PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Computer Science & Engineering

Title: SECRET AGENT CHAT

Team Member Details

PES2UG21CS086 - Aravilli Atchuta Ram

PES2UG21CS232 - Kolisetty Venkata Aditya

PES2UG21CS285 - Mayank Kashyap

Abstract

During the last decades, information security has become a major issue. Encrypting and decrypting data have recently been widely investigated and developed because there is a demand for a stronger encryption and decryption which is very hard to crack. Cryptography plays major roles to fulfilment these demands. Nowadays, many of researchers have proposed many of encryption and decryption algorithms such as AES,DES, RSA, and others

Security of data in a computer is needed to protect critical data and information from other parties. One way to protect data is to apply the science of cryptography to perform data encryption. There are wide variety of algorithms used for encryption of data, this study used a one-time pad algorithm for encrypting data. One time pad algorithm uses symmetric key encryption. Algorithm One Time Pad uses the same key in the encryption process and a decryption of the data. An encrypted data will be transformed into cipher text so that the only person who has the key can open that data. Therefore, analysis will be done for an application that implements a one-time pad algorithm for encrypting data. The application that implements the one time pad algorithm can help users to store data securely.

Table of Contents

Introduction

In network security,cryptography has a long history by provides a way to store sensitive information or transmit it across insecure networks (i.e. the Internet) so that it cannot be read by anyone except the intended recipient, where the cryptosystem is a set of algorithms combined with keys to convert the original message (Plain-text) to encrypted message (Cipher-text) and convert it back in the intended recipient side to the original message (Plain-text)

Security of data in a computer is very important to protect the data from other parties that do not have the authority to determine the content of the data . If the data has a very high value that has been stored in the computer and then opened by another party then it would be very detrimental.

The advantage of the one-time pad algorithm is to perform the encryption process and a decryption of each character plaintext, using each character in the key.

One time pad algorithm is only used one time for one key encryption key then it will be destroyed and not used again to encrypt other data.

This algorithm was found in 1917 by Major Joseph Mauborgne as the improvement of the Vernam cipher to produce the perfect security. Mauborgne proposes the use of one-time pad (pad = paper notebooks) which contains the generation of random sequences of characters - a key character. To encrypt a message pad, it is simply used once (one-time), afterwards to encrypt messages, the pad that has been used can be destroyed in order that no one can use it.

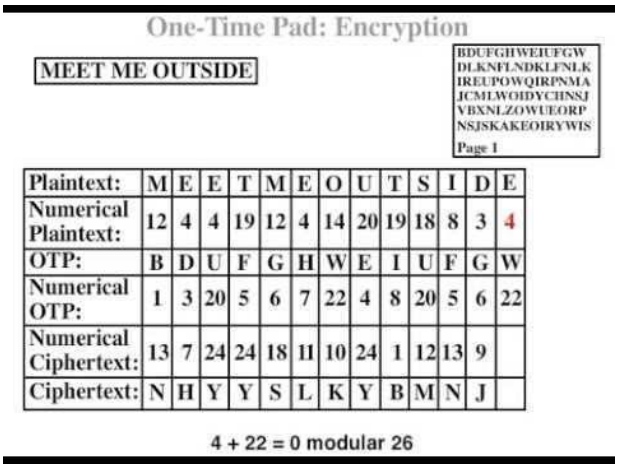
The user has to enter the plain text to be encrypted and give the maximum length of the message, the one-time pads of the corresponding length are generated and the text is encrypted(both one-time pads and encrypted text can be saved into .txt files).

If the user wants to decrypt a message , the user has to choose the encrypted text message file and select the correspoding one-time pad file that was used and can decrypt the message.

The main purpose of GUI is to enhance the usability of the program and for easier understanding. There have been many components of Python used in the project such as lists, random module, tkinter GUI module, dictionaries, basic constructs i.e. if-else construct, file functions(read,readlines).

