Experiment no.1: Design and implement of a product cipher using Substitution and Transposition Cipher

Learning Objective: Student should be able to design and implementation of a product cipher using Substitution and Transposition Cipher.

Tools: C/C++/Java/Python or any computational software

Theory:

A substitution cipher is a method of encoding by which units of plaintext are replaced with ciphertext, according to a fixed system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing the inverse substitution.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polyalphabetic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa.

The function for Additive/Shift/Generalized Caesar Cipher is given as follows:

- It can use any shift from 1 to 25, i.e., replace each letter by a letter a fixed distance away.
- Ci=E(Pi)=(Pi+k) mod 26 and Pi=D(Ci)=(Ci-k) mod 26.

In Cryptography, a Caesar Cipher, also known as Caesar's Cipher, the Shift Cipher, Caesar's Code or Caesar Shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, A would be replaced by D, E would become H, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

The function for Caesar Cipher is defined as follows:

- Ci= E(Pi)=(Pi+3) mod 26.

- Pi=D(Ci)=(Ci-3) mod 26.
- Example:
- Plain Text: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Cipher Text: DEFGHIJKLMNOPQRSTUVWXYZABC

Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

A transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed (the plaintext is reordered). Mathematically a bijective function is used on the characters' positions to encrypt and an inverse function to decrypt.

Transposition Ciphers does not substitute one symbol for another, instead it changes the location of the symbols.

A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eight position in the plaintext may appear in the first position of the ciphertext. **A transposition cipher reorders (transposes) the symbols.** Simple transposition ciphers, which were used in the past, are keyless.

There are two methods for permutation of characters. In the first method, the text is written into a table column by column and then transmitted row by row. In the second method, the text is written into a table row by row and then transmitted column by column.

Caesar Cipher :

Program-

```
import random

def caesar_cipher(text, key, mode='encrypt'):
    result = ""
    process_steps = []

    for char in text:
        if char.isalpha():
            is_upper = char.isupper()

            # Convert the character to a number where A=0, B=1, ..., Z=25
            char_index = ord(char) - ord('A') if is_upper else ord(char) - ord('a')

            # Shift the number by the specified key
            shifted_index = (char_index + key) % 26 if mode == 'encrypt' else (char_index - key) % 26

            # Convert the shifted number back to a character
            shifted_char = chr(shifted_index + ord('A')) if is_upper else chr(shifted_index + ord('a'))

            result += shifted_char
            process_steps.append((char, char_index, shifted_char))
        else:
            result += char

    return result, process_steps

# Get user input
plaintext = input("Enter the text to process: ")

# Generate a random key within the range of 0 to 25
key = random.randint(0, 25)

# Encrypt the user input and show the encryption process
encrypted_text, encryption_steps = caesar_cipher(plaintext, key, mode='encrypt')
print(f"Original Text: {plaintext}")
print(f"Generated Shift (Key) for Encryption: {key}")
print(f"Encryption Process:")
for step in encryption_steps:
    print(f"{step[0]} ({step[1]}) -> {step[2]}")
print(f"Encrypted Text: {encrypted_text}")

# Decrypt the encrypted text and show the decryption process
decrypted_text, decryption_steps = caesar_cipher(encrypted_text, key, mode='decrypt')
print("\nDecryption Process:")
for step in decryption_steps:
    print(f"{step[2]} ({step[1]}) -> {step[0]}")
print(f"Shift (Key) for Decryption: {key}")
print(f"Decrypted Text: {decrypted_text}")
```

Output:

```
Enter the text to process: pratham
Original Text: pratham
Generated Shift (Key) for Encryption: 19
Encryption Process:
p (15) -> i
r (17) -> k
a (0) -> t
t (19) -> m
h (7) -> a
a (0) -> t
m (12) -> f
Encrypted Text: iktmatf

Decryption Process:
p (8) -> i
r (10) -> k
a (19) -> t
t (12) -> m
h (0) -> a
a (19) -> t
m (5) -> f
Shift (Key) for Decryption: 1
Decrypted Text: pratham
```

Rail Fence Cipher :

Program-

```
import random

def transpose_cipher(text, num_rows):
    clean_text = ''.join(text.split()).upper()
    length = len(clean_text)
    num_columns = (length + num_rows - 1) // num_rows
    padded_text = clean_text.ljust(num_rows * num_columns)
    matrix = [list(padded_text[i:i+num_columns]) for i in range(0, len(padded_text), num_columns)]

    # Rearrange letters randomly in each column
    for col_index in range(num_columns):
        col = [row[col_index] for row in matrix]
        random.shuffle(col)
        for row_index in range(num_rows):
            matrix[row_index][col_index] = col[row_index]

    # Display the matrix
    print("\nMatrix:")
    for row in matrix:
        print(' '.join(row))

    encrypted_text = ''.join([''.join(row) for row in matrix])
    return encrypted_text

# Get user input
plaintext = input("Enter the text to encrypt: ")
num_rows = int(input("Enter the number of rows: "))

# Encrypt the user input
encrypted_text = transpose_cipher(plaintext, num_rows)

# Display the result
print(f"\nEncrypted Text: {encrypted_text}")
```

Output:

Enter the text to encrypt: pratham

Enter the number of rows: 4
Matrix:
M
H A
A R
P T

Encrypted Text: M HAARPT

Applications:

1. Ciphers are most commonly used in secure online communications to prevent unauthorized access. They're also incorporated into many different network protocols such as 'Secure Sockets Layer', 'TLS', HTTPS, etc.
2. In daily life we see applications of cryptography in ATM services, emails, etc

Result and Discussion:

In vignered substitution cipher it's possible to encrypt more than one characters to a common character and is dependent on the keystream used. The keystream is used repeatedly until all the plaintext characters are encrypted. The same keystream is necessary in order to decrypt the ciphertext back to original plain text. In keyed transposition cipher the keys used are actually mappers used to map each character in the plaintext to a specific index less than the length of plaintext. Original plaintext is divided in chunks of length equal to the number of keys which are then encrypted using the keys. If the last chunk doesn't have length equal to the number of keys then we add some bogus characters to the last chunk until desired length is obtained and encrypt the message.

Learning Outcomes: The student should have the ability to design & implement product cipher using Substitution and Transposition Cipher

LO1: To describe & understand about Substitution and Transposition cipher techniques

LO2: To implement Substitution and Transposition cipher techniques

Course Outcomes: Upon completion of the course students will be able to understand & implement Substitution and Transposition Cipher.

Conclusion:

In this experiment, we implemented a substitution cipher called 'Caesar Cipher' and a transposition cipher called 'Keyed Transposition cipher' both of which included the use of anumber of keys for encryption instead of standard one key which enhances the security and makes it extremely difficult to decrypt the ciphertext without the knowledge of keys. The complexity of blind decryption increases with the increase in number of keys used.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				