ST. FRANCIS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

SECURITY LAB (SL)

Experiment – 1: Implementation of Shift (Caesar/Additive) Cipher

Aim: Write a program to implement Shift Cipher Technique and understand cryptanalysis of the same.

Objective: After performing the experiment, the students will be able to –

- To understand the encryption and decryption fundamentals.
- To understand that secure encryption is not possible with small key space.

Lab objective mapped: L502.1: Students should be able to apply the knowledge of symmetric cryptography to implement simple ciphers.

Prerequisite: Basic knowledge of cryptography.

Requirements: PYTHON

Pre-Experiment Theory:

1. **Caesar Cipher**: In cryptography, a Caesar cipher, also known as a 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 shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals.
   E.g Plaintext: "Welcome to third year" when encrypted using Caesar cipher will give Ciphertext "ZHOFRPH WR WKLUG BHDU".

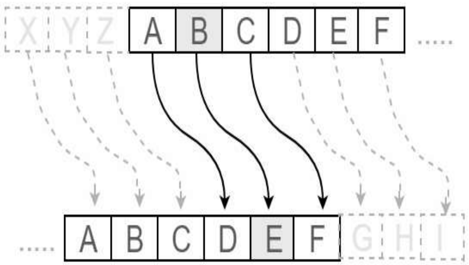
2. **Mathematical Description**

   First we translate all of our characters to numbers, 'a'=0, 'b'=1, 'c'=2, ..., 'z'=25. We can now represent the caesar cipher encryption function, e(x), where x is the character we are encrypting, as:

$$e(x) = (x + k) \pmod{26}$$

   Where, k is the key (the shift) applied to each letter. After applying this function, the result is a number which must then be translated back into a letter.
   The decryption function is:

$$e(x) = (x - k) \pmod{26}$$



- **Breaking of Caesar cipher:**

  With a Caesar cipher, there are only 26 possible keys, of which only 25 are of any use since mapping A to A etc. doesn't really encrypt the message. The hacker can try each of the keys (shifts) in turn, until he recognizes the original message.
  Note: The hacker needs to be able to recognize when he gets an original message (i.e. is in English or other language). This is usually easy for humans, but hard for computers. Cryptanalysis using shift cipher is much harder with compressed data.

  Example "GCUA VQ DTGCM" when broken gives "easy to break", with a shift of 2.

Output:

1. Attach complete program with comments performing encryption and decryption of shift cipher.
2. Attach screenshots of program output (for encryption & decryption) and its validation using a virtual lab tool.
3. Attach screenshots of examples from post experiment exercises.

Post Experimental Exercise- *(to be handwritten on ruled journal sheets)*

1. Explain substitution cipher technique (Ceasar) with an example ***[theoretical result and code output attached should match]***.
2. **Solve the following manually as well as using TOOL (in the references) given** (attach screenshots)
   a. Encrypt the following plain text using key k = 7.
      Plain Text: Lord Rama was a good king.
   b. Given a cipher text, find out the corresponding plain text using brute force attack.
      Cipher text: HAAHJR HA KHDU

Conclusion:

In this experiment we learned the basic features of Shift Cipher by implementing a code for encryption and decryption. We also observed the decryption when the key is known and understood, breaking the cipher when key space is very small by performing cryptanalysis of ciphertext.

References:

Mention your references here.

