FILE SECURITY USING RSA ENCRYPTION

ANUSHKA PANDEY

DODDAMANI APURV SANGMESH

KANCHARLA LOKESH BABU

Abstract:

Our project aim is secure data using one of the 'Data Encryption' technique that is 'RSA Data Encryption' technique. Data encryption is done to prevent any tampering or unauthorized access. In this we convert normal text file/image file into protected Cipher Text using RSA algorithm. In between to make data more secure we have added password barrier. Whenever anyone wants to login the system to access the files, they should have password. After logging in user can encrypt the file, data encryption is what happens when you take the text or data you use and convert it to a code (also called 'ciphertext') that can't be understood by those who do not have the correct key. For the data/file to be useable, it must be changed back from encrypted form or needs to be decrypted, so to make that secured, before decrypting any file person needs to enter correct password.

Introduction about the project:

So, as we are using RSA encryption algorithm, it is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e., **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and Private key is kept private.

The public key consists of two numbers where one number is multiplication of two prime numbers(n) and another number is an integer (e). And private key(d) is also derived from the same two prime numbers. So, while registering the user is asked to enter two numbers and using those two numbers, we find two nearest next prime numbers and use that to find public and private key that will be used for encryption and decryption.

User have to first get registered, register form will ask to enter username, password and any two random numbers. After registering user can login to the system using that username and password and then they can encrypt and decrypt files after entering password. Encryption and decryption will be performed using public and private keys which has been calculated using those two numbers that user have given while registering.

Generating Public Key and Private Key:

Generating Public Key:

* Select two prime numbers. Suppose P and Q.

- Now First part of the Public key: **n = P*Q**.
- We also need a small exponent say e: But e Must be
    1. An integer.
    2. Not be a factor of n.
    3. **$1 < e < \Phi(n)$** [$\Phi(n)$ is discussed below],
- Our Public Key is made of n and e

Generating Private Key:

- We need to calculate $\Phi(n)$: Such that **$\Phi(n) = (P-1)(Q-1)$**
- Now calculate Private Key, d: **$d = (k*\Phi(n) + 1) / e$** for some integer k

So, by this we get our – Public Key (n and e) and Private Key(d)

Converting Letters to Digit Form:

Suppose we have to encrypt **“HI”**:

- Convert letters to numbers: H = 72 and I = 73(Alphabet are converted into numbers using ASCII table)

- Thus, **Encrypted Data for H = 72^e mod n and I = 73^e mod n**. Thus, our Encrypted Data comes out to be 183,57

- Now we will decrypt:

Decrypted Data = c^d mod n.

Thus, our Encrypted Data comes out to be 72 **= H and I = 73, i.e., "HI".**

Module Description

In this particular part we will talk about the description of various modules which have been built by us.

Main Module main_screen():

```
def main_screen(): #part_1
```

This is the most important module which houses both the login and register button for our display page. On clicking the login button it takes to the login page where you have to insert the username and password, and on clicking the registration button it takes you to a new registration page where you have to enter new username, password, **p** and **q values** used for creating public and private keys.

Login Module:

```
def login():  #part_2
```

This module has the login page where you have to insert the username and password. On successful registration it takes you to encryption and decryption page.

Registration Module:

```
def register():   #part_3
```

This module has the registration part on clicking the registration button it takes you to a new registration page where you have to enter new username, password, **p** and **q values** used for creating public and private keys. On successful registration it takes you back to login page so that you can login.

RSA Module:

```
def rsa(p, q):
```

After successful registration the **p** and **q values** given will be sent to this function which is used to create private and public key. The numbers entered initially are sent to next-prime function which rounds them to the nearest prime numbers. This function then multiplies both the values to find n and multiplies (p-1) and (n-1) to find phi value. The phi value is used in conjunction with e value to find the public key. After finding public key, we find k value using random function which is used to d value which is the private key.

Prime Generation Module:

```
def nxtprimegen(n):
```

As said earlier the **p** and **q values** given will be sent to this function for the creation of next nearest prime values of those two numbers so that we can get two prime numbers which will be used for getting public and private keys.

