EXPERIMENT 01

Aim: Design and Implementation of a product cipher using Substitution and Transposition ciphers.

Theory :

Substitution Ciphers: These replace each letter in the plaintext with another letter or symbol according to a predetermined key. Examples include Caesar cipher, Atbash cipher, and the more complex polyalphabetic ciphers like the Vigenère cipher.

Transposition Ciphers: Instead of replacing characters, these ciphers rearrange the order of characters in the plaintext according to a specific rule. Examples include the Rail Fence cipher and Columnar Transposition cipher.

A product cipher combines multiple cryptographic techniques, such as substitution and transposition ciphers, to enhance security.

Below is a Python implementation of a product cipher that combines a substitution cipher (Caesar cipher) and a transposition cipher (Rail Fence cipher):

Programm:

```
def caesar_cipher_encrypt(text, shift):
    encrypted_text = ""

    for char in text:
        # Encrypt uppercase letters
        if char.isupper():
            encrypted_text += chr((ord(char) - 65 + shift) % 26 + 65)
        # Encrypt lowercase letters
        elif char.islower():
            encrypted_text += chr((ord(char) - 97 + shift) % 26 + 97)
        # Leave other characters unchanged
        else:
            encrypted_text += char

    return encrypted_text

def rail_fence_cipher_encrypt(text, rails):
    fence = [[] for _ in range(rails)]
    rail = 0
```

```
    direction = 1

    for char in text:
        fence[rail].append(char)
        rail += direction

        if rail == rails - 1 or rail == 0:
            direction *= -1

    encrypted_text = ""
    for rail in fence:
        encrypted_text += ''.join(rail)

    return encrypted_text

def product_cipher_encrypt(plaintext, caesar_shift, rail_fence_rails):
    # Step 1: Apply Caesar cipher encryption
    caesar_encrypted_text = caesar_cipher_encrypt(plaintext, caesar_shift)

    # Step 2: Apply Rail Fence cipher encryption
    product_cipher_text = rail_fence_cipher_encrypt(caesar_encrypted_text, rail_fence_rails)

    return product_cipher_text

def main():
    plaintext = input("Enter the plaintext to encrypt: ")
    caesar_shift = int(input("Enter the Caesar cipher shift value (positive integer): "))
    rail_fence_rails = int(input("Enter the number of rails for Rail Fence cipher (positive integer):
"))

    encrypted_text = product_cipher_encrypt(plaintext, caesar_shift, rail_fence_rails)
    print("Encrypted text:", encrypted_text)

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

Here's a brief overview of how the program works:

1. The 'caesar_cipher_encrypt' function encrypts the plaintext using the Caesar cipher with a specified shift value.

2. The 'rail_fence_cipher_encrypt' function encrypts the text using the Rail Fence cipher with a specified number of rails.

3. The 'product_cipher_encrypt' function applies both the Caesar cipher and the Rail Fence cipher to the plaintext in sequence.

4. The 'main' function prompts the user to enter the plaintext, Caesar cipher shift value, and the number of rails for the Rail Fence cipher.

5. It then calls the 'product_cipher_encrypt' function with the provided input and prints the encrypted text.

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR  COMMENTS

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\student\Desktop\41SaurabhPatil>  & 'c:\Python312\python.exe' 'c:\Users\student\.vscode\extensions\ms-python.debugpy-2024.2.0-win32-
:\Users\student\Desktop\41SaurabhPatil\41_CIPHER.py'
Enter the plaintext to encrypt: helloworld
Enter the Caesar cipher shift value (positive integer): 9
Enter the number of rails for Rail Fence cipher (positive integer): 6
Encrypted text: qnmuuuaxxf
PS C:\Users\student\Desktop\41SaurabhPatil>  c:; cd 'c:\Users\student\Desktop\41SaurabhPatil'; & 'c:\Python312\python.exe' 'c:\Users\student\.v
py\adapter/../..\debugpy\launcher' '50484' '--' 'c:\Users\student\Desktop\41SaurabhPatil\41_CIPHER.py'
Enter the plaintext to encrypt: computer
Enter the Caesar cipher shift value (positive integer): 5
Enter the number of rails for Rail Fence cipher (positive integer): 2
Encrypted text: hrzjtuyw
PS C:\Users\student\Desktop\41SaurabhPatil>
```

Conclusion:

Q. What is the benefit of implementing Substitution and Transposition ciphers together ?

Ans:

The implementation of a product cipher combining both substitution (Caesar cipher) and transposition (Rail Fence cipher) techniques provides an effective means of enhancing the security of plaintext data. By employing the Caesar cipher first to substitute characters based on a specified shift value, followed by the Rail Fence cipher to transpose the text across a configurable number of rails, the resulting encryption scheme offers increased complexity and resistance to cryptographic attacks.