**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”.

1. **Mathematical Description**

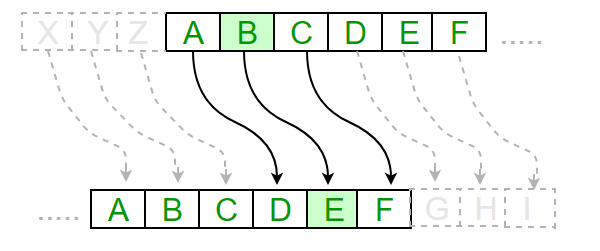
First we translate all of our characters to numbers, 'a'=0, 'b'=1, 'c'=2, ..., 'z'=25. We can now represent the caesar cipher encryption function, e(x), where x is the character we are encrypting, as:



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:





* **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)
   1. Encrypt the following plain text using key k = 7.  
      Plain Text: Lord Rama was a good king.
   2. 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)*

**OUTPUT:**

1. **Attach complete program with comments performing encryption and decryption of shift cipher.**

**CODE:**

#Encrypts the given plaintext using a Caesar cipher with the specified shift.

def encrypt(text, shift):

encrypted\_text = ""

for char in text:

if char.isalpha(): #

shifted = ord(char) + shift #Calculates the ASCII value after applying the shift

if char.islower():

if shifted > ord('z'): #Wraps around if the shifted value exceeds 'z'

shifted = shifted - 26

elif shifted < ord('a'): #Wraps around if the shifted value is less than 'a'

shifted = shifted + 26

elif char.isupper():

if shifted > ord('Z'): #Wraps around if the shifted value exceeds 'Z'

shifted = shifted - 26

elif shifted < ord('A'):

shifted = shifted + 26

encrypted\_text += chr(shifted) #Converts the shifted value back to a character

#The keeps shifted letters from going out of bounds by wrapping them around if they go past 'z' or 'Z',

#or below 'a' or 'A'. This way, it keeps the letters in their correct alphabet range.

else:

encrypted\_text += char

return encrypted\_text

#Decrypts the given ciphertext using a Caesar cipher with the specified shift(key value).

def decrypt(text, shift):

decrypted\_text = ""

for char in text: # Loops through each character in the input text

if char.isalpha(): # Checks if the character is a letter

shifted = ord(char) - shift # Calculates the ASCII value after reversing the shift

if char.islower():

if shifted < ord('a'):

shifted = shifted + 26

elif char.isupper():

if shifted < ord('A'):

shifted = shifted + 26

decrypted\_text += chr(shifted) # Converts the shifted value back to a character

else:

decrypted\_text += char

#THUS THE NON ALPHABET CHARACTERS ARE UNCHANGED AND WE GET THE DECRYPTED TEXT

return decrypted\_text

#Decrypts the given ciphertext with all possible keys and prints the results.

def decrypt\_with\_all\_keys(ciphertext):

print("Decryption with all possible keys:")

for shift in range(1, 26): #The range is of keys 1 to 25

decrypted\_text = decrypt(ciphertext, shift) # Decrypts the ciphertext with the current key

print(f"Key {shift}: {decrypted\_text}")

def main():

while True:

choice = input("Enter '1' for encryption,\n'2' for decryption with a key,\nor '3' for Brute Force Attack:")

#Here we prompt user for their choice

if choice == '1':

plaintext = input("Enter the plaintext: ")

shift = int(input("Enter the shift value (integer): "))

encrypted = encrypt(plaintext, shift)

print("Encrypted:", encrypted)

break

#Here, we take the input from the user and encrypt it using the key value.

elif choice == '2':

ciphertext = input("Enter the ciphertext: ")

shift = int(input("Enter the shift value (integer): "))

decrypted = decrypt(ciphertext, shift)

print("Decrypted:", decrypted)

break

#The input from the user is taken along with the key number and the decrypted text is delivered

elif choice == '3': # If the user chooses decryption with all keys

ciphertext = input("Enter the ciphertext: ")

decrypt\_with\_all\_keys(ciphertext)

break

#This is the BRUTE FORCE ATTACK method where we decrypt the ciphertext using all the keys and infer them.

else:

print("Invalid choice. Please enter '1', '2', or '3'.")

