Kingdom of Saudi Arabia

Ministry of Higher Education

l-Imam Muhammad Ibn Saud Islamic University

College of Computer and Information Sciences

Information Security CS 334

Section: 372

Encryption and Decryption Application Project

Date of submission: 08/05/2022

Name	Student ID
Hanan Almutairi	

Instructor:

Table of contents

Table of Figures:

○ Overview of the project design

In the Information Security course, we learned the classic encryption methods and detailed information about them. One of the technologies is The Advanced Encryption Standard (AES) is a symmetric block cipher implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity, and electronic data protection [1].

And the way it works in the following scenario:

Bob and Alice have the same encryption key. In the following, Bob and Alice share an encryption key, and where Bob converts his plaintext into ciphertext, and then Alice converts the ciphertext back into plaintext using a shared secret key [2].

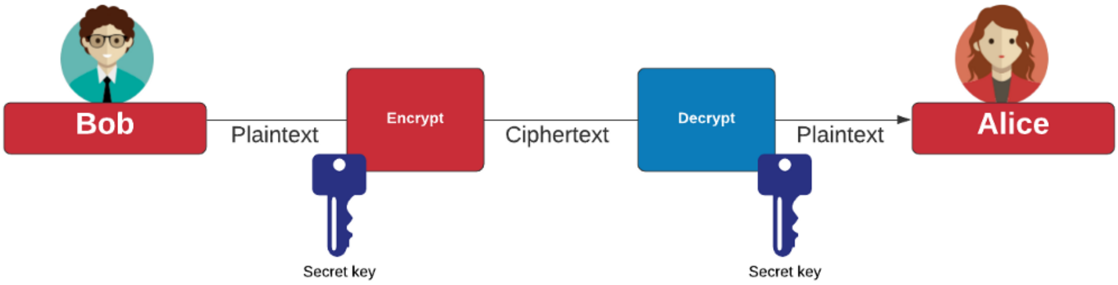


Figure 1 : AES Working

Another important technique we want to focus on is The Rivest-Shamir-Adleman (RSA) encryption algorithm an asymmetric encryption algorithm that is widely used in many products and services. Asymmetric encryption uses a key pair that is mathematically linked to encrypt and decrypt data. A private and public key are created, with the public key being accessible to anyone and the private key being a

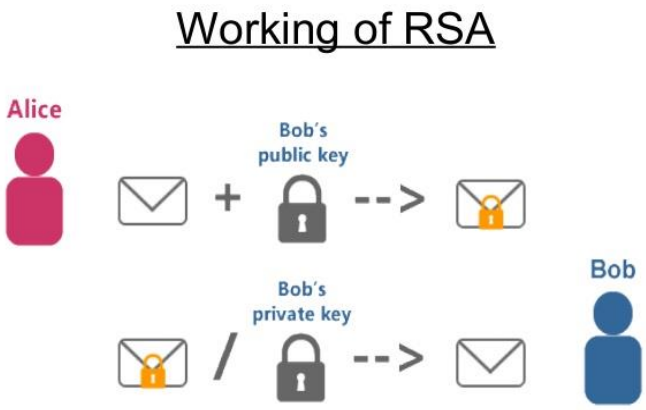


Figure 2 :RSA Working

secret known only by the key pair creator. With RSA, either the private or public key can encrypt the data, while the other key decrypts it. This is one of the reasons RSA is the most used asymmetric encryption algorithm [3]. Another important technique we want to focus on is The Rivest-Shamir-Adleman (RSA) encryption algorithm an asymmetric encryption algorithm that is widely used in many products and services. Asymmetric encryption uses a key pair that is mathematically linked to encrypt and decrypt data. A private and public key are created, with the public key being accessible to anyone and the private key being a secret known only by the key pair creator. With RSA, either the private or public key can encrypt the data, while the other key decrypts it. This is one of the reasons RSA is the most used asymmetric encryption algorithm [3].

So, we will take charge from *XSecure*, a great cyber security company to design a secure file-sharing application.

○ Approach and steps to implementation.

1. The language:

We used the Python language for ease of handling and because it contains many

useful libraries that made it easy for us to program the application

2. Application building:

First, we programmed the GUI and the AES, RSA algorithms, and the login system separately and then collected them in one file to complete the application.

2.1 GUI:

To program the GUI, we used some useful libraries from python, where we used the ***Tkinter*** library in addition to the ***messagebox*** from ***Tkinter*** [4-8].

- First, Easy-to-understand graphic user interface design to interact with users who want to encrypt/decrypt files.

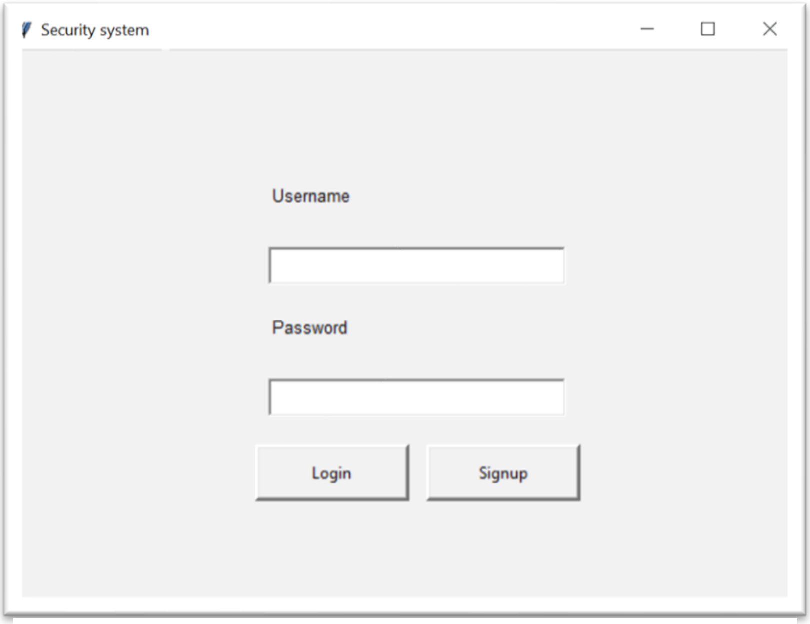
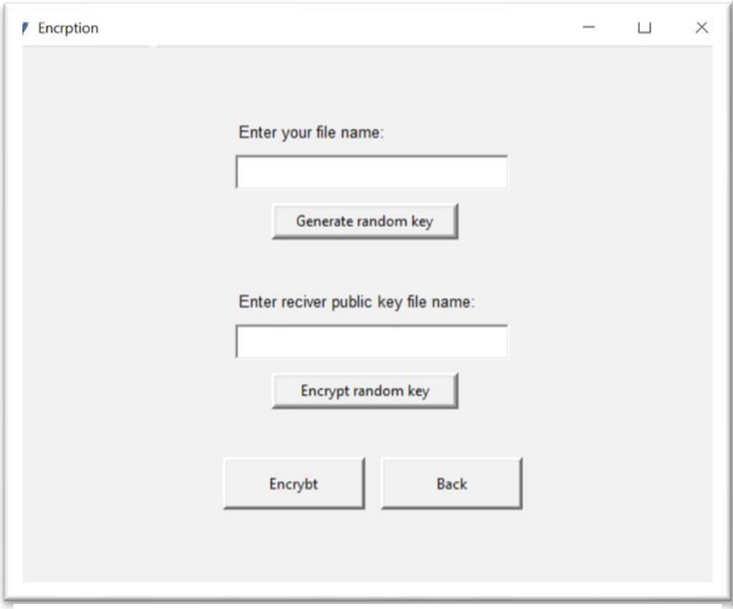


Figure 3 :GUI Main Page

- Second, Using the AES algorithm to encrypt/decrypt the shared files.



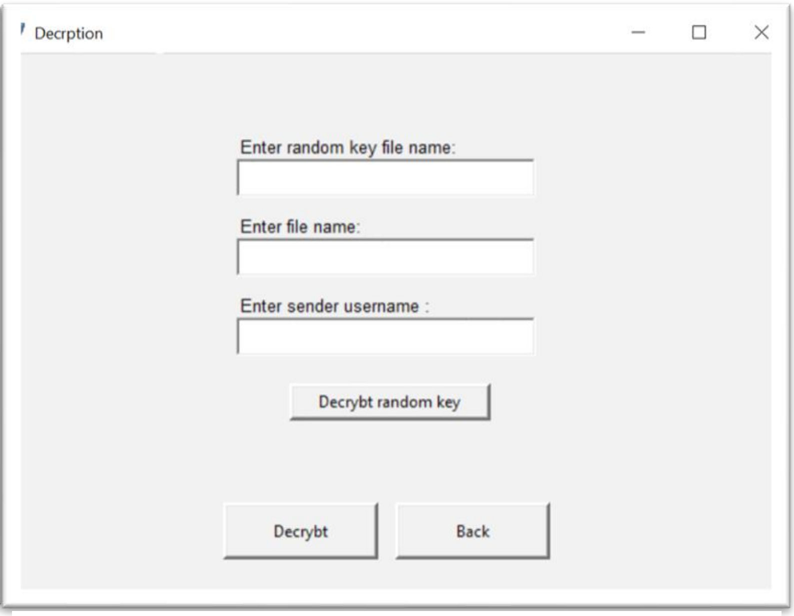
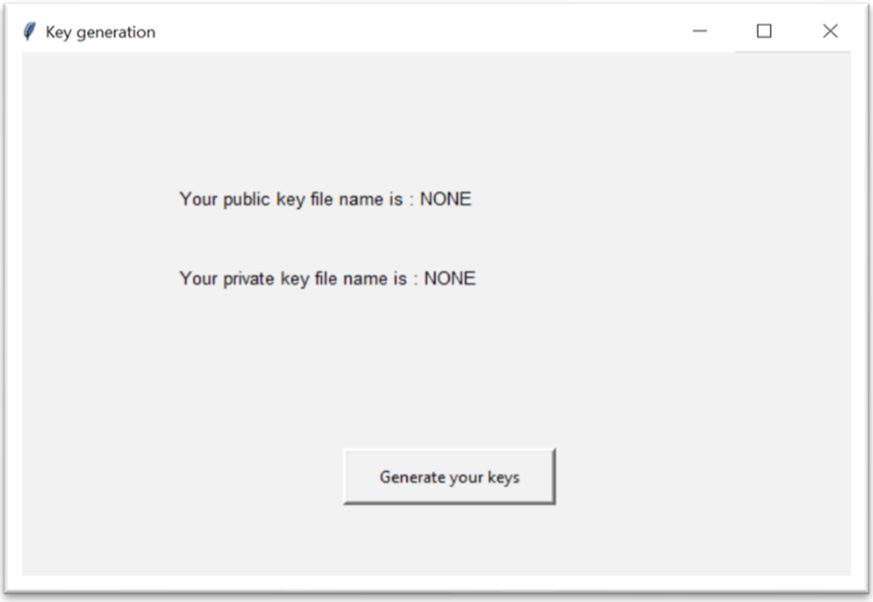


Figure 4: Decryption page

- Third, Using the RSA algorithm to generate Public & Private keys.



