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 Vigenere 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 Vigenere 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 'VigenereCipher.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 VigenereCipher.py**
  4. Select 'ENCRYPTION' or 'DECRYPTION' to navigate and perform respective operation

2. Description of the algorithm:

The Vigenère cipher is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword. It employs a form of polyalphabetic substitution.

In a Caesar cipher, each letter of the alphabet is shifted along some number of places. For example, in a Caesar cipher of shift 3, A would become D, B would become E, Y would become B and so on. The Vigenère cipher has several Caesar ciphers in sequence with different shift values.

To encrypt, a table of alphabets can be used, termed a tabula recta, Vigenère square or Vigenère table. It has the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword. For example, suppose that the plaintext to be encrypted is

ATTACKATDAWN.

The person sending the message chooses a keyword and repeats it until it matches the length of the plaintext, for example, the keyword "LEMON":

LEMONLEMONLE

Cipher Text: LXFOPVEFRNHR

3. Algorithm:

Vigenère can also be described algebraically. If the letters A – Z are taken to be the numbers 0–25 ($A \mathrel{\widehat{=}} 0$, $B \mathrel{\widehat{=}} 1$, etc.), and addition is performed modulo 26, Vigenère encryption E using the key K can be written as

$$C_i = E_K(M_i) = (M_i + K_i) \bmod 26$$

and decryption D using the key K as

$$M_i = D_K(C_i) = (C_i - K_i) \bmod 26,$$

in which $M = M_1 \dots M_n$ is the message, $C = C_1 \dots C_n$ is the ciphertext and $K = K_1 \dots K_n$ is the key obtained by repeating the keyword $\lceil n/m \rceil$ times in which m is the keyword length.

Thus, by using the previous example, to encrypt $A \mathrel{\widehat{=}} 0$ with key letter $L \mathrel{\widehat{=}} 11$ the calculation would result in $11 \mathrel{\widehat{=}} L$.

$$11 = (0 + 11) \bmod 26$$

Therefore, to decrypt $R \mathrel{\widehat{=}} 17$ with key letter $E \mathrel{\widehat{=}} 4$, the calculation would result in $13 \mathrel{\widehat{=}} N$.

$$13 = (17 - 4) \bmod 26$$

In general, if Σ is the alphabet of length ℓ, and m is the length of key, Vigenère encryption and decryption can be written:

$$C_i = E_K(M_i) = (M_i + K_{(i \bmod m)}) \bmod \ell,$$

$$M_i = D_K(C_i) = (C_i - K_{(i \bmod m)}) \bmod \ell.$$

M_i denotes the offset of the *i*-th character of the plaintext M in the alphabet Σ. For example, by taking the 26 English characters as the alphabet $\Sigma = (A, B, C, \dots, X, Y, Z)$, the offset of A is 0, the offset of B is 1 etc. C_i and K_i are similar.

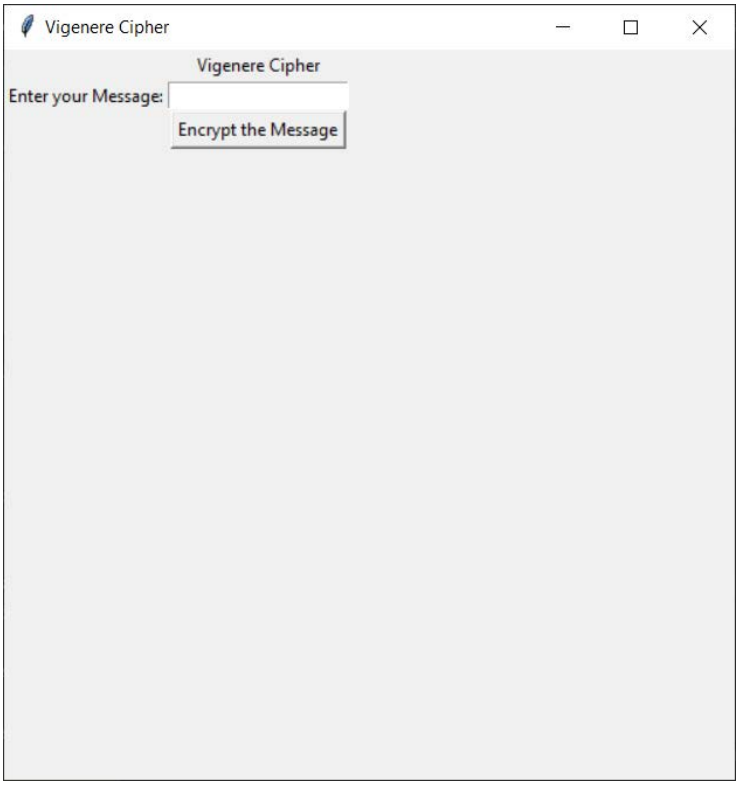
4. Implementation of algorithm(code):