Encryption Module:

```
def encryption():
```

This module is called after successful login. This opens a dialog box to select a file (.txt or .png) after selecting a file. It starts the conversion of the file by splitting the text lines and then words. The words are then converted into ASCII values which are then converted using the pow function (i: e (ASCII number)^public key % n).This value is then placed back into file instead of original text file. The original file is then deleted. Only the encrypted file remains

Decryption Module:

```
def decryption():
```

This module is called after successful login and once a particular module has been encrypted. This opens a dialog box to select a file (.txt or .png) after selecting a file. It starts the conversion of the file by splitting the text lines and then words. The number which we got from encryption need to be decrypted back using the pow function (i: e (Encrypted number)^private key % n).This value is then placed back into the file. This way the file is once again converted to its original form

Hardware /software used:

We have used software PyCharm Community 2021.1 (Python language) for implementing our project.

Sample Code:

```
import base64

import random

from tkinter import *

from tkinter import messagebox

from tkinter import filedialog

import os


import math


def nxtprimegen(n):

   m = n

   n = n + 100

   prime = [True for i in range(n + 1)]

   p = 2

   while (p * p <= n):


      if (prime[p] == True):

         for i in range(p * 2, n + 1, p):

            prime[i] = False

      p += 1

   for p in range(2, n):

      if prime[p]:

         if p > m:

            return p

         else:

            val = p
```

```
    return val



def rsa(p, q):
    p = int(p)
    q = int(q)
    p = nxtprimegen(p)
    q = nxtprimegen(q)
    print("prime: ", p, " ", q)
    n = p * q
    print("n: ", n)
    phi = (p - 1) * (q - 1)
    e = 2
    while (1):
        if math.gcd(e, phi) == 1:
            break
        else:
            e += 1
    print("public key: ", e)
    k=random.randint(1,9)
    d = (((k * phi) + 1)// e)
    print("private key: ", d)
    return e, d, n


def encryption():
    global screen6,ee,dd,nn
    # screen3.destroy()
    screen6 = Toplevel(screen)
    screen6.title("Encryption")
```

```
screen6.geometry("300x400")
file1 = filedialog.askopenfilename()


with open(username1, "r") as f:
    lines = f.read().splitlines()
ee = int(float(lines[2]))
dd = int(float(lines[3]))
nn = int(float(lines[4]))



with open(file1, "rb") as image:
    f = image.read()
    msg = bytearray(f)
en = []
de = []

print("Public Key", ee)
print("Private Key", dd)
print("n:", nn)

for x in msg:
    en.append(pow(x, ee, nn))
# print("\nencrypt msg: ", en)
ss = ""
f3 = open(file1 + '.bin', 'w')
for x in en:
    ss += (str(x) + " ")

f3.write(ss)
f3.close()
```

```
    os.remove(file1)

    messagebox.showinfo("Success!", "Encryption Complete")


    screen6.destroy()


def decryption():
    # screen3.destroy()
    global screen7
    screen7 = Toplevel(screen)
    screen7.title("Decryption")
    screen7.geometry("300x400")
    file1 = filedialog.askopenfilename()
    print(file1)
    # Label(screen7, text = file1).pack()

    with open(username1, "r") as f:
        lines = f.read().splitlines()
    ee = int(float(lines[2]))
    dd = int(float(lines[3]))
    nn = int(float(lines[4]))

    print("Public Key", ee)
    print("Private Key", dd)
    print("n:", nn)

    en = []
    de = []
    f3 = open(file1, 'r')

    str2 = f3.read()

    en2 = str2.split(" ")

    f3.close()

    for x in en2:

        if len(x) > 0:

            en.append(int(x))

    for x in en:

        de.append(pow(x, dd, nn))

    bytearray(de[:4])

    f2 = open(file1[:-4], 'wb')

    f2.write(bytearray(de))

    f2.close()

    # print("decrypt msg: ", de)

    os.remove(file1)

    messagebox.showinfo("Success!", "Decryption Complete")

    screen7.destroy()


def decryption_password():

    global screen8

    screen8 = Toplevel(screen)

    screen8.title("Password verification")

    screen8.geometry("300x400")

    Label(screen8, text="Please enter password below to decrypt", bg="wheat", width="300", height="2",

          font=("Calibri", 11)).pack()

    Label(screen8, text="").pack()



    global password_verify
    password_verify = StringVar()



    global password_entry1



    Label(screen8, text="Password * ", font=("Calibri", 11)).pack()
    password_entry1 = Entry(screen8, textvariable=password_verify,show='*')
    password_entry1.pack()
    Label(screen8, text="").pack()
    Button(screen8, text="Decrypt", width=10, height=1, command=login_decryption).pack()



def login_decryption():

    password1 = password_verify.get()

    password_entry1.delete(0, END)

    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            decryption()
        else:
            password_not_recognised()
    else :
        main_screen()
```