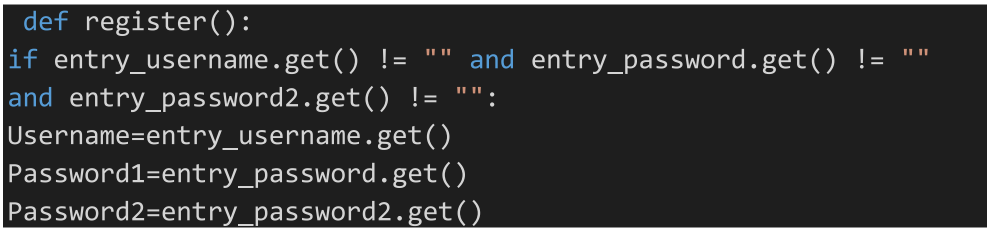
Your public key file name is : NONE

Your private key file name is : NONE

Figure6: Key Generation

2.2 Signup

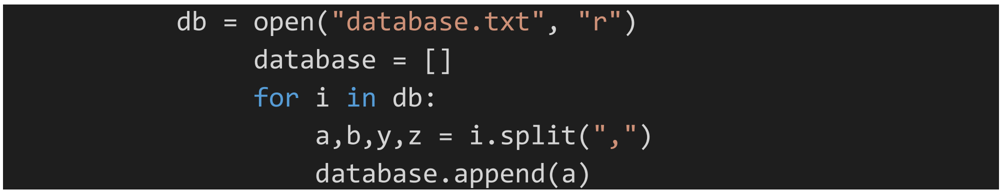
- The first thing you need to do if you are a new user is to register
  We will review the methods for signing up here.



```
 def register():
if entry_username.get() != "" and entry_password.get() != ""
and entry_password2.get() != "":
Username=entry_username.get()
Password1=entry_password.get()
Password2=entry_password2.get()
```

Figure 7: Register Method Code1

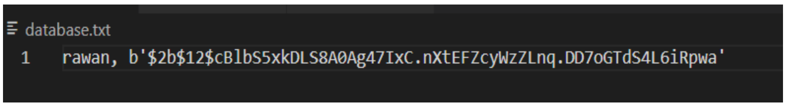
- First, we will check if the user has entered empty values or not
  Otherwise, it will take the values from the user, which is the username and password, and require the user to type the password again.



```
            db = open("database.txt", "r")
                 database = []
                 for i in db:
                     a,b,y,z = i.split(",")
                     database.append(a)
```

Figure 8: register Method Code2

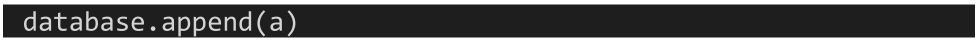
- Then we will open a database to store user data , and we need to create new array .
- In this file, we will save the data separated by commas, and here we will show what data we have saved so far
  a "username", b "password", y "public key", z "private key"



database.txt

```
rawan, b'$2b$12$cBlbS5xkDLS8A0Ag47IxC.nXtEFZcyWzZLnq.DD7oGTdS4L6iRpwa'
```

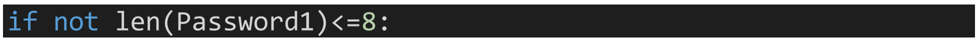
Figure 9:database file



```
 database.append(a)
```

Figure 10: register Method Code3

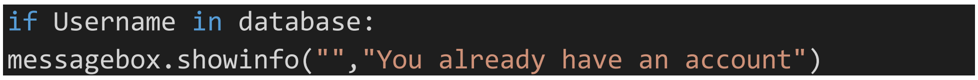
- The first value we will save is the username.



```
if not len(Password1)<=8:
```

Figure 11:register Method Code4

- In order to increase security, we have specified a password length of 8 characters or more.

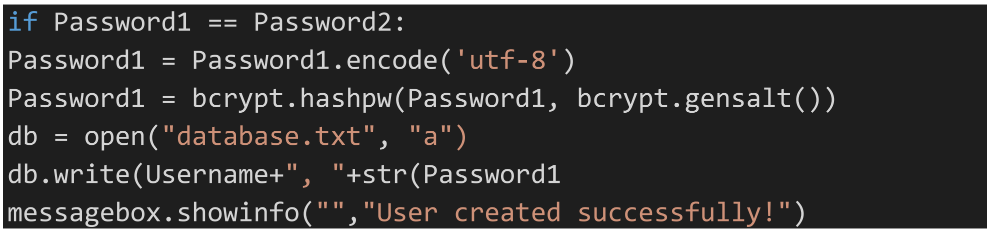


```
if Username in database:
messagebox.showinfo("","You already have an account")
```

Figure 12:register Method Code5

- After that, it will check if the user has already been registered in the database.

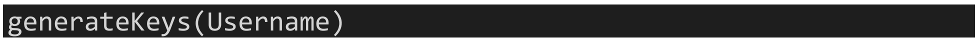
If your account already exists, a message will appear to him telling him that you have one.



```
if Password1 == Password2:
Password1 = Password1.encode('utf-8')
Password1 = bcrypt.hashpw(Password1, bcrypt.gensalt())
db = open("database.txt", "a")
db.write(Username+", "+str(Password1
messagebox.showinfo("","User created successfully!")
```

Figure 13:register Method Code6

- If he does not have an account before, he will check if the password he entered matches or not
- The encode() method returns the byte of the password, while the hashpw() method hashes the password.
- Then, open the database file that contains all the users' data for writing
- Enter the username and password as a string and separate them with a comma.



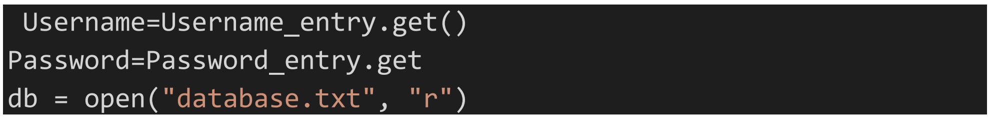
```
generateKeys(Username)
```

Figure 14:register Method Code7

- After completing his data entry, he must generate the keys using this method.

2.3 LogIn

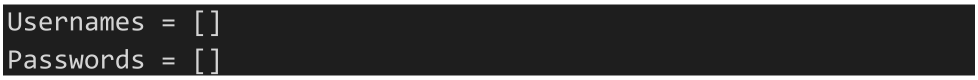
In case the user has an existing account, he must log in.



```
 Username=Username_entry.get()
Password=Password_entry.get
db = open("database.txt", "r")
```

Figure 15:gainAccess Method Code1

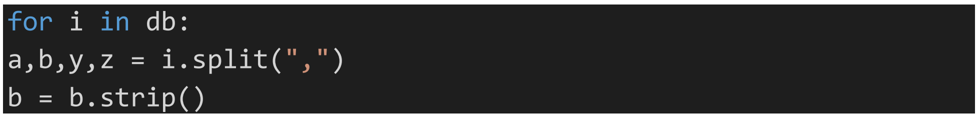
- Once the user's data is obtained, the database is opened.



```
Usernames = []
Passwords = []
```

Figure 16:gainAccess Method Code2

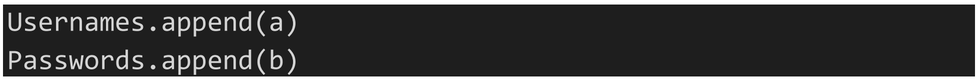
- create a new array to save all the usernames and password on it.



```
for i in db:
a,b,y,z = i.split(",")
b = b.strip()
```

Figure 17:gainAccess Method Code3

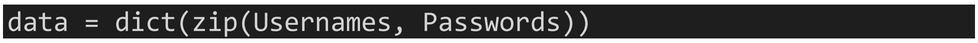
- Returns each line in the file and The split() function returns the strings as a list seperated by commas. The strip() method removes whitespace from the beginning and at the end of the string.



```
Usernames.append(a)
Passwords.append(b)
```

Figure 18:gainAccess Method Code4

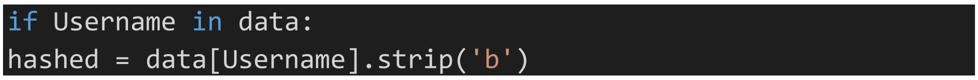
- Then add your username and password in the new array



```
data = dict(zip(Usernames, Passwords))
```

Figure 19:gainAccess Method Code5

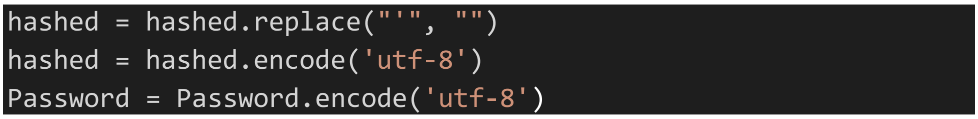
- There is a zip() function that will aggregate elements from two or more and a dict() function for storing values in key:value pairs.



```
if Username in data:
hashed = data[Username].strip('b')
```

Figure 20:gainAccess Method Code6

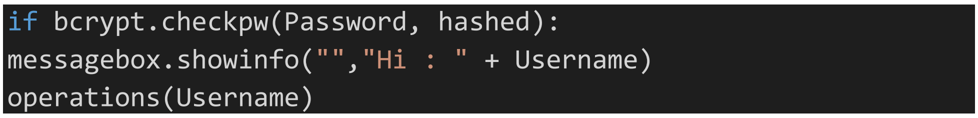
- If the user is in the database, take the second item in the list that has the same username.



```
hashed = hashed.replace("'", "")
hashed = hashed.encode('utf-8')
Password = Password.encode('utf-8')
```

Figure 21:gainAccess Method Code7

- Remove all backticks from the hashed password, then convert it from string to byte, both from the password entered by the user and from the database.



```
if bcrypt.checkpw(Password, hashed):
messagebox.showinfo("","Hi : " + Username)
operations(Username)
```

Figure 22:gainAccess Method Code8

- Check that an unhashed password matches the hashed password, and if they do, display a welcome message before directing the user to the options page.

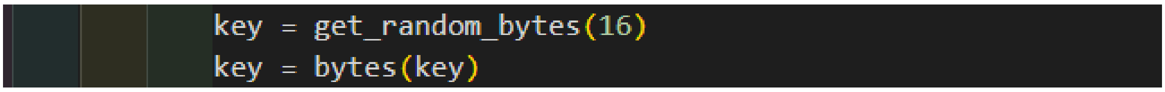
2.4 AES Algorithm:

To apply the AES algorithm programmatically, we used ***Crypto.Cipher*** and several help libraries, which are ***Crypto.Random***, ***os***, ***struct*** [9,10]. Our use of it is detailed in the next sections.

2.4.1 Files Encryption using AES algorithm:

2.4.1.1 Generating a Key:

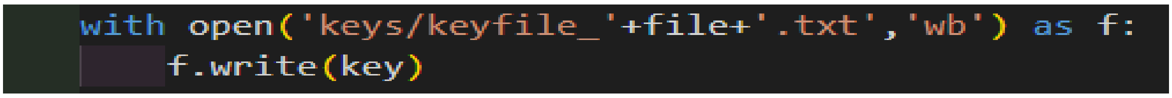
- in these two lines we generate a 16-byte random key using the ***get_random_bytes()*** function from ***Crypto.Random*** library.



```
key = get_random_bytes(16)
key = bytes(key)
```

Figure 23:Generating The Key Method Code1

- Then the random key is written to a file with the name consist the ***keyfile*** + the name of the file to be encrypted so that it can be accessed later.

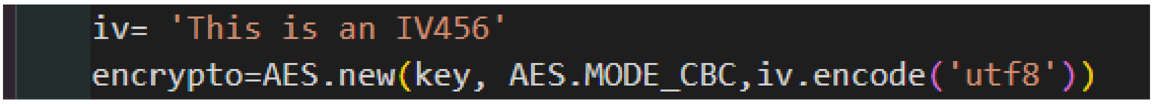


```
with open('keys/keyfile_'+file+'.txt','wb') as f:
    f.write(key)
```

Figure 24:Generating The Key Method Code2

2.4.1.2 Initialization Vector:

- A 16-byte ***initialization vector*** is generated as shown in, and its purpose is to produce different encrypted data.
- AES encryption cipher(***encrypto***) is generated with ***CBC Mode***(wherein each block is “chained” to the previous block in the stream) and passed to it the Key and the ***Initialization Vector***

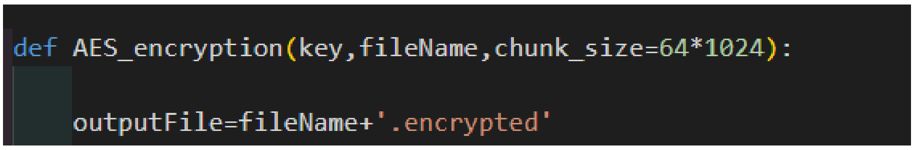


```
iv= 'This is an IV456'
encrypto=AES.new(key, AES.MODE_CBC,iv.encode('utf8'))
```

Figure 25:Initialization Vector and encryption cipher generation

2.4.1.3 Encrypting The file:

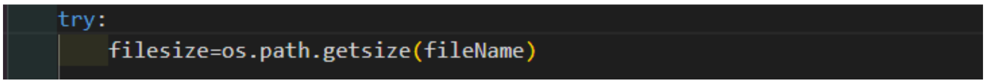
- Encryption method receives the key after it has been read from the file, and the name of the file to be encrypted.
- Extension "***.encrypted***" is added to the name of the file to be encrypted to mark the file after the encryption process.



```
def AES_encryption(key,fileName,chunk_size=64*1024):

    outputFile=fileName+'.encrypted'
```

Figure 26:AES Encryption Method Code1

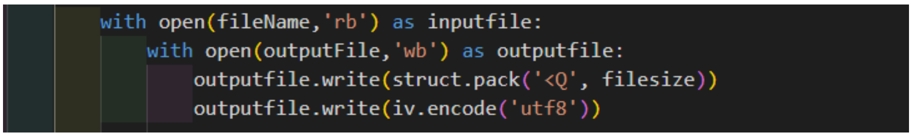
- We have to write the size of the file being encrypted to the output file, so first, we determine the size by the ***getsize()*** Function.



```
try:
    filesize=os.path.getsize(fileName)
```

Figure 27:AES Encryption Method Code2

- after opening the output file, we write the size of the file by using the ***struct.pack().***
- The ***initialization vector*** is also written to the output file
- The file to be encrypted is also opened to read the data from it.