Design/Implementation

Example of the algorithm



Plaintext:	M	E	E	T	M	E	O	U	T	S	I	D	E
Numerical Plaintext:	12	4	4	19	12	4	14	20	19	18	8	3	4
OTP:	B	D	U	F	G	H	W	E	I	U	F	G	W
Numerical OTP:	1	3	20	5	6	7	22	4	8	20	5	6	22
Numerical Ciphertext:	13	7	24	24	18	11	10	24	1	12	13	9	
Ciphertext:	N	H	Y	Y	S	L	K	Y	B	M	N	J	

Code:

#Keys Generation

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
from tkinter import scrolledtext
from random import randint
from tkinter import filedialog

# from tkinter.scrolledtext import ScrolledText
ALPHABET = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'


def saveFile():
    myFile = filedialog.asksaveasfile(mode="w", defaultextension='.txt')
    if myFile is None:
        return
    # data=keysEditor.get('1.0','end')
    data = otpEntries.get()
    myFile.write(data)
    myFile.close()


def genOTPs(*args):
    try:
        numotps = int(otpEntry.get())
        outval = ""
        for i in range(1, numotps + 1):
```

```
            otp = str(randint(0, 26))
            otps.append(otp)
        outval = ",".join(otps)
        # keysEditor.insert(1.0,outval)
        otpEntries.set(outval)
    except ValueError:
        pass


def write_tofile(k):
    with open(k, 'a+') as myfile:
        for c in otps:
            myfile.write(str(c) + '\n')


root = Tk()
width = root.winfo_screenwidth()
height = root.winfo_screenheight()
root.geometry("%dx%d" % (width, height))
root.title("Keys Generation Window")

otps = []
otpEntry = StringVar()
otpLabel = StringVar()
otpEntries = StringVar()
```

```
tk.Label(root, text="Maximum Length of Message", background='black', fg='white',
font=25).place(x=650, y=90)  # ----

tk.Entry(root, textvariable=otpEntry, width=40,font=18).place(x=570, y=140)


tk.Label(root, text="Generated Keys are", background='green',
foreground="white",font=25).place(x=685, y=250)


# Creating scrolled text

# area widget

tk.Entry(root, width=40, textvariable=otpEntries,font=18).place(x=570, y=300)


tk.Button(root, text="Click to Generate OTPs", width=40,
command=genOTPs,font=25,background='pink').place(x=570, y=350)


# keysEditor = tk.Text(root,height=6,width=24)

# keysEditor.grid(column=1, row=3, sticky=E)



tk.Button(root, text="Save to File", width=10,
command=saveFile,font=25,background='pink').place(x=705, y=400)