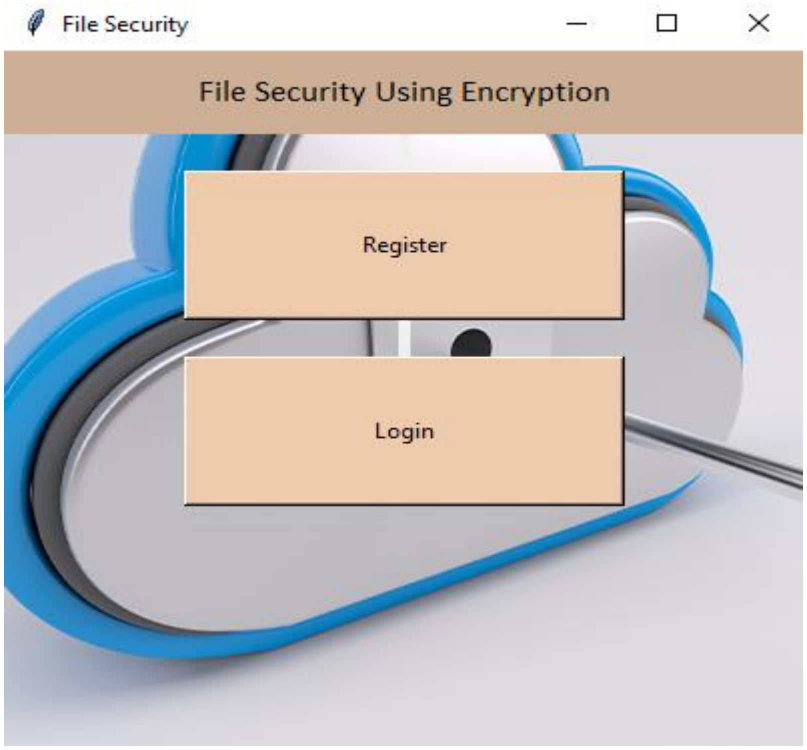
```
def delete2():
    screen3.destroy()



def delete3():
    screen4.destroy()



def delete4():
    screen5.destroy()



def logout():
    screen3.destroy()
    screen.destroy()
    main_screen()



def login_sucess():
    global screen3
    screen2.destroy()
    screen3 = Toplevel(screen)
    screen3.title("Encrypt or Decrypt")
    screen3.geometry("400x400")
    Label(screen3, text="Login Successful",bg="peachpuff3", height="3",width="300", font=("Calibri", 11)).pack()
    Label(screen3, text="").pack()
    Button(screen3, text="Encrypt",bg="peachpuff2", height="3",width="30", font=("Calibri", 11),
command=encryption).pack()
    Label(screen3, text="").pack()
    Button(screen3, text="Decrypt",bg="peachpuff2", height="3",width="30", font=("Calibri", 11),
command=decryption_password).pack()

    Label(screen3, text="").pack()

    Button(screen3, text="Logout",bg="peachpuff2", height="3",width="30", font=("Calibri", 11),
command=logout).pack()

    screen3.mainloop()




def password_not_recognised():

    global screen4

    screen4 = Toplevel(screen)

    screen4.title("Success")

    screen4.geometry("150x100")

    Label(screen4, text="Password Error").pack()

    Button(screen4, text="OK", command=delete3).pack()




def user_not_found():

    global screen5

    screen5 = Toplevel(screen)

    screen5.title("Success")

    screen5.geometry("150x100")

    Label(screen5, text="User Not Found").pack()

    Button(screen5, text="OK", command=delete4).pack()




def register_user():

    print("working")



    username_info = username.get()

    password_info = password.get()
    exists = os.path.isfile(username.get())
    if exists:
        messagebox.showerror("Error!", "Username already exists")
    else:
        primep_info = primep.get()
        primeq_info = primeq.get()
        file = open(username_info, "w")
        file.write(username_info + "\n")
        file.write(password_info + "\n")
        e, d, n = rsa(primep.get(), primeq.get())
        file.write(str(e) + "\n")
        file.write(str(d) + "\n")
        file.write(str(n) + "\n")
        file.close()

        username_entry.delete(0, END)
        password_entry.delete(0, END)
        primep_entry.delete(0, END)
        primeq_entry.delete(0, END)
        messagebox.showinfo("Complete", "Signup Successful")
        screen1.destroy()


def login_verify():
    global username1
    username1 = username_verify.get()
    print(username1)
    password1 = password_verify.get()
    username_entry1.delete(0, END)
    password_entry1.delete(0, END)