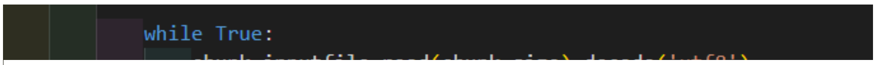


```
with open(fileName,'rb') as inputfile:
    with open(outputFile,'wb') as outputfile:
        outputfile.write(struct.pack('<Q', filesize))
        outputfile.write(iv.encode('utf8'))
```

Figure 28:AES Encryption Method Code3

-

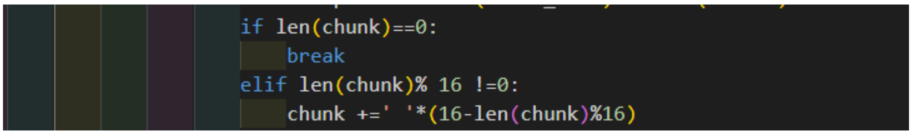
- The data is read from the file as chunks multiple 16 bytes in size.
- And it is converted into a String before being saved in the variable ***chunk*** by ***decode***() function so that it is easier to deal with it in case it needs to be padded.



```
while True:
    chunk=inputfile.read(chunk_size).decode('utf8')
```

Figure 29:AES Encryption Method Code4

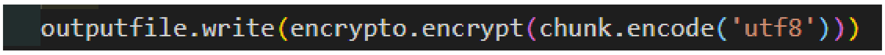
- If there is no data to read, ***break***.
- If there is data and it's not multiple of 16 bytes in size, then padding it



```
if len(chunk)==0:
    break
elif len(chunk)% 16 !=0:
    chunk +=' '*(16-len(chunk)%16)
```

Figure 30:AES Encryption Method Code5

- Then the data is encrypted using ***encrypt()*** function and written after converting it to bytes in the ***outputfile***.



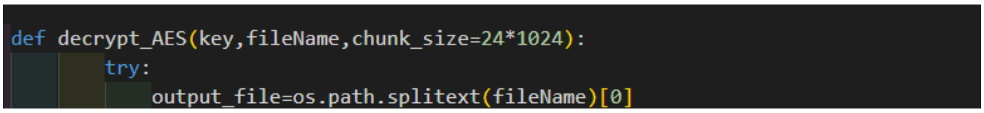
```
outputfile.write(encrypto.encrypt(chunk.encode('utf8')))
```

Figure 31:AES Encryption Method Code6

2.4.2 Files Decryption using AES algorithm:

After decrypting the key with the RSA algorithm, we decrypt the file encrypted with that key .*For the key decryption process see section*

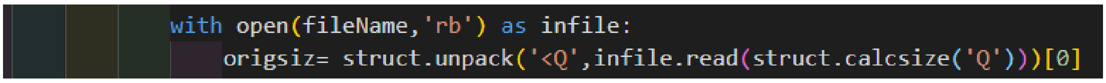
- First, the extension that was added previously is removed by ***splittext***() function so that the file is distinguished after decryption.



```
def decrypt_AES(key,fileName,chunk_size=24*1024):
    try:
        output_file=os.path.splitext(fileName)[0]
```

Figure 32:AES Decryption Method Code1

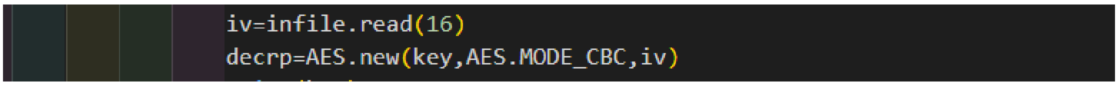
- The Encrypted file is opened and the size of the file we wrote previously is read and saved to ***origsize*** variable.



```
with open(fileName,'rb') as infile:
    origsiz= struct.unpack('<Q',infile.read(struct.calcsize('Q')))[0]
```

Figure 33:AES Decryption Method Code2

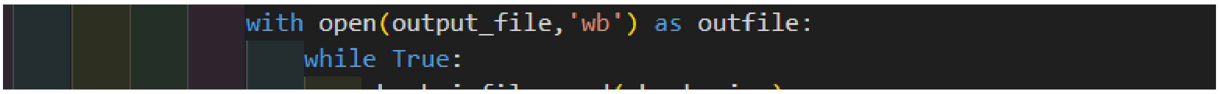
- The ***initialization vector*** is read from the encrypted file
- the Decryption cipher (***decrp***) using the ***key*** and the ***iV*** is created



```
iv=infile.read(16)
decrp=AES.new(key,AES.MODE_CBC,iv)
```

Figure 34:AES Decryption Method Code3

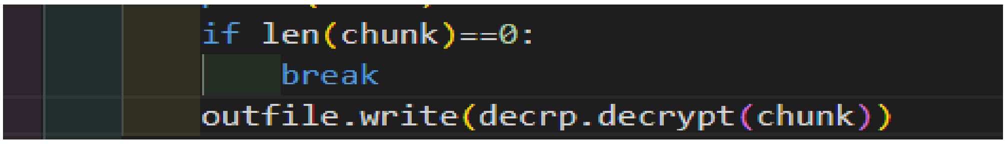
- The ***output_file*** is opened, and the data is read as chunks of multiple 16 bytes in size



```
with open(output_file,'wb') as outfile:
    while True:
```

Figure 35:AES Decryption Method Code4

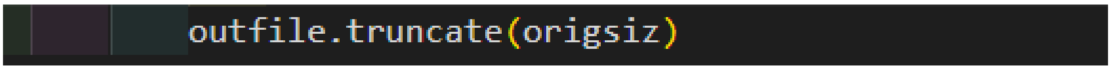
- If there is no data to read, then ***break***
- If there is data, the data is decrypted using the ***decrypt***() function, Then the text after decryption is written to the ***outputFile***.



```
if len(chunk)==0:
    break
outfile.write(decrp.decrypt(chunk))
```

Figure 36:AES Decryption Method Code5

- The file is truncated using the ***truncate***() function after decryption to the original size before encryption, the padding is removed and returned to the original size.



```
outfile.truncate(origsiz)
```

Figure 37:AES Decryption Method Code37

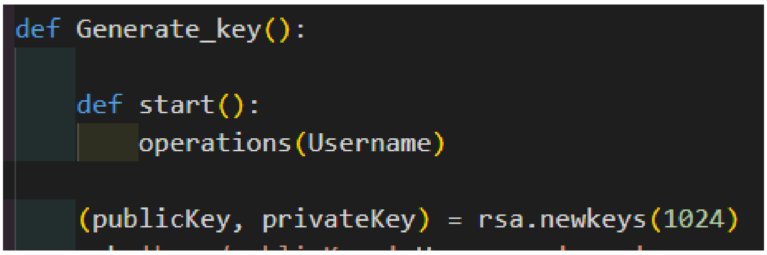
2.5 RSA Algorithm:

Using the rsa library from Python, we implemented the RSA algorithm programmatically [11]. We have detailed our use of its methods in the following sections.

2.5.1 Key Encryption using RSA algorithm:

2.5.1.1 Generating the Keys

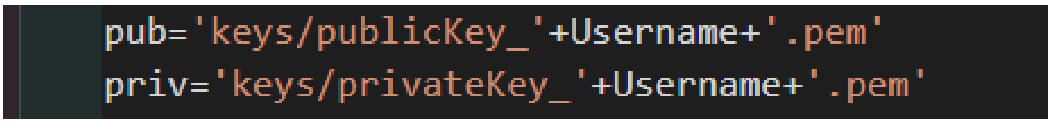
by using ***rsa.newkeys(1024)*** we create keys of 1024 bits and will save them into a tuple of public and private keys.



```
def Generate_key():

    def start():
        operations(Username)

    (publicKey, privateKey) = rsa.newkeys(1024)
```

Figure 38:RSA Generating The keys Method Code1

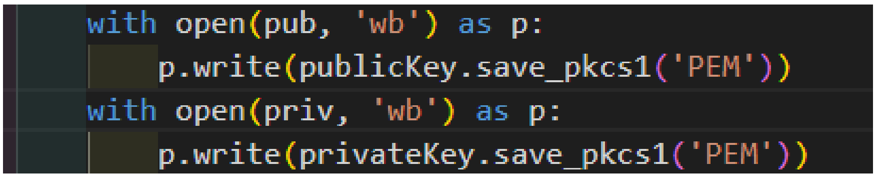
- Then a ***key*** folder is created that contains two files, one for the public key and the other for the private. The file names are modified to contain the user name in order to facilitate access to it.



```
pub='keys/publicKey_'+Username+'.pem'
priv='keys/privateKey_'+Username+'.pem'
```

Figure 39:RSA Generating The keys Method Code2

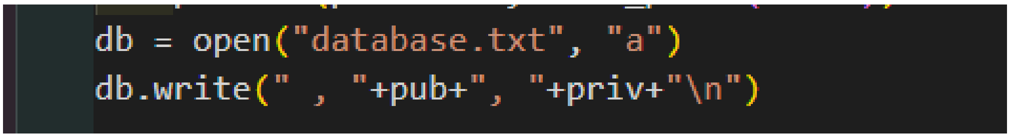
- Files assigned to keys are opened and the keys are saved in ***pem*** format by using the ***save_pkcs1('PEM')*** function, where they are encoded in ***base 64*** before being written to the files.



```
with open(pub, 'wb') as p:
    p.write(publicKey.save_pkcs1('PEM'))
with open(priv, 'wb') as p:
    p.write(privateKey.save_pkcs1('PEM'))
```

Figure 40:RSA Generating The keys Method Code2

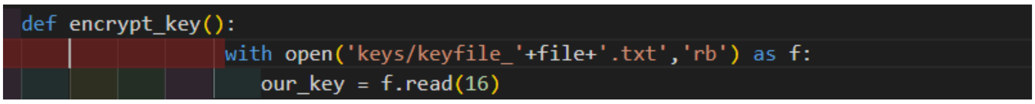
- The database is opened and the key file names are saved along with the rest of the user's information.



```
db = open("database.txt", "a")
db.write(" , "+pub+", "+priv+"\n")
```

2.5.1.2 Encrypt The Key

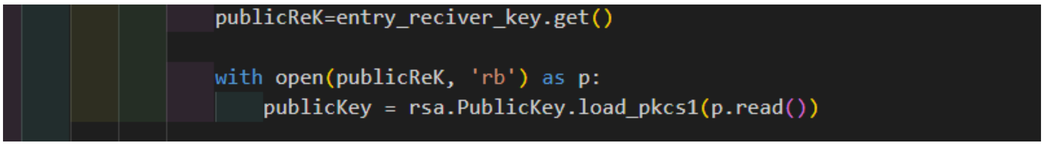
- The AES key file used to encrypt the files is opened and the key is read to start the encryption process



```
def encrypt_key():
        with open('keys/keyfile_'+file+'.txt','rb') as f:
            our_key = f.read(16)
```

Figure 41: RSA Encrypt The Key Method Code1

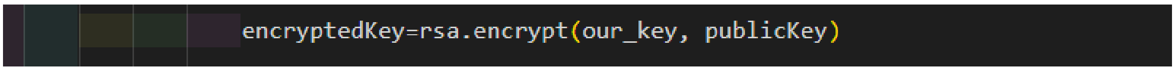
- The receiver's public key file name is received. and The file is opened, the key is read and decoded before it is saved in the ***publickey*** variableble.



```
publicReK=entry_reciver_key.get()

with open(publicReK, 'rb') as p:
    publicKey = rsa.PublicKey.load_pkcs1(p.read())
```

Figure42:RSA Encrypt The Key Method Code2

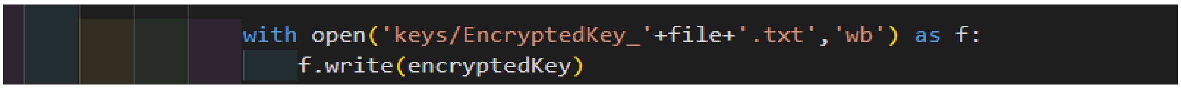
- With the ***rsa.encrypt***() function, the key is encrypted. This function receives the key to be encrypted and the public key of the receiver, The encrypted key is saved in the variable ***encryptedKey***



```
encryptedKey=rsa.encrypt(our_key, publicKey)
```

Figure 43:RSA Encrypt The Key Method Code3

- The encrypted key is written to a file whose name consists of ***EncryptedKey*** + the name of the file that was encrypted with this key.