1. Virtual Lab Tool: https://cse29-iiith.vlabs.ac.in/
2. *https://www.youtube.com/watch?v=1Ri7t7VIQJA*
3. *(Add your references)*

Name: Vishal Rajesh Mahajan SL EXP 1
Class: TE IT A Roll No: 62

In Lab Exercise (Implementation of Shift (Caesar/Additive) Cipher):

```
def plaintocipher(Plain, key):
    """
    Converts plaintext to ciphertext using the Caesar cipher method.

    Parameters:
    Plain (str): The plaintext message to be encrypted.
    key (int): The number of positions each letter in the plaintext is
shifted.

    Returns:
    str: The encrypted ciphertext.
    """
    cipher = ""
    for i in Plain:
        if i == " ":  # Preserve spaces as is
            cipher += " "
      # Check if the character is uppercase and convert the character
using the key and add to the cipher string
        elif i.isupper():
             cipher += chr((ord(i) + key - 65) % 26 + 65)
      # Convert the character using the key and add to the cipher
string
        else:
             cipher += chr((ord(i) + key - 97) % 26 + 97)
    return cipher



def ciphertoplain(cipher, key):
    """
    Converts ciphertext back to plaintext using the Caesar cipher
method.

    Parameters:
    cipher (str): The ciphertext message to be decrypted.
    key (int): The number of positions each letter in the ciphertext
is shifted.
```

```
    Returns:
    str: The decrypted plaintext.
    """
    plain = ""
    for i in cipher:
        if i == " ":  # Preserve spaces as is
            plain += " "
     # Check if the character is uppercase and convert the character
using the key and add to the plain string
        elif i.isupper():
            plain += chr((ord(i) - key - 65) % 26 + 65)
     # Convert the character using the key and add to the plain string
        else:
            plain += chr((ord(i) - key - 97) % 26 + 97)
    return plain


# Main program loop
while True:
    # Display the menu and get the user's choice
    choice = int(input("\n\nMenu \n1.PlainText to CipherText
\n2.CipherText to PlainText \n3.BruteForce \n4.Exit\nEnter your
Choice:"))

    # Encrypt plaintext to ciphertext
    if choice == 1:
        Plain = input("Enter the Plain Text: ")
        key = int(input("Enter the Key: "))
        print(Plain, "in cipher with", key, "is", plaintocipher(Plain,
key))

    # Decrypt ciphertext to plaintext
    elif choice == 2:
        Cipher = input("Enter the Cipher Text: ")
        key = int(input("Enter the Key: "))
        print(Cipher, "in plain with", key, "is",ciphertoplain(Cipher,
key))
```

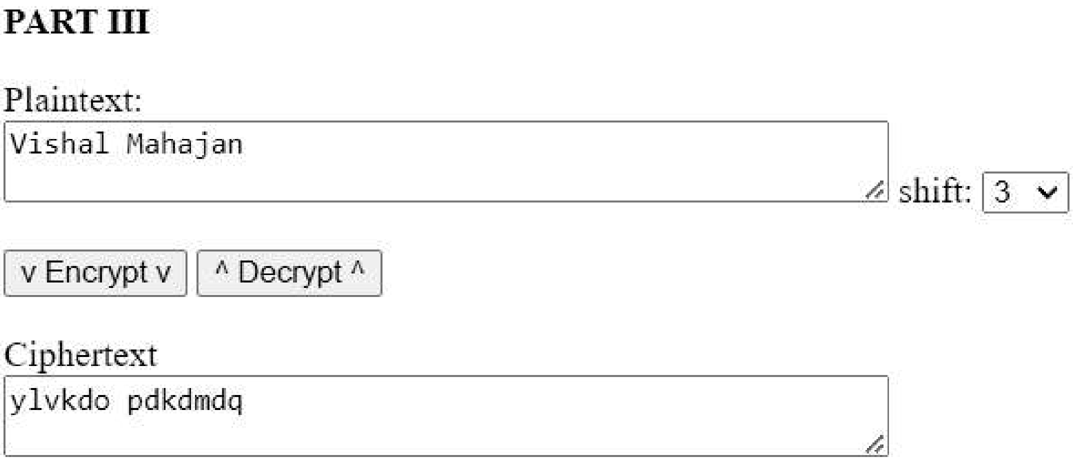
```
# Brute-force decryption
elif choice == 3:
        cipher = input("Enter the Cipher Text: ")
        # Try all possible keys from 0 to 25
        for i in range(26):
            brute = ""
            for j in cipher:
                if j == " ":  # Preserve spaces as is
                    brute += " "
        # Check if the character is uppercase and convert the
character using the key and add to the brute string
                elif j.isupper():
                    brute += chr((ord(j) - i - 65) % 26 + 65)
     # Convert the character using the key and add to the brute string
                else:
                      brute += chr((ord(j) - i - 97) % 26 + 97)
            print("Key:", i, "->", brute)

    # Exit the program
    elif choice == 4:
               break

    # Handle invalid menu choices
    else:
        print("Enter a Valid Option")
```