    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            login_sucess()
        else:
            password_not_recognised()

    else:
        user_not_found()



def register():    #part_2
    global screen1
    screen1 = Toplevel(screen)
    screen1.title("Register")
    screen1.geometry("400x400")

    global username
    global password
    global username_entry
    global password_entry
    global primep
    global primeq
    global primep_entry
    global primeq_entry
    username = StringVar()
    password = StringVar()

    primep = StringVar()

    primeq = StringVar()



    Label(screen1, text="Please enter details below", bg="peachpuff3", width="300", height="2", font=("Calibri",
13)).pack()

    Label(screen1, text="").pack()



    Label(screen1, text="Username * ", font=("Calibri", 11)).pack()

    username_entry = Entry(screen1, textvariable=username)

    username_entry.pack()



    Label(screen1, text="Password * ", font=("Calibri", 11)).pack()

    password_entry = Entry(screen1, textvariable=password)

    password_entry.pack()



    Label(screen1, text="1st Number * ", font=("Calibri", 11)).pack()

    primep_entry = Entry(screen1, textvariable=primep)

    primep_entry.pack()



    Label(screen1, text="2nd Number * ", font=("Calibri", 11)).pack()

    primeq_entry = Entry(screen1, textvariable=primeq)

    primeq_entry.pack()



    Label(screen1, text="").pack()

    Button(screen1, text="Register", width=10, height=1, font=("Calibri", 11), command=register_user).pack()##



def call1():

    password_entry1 = Entry(screen2, textvariable=password_verify, show=').pack()



def login():  #part_3
```

```
global screen2
screen2 = Toplevel(screen)
screen2.title("Login")
screen2.geometry("400x400")
Label(screen2, text="Please enter details below to login", bg="peachpuff3", width="300", height="2",
    font=("Calibri", 11)).pack()
Label(screen2, text="").pack()

global username_verify
global password_verify

username_verify = StringVar()
password_verify = StringVar()

global username_entry1
global password_entry1

Label(screen2, text="Username * ", font=("Calibri", 11)).pack()
username_entry1 = Entry(screen2, textvariable=username_verify)
username_entry1.pack()
Label(screen2, text="").pack()
Label(screen2, text="Password * ", font=("Calibri", 11)).pack()
password_entry1 = Entry(screen2, textvariable=password_verify,show='*')
password_entry1.pack()
Button(screen2, text="SHOW", width=10, height=1, command=call1).pack()
Label(screen2, text="").pack()
Label(screen2, text="").pack()
Button(screen2, text="Login", width=10, height=1, command=login_verify).pack()
```