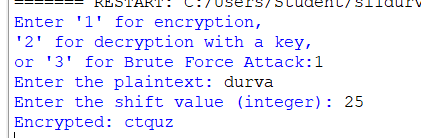
if \_\_name\_\_ == "\_\_main\_\_":

main()

This code implements a cipher for encryption and decryption. The encrypt function shifts each letter of the plaintext by a specified integer value, wrapping around if the shift goes past z. The decrypt function reverses this shift to retrieve the original text. The decrypt\_with\_all\_keys function attempts to decrypt a given ciphertext using all possible keys (from 1 to 25) to demonstrate how a brute force attack might work.

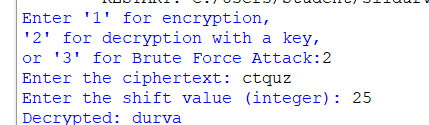
1. **Attach screenshots of program output (for encryption & decryption) and its validation using a virtual lab tool.**

ENCRYPTION**:**

****

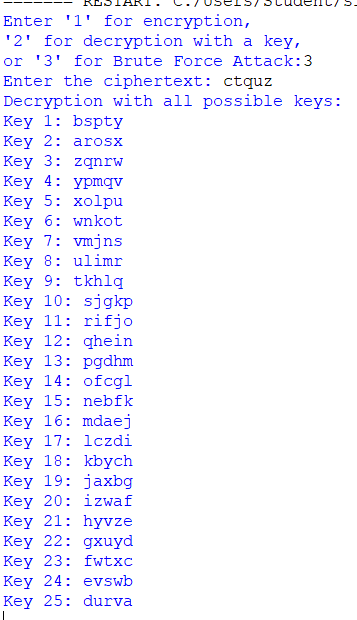
The output shows us the encrypted text with key value 25.

DECRYPTION**:**

****

The output shows us the decrypted text with key value 25.

BRUTE FORCE ATTACK**:**

****

This output shows us all possible keys for the given input and we can determine which the output is on the key that gives us a sensible plaintext.

**VALIDATION USING VIRTUAL TOOL:**

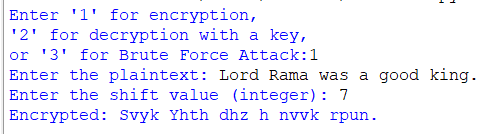
ENCRYPTION: DECRYPTION:

** **

To validate the output of the code, we input it into a virtual lab tool. Since we got the same results this step-by-step validation ensured that the code worked as intended for various inputs and functionalities.

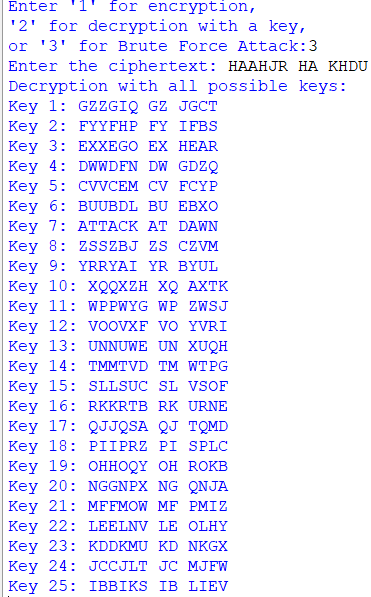
1. **Attach screenshots of examples from post experiment exercises.**
2. Encrypt the following plain text using key k = 7.

Plain Text: Lord Rama was a good king.

****

1. Given a cipher text, find out the corresponding plain text using brute force attack.

Cipher text: HAAHJR HA KHDU

****