Plaintext: Vishal Mahajan
key: 3

Output Using Tool



PART III

Plaintext:

Vishal Mahajan

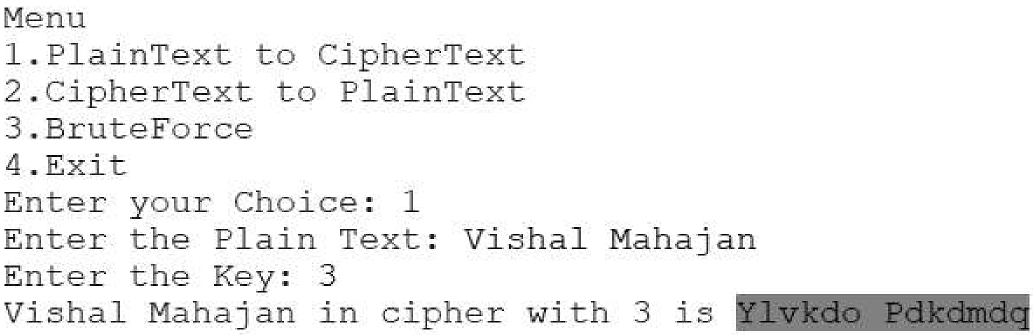
shift: 3

Ciphertext

ylvkdo pdkdmdq

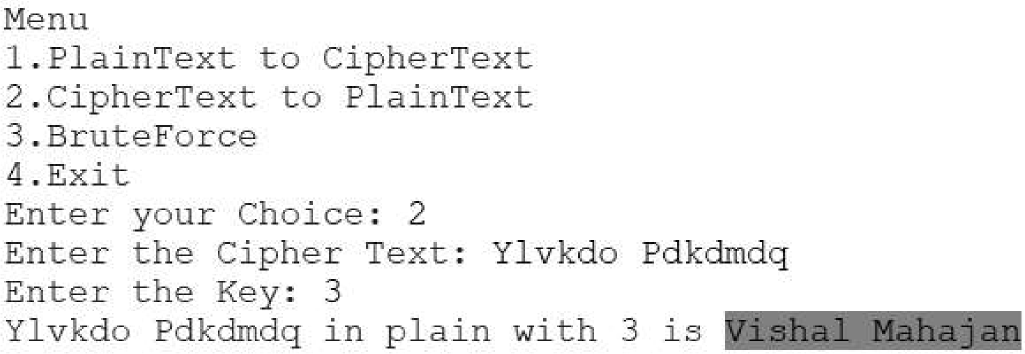
Output Using Code:

Encryption:



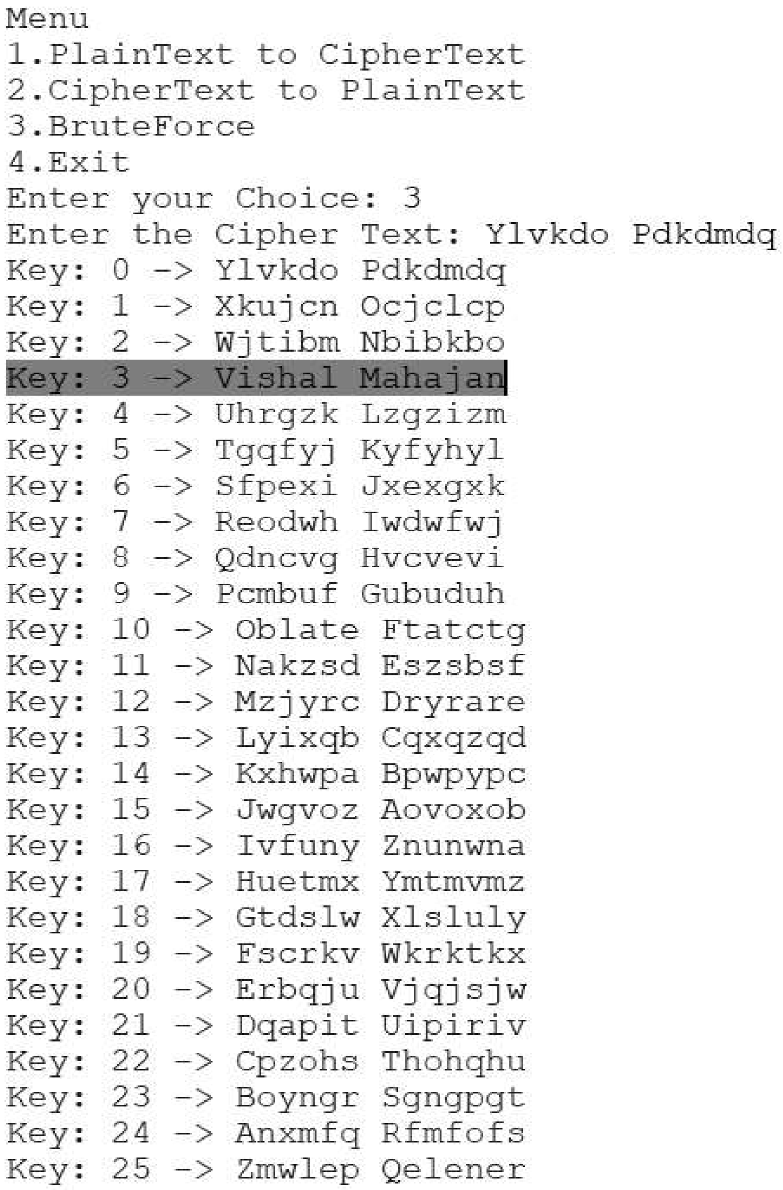
```
Menu
1.PlainText to CipherText
2.CipherText to PlainText
3.BruteForce
4.Exit
Enter your Choice: 1
Enter the Plain Text: Vishal Mahajan
Enter the Key: 3
Vishal Mahajan in cipher with 3 is Ylvkdo Pdkdmdq
```

Decryption:



```
Menu
1.PlainText to CipherText
2.CipherText to PlainText
3.BruteForce
4.Exit
Enter your Choice: 2
Enter the Cipher Text: Ylvkdo Pdkdmdq
Enter the Key: 3
Ylvkdo Pdkdmdq in plain with 3 is Vishal Mahajan
```