root.mainloop()
```

Code

#Encryption of message

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from random import randint
from tkinter.scrolledtext import ScrolledText
from tkinter import filedialog

##k=r"C:\Users\atchu\Documents\Jupyter_nb\otp-tk19.txt"
key_file_name = ""
otps = []


def open_Kyes_file():
    # file type
    filetypes = (
        ('text files', '*.txt'),
        ('All files', '*.*')
    )
    # show the open file dialog
    key_file_name = filedialog.askopenfile(filetypes=filetypes)
    # read the text file and show its content on the Text

    if key_file_name is None:  # askopenfile() returns `None` if dialog closed with "cancel".
        return
    else:
        otps = key_file_name.read().splitlines()
```

```
    keysFile.insert('1.0', otps)
    keysFile.config(state=DISABLED)

    encryptMessage(otps)


def encryptMessage(otps):
    ALPHABET = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
    # data=keysFile.get('1.0','end')
    # print(data)

    otps_loc = otps[0].split(',')
    # print(otps_loc[0])
    # print(otps_loc[1])

    try:
        ciphertext = ''
        plaintext = plain.get('1.0', tk.END)

        for position, character in enumerate(plaintext):
            print(position)
            print(character)
            print(otps_loc[position])
            if character not in ALPHABET:
                ciphertext += character
            else:
                encrypted = (ALPHABET.index(character) + int(otps_loc[position])) % 52
```

```
            ciphertext += ALPHABET[encrypted]

        enceditor.insert('1.0', ciphertext)
        enceditor.config(state=DISABLED)

    except Exception as e:
        print(e)
        pass


def saveEncryptedDataFile():
    myFile = filedialog.asksaveasfile(mode="w", defaultextension='.txt')
    if myFile is None:
        return
    data = enceditor.get('1.0', tk.END)
    myFile.write(data)
    myFile.close()


root = tk.Tk()
width = root.winfo_screenwidth()  # 1
height = root.winfo_screenheight()  # 2
root.geometry("%dx%d" % (width, height))
root.title("Encryption Window")

# In[6]:
```

```
tk.Label(root, text="Enter Message to
Encrypt",font=25,background='black',foreground='white').grid(column=2, row=1)

plain = tk.Text(root, height=6, width=24,font=15)

plain.focus()

plain.grid(column=4, row=1)


tk.Button(root, text="Select Keys File to Encrypt",
command=open_Kyes_file,font=21,background='magenta').grid(column=2, row=2)

keysFile = tk.Text(root, height=6, width=24,font=15)

keysFile.grid(column=4, row=2)


tk.Label(root, text="Encrypted Data",font=25,background='green',
foreground="white").grid(column=2, row=3)

enceditor = tk.Text(root, height=12, width=24,font=15)

enceditor.grid(column=4, row=3)


# tk.Button(root, text="Encrypt", command=encryptMessage).grid(column=1, row=5)


tk.Button(root, text="Save to file",
command=saveEncryptedDataFile,font=30,background='pink').grid(column=4,row=4)


root.mainloop()
```

Code

 # Decryption of message

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from random import randint
from tkinter.scrolledtext import ScrolledText
from tkinter import filedialog
from tkinter import filedialog as fd



def open_Keys_file():
    # file type
    filetypes = (
        ('text files', '*.txt'),
        ('All files', '*.*')
    )
    # show the open file dialog
    key_file_name = filedialog.askopenfile(filetypes=filetypes)
    # read the text file and show its content on the Text

    if key_file_name is None:  # askopenfile() returns `None` if dialog closed with "cancel".
        return
    else:
        otps = key_file_name.read().splitlines()

    keysFile.set(otps)
```

```
    decryptMessage(otps)



def decryptMessage(otps):
    ALPHABET = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
    # data=keysFile.get('1.0','end')
    # print(data)

    otps_loc = otps[0].split(',')
    # print(otps_loc[0])
    # print(otps_loc[1])

    try:
        plaintext = ''
        ciphertext = cipher.get()
        print(ciphertext)
        for position, character in enumerate(ciphertext):
            if character not in ALPHABET:
                plaintext += character
            else:
                decrypted = (ALPHABET.index(character) - int(otps_loc[position])) % 52
                plaintext += ALPHABET[decrypted]
        deceditor.set(plaintext)

    except ValueError:
        print()
        pass
```

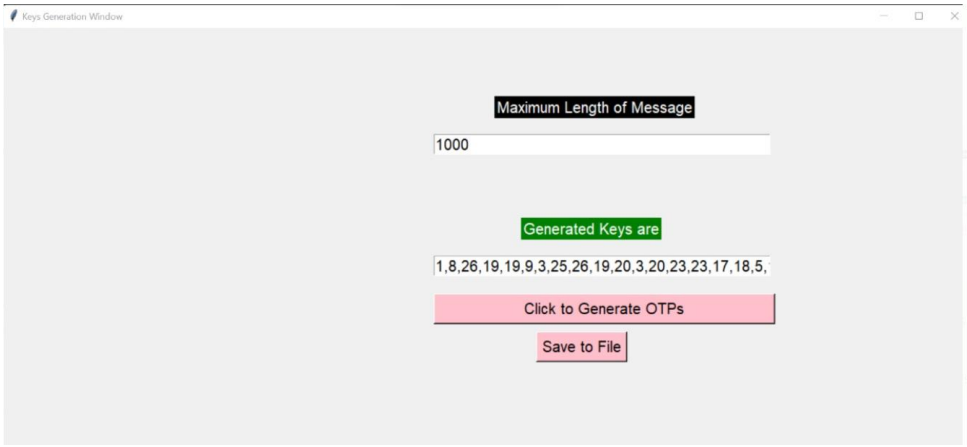
```
def open_text_file():
    # file type
    filetypes = (
        ('text files', '*.txt'),
        ('All files', '*.*')
    )
    # show the open file dialog
    f = filedialog.askopenfile(filetypes=filetypes)
    # read the text file and show its content on the Text

    encText = f.read()

    cipher.set(encText)