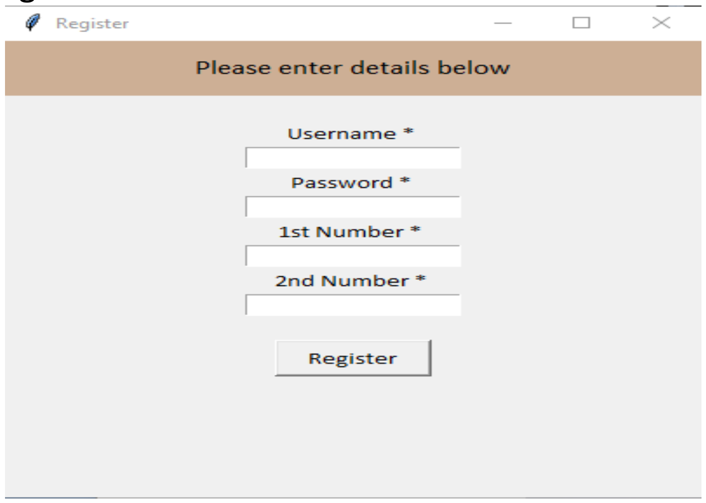
```
def main_screen(): #part_1

    global screen

    screen = Tk()

    screen.geometry("400x400")

    bg = PhotoImage(file="sec_1.png")

    my_label = Label(screen, image=bg)

    my_label.place(x=0, y=0, relwidth=1, relheight=1)

    screen.title("File Security ")

    Label(text="File Security Using Encryption", bg="peachpuff3", width="300", height="2", font=("Calibri",
13)).pack()

    Label(text="").pack()

    Button(text="Register", bg="peachpuff2", height="5", width="30", command=register).pack()

    Label(text="").pack()

    Button(text="Login",bg="peachpuff2", height="5", width="30", command=login).pack()






    screen.mainloop()




main_screen()
```