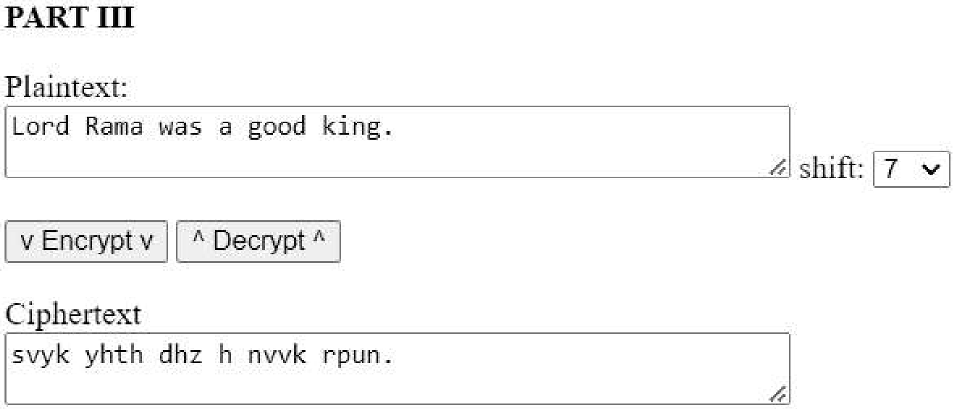
BruteForce:



```
Menu
1.PlainText to CipherText
2.CipherText to PlainText
3.BruteForce
4.Exit
Enter your Choice: 3
Enter the Cipher Text: Ylvkdo Pdkdmdq
Key: 0 -> Ylvkdo Pdkdmdq
Key: 1 -> Xkujcn Ocjclcp
Key: 2 -> Wjtibm Nbibkbo
Key: 3 -> Vishal Mahajan
Key: 4 -> Uhrgzk Lzgzizm
Key: 5 -> Tgqfyj Kyfyhyl
Key: 6 -> Sfpexi Jxexgxk
Key: 7 -> Reodwh Iwdwfwj
Key: 8 -> Qdncvg Hvcvevi
Key: 9 -> Pcmbuf Gubuduh
Key: 10 -> Oblate Ftatctg
Key: 11 -> Nakzsd Eszsbsf
Key: 12 -> Mzjyrc Dryrare
Key: 13 -> Lyixqb Cqxqzqd
Key: 14 -> Kxhwpa Bpwpypc
Key: 15 -> Jwgvoz Aovoxob
Key: 16 -> Ivfuny Znunwna
Key: 17 -> Huetmx Ymtmvmz
Key: 18 -> Gtdslw Xlsluly
Key: 19 -> Fscrkv Wkrktkx
Key: 20 -> Erbqju Vjqjsjw
Key: 21 -> Dqapit Uipiriv
Key: 22 -> Cpzohs Thohqhu
Key: 23 -> Boyngr Sgngpgt
Key: 24 -> Anxmfq Rfmfofs
Key: 25 -> Zmwlep Qelener
```

POST LAB EXERCISE:

1. **Encrypt the following plain text using key k = 7.**
   Plain Text: Lord Rama was a good king.
   Output using Tool :



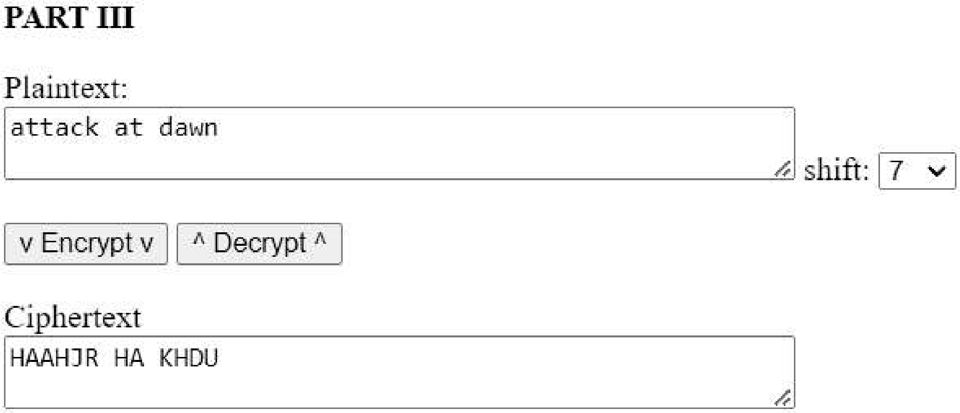
PART III

Plaintext:
Lord Rama was a good king.

shift: 7

Ciphertext
svyk yhth dhz h nvvk rpun.

2. **Given a cipher text, find out the corresponding plain text using brute force attack.**
   Cipher text: HAAHJR HA KHDU
   Output using Tool:



PART III

Plaintext:
attack at dawn

shift: 7

Ciphertext
HAAHJR HA KHDU