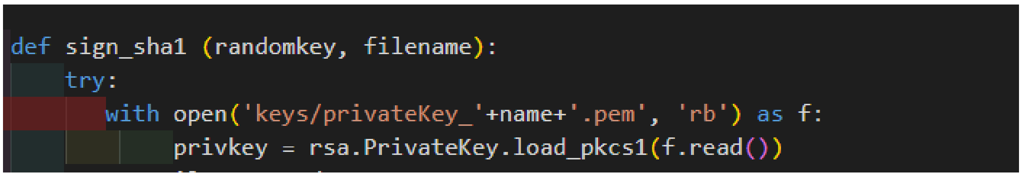
```
with open('keys/EncryptedKey_'+file+'.txt','wb') as f:
    f.write(encryptedKey)
```

Figure 44:RSA Encrypt The Key Method Code4

2.5.1.3 Sign The key

- The function ***sign_sha1()*** receives the encrypted key and the name of the file that is encrypted with the encrypted key.

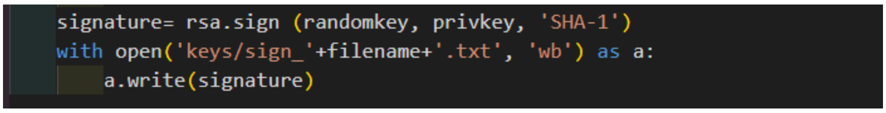
- The user's private key file is opened, and the key is read and saved to ***privkey*** variable.



```
def sign_sha1 (randomkey, filename):
    try:
        with open('keys/privateKey_'+name+'.pem', 'rb') as f:
            privkey = rsa.PrivateKey.load_pkcs1(f.read())
```

Figure 45:RSA Sign The key Code1

- Using ***rsa.sign()*** the encrypted key is signed where the encrypted key and the private key of the sender are sent to this function and by using the ***SHA-1*** hashing algorithm the signature is created.
- A file is created whose name consists of ***sign*** + the name of the encrypted file, and the signature is written in this file so that the receiver can verify it later.



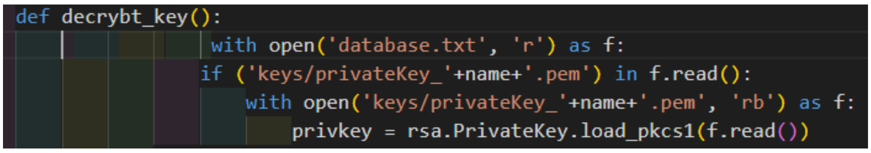
```
    signature= rsa.sign (randomkey, privkey, 'SHA-1')
    with open('keys/sign_'+filename+'.txt', 'wb') as a:
        a.write(signature)
```

Figure 46:RSA Sign The key Code2

2.5.2 Key Decryption using RSA algorithm

2.5.2.1 Decrypt The key

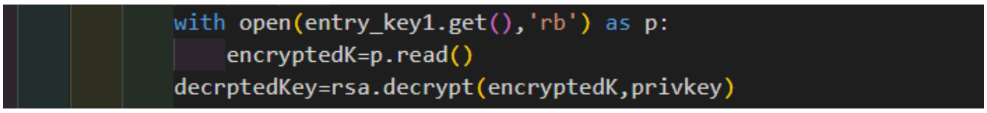
- First, open the database and make sure that the receiver's private key file exists
- The private file is opened and the key is saved in the ***privkey*** variable.



```
def decrybt_key():
                with open('database.txt', 'r') as f:
               if ('keys/privateKey_'+name+'.pem') in f.read():
                   with open('keys/privateKey_'+name+'.pem', 'rb') as f:
                       privkey = rsa.PrivateKey.load_pkcs1(f.read())
```

Figure 47:RSA Decrypt The key Method Code1

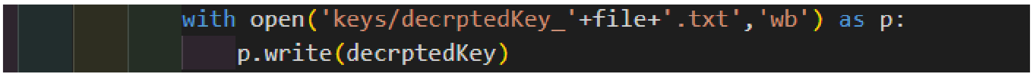
- The name of the encrypted key is received so that the file is opened and the encrypted key is read
- Using the ***rsa.decrypt()*** function, the key is decrypted, and the encrypted key and the private key of the receiver are sent to this function.



```
            with open(entry_key1.get(),'rb') as p:
                encryptedK=p.read()
            decrptedKey=rsa.decrypt(encryptedK,privkey)
```

Figure 48:RSA Decrypt The key Method Code2

- A file is created in which the decrypted key is written and the name of this file consists of ***decriptedKey*** + the name of the file to be decrypted using this key.

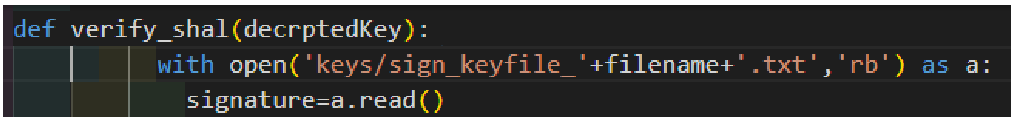


```
            with open('keys/decrptedKey_'+file+'.txt','wb') as p:
                p.write(decrptedKey)
```

Figure49:RSA Decrypt The key Method Code3

2.5.2.2 Verify The Key:

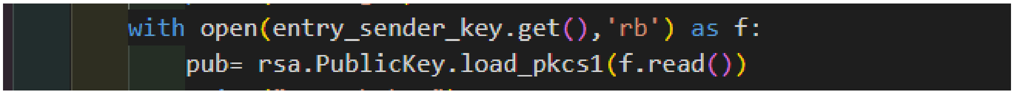
- The file containing the signature is opened and the signature is read from it



```
def verify_shal(decrptedKey):
        with open('keys/sign_keyfile_'+filename+'.txt','rb') as a:
            signature=a.read()
```

Figure 50:RSA Verify The Key Method Code1

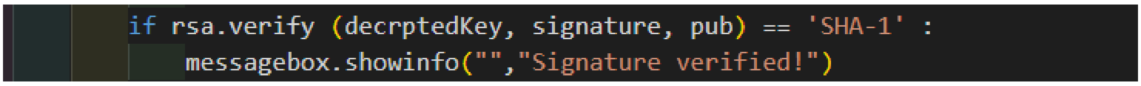
- The name of the sender's public key is received for use in key verification.
- Then the file public key of the sender is read and saved in the ***pub*** variable



```
        with open(entry_sender_key.get(),'rb') as f:
            pub= rsa.PublicKey.load_pkcs1(f.read())
```

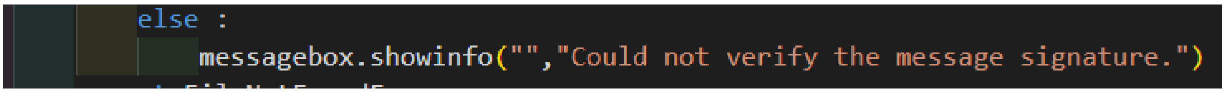
Figure 51:RSA Verify The Key Method Code2

- With ***rsa.verify()*** function, the decrypted key, signature, and public key of the sender are received, and verification of the decrypted key begins.
- This verification method returns the hash algorithm used in the signature. So if the used algorithm equals ***SHA-1***.
- then the signature is authentic and shows a success message



```
        if rsa.verify (decrptedKey, signature, pub) == 'SHA-1' :
            messagebox.showinfo("","Signature verified!")
```

Figure 52:RSA Verify The Key Method Code3



```
        else :
            messagebox.showinfo("","Could not verify the message signature.")
```

Figure 53:RSA Verify The Key Method Code4

- In case there is an exception, the verification has failed message will show. This means either the message or the signature was manipulated and is not authentic.

- **Code**.

1. The Application Flow:

In the picture below, we explained the interrelationship of the application methods to implement the tasks provided by the application, which are encryption, decryption, and finding public keys for users.