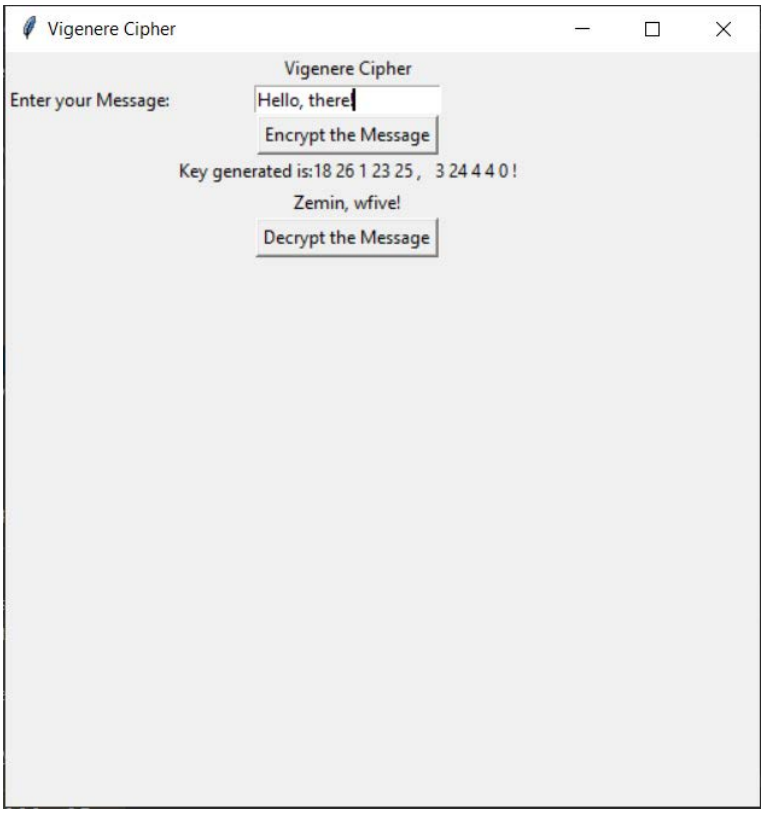
```
from tkinter import *
from random import *

vigenere_cipher = Tk()
vigenere_cipher.title("Vigenere Cipher")
vigenere_cipher.minsize(500, 500)

firstLabel = Label(vigenere_cipher, text="Vigenere Cipher")
firstLabel.grid(row=0, column=250)

msg_label = Label(vigenere_cipher, text="Enter your Message:")
msg_label.grid(row=20, column=150)

msg = Entry(vigenere_cipher)
msg.grid(row=20, column=250)


def generate_key(message):
    key = []
    for i in message:
        if i.isalpha():
            key.append(randint(0, 26))
        else:
            key.append(i)
    return key


def encrypt():
    message = msg.get()
    key = generate_key(message)
    key_label = Label(vigenere_cipher, text="Key generated is:" + " ".join(str(x) for x in
key))
    key_label.grid(column=250)

    cipher_text = ""
    for i in range(len(message)):
        char = message[i]
        if char.isalpha():
            if char.isupper():
                cipher_text += chr((ord(char) + key[i] - 65) % 26 + 65)
            else:
                cipher_text += chr((ord(char) + key[i] - 97) % 26 + 97)
```