    # v1.config(state=DISABLED) #changes



root = tk.Tk()
root.title("Decryption Window")
width = root.winfo_screenwidth()  # 1
height = root.winfo_screenheight()  # 2
root.geometry("%dx%d" % (width, height))

cipher = StringVar()
keysFile = StringVar()
deceditor = StringVar()
```

```
open_button = tk.Button(root, text='Open encrypted Data File',
command=open_text_file,font=30,background='yellow')

open_button.place(x=680,y=100)


tk.Entry(root, textvariable=cipher, state=DISABLED,font=20,width=40).place(x=600,y=150)  #
change

# open_button.grid(column=1, row=2, sticky='w', padx=10, pady=10)


tk.Button(root, text="Select Keys File to Decrypt",
command=open_Keys_file,font=30,background='cyan').place(x=680,y=250)

tk.Entry(root, textvariable=keysFile,font=30,width=30,state=DISABLED).place(x=640,y=300)


tk.Label(root, text="   Decrypted Content    ", width=40,bg='black',
fg='white',font=50,background='green').place(x=595,y=400)

#tk.Button(root,text="Click to Decrypt ").place(x=250,y=285)

tk.Entry(root,textvariable=deceditor,width=40,font=50,fg='orange',insertontime=0).place(x=595,y=
450)