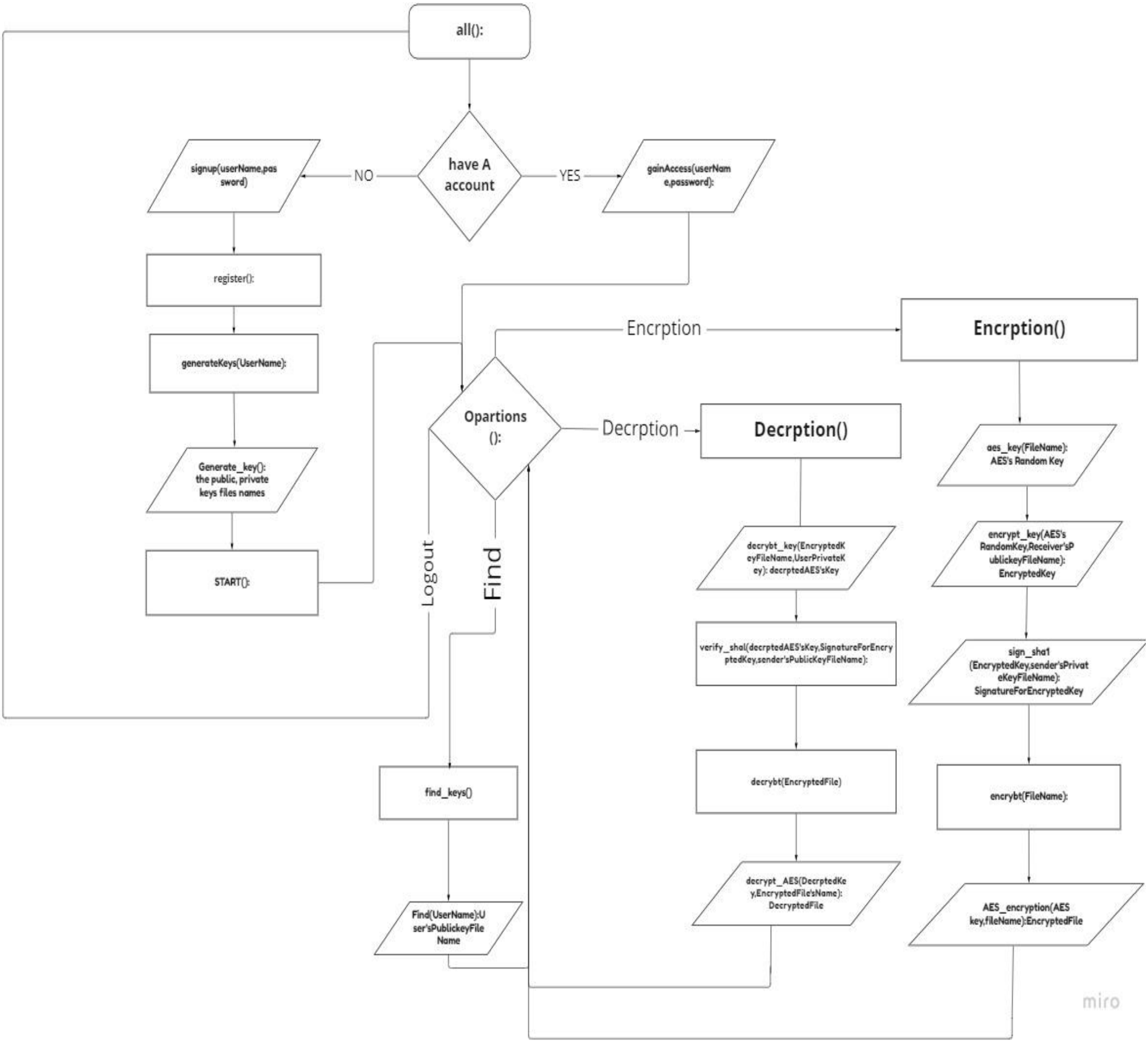
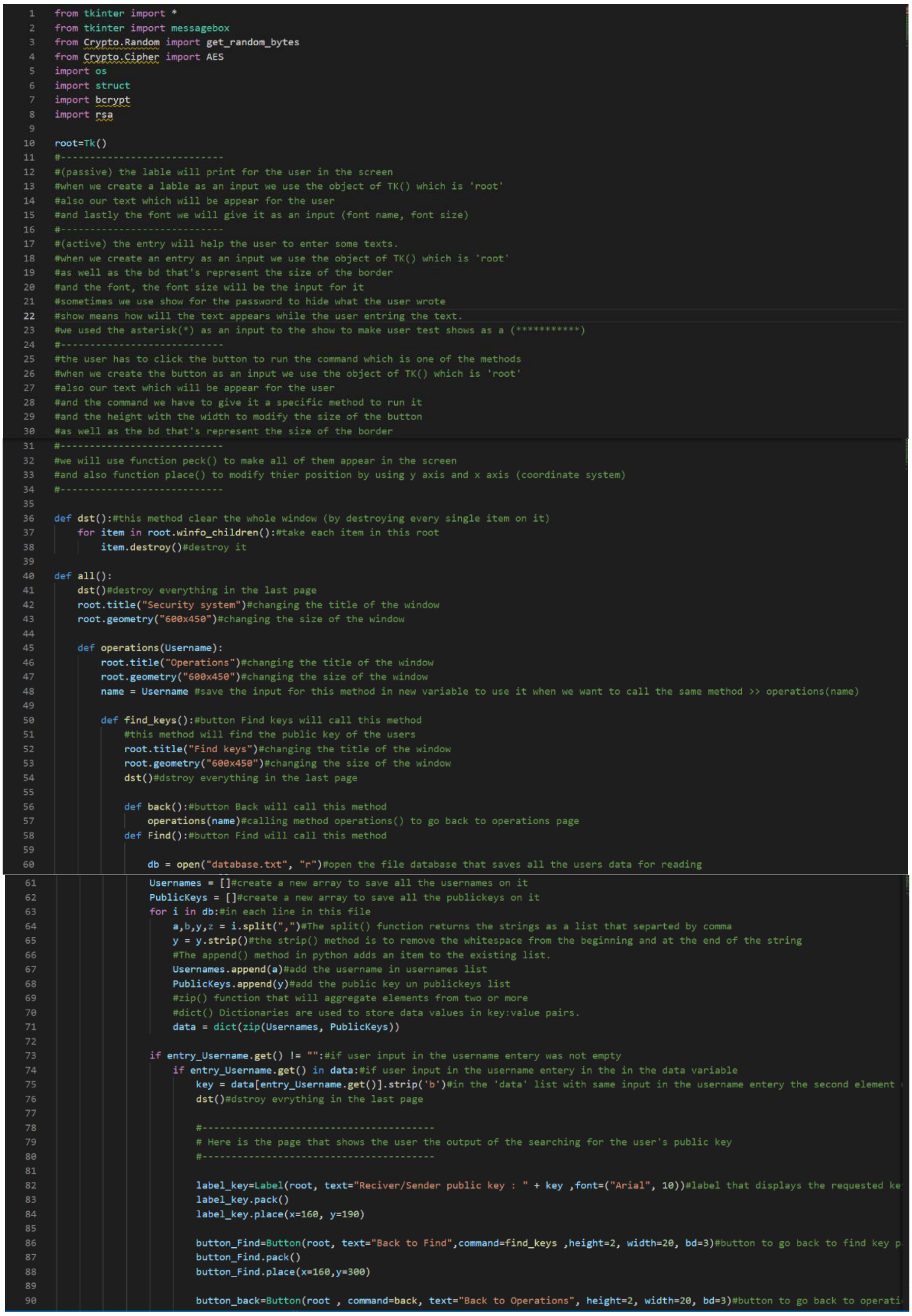
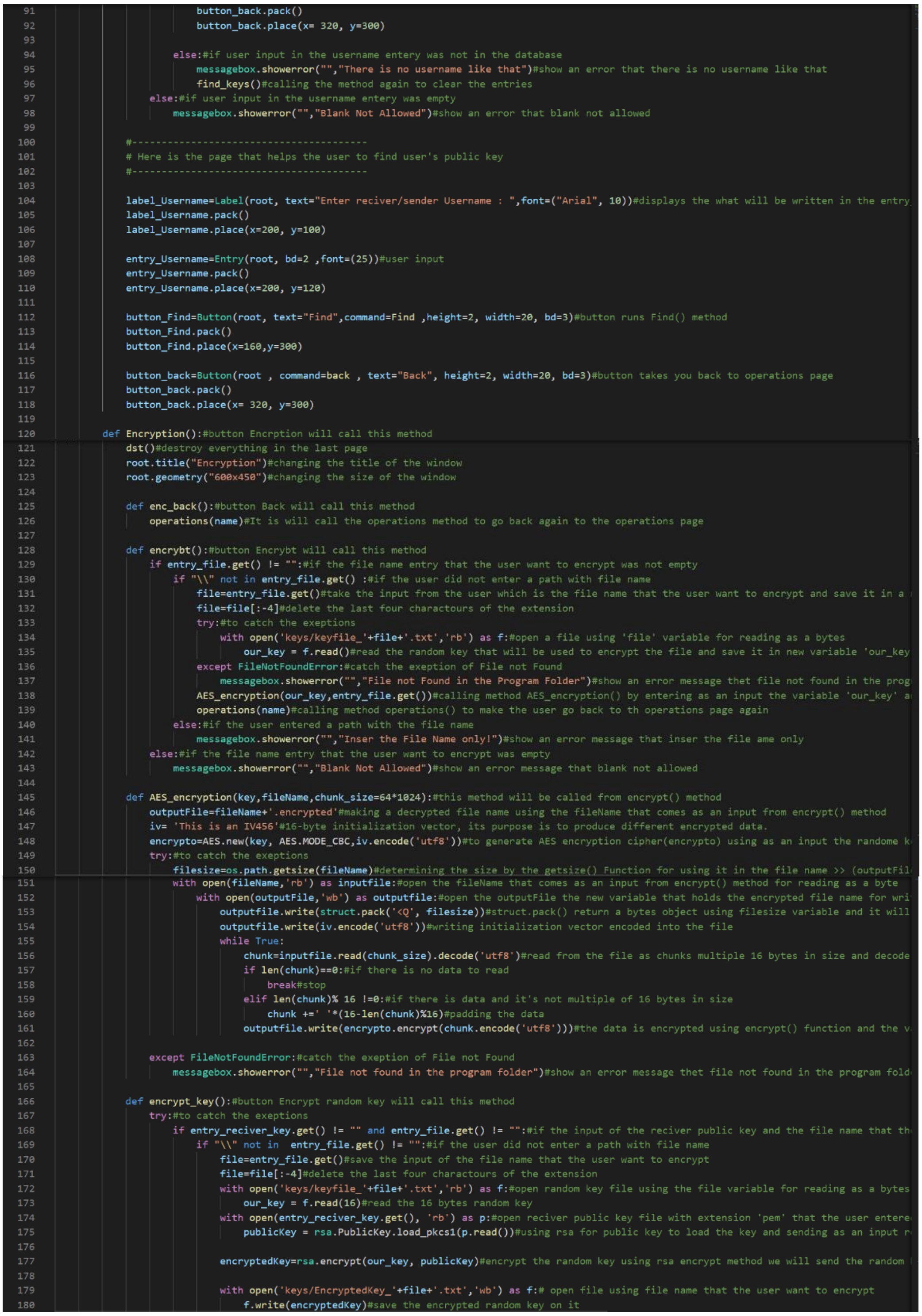

all():
have A account
NO
YES
signup(userName,password)
gainAccess(userName,password):
register():
generateKeys(UserName):
Generate_key(): the public, private keys files names
START():
Oparions ():
Encrption
Encrption()
Decrption
Decrption()
Logout
Find
aes_key(FileName): AES's Random Key
encrypt_key(AES's RandomKey,Receiver'sPublickeyFileName): EncryptedKey
sign_sha1 (EncryptedKey,sender'sPrivateKeyFileName): SignatureForEncryptedKey
encrybt(FileName):
AES_encryption(AES key,fileName):EncryptedFile
decrybt_key(EncryptedKeyFileName,UserPrivateKey): decrptedAES'sKey
verify_shal(decrptedAES'sKey,SignatureForEncryptedKey,sender'sPublicKeyFileName):
decrybt(EncryptedFile)
decrypt_AES(DecrptedKey,EncryptedFile'sName): DecryptedFile
find_keys()
Find(UserName):User'sPublickeyFileName
miro

Figure 54:Application Flow

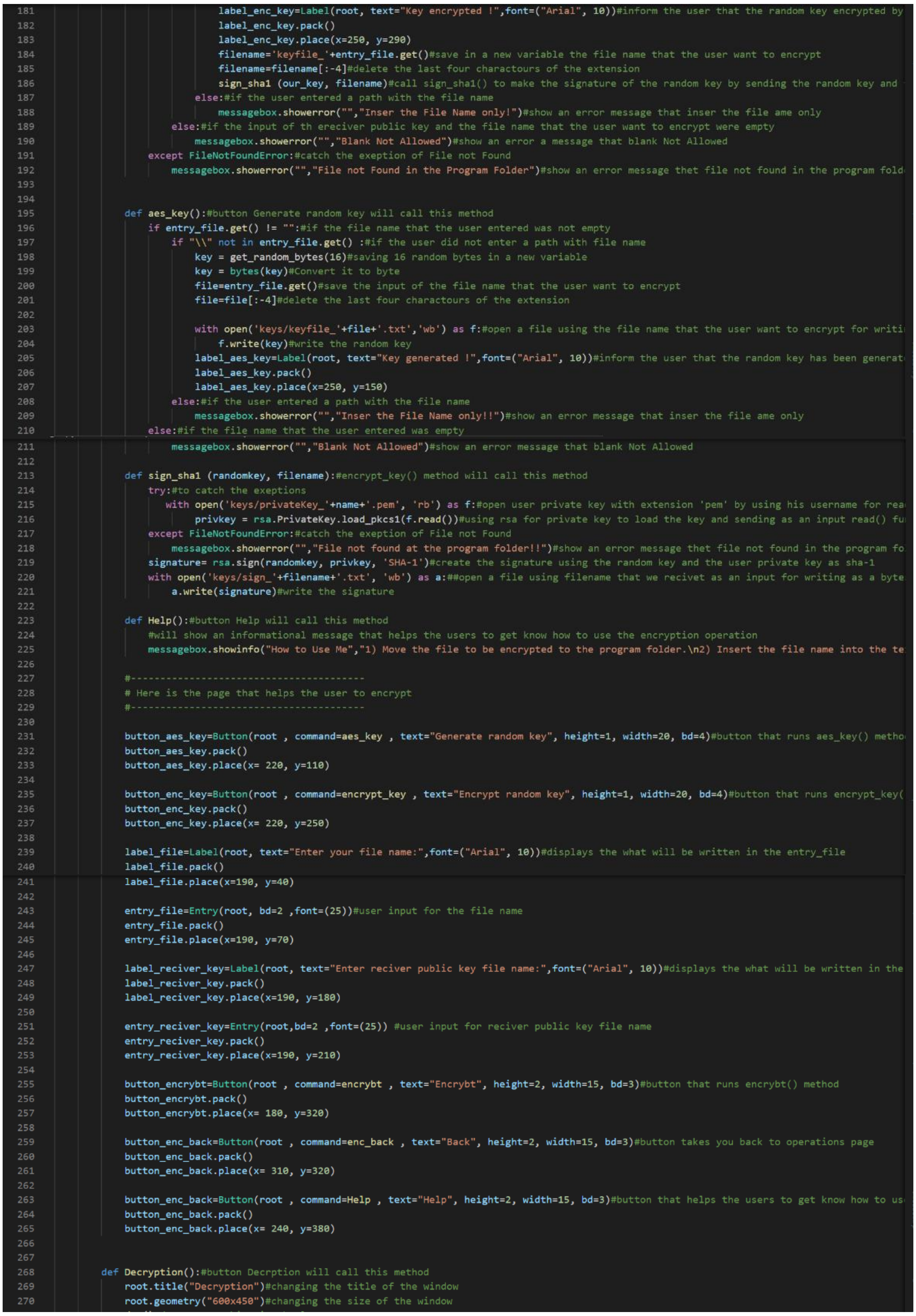
2. Code screenshot



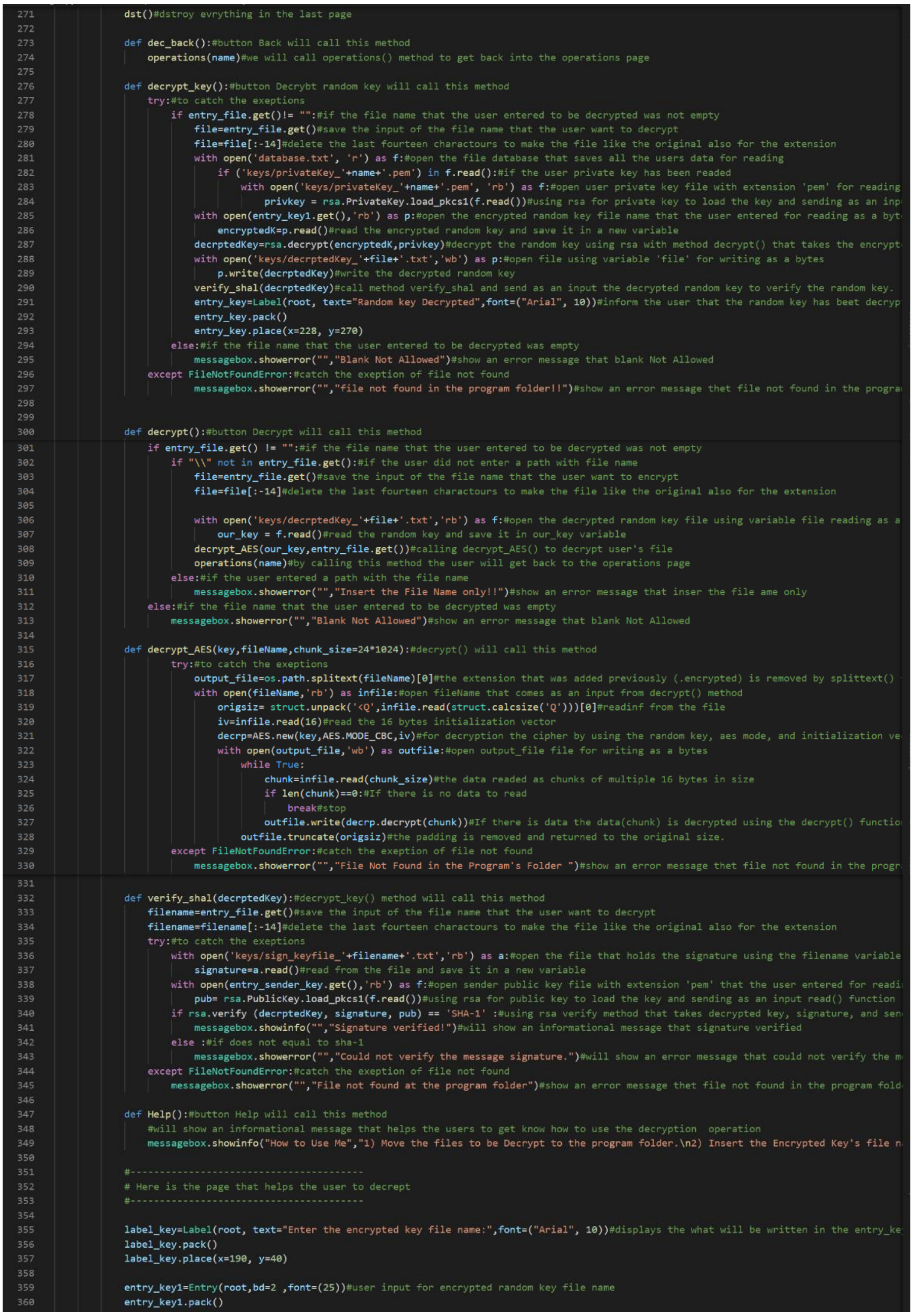
```python
from tkinter import *
from tkinter import messagebox
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
import os
import struct
import bcrypt
import rsa

root=Tk()
#---------------------------
#(passive) the lable will print for the user in the screen
#when we create a lable as an input we use the object of TK() which is 'root'
#also our text which will be appear for the user
#and lastly the font we will give it as an input (font name, font size)
#---------------------------
#(active) the entry will help the user to enter some texts.
#when we create an entry as an input we use the object of TK() which is 'root'
#as well as the bd that's represent the size of the border
#and the font, the font size will be the input for it
#sometimes we use show for the password to hide what the user wrote
#show means how will the text appears while the user entring the text.
#we used the asterisk(*) as an input to the show to make user test shows as a (***********)
#---------------------------
#the user has to click the button to run the command which is one of the methods
#when we create the button as an input we use the object of TK() which is 'root'
#also our text which will be appear for the user
#and the command we have to give it a specific method to run it
#and the height with the width to modify the size of the button
#as well as the bd that's represent the size of the border
#---------------------------
#we will use function peck() to make all of them appear in the screen
#and also function place() to modify thier position by using y axis and x axis (coordinate system)
#---------------------------

def dst():#this method clear the whole window (by destroying every single item on it)
    for item in root.winfo_children():#take each item in this root
        item.destroy()#destroy it

def all():
    dst()#destroy everything in the last page
    root.title("Security system")#changing the title of the window
    root.geometry("600x450")#changing the size of the window

    def operations(Username):
        root.title("Operations")#changing the title of the window
        root.geometry("600x450")#changing the size of the window
        name = Username #save the input for this method in new variable to use it when we want to call the same method >> operations(name)

        def find_keys():#button Find keys will call this method
            #this method will find the public key of the users
            root.title("Find keys")#changing the title of the window
            root.geometry("600x450")#changing the size of the window
            dst()#dstroy everything in the last page

            def back():#button Back will call this method
                operations(name)#calling method operations() to go back to operations page
            def Find():#button Find will call this method

                db = open("database.txt", "r")#open the file database that saves all the users data for reading
                Usernames = []#create a new array to save all the usernames on it
                PublicKeys = []#create a new array to save all the publickeys on it
                for i in db:#in each line in this file
                    a,b,y,z = i.split(",")#The split() function returns the strings as a list that separted by comma
                    y = y.strip()#the strip() method is to remove the whitespace from the beginning and at the end of the string
                    #The append() method in python adds an item to the existing list.
                    Usernames.append(a)#add the username in usernames list
                    PublicKeys.append(y)#add the public key un publickeys list
                    #zip() function that will aggregate elements from two or more
                    #dict() Dictionaries are used to store data values in key:value pairs.
                    data = dict(zip(Usernames, PublicKeys))

                if entry_Username.get() != "":#if user input in the username entery was not empty
                    if entry_Username.get() in data:#if user input in the username entery in the in the data variable
                        key = data[entry_Username.get()].strip('b')#in the 'data' list with same input in the username entery the second element
                        dst()#dstroy evrything in the last page

                        #--------------------------------------
                        # Here is the page that shows the user the output of the searching for the user's public key
                        #--------------------------------------

                        label_key=Label(root, text="Reciver/Sender public key : " + key ,font=("Arial", 10))#label that displays the requested ke
                        label_key.pack()
                        label_key.place(x=160, y=190)

                        button_Find=Button(root, text="Back to Find",command=find_keys ,height=2, width=20, bd=3)#button to go back to find key p
                        button_Find.pack()
                        button_Find.place(x=160,y=300)

                        button_back=Button(root , command=back, text="Back to Operations", height=2, width=20, bd=3)#button to go back to operati
```



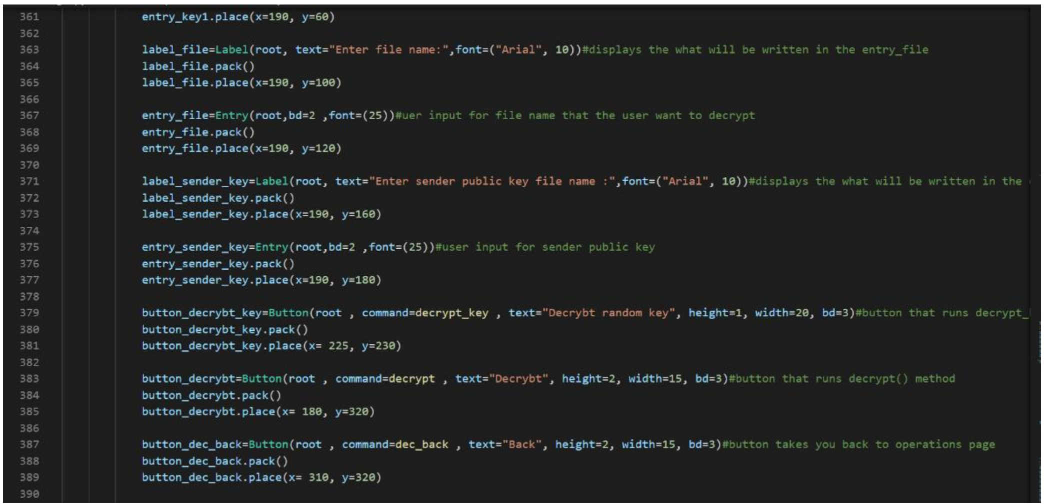
```
                        button_back.pack()
                        button_back.place(x= 320, y=300)

                    else:#if user input in the username entery was not in the database
                        messagebox.showerror("","There is no username like that")#show an error that there is no username like that
                        find_keys()#calling the method again to clear the entries
                else:#if user input in the username entery was empty
                    messagebox.showerror("","Blank Not Allowed")#show an error that blank not allowed

            #---------------------------------------
            # Here is the page that helps the user to find user's public key
            #---------------------------------------

            label_Username=Label(root, text="Enter reciver/sender Username : ",font=("Arial", 10))#displays the what will be written in the entry
            label_Username.pack()
            label_Username.place(x=200, y=100)

            entry_Username=Entry(root, bd=2 ,font=(25))#user input
            entry_Username.pack()
            entry_Username.place(x=200, y=120)

            button_Find=Button(root, text="Find",command=Find ,height=2, width=20, bd=3)#button runs Find() method
            button_Find.pack()
            button_Find.place(x=160,y=300)

            button_back=Button(root , command=back , text="Back", height=2, width=20, bd=3)#button takes you back to operations page
            button_back.pack()
            button_back.place(x= 320, y=300)

        def Encryption():#button Encrption will call this method
            dst()#destroy everything in the last page
            root.title("Encryption")#changing the title of the window
            root.geometry("600x450")#changing the size of the window

            def enc_back():#button Back will call this method
                operations(name)#It is will call the operations method to go back again to the operations page

            def encrybt():#button Encrybt will call this method
                if entry_file.get() != "":#if the file name entry that the user want to encrypt was not empty
                    if "\\" not in entry_file.get() :#if the user did not enter a path with file name
                        file=entry_file.get()#take the input from the user which is the file name that the user want to encrypt and save it in a
                        file=file[:-4]#delete the last four charactours of the extension
                        try:#to catch the exeptions
                            with open('keys/keyfile_'+file+'.txt','rb') as f:#open a file using 'file' variable for reading as a bytes
                                our_key = f.read()#read the random key that will be used to encrypt the file and save it in new variable 'our_key
                        except FileNotFoundError:#catch the exeption of File not Found
                            messagebox.showerror("","File not Found in the Program Folder")#show an error message thet file not found in the prog
                        AES_encryption(our_key,entry_file.get())#calling method AES_encryption() by entering as an input the variable 'our_key' a
                        operations(name)#calling method operations() to make the user go back to th operations page again
                    else:#if the user entered a path with the file name
                        messagebox.showerror("","Inser the File Name only!")#show an error message that inser the file ame only
                else:#if the file name entry that the user want to encrypt was empty
                    messagebox.showerror("","Blank Not Allowed")#show an error message that blank not allowed

            def AES_encryption(key,fileName,chunk_size=64*1024):#this method will be called from encrypt() method
                outputFile=fileName+'.encrypted'#making a decrypted file name using the fileName that comes as an input from encrypt() method
                iv= 'This is an IV456'#16-byte initialization vector, its purpose is to produce different encrypted data.
                encrypto=AES.new(key, AES.MODE_CBC,iv.encode('utf8'))#to generate AES encryption cipher(encrypto) using as an input the randome k
                try:#to catch the exeptions
                    filesize=os.path.getsize(fileName)#determining the size by the getsize() Function for using it in the file name >> (outputFil
                    with open(fileName,'rb') as inputfile:#open the fileName that comes as an input from encrypt() method for reading as a byte
                        with open(outputFile,'wb') as outputfile:#open the outputFile the new variable that holds the encrypted file name for wri
                            outputfile.write(struct.pack('<Q', filesize))#struct.pack() return a bytes object using filesize variable and it will
                            outputfile.write(iv.encode('utf8'))#writing initialization vector encoded into the file
                            while True:
                                chunk=inputfile.read(chunk_size).decode('utf8')#read from the file as chunks multiple 16 bytes in size and decode
                                if len(chunk)==0:#if there is no data to read
                                    break#stop
                                elif len(chunk)% 16 !=0:#if there is data and it's not multiple of 16 bytes in size
                                    chunk +=' '*(16-len(chunk)%16)#padding the data
                                outputfile.write(encrypto.encrypt(chunk.encode('utf8')))#the data is encrypted using encrypt() function and the v

                except FileNotFoundError:#catch the exeption of File not Found
                    messagebox.showerror("","File not found in the program folder")#show an error message thet file not found in the program fold

            def encrypt_key():#button Encrypt random key will call this method
                try:#to catch the exeptions
                    if entry_reciver_key.get() != "" and entry_file.get() != "":#if the input of the reciver public key and the file name that th
                        if "\\" not in  entry_file.get() != "":#if the user did not enter a path with file name
                            file=entry_file.get()#save the input of the file name that the user want to encrypt
                            file=file[:-4]#delete the last four charactours of the extension
                            with open('keys/keyfile_'+file+'.txt','rb') as f:#open random key file using the file variable for reading as a bytes
                                our_key = f.read(16)#read the 16 bytes random key
                            with open(entry_reciver_key.get(), 'rb') as p:#open reciver public key file with extension 'pem' that the user entere
                                publicKey = rsa.PublicKey.load_pkcs1(p.read())#using rsa for public key to load the key and sending as an input r

                            encryptedKey=rsa.encrypt(our_key, publicKey)#encrypt the random key using rsa encrypt method we will send the random

                            with open('keys/EncryptedKey_'+file+'.txt','wb') as f:# open file using file name that the user want to encrypt
                                f.write(encryptedKey)#save the encrypted random key on it
```



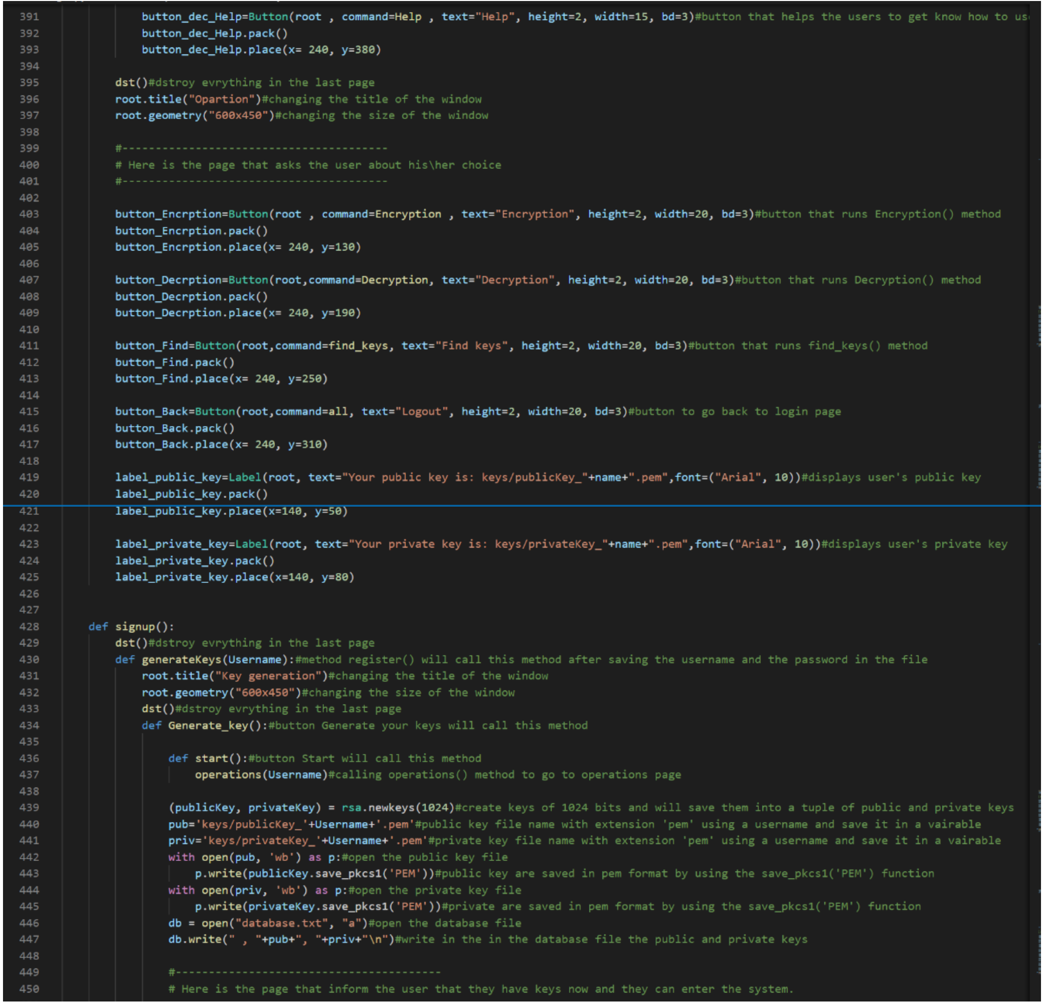
```
                        label_enc_key=Label(root, text="Key encrypted !",font=("Arial", 10))#inform the user that the random key encrypted by
                        label_enc_key.pack()
                        label_enc_key.place(x=250, y=290)
                        filename='keyfile_'+entry_file.get()#save in a new variable the file name that the user want to encrypt
                        filename=filename[:-4]#delete the last four charactours of the extension
                        sign_sha1 (our_key, filename)#call sign_sha1() to make the signature of the random key by sending the random key and
                    else:#if the user entered a path with the file name
                        messagebox.showerror("","Inser the File Name only!")#show an error message that inser the file ame only
                else:#if the input of th ereciver public key and the file name that the user want to encrypt were empty
                    messagebox.showerror("","Blank Not Allowed")#show an error a message that blank Not Allowed
            except FileNotFoundError:#catch the exeption of File not Found
                messagebox.showerror("","File not Found in the Program Folder")#show an error message thet file not found in the program fold


        def aes_key():#button Generate random key will call this method
            if entry_file.get() != "":#if the file name that the user entered was not empty
                if "\\" not in entry_file.get() :#if the user did not enter a path with file name
                    key = get_random_bytes(16)#saving 16 random bytes in a new variable
                    key = bytes(key)#Convert it to byte
                    file=entry_file.get()#save the input of the file name that the user want to encrypt
                    file=file[:-4]#delete the last four charactours of the extension

                    with open('keys/keyfile_'+file+'.txt','wb') as f:#open a file using the file name that the user want to encrypt for writi
                        f.write(key)#write the random key
                    label_aes_key=Label(root, text="Key generated !",font=("Arial", 10))#inform the user that the random key has been generat
                    label_aes_key.pack()
                    label_aes_key.place(x=250, y=150)
                else:#if the user entered a path with the file name
                    messagebox.showerror("","Inser the File Name only!!")#show an error message that inser the file ame only
            else:#if the file name that the user entered was empty
                messagebox.showerror("","Blank Not Allowed")#show an error message that blank Not Allowed

        def sign_sha1 (randomkey, filename):#encrypt_key() method will call this method
            try:#to catch the exeptions
               with open('keys/privateKey_'+name+'.pem', 'rb') as f:#open user private key with extension 'pem' by using his username for rea
                    privkey = rsa.PrivateKey.load_pkcs1(f.read())#using rsa for private key to load the key and sending as an input read() fu
            except FileNotFoundError:#catch the exeption of File not Found
                messagebox.showerror("","File not found at the program folder!!")#show an error message thet file not found in the program fo
            signature= rsa.sign(randomkey, privkey, 'SHA-1')#create the signature using the random key and the user private key as sha-1
            with open('keys/sign_'+filename+'.txt', 'wb') as a:##open a file using filename that we recivet as an input for writing as a byte
                a.write(signature)#write the signature

        def Help():#button Help will call this method
            #will show an informational message that helps the users to get know how to use the encryption operation
            messagebox.showinfo("How to Use Me","1) Move the file to be encrypted to the program folder.\n2) Insert the file name into the te

        #---------------------------------------
        # Here is the page that helps the user to encrypt
        #---------------------------------------

        button_aes_key=Button(root , command=aes_key , text="Generate random key", height=1, width=20, bd=4)#button that runs aes_key() metho
        button_aes_key.pack()
        button_aes_key.place(x= 220, y=110)

        button_enc_key=Button(root , command=encrypt_key , text="Encrypt random key", height=1, width=20, bd=4)#button that runs encrypt_key(
        button_enc_key.pack()
        button_enc_key.place(x= 220, y=250)

        label_file=Label(root, text="Enter your file name:",font=("Arial", 10))#displays the what will be written in the entry_file
        label_file.pack()
        label_file.place(x=190, y=40)

        entry_file=Entry(root, bd=2 ,font=(25))#user input for the file name
        entry_file.pack()
        entry_file.place(x=190, y=70)

        label_reciver_key=Label(root, text="Enter reciver public key file name:",font=("Arial", 10))#displays the what will be written in the
        label_reciver_key.pack()
        label_reciver_key.place(x=190, y=180)

        entry_reciver_key=Entry(root,bd=2 ,font=(25)) #user input for reciver public key file name
        entry_reciver_key.pack()
        entry_reciver_key.place(x=190, y=210)

        button_encrybt=Button(root , command=encrybt , text="Encrybt", height=2, width=15, bd=3)#button that runs encrybt() method
        button_encrybt.pack()
        button_encrybt.place(x= 180, y=320)

        button_enc_back=Button(root , command=enc_back , text="Back", height=2, width=15, bd=3)#button takes you back to operations page
        button_enc_back.pack()
        button_enc_back.place(x= 310, y=320)

        button_enc_back=Button(root , command=Help , text="Help", height=2, width=15, bd=3)#button that helps the users to get know how to us
        button_enc_back.pack()
        button_enc_back.place(x= 240, y=380)


    def Decryption():#button Decrption will call this method
        root.title("Decryption")#changing the title of the window
        root.geometry("600x450")#changing the size of the window
```



