K. S. Institute of Technology

Department of Computer Science & Engineering

Mini Project Report

Academic Year:	2019-20(Even)		
Course Name(Code):	Cryptography Network Security & Cyber Law (17CS61)		
Mini Project Title:	Implementation of Ceasar Cipher Algorithm		
Group No:	07		
Group Members:	Divya Yashaswi Kanney	1KS17CS023	
Semester/Section	VI/A		

1. Description of the mini project:

- The aim of the project is to implement the Caesar Cipher algorithm to encrypt plain text into cipher text with the help of a key and to decrypt cipher text into plain text with the help of a key.
- **Front end:** Tkinter Python
- **Back end:** Python

- **Working of the mini project:** The mini project contains a python file named 'CaesarCipher.py'.

- **Prerequisites:**
  1. Python (Download)
  2. Tkinter module

- **Steps to run:**
  1. Launch the terminal or the command prompt
  2. Change the directory to the directory containing the files of the mini project
  3. Type **python CaesarCipher.py**
  4. Select 'ENCRYPTION' or 'DECRYPTION' to navigate and perform respective operation

2. Description of the algorithm:

The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals.

Example

The text we will encrypt is 'go to war', with a shift (key) of 2 .

```
plaintext:    go to war
ciphertext: iq vq yct
```

It is easy to see how each character in the plaintext is shifted up the alphabet. Decryption is just as easy, by using an offset of -2.

3. Algorithm:

Encryption:

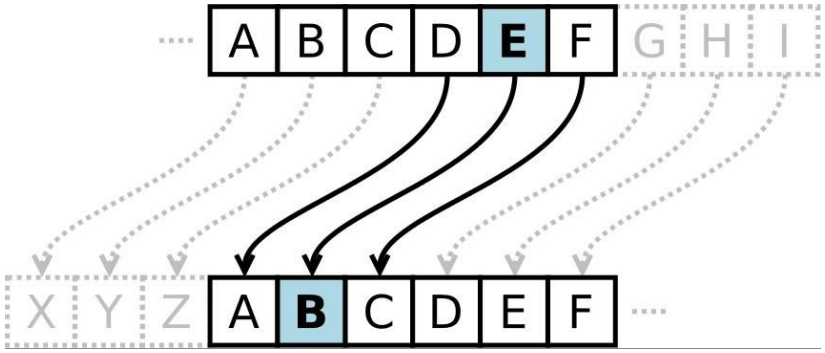
First we translate all of our characters to numbers, 'a'=0, 'b'=1, 'c'=2, ... , 'z'=25. We can now represent the caesar cipher encryption function, e(x), where x is the character we are encrypting, as:

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

Decryption:

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 :

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



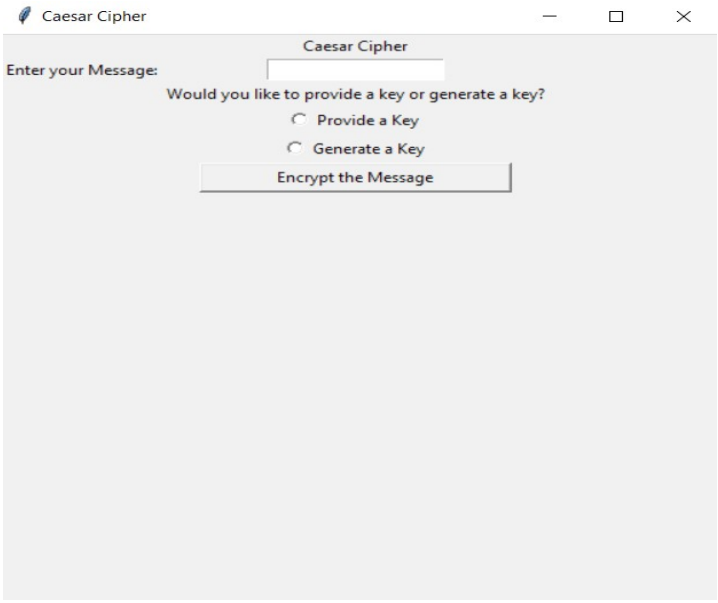
4. Implementation of algorithm(code):