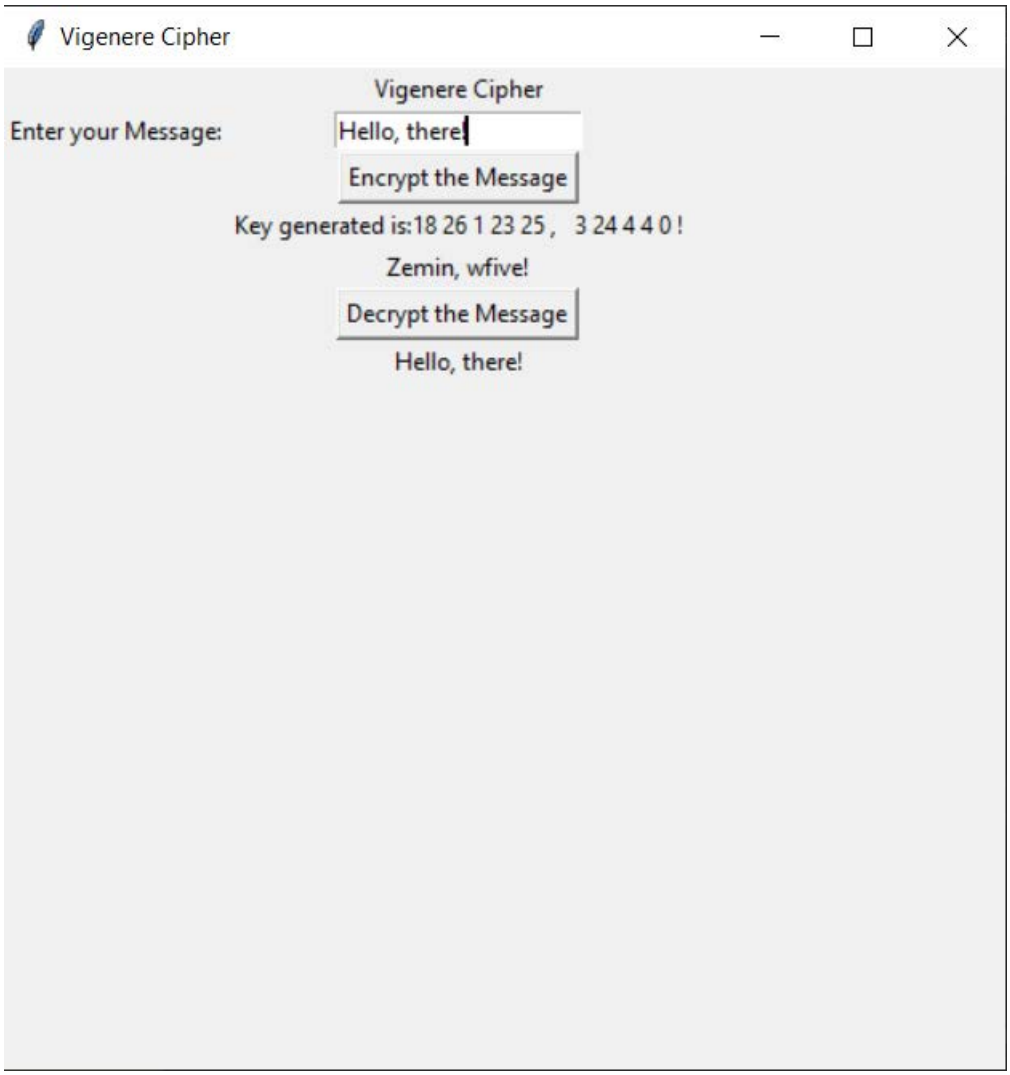
4. Implementation of algorithm(code):
Continued...

```
        else:
            cipher_text += char

    label = Label(vigenere_cipher, text=cipher_text)
    label.grid(column=250)

    decrypt_button = Button(vigenere_cipher, text="Decrypt the Message",
command=lambda: decrypt(cipher_text, key))
    decrypt_button.grid(column=250)


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


encrypt_button = Button(vigenere_cipher, text="Encrypt the Message",
command=encrypt)
encrypt_button.grid(column=250)

vigenere_cipher.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 Vigenere cipher algorithm.

7. References:

- https://www.geeksforgeeks.org/vigenere-cipher/
- 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
- https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher