**INFORMATION SECURITY LAB**

**BCA-VI SEMESTER**

**LAB SHEET 2**

**Name: Tanmay Dangat**

**PRN NO: 20220801070**

**Objective**

To implement the **Rail Fence Cipher** algorithm in Python for **encryption and decryption** of a given plaintext using a specified depth

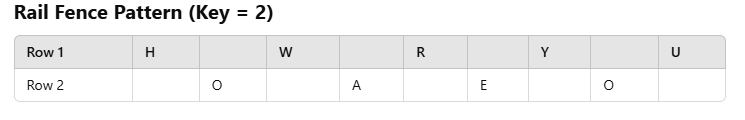
**Rail Fence Cipher**

The Rail Fence Cipher is a transposition cipher that writes the plaintext in a zigzag pattern along a number of "rails" and reads it row by row for encryption. To decrypt, the zigzag pattern is reconstructed to retrieve the original message

**Key Concept**

1. **Encryption**:
   * A zigzag pattern is formed using the number of rails (key).
   * Characters are written in this zigzag pattern.
   * The encrypted text is generated by reading the zigzag row by row.
2. **Decryption**:
   * A zigzag pattern is reconstructed by marking the positions of characters.
   * The cipher text is filled into these marked positions.
   * The zigzag is read row by row to retrieve the original message.

Here’s the table representation of the Rail Fence Cipher for **"HOW ARE YOU"** with **depth = 2**:



**Tasks**

1. Run the **Rail Fence Cipher** Python program in your IDE.

2. Test the encryption & decryption with different depth values.

3. Paste your Python code & output below with the date & time of execution

**Python Code:**

import datetime

def rail\_fence\_encrypt(text, depth):

    if depth <= 1:

        return text

    rail = [['\n' for \_ in range(len(text))] for \_ in range(depth)]

    direction\_down = False

    row, col = 0, 0

    for char in text:

        if row == 0 or row == depth - 1:

            direction\_down = not direction\_down

        rail[row][col] = char

        col += 1

        row += 1 if direction\_down else -1

    encrypted\_text = ''.join(char for row in rail for char in row if char != '\n')

    return encrypted\_text

def rail\_fence\_decrypt(cipher\_text, depth):

    if depth <= 1:

        return cipher\_text

    rail = [['\n' for \_ in range(len(cipher\_text))] for \_ in range(depth)]

    direction\_down = None

    row, col = 0, 0

    for \_ in range(len(cipher\_text)):

        if row == 0:

            direction\_down = True

        elif row == depth - 1:

            direction\_down = False

        rail[row][col] = '\*'

        col += 1

        row += 1 if direction\_down else -1

    index = 0

    for i in range(depth):

        for j in range(len(cipher\_text)):

            if rail[i][j] == '\*' and index < len(cipher\_text):

                rail[i][j] = cipher\_text[index]

                index += 1

    result = []

    row, col = 0, 0

    for \_ in range(len(cipher\_text)):

        if row == 0:

            direction\_down = True

        elif row == depth - 1:

            direction\_down = False

        result.append(rail[row][col])

        col += 1

        row += 1 if direction\_down else -1

    return ''.join(result)

# User Input

plaintext = input("Enter text to encrypt or decrypt: ")

depth = int(input("Enter depth (rails): "))

operation = input("Enter e for encrypt and d for decrypt: ").lower()

if (operation == 'e'):

    encrypt = True

    encrypted\_text = rail\_fence\_encrypt(plaintext, depth)

    print(f"Encrypted Text: {encrypted\_text}")

else:

    encrypt = False

    decrypted\_text = rail\_fence\_decrypt(plaintext, depth)

    print(f"Decrypted Text: {decrypted\_text}") print("Execution Date & Time:", datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")) A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Output:**

Enter text to encrypt or decrypt: Hello World!

Enter depth (rails): 4

Enter e for encrypt and d for decrypt: e

Encrypted Text: HWe o!lordll

Execution Date & Time: 2025-02-13 22:54:49

Enter text to encrypt or decrypt: HWe o!lordll

Enter depth (rails): 4

Enter e for encrypt and d for decrypt: d

Decrypted Text: Hello World!

Execution Date & Time: 2025-02-13 22:59:05