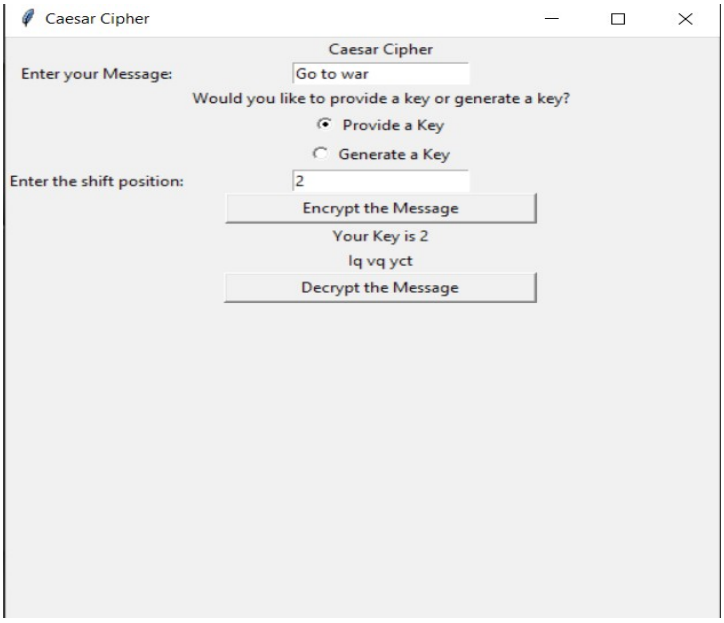
```
from tkinter import *
from random import *

root = Tk()
root.title("Caesar Cipher")
root.minsize(500, 500)
firstLabel = Label(root, text="Caesar Cipher")
firstLabel.grid(row=25, column=250)
msg_label = Label(root, text="Enter your Message:")
msg_label.grid(row=30, column=150)
msg = Entry(root)
msg.grid(row=30, column=250, columnspan=2)
radio = IntVar()
question_label = Label(root, text="Would you like to provide a key or generate a key?")
question_label.grid(column=250)
key_shift = Entry(root)

def print_key():
    key_label = Label(root, text="Enter the shift position:")
    key_label.grid(row =150, column=150)
    # key_shift = Entry(root)
    global key_shift
    key_shift.grid(row=150, column=250, columnspan=2)


provide = Radiobutton(root, text="Provide a Key", variable=radio, value=1, command=print_key)
provide.grid(column=250)


def generate_key():
    keyshift = randint(1, 26)
    # global key
    # key = key_shift
    return keyshift


generate = Radiobutton(root, text="Generate a Key", variable=radio, value=2, command=generate_key)
generate.grid(column=250)
generate.deselect()
provide.deselect()
```

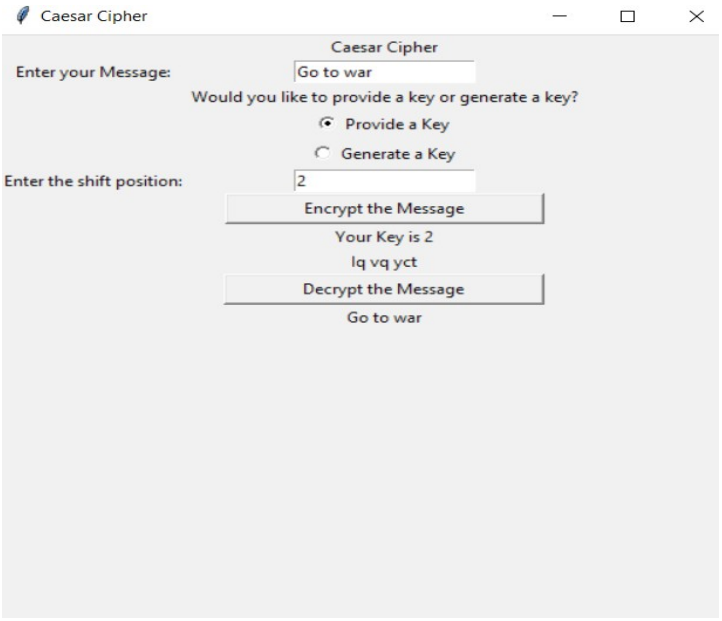
4. Implementation of algorithm(code): Continued...