Screen Shots:

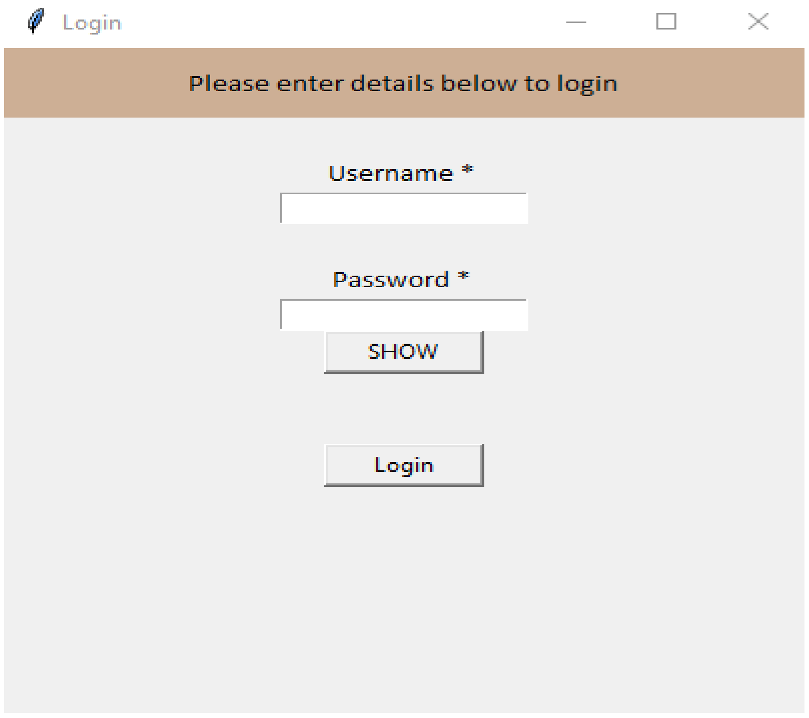
* Main Screen



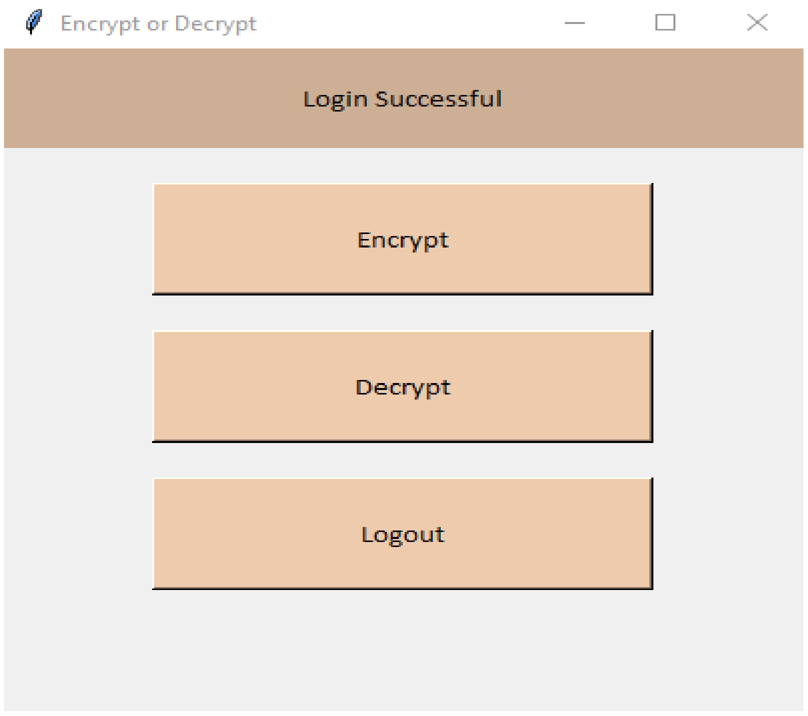
* Registration



* Login

