Vivekanand Education Society's Institute of Technology

Department of AI&DS Engineering

Subject: Cryptography and System Security

Class: D11AD

Roll No: 46	**Name: Ashish Patil**
Practical No:1	**Title:** Basics of cryptography
DOP:	**DOS:**
Grades:	**LOs Mapped:**
Signature:	

Title: Basics of Cryptography
DOP: 15/01/24
DOS: /01/24

Aim: To implement Substitution and Transposition cipher.

Theory: 1. Substitution Cipher:
A substitution cipher is a method of encrypting plaintext by replacing each letter with another letter or symbol according to a fixed system. In a substitution cipher, the letters in the plaintext are systematically replaced with other letters or symbols. The most well-known example of a substitution cipher is the Caesar cipher.

2. Caesar Cipher:
The Caesar cipher 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 shifted a certain number of places down or up the alphabet. For example, with a shift of 3, 'A' would be replaced by 'D', 'B' would become 'E', and so on. The method is named after Julius Caesar, who is reported to have used it to communicate with his generals.

Example:
Suppose we want to encrypt the message "HELLO" using a Caesar cipher with a shift of 3.

- Plaintext: H E L L O
- Encrypted: K H O O R

Explanation:
- 'H' is shifted 3 positions to the right, becoming 'K'.
- 'E' is shifted 3 positions to the right, becoming 'H'.
- 'L' is shifted 3 positions to the right, becoming 'O'.
- 'O' is shifted 3 positions to the right, becoming 'R'.

So, the encrypted message is "KHOOR".

3. Transposition Cipher:
A transposition cipher is a method of encryption where the positions of the characters in the plaintext are systematically rearranged according to a specific scheme, but the characters themselves are not changed. Unlike substitution ciphers, which replace each letter with another letter or symbol, transposition ciphers only change the order of the characters.

4. Columnar Transposition (with key):
Columnar transposition is a type of transposition cipher where the plaintext is written out in rows of a fixed length, and then arranged into a rectangular matrix. The columns of this matrix are then rearranged according to a permutation of numbers, which serves as the key. The ciphertext is formed by reading the columns of the matrix in the order specified by the key.

Example:
Suppose we want to encrypt the message "HELLO" using columnar transposition with the key "321".

- Plaintext: H E L L O
- Matrix:
  ```
  3 2 1
  H E L
  L O X
  ```

Explanation:
- We arrange the plaintext "HELLO" in rows of length 3, and pad with an extra character 'X' to fill the last row.
- The key "321" indicates the order of the columns.
- Reading the columns in the order specified by the key, we get the ciphertext: "LXHEOL".

So, the encrypted message is "LXHEOL".

Program: Caesar Cipher

```
Click here to ask Blackbox to help you code faster
def caesar_cipher(text, shift, mode):
    """
    Function to perform Caesar cipher encryption or decryption.

    Parameters:
    text (str): The text to be encrypted or decrypted.
    shift (int): The number of positions to shift the characters in the alphabet.
    mode (str): 'encrypt' to perform encryption, 'decrypt' to perform decryption.

    Returns:
    str: The encrypted or decrypted text.
    """
    result = ''
    for char in text:
        if char.isalpha():
            # Determine the position of the character in the alphabet
            ascii_offset = 65 if char.isupper() else 97
            char_index = ord(char) - ascii_offset

            # Apply the shift based on the mode
            if mode == 'encrypt':
                shifted_index = (char_index + shift) % 26
            elif mode == 'decrypt':
                shifted_index = (char_index - shift) % 26

            # Convert back to a character
            shifted_char = chr(shifted_index + ascii_offset)
            result += shifted_char
        else:
            # If the character is not an alphabet character, just append it unchanged
            result += char
    return result

def main():
    print("Welcome to Caesar Cipher Program")
    mode = input("Do you want to encrypt or decrypt? (Type 'encrypt' or 'decrypt'): ").lower()

    if mode == 'encrypt' or mode == 'decrypt':
        text = input("Enter the text: ")
        shift = int(input("Enter the shift value (a positive integer): "))
        if mode == 'encrypt':
            encrypted_text = caesar_cipher(text, shift, 'encrypt')
```

Output:

```
Welcome to Caesar Cipher Program
Do you want to encrypt or decrypt? (Type 'encrypt' or 'decrypt'): encrypt
Enter the text: this is the attack place
Enter the shift value (a positive integer): 56
```

Program : columnar Transposition with key

```python
import math

def generate_key_order(key):
    """
    Generate the order of columns based on the key.

    Parameters:
    key (str): The key for columnar transposition.

    Returns:
    list: A list representing the order of columns based on the key.
    """
    # Create a sorted version of the key
    sorted_key = sorted(key)
    key_order = []
    for char in key:
        # Find the index of the character in the sorted key
        index = sorted_key.index(char)
        # Store the original index of the character
        key_order.append(index)
        # Replace the character with None to handle duplicates
        sorted_key[index] = None
    return key_order

def encrypt_columnar_transposition(text, key):
    """
    Encrypt the text using columnar transposition cipher with a given key.

    Parameters:
    text (str): The text to be encrypted.
    key (str): The key for columnar transposition.

    Returns:
    str: The encrypted text.
    """
    key_order = generate_key_order(key)
    num_columns = len(key)
    num_rows = int(math.ceil(len(text) / num_columns))
    # Fill the incomplete columns with spaces
    padded_text = text.ljust(num_columns * num_rows)

    # Create the matrix for transposition
    matrix = [list(padded_text[i:i+num_columns]) for i in range(0, len(padded_text), num_columns)]
```

```python
    Parameters:
    text (str): The text to be decrypted.
    key (str): The key for columnar transposition.

    Returns:
    str: The decrypted text.
    """
    key_order = generate_key_order(key)
    num_columns = len(key)
    num_rows = int(math.ceil(len(text) / num_columns))

    # Determine the number of characters to remove from the end
    padding = num_columns * num_rows - len(text)
    if padding > 0:
        padding_char = ' ' if text[-1] == ' ' else 'X'
        text = text[:-padding]

    # Create the matrix for transposition
    matrix = [list(text[i:i+num_rows]) for i in range(0, len(text), num_rows)]

    # Perform columnar transposition decryption
    decrypted_text = ''
    for row in range(num_rows):
        for col in range(num_columns):
            decrypted_text += matrix[row][key_order.index(col)]
    return decrypted_text.rstrip()

f main():
    print("Welcome to Columnar Transposition Cipher Program")S

    while True:
        choice = input("Do you want to encrypt or decrypt? (Type 'encrypt' or 'decrypt' or 'exit'): ").lower()

        if choice == 'encrypt':
            plaintext = input("Enter the text to encrypt: ").replace(" ", "").upper()
            key = input("Enter the key: ").upper()
            encrypted_text = encrypt_columnar_transposition(plaintext, key)
            print("Encrypted:", encrypted_text)
        elif choice == 'decrypt':
            ciphertext = input("Enter the text to decrypt: ").replace(" ", "").upper()
            key = input("Enter the key: ").upper()
```

Output:

```
Welcome to Columnar Transposition Cipher Program
Do you want to encrypt or decrypt? (Type 'encrypt' or 'decrypt' or 'ex
Enter the text to encrypt: this is the day
Enter the key: 123456
```

Conclusion: We have successfully implemented two ciphers , ceaser cipher and keyed columnar transposition.