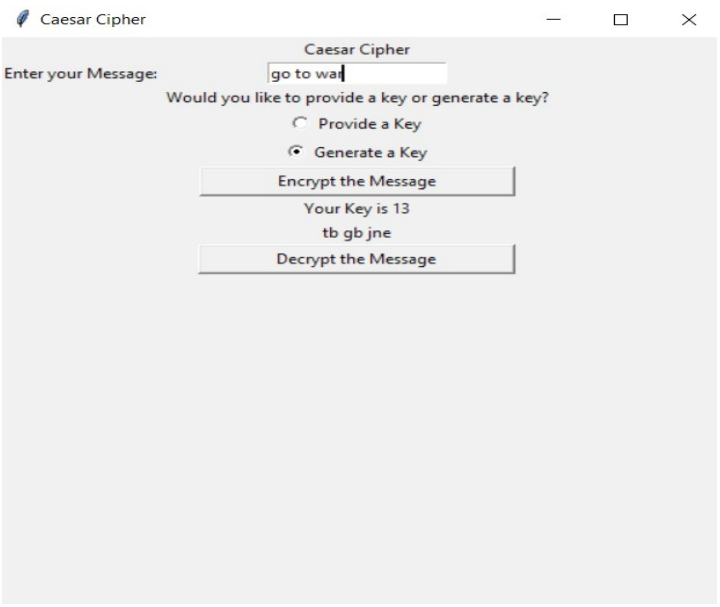
```
def encrypt():
    key = ""
    option = radio.get()
    if option == 1:
        key = key_shift.get()
    elif option == 2:
        key = str(generate_key())
    else:
        label = Label(root, text="Please select an option")
        label.grid(column=250)
    # shift = IntVar()
    message = msg.get()
    if message == "" or key == "":
        label = Label(root, text="Please Enter a Message and/or Key!")
        label.grid(column=250)
    else:
        cipher_text = ""
        shift = int(key)
        key_label = Label(root, text="Your Key is " + key)
        key_label.grid(column=250)
        for i in range(len(message)):
            char = message[i]
            if char.isalpha():
                if char.isupper():
                    cipher_text += chr((ord(char) + shift - 65) % 26 + 65)
                else: cipher_text += chr((ord(char) + shift - 97) % 26 + 97)
            else: cipher_text += char
        label = Label(root, text=cipher_text)
        label.grid(column=250)
        decrypt_button = Button(root, text="Decrypt the Message", padx=50,
            command=lambda: decrypt(shift, cipher_text))
        decrypt_button.grid(column=250)

def decrypt(shift, cipher_text):
    plain_text = ""
    for i in range(len(cipher_text)):
        char = cipher_text[i]
        if char.isalpha():
            if char.isupper():
                plain_text += chr((ord(char) - shift - 65) % 26 + 65)
            else:
                plain_text += chr((ord(char) - shift - 97) % 26 + 97)
        else:
            plain_text += char
    label = Label(root, text=plain_text)
    label.grid(column=250)

encrypt_button = Button(root, text="Encrypt the Message", padx=50, command=encrypt)
encrypt_button.grid(row=250, column=250)
root.mainloop()
```

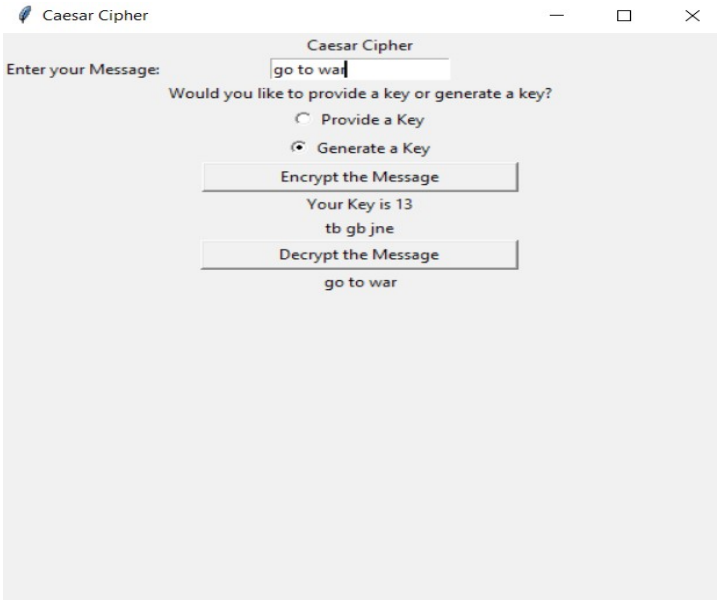
5.Snapshots of output:











6. Conclusion:

The project works successfully to encrypt plain texts to cipher texts and decrypt the cipher text to plain text using caesar cipher algorithm.

7. References:

- https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/
- https://www.geeksforgeeks.org/python-tkinter-grid_location-and-grid_size-method/
- https://www.delftstack.com/howto/python-tkinter/how-to-pass-arguments-to-tkinter-button-command/
- https://www.youtube.com/watch?v=YXPyB4XeYLA&t=2800s