root.mainloop()
```

Testing

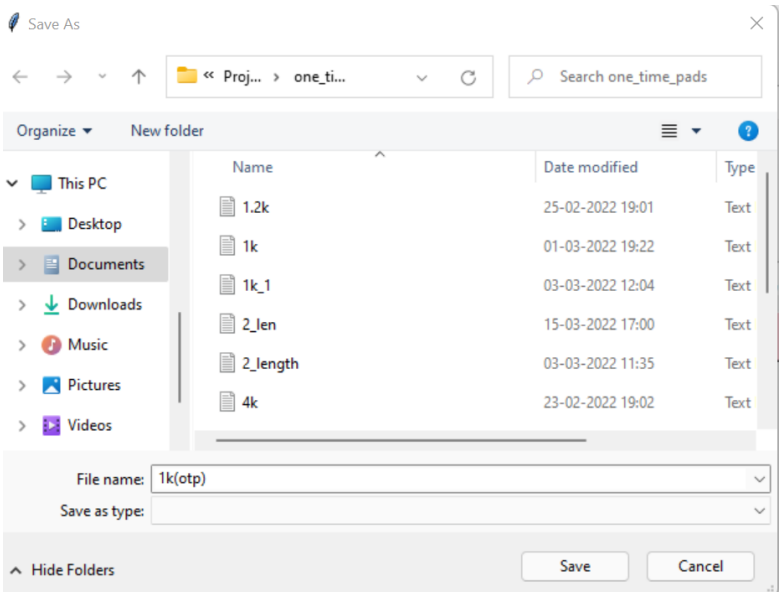
Test Cases	Results
1.Maximum length of message was given 2 but more than 2 characters were entered in the plaintext Entry Box	Error displayed as list index out of range
2.Input message was entered but wrong key file was chosen during decryption	No Error but expected output is different from actual output
3.Input message was entered , otp1.txt was chosen for keys generation and file was saved as t1//.txt	Error as invalid file name was entered
4.Input message was entered , keys were generated , message encrypted and during decryption the same message came back	Program Execution Successful

Result and Analysis

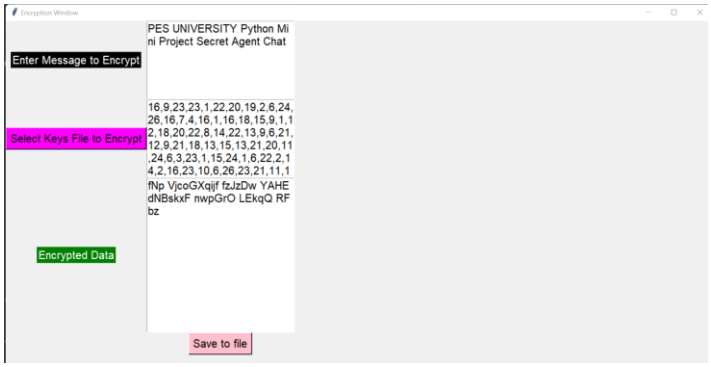


Keys Generation Stage

Maximum Length Of message was given as 1000 and 1000 keys are generated.

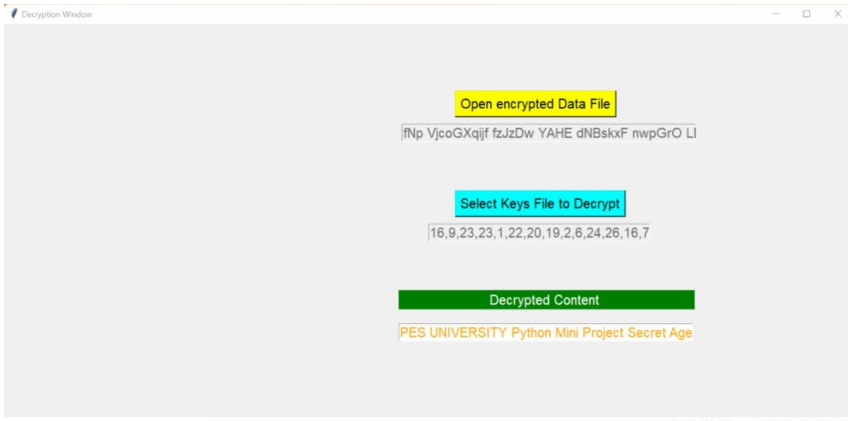


File is saved as 1k(otp).txt



Encryption Stage

Message is encrypted, 1k(otp).txt file is selected and the corresponding encrypted data is generated, which is saved into a file m1(proj).txt



Decryption Stage

Corresponding encrypted data file is selected and keys file is selected , the decrypted content is displayed in a scrollable text format.

Conclusions and Future Enhancements

Secret Agent Chat helps you encrypt and decrypt messages with the help of tkinter GUI , which is easily usable and generates precise output.

Just with few lines of code running you have the power to encrypt and decrypt messages using a simple yet efficient algorithm and secure your day to day communications.

This project is very useful for defence, corporate, banking, communication and different government portals where information exchange is more crucial.

This project has a vast scope and numerous complex algorithms of cryptography can be utilized to enhance the security. In the future we plan to link this directly to our day to day communication apps like WhatsApp, Facebook , Instagram, G-Mail etc, for our own end to end encryption.

Also instead of the random module the os.urandom module of python is better and more efficient.

References

- https://science.jrank.org/computer-science/Encryption-2.html
- https://en.wikibooks.org/wiki/Cryptography/One_time_pads
- https://projects.raspberrypi.org
- www.geeksforgeeks.org
- Modern Tkinter for Busy Python Developers: By Mark Roseman
- https://docs.python.org/3/library/tkinter.html
- https://engineering.fb.com/2021/09/10/security/whatsapp-e2ee-backups/
- https://www.youtube.com/watch?v=yG0fAUn2uB0
- Images from Google Images.
- https://www.researchgate.net/publication/273011532_A_New_Approach_for_Complex_Encrypting_and_Decrypting_Data
- https://thesai.org/Downloads/Volume6No9/Paper_39-An_Analysis_Encryption_and_Description_Application_by_using_One_Time_Pad_Algorithm.pdf