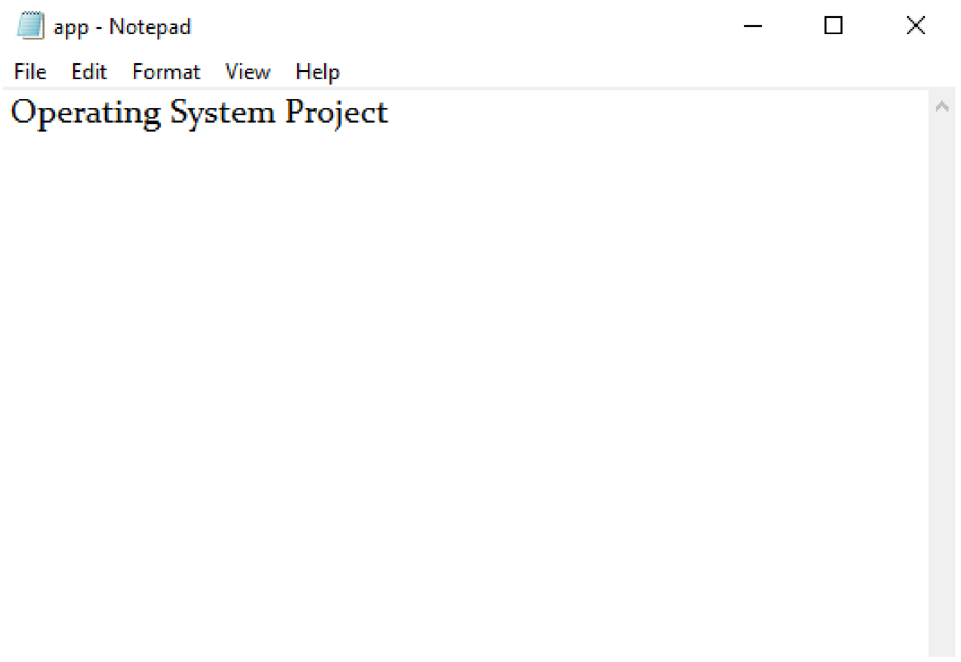


* Encryption and Decryption



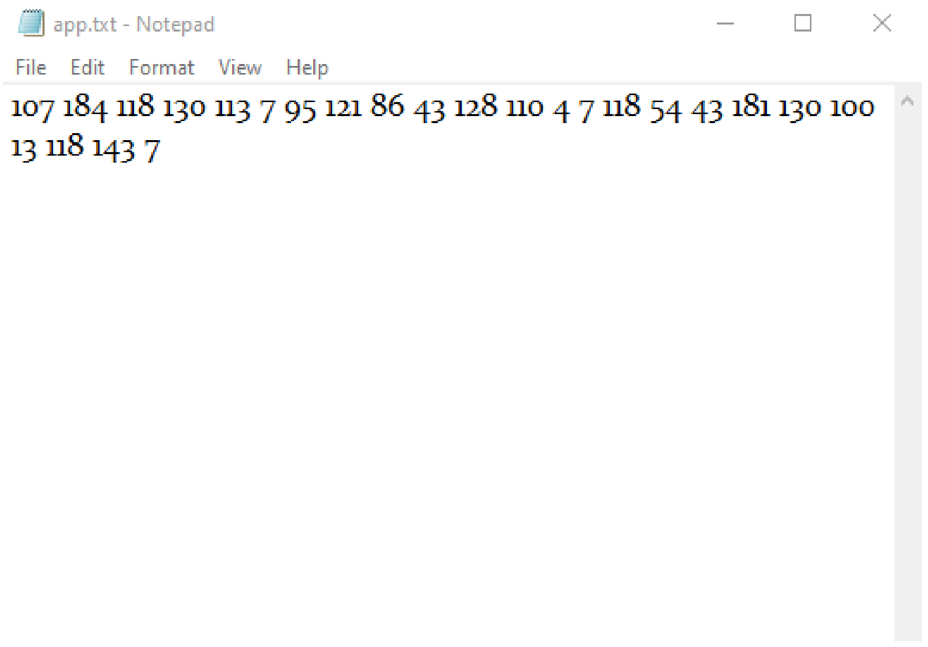
Sample File Encryption and Decryption:

