BARI ANKIT (56)

Exp 1 : Design and Implementation of a product cipher using Substitution and Transposition ciphers.

Design and Implementation of a Product Cipher using Substitution and Transposition Ciphers

A **product cipher** is a cipher that combines multiple encryption techniques, in this case, **Substitution** and **Transposition** ciphers, to increase security. The main idea is to apply one cipher after the other. The first cipher (Substitution) transforms the plaintext, and then the second cipher (Transposition) permutes the output of the first cipher.

Steps for Design and Implementation:

1. **Substitution Cipher**:
    - This cipher replaces each letter of the plaintext with another letter according to a fixed substitution rule, often based on a key.
    - A typical example is the **Caesar Cipher** (shift cipher), but more advanced ciphers like **Vigenère Cipher** can be used as well.
2. **Transposition Cipher**:
    - This cipher rearranges the characters of the plaintext (or ciphertext from the Substitution Cipher) according to a specific pattern or key.
    - Common transposition ciphers include the **Rail Fence Cipher** and **Columnar Transposition**.

Design:

1. **Substitution Cipher**:
    - **Key**: A 26-letter key for simple substitution or a keyword for Vigenère Cipher.
    - The plaintext is mapped to ciphertext by shifting or substituting the letters according to the key.
2. **Transposition Cipher**:
    - **Key**: A number or a set of numbers that determines the pattern for rearranging the text.
    - The ciphertext from the Substitution Cipher is then rearranged based on the pattern defined by the transposition key.

Example of a Product Cipher Implementation:

Let's say we use:

1. **Caesar Cipher** for Substitution (shift by 3).
2. **Rail Fence Cipher** for Transposition (2 rails).

Implementation Steps:

1. **Step 1**: Implement a Caesar Cipher (Substitution Cipher).
2. **Step 2**: Implement a Rail Fence Cipher (Transposition Cipher).
3. **Step 3**: Combine the two ciphers in sequence, where the text is first encrypted using the Caesar Cipher and then passed through the Rail Fence Cipher.

Code :

```
# Caesar Cipher (Substitution)
def caesar_cipher_encrypt(plaintext, shift):
    result = []
    for char in plaintext:
        if char.isalpha():
            start = ord('A') if char.isupper() else ord('a')
            result.append(chr((ord(char) - start + shift) % 26 + start))
        else:
            result.append(char)
    return ''.join(result)

# Rail Fence Cipher (Transposition)
def rail_fence_encrypt(plaintext, rails):
    # Create empty 2D array for rails
    rail = [['\n' for _ in range(len(plaintext))] for _ in range(rails)]

    # Find direction to fill the rail
    row, col, down = 0, 0, True
    for char in plaintext:
        rail[row][col] = char
        col += 1
        if down:
            row += 1
            if row == rails:
                down = False
                row -= 2
        else:
            row -= 1
```

```
            if row == -1:
                down = True
                row += 2

    # Construct ciphertext by reading rails
    ciphertext = ''.join([rail[i][j] for i in range(rails) for j in range(len(plaintext)) if rail[i][j] != '\n'])
    return ciphertext

# Product Cipher: Combine Caesar and Rail Fence
def product_cipher_encrypt(plaintext, shift, rails):
    # Step 1: Apply Caesar Cipher
    substituted_text = caesar_cipher_encrypt(plaintext, shift)
    print(f"Substituted text (Caesar Cipher): {substituted_text}")

    # Step 2: Apply Rail Fence Cipher
    ciphertext = rail_fence_encrypt(substituted_text, rails)
    print(f"Final ciphertext (Rail Fence Cipher): {ciphertext}")

    return ciphertext
```

Output :

```
plaintext = "HELLO WORLD"
shift = 3  # Caesar Cipher shift value
rails = 2  # Rail Fence Cipher rails value

ciphertext = product_cipher_encrypt(plaintext, shift, rails)
```