplicity makes it ideal for educational purposes and basic applications. This project successfully implemented the encryption and decryption processes, highlighting both the working and limitations of the Rail Fence Cipher.