* Before Encryption



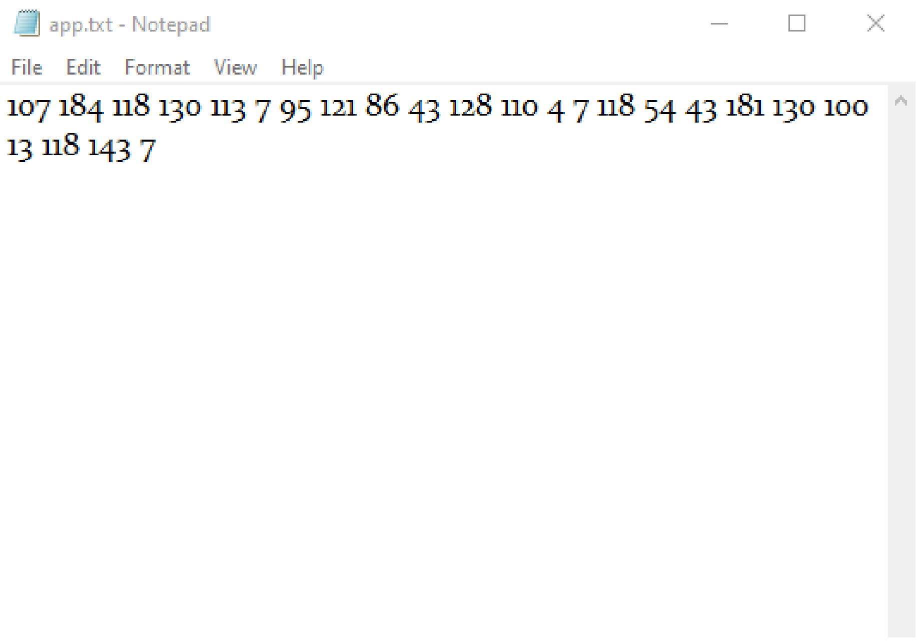
```
Operating System Project
```

* After Encryption



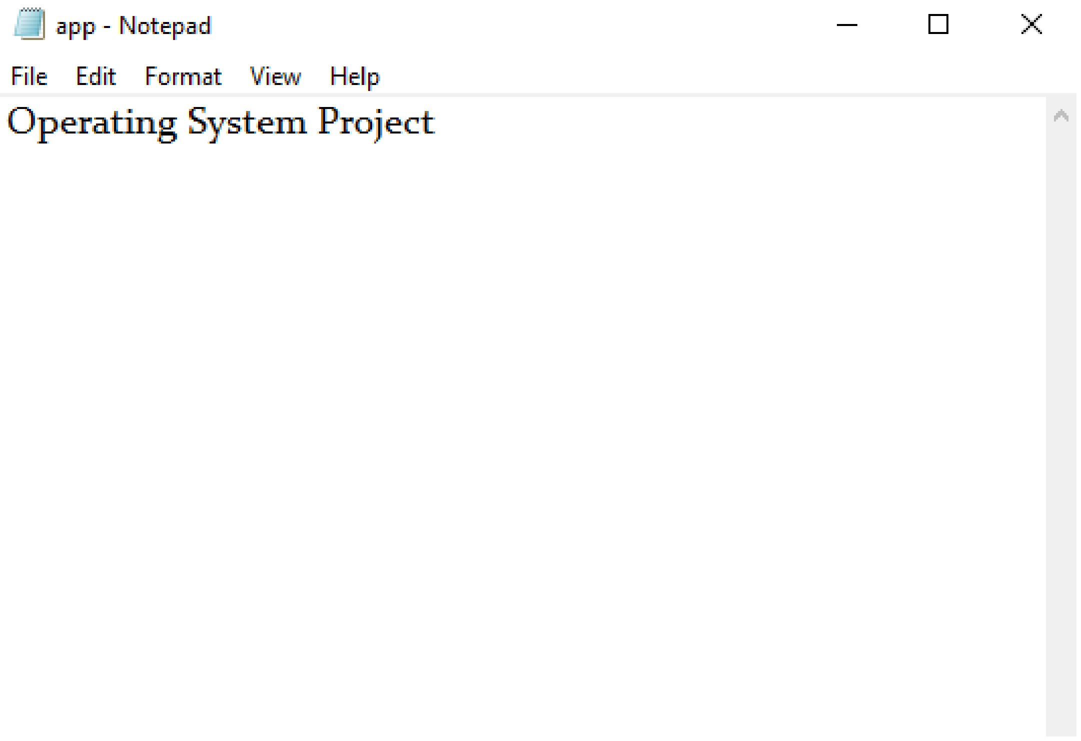
```
107 184 118 130 113 7 95 121 86 43 128 110 4 7 118 54 43 181 130 100
13 118 143 7
```

* Before Decryption



```
107 184 118 130 113 7 95 121 86 43 128 110 4 7 118 54 43 181 130 100
13 118 143 7
```

* After Decryption



```
Operating System Project
```

<u>Conclusion :</u>

The public key crypto system solves one of the biggest problems in key management by allowing public part of the key pair available to every one. This cuts down the key management overhead enormously. Public key- based systems provide more flexibility and ensure that node compromise does not affect uncompromised network nodes. Group communication confidentiality requires that only valid users could decrypt the multicast data even if the data is broadcast to the entire network. Therefore, a pair of keys such as public and private keys must be shared securely between the valid group members. The public key is used to encrypt data by the source and private key is used to decrypt it by valid group members to recover original data. The public and private keys were generated using RSA-Public key cryptographic algorithm.

During at registration we are supposed to give any two numbers those two numbers are converted to nearest prime numbers and the private and public keys are calculated from them.
We made a login portal from where a user will be directed to Encryption or Decryption page. To encrypt a file we need to select Encryption on this page and we need to select the file that is need to be encrypted then the file will get encrypted using public key in RSA algorithm. To decrypt a file we need to go back to the Encryption or Decryption page and select Decryption then we need to enter the password (the password we used to login) then if it is correct we need to select a file to decrypt using private key the file will be decrypted.

References :

https://www.sciencedirect.com/topics/computer-science/file-encryption
https://www.geeksforgeeks.org/rsa-algorithm-cryptography/
https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_understanding_rsa_algorithm.htm