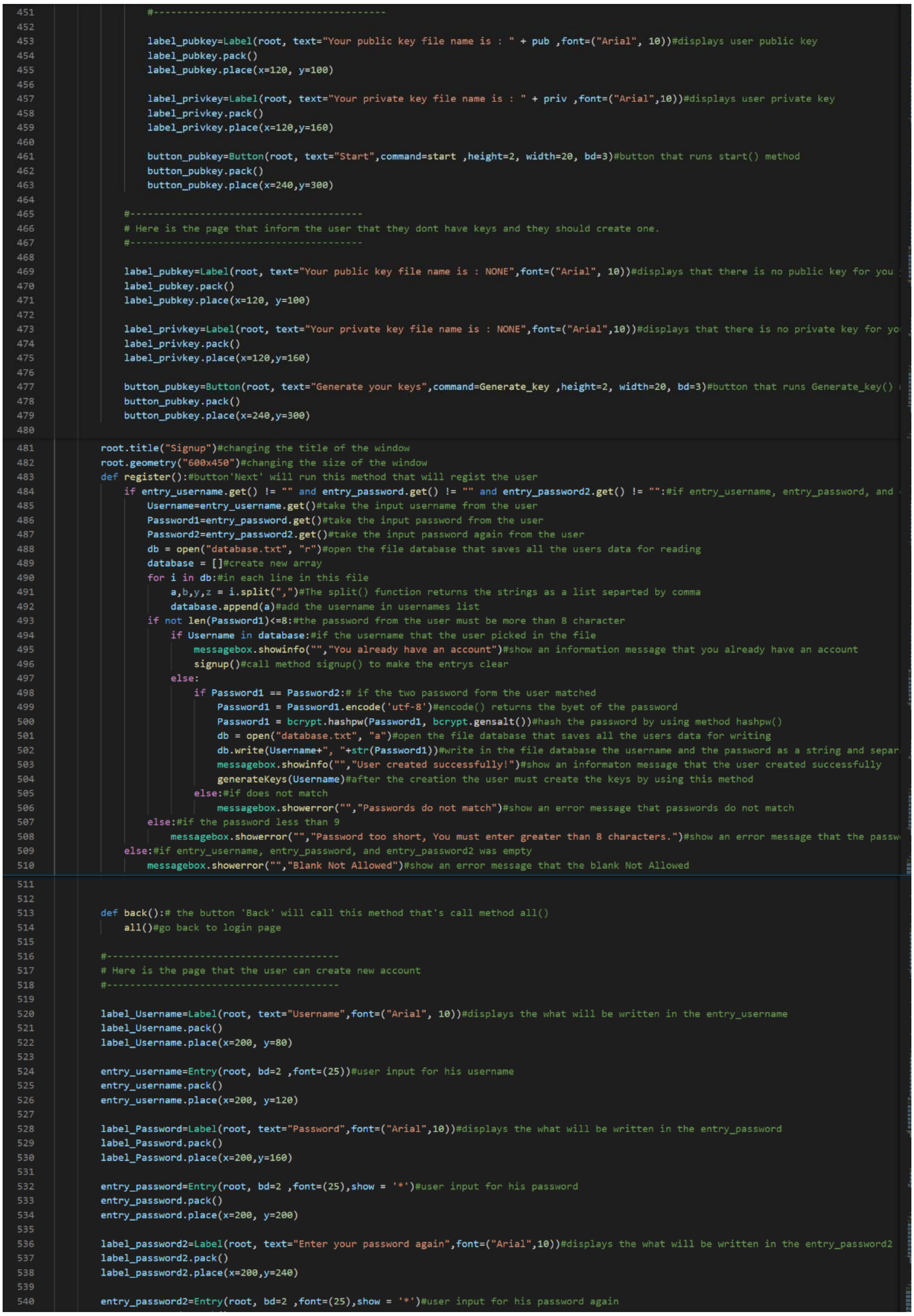
```
dst()#dstroy evrything in the last page

def dec_back():#button Back will call this method
    operations(name)#we will call operations() method to get back into the operations page

def decrypt_key():#button Decrybt random key will call this method
    try:#to catch the exeptions
        if entry_file.get()!= "":#if the file name that the user entered to be decrypted was not empty
            file=entry_file.get()#save the input of the file name that the user want to decrypt
            file=file[:-14]#delete the last fourteen charactours to make the file like the original also for the extension
            with open('database.txt', 'r') as f:#open the file database that saves all the users data for reading
                if ('keys/privateKey_'+name+'.pem') in f.read():#if the user private key has been readed
                    with open('keys/privateKey_'+name+'.pem', 'rb') as f:#open user private key file with extension 'pem' for reading
                        privkey = rsa.PrivateKey.load_pkcs1(f.read())#using rsa for private key to load the key and sending as an inp
            with open(entry_key1.get(),'rb') as p:#open the encrypted random key file name that the user entered for reading as a byt
                encryptedK=p.read()#read the encrypted random key and save it in a new variable
            decrptedKey=rsa.decrypt(encryptedK,privkey)#decrypt the random key using rsa with method decrypt() that takes the encrypt
            with open('keys/decrptedKey_'+file+'.txt','wb') as p:#open file using variable 'file' for writing as a bytes
                p.write(decrptedKey)#write the decrypted random key
            verify_shal(decrptedKey)#call method verify_shal and send as an input the decrypted random key to verify the random key.
            entry_key=Label(root, text="Random key Decrypted",font=("Arial", 10))#inform the user that the random key has beet decryp
            entry_key.pack()
            entry_key.place(x=228, y=270)
        else:#if the file name that the user entered to be decrypted was empty
            messagebox.showerror("","Blank Not Allowed")#show an error message that blank Not Allowed
    except FileNotFoundError:#catch the exeption of file not found
            messagebox.showerror("","file not found in the program folder!!")#show an error message thet file not found in the progra


def decrypt():#button Decrypt will call this method
    if entry_file.get() != "":#if the file name that the user entered to be decrypted was not empty
        if "\\" not in entry_file.get():#if the user did not enter a path with file name
            file=entry_file.get()#save the input of the file name that the user want to encrypt
            file=file[:-14]#delete the last fourteen charactours to make the file like the original also for the extension

            with open('keys/decrptedKey_'+file+'.txt','rb') as f:#open the decrypted random key file using variable file reading as a
                our_key = f.read()#read the random key and save it in our_key variable
            decrypt_AES(our_key,entry_file.get())#calling decrypt_AES() to decrypt user's file
            operations(name)#by calling this method the user will get back to the operations page
        else:#if the user entered a path with the file name
            messagebox.showerror("","Insert the File Name only!!")#show an error message that inser the file ame only
    else:#if the file name that the user entered to be decrypted was empty
        messagebox.showerror("","Blank Not Allowed")#show an error message that blank Not Allowed

def decrypt_AES(key,fileName,chunk_size=24*1024):#decrypt() will call this method
        try:#to catch the exeptions
            output_file=os.path.splitext(fileName)[0]#the extension that was added previously (.encrypted) is removed by splittext()
            with open(fileName,'rb') as infile:#open fileName that comes as an input from decrypt() method
                origsiz= struct.unpack('<Q',infile.read(struct.calcsize('Q')))[0]#readinf from the file
                iv=infile.read(16)#read the 16 bytes initialization vector
                decrp=AES.new(key,AES.MODE_CBC,iv)#for decryption the cipher by using the random key, aes mode, and initialization ve
                with open(output_file,'wb') as outfile:#open output_file file for writing as a bytes
                    while True:
                        chunk=infile.read(chunk_size)#the data readed as chunks of multiple 16 bytes in size
                        if len(chunk)==0:#If there is no data to read
                            break#stop
                        outfile.write(decrp.decrypt(chunk))#If there is data the data(chunk) is decrypted using the decrypt() functio
                    outfile.truncate(origsiz)#the padding is removed and returned to the original size.
        except FileNotFoundError:#catch the exeption of file not found
            messagebox.showerror("","File Not Found in the Program's Folder ")#show an error message thet file not found in the progr

def verify_shal(decrptedKey):#decrypt_key() method will call this method
    filename=entry_file.get()#save the input of the file name that the user want to decrypt
    filename=filename[:-14]#delete the last fourteen charactours to make the file like the original also for the extension
    try:#to catch the exeptions
        with open('keys/sign_keyfile_'+filename+'.txt','rb') as a:#open the file that holds the signature using the filename variable
            signature=a.read()#read from the file and save it in a new variable
        with open(entry_sender_key.get(),'rb') as f:#open sender public key file with extension 'pem' that the user entered for readi
            pub= rsa.PublicKey.load_pkcs1(f.read())#using rsa for public key to load the key and sending as an input read() function
        if rsa.verify (decrptedKey, signature, pub) == 'SHA-1' :#using rsa verify method that takes decrypted key, signature, and sen
            messagebox.showinfo("","Signature verified!")#will show an informational message that signature verified
        else :#if does not equal to sha-1
            messagebox.showerror("","Could not verify the message signature.")#will show an error message that could not verify the m
    except FileNotFoundError:#catch the exeption of file not found
        messagebox.showerror("","File not found at the program folder")#show an error message thet file not found in the program fold

def Help():#button Help will call this method
    #will show an informational message that helps the users to get know how to use the decryption  operation
    messagebox.showinfo("How to Use Me","1) Move the files to be Decrypt to the program folder.\n2) Insert the Encrypted Key's file n

#---------------------------------------
# Here is the page that helps the user to decrept
#---------------------------------------

label_key=Label(root, text="Enter the encrypted key file name:",font=("Arial", 10))#displays the what will be written in the entry_ke
label_key.pack()
label_key.place(x=190, y=40)

entry_key1=Entry(root,bd=2 ,font=(25))#user input for encrypted random key file name
entry_key1.pack()
```



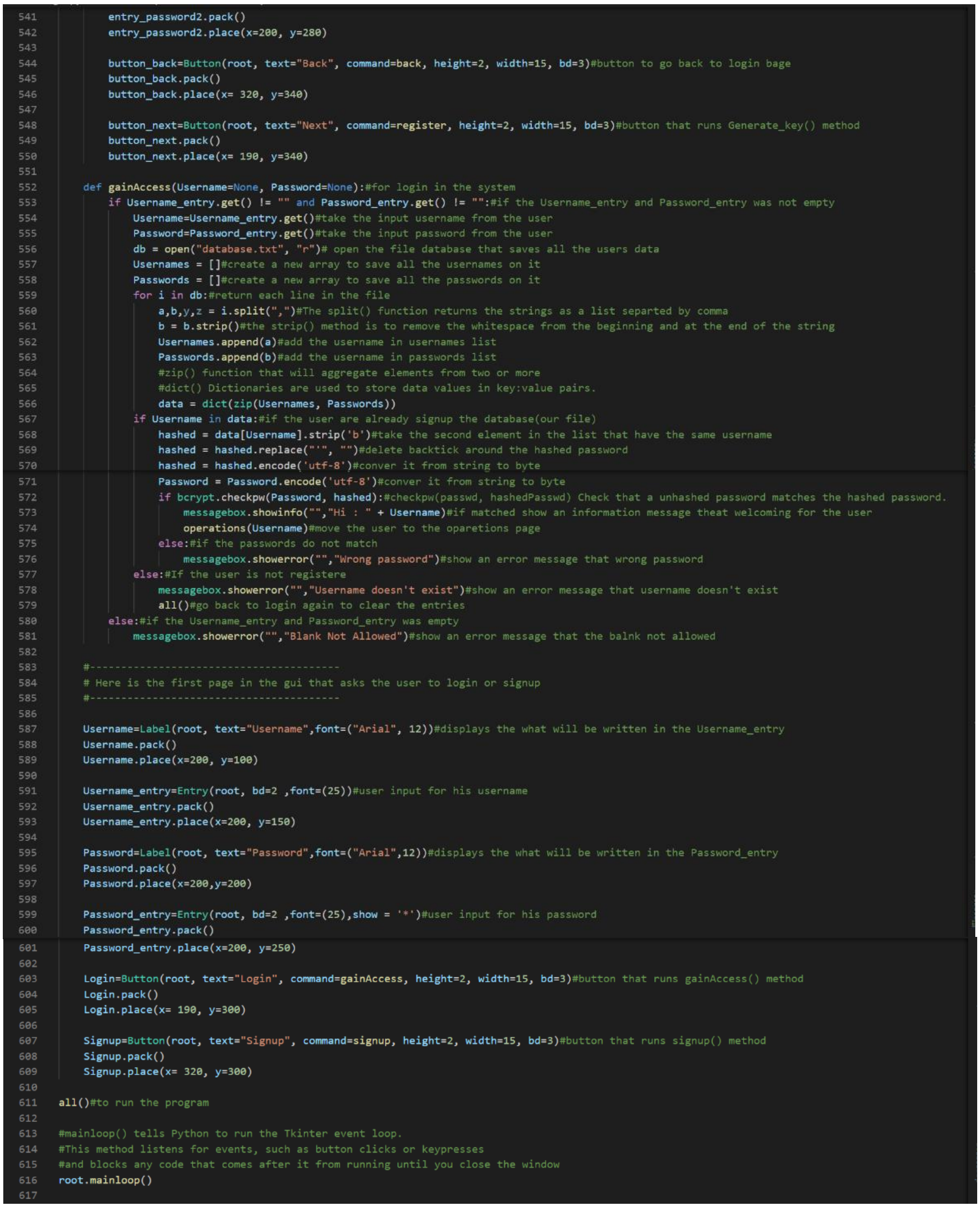
```
            entry_key1.place(x=190, y=60)

            label_file=Label(root, text="Enter file name:",font=("Arial", 10))#displays the what will be written in the entry_file
            label_file.pack()
            label_file.place(x=190, y=100)

            entry_file=Entry(root,bd=2 ,font=(25))#uer input for file name that the user want to decrypt
            entry_file.pack()
            entry_file.place(x=190, y=120)

            label_sender_key=Label(root, text="Enter sender public key file name :",font=("Arial", 10))#displays the what will be written in the
            label_sender_key.pack()
            label_sender_key.place(x=190, y=160)

            entry_sender_key=Entry(root,bd=2 ,font=(25))#user input for sender public key
            entry_sender_key.pack()
            entry_sender_key.place(x=190, y=180)

            button_decrybt_key=Button(root , command=decrypt_key , text="Decrybt random key", height=1, width=20, bd=3)#button that runs decrypt_
            button_decrybt_key.pack()
            button_decrybt_key.place(x= 225, y=230)

            button_decrybt=Button(root , command=decrypt , text="Decrybt", height=2, width=15, bd=3)#button that runs decrypt() method
            button_decrybt.pack()
            button_decrybt.place(x= 180, y=320)

            button_dec_back=Button(root , command=dec_back , text="Back", height=2, width=15, bd=3)#button takes you back to operations page
            button_dec_back.pack()
            button_dec_back.place(x= 310, y=320)

```



```
            button_dec_Help=Button(root , command=Help , text="Help", height=2, width=15, bd=3)#button that helps the users to get know how to us
            button_dec_Help.pack()
            button_dec_Help.place(x= 240, y=380)

        dst()#dstroy evrything in the last page
        root.title("Opartion")#changing the title of the window
        root.geometry("600x450")#changing the size of the window

        #---------------------------------------
        # Here is the page that asks the user about his\her choice
        #---------------------------------------

        button_Encrption=Button(root , command=Encryption , text="Encryption", height=2, width=20, bd=3)#button that runs Encryption() method
        button_Encrption.pack()
        button_Encrption.place(x= 240, y=130)

        button_Decrption=Button(root,command=Decryption, text="Decryption", height=2, width=20, bd=3)#button that runs Decryption() method
        button_Decrption.pack()
        button_Decrption.place(x= 240, y=190)

        button_Find=Button(root,command=find_keys, text="Find keys", height=2, width=20, bd=3)#button that runs find_keys() method
        button_Find.pack()
        button_Find.place(x= 240, y=250)

        button_Back=Button(root,command=all, text="Logout", height=2, width=20, bd=3)#button to go back to login page
        button_Back.pack()
        button_Back.place(x= 240, y=310)

        label_public_key=Label(root, text="Your public key is: keys/publicKey_"+name+".pem",font=("Arial", 10))#displays user's public key
        label_public_key.pack()
        label_public_key.place(x=140, y=50)

        label_private_key=Label(root, text="Your private key is: keys/privateKey_"+name+".pem",font=("Arial", 10))#displays user's private key
        label_private_key.pack()
        label_private_key.place(x=140, y=80)


    def signup():
        dst()#dstroy evrything in the last page
        def generateKeys(Username):#method register() will call this method after saving the username and the password in the file
            root.title("Key generation")#changing the title of the window
            root.geometry("600x450")#changing the size of the window
            dst()#dstroy evrything in the last page
            def Generate_key():#button Generate your keys will call this method

                def start():#button Start will call this method
                    operations(Username)#calling operations() method to go to operations page

                (publicKey, privateKey) = rsa.newkeys(1024)#create keys of 1024 bits and will save them into a tuple of public and private keys
                pub='keys/publicKey_'+Username+'.pem'#public key file name with extension 'pem' using a username and save it in a vairable
                priv='keys/privateKey_'+Username+'.pem'#private key file name with extension 'pem' using a username and save it in a vairable
                with open(pub, 'wb') as p:#open the public key file
                    p.write(publicKey.save_pkcs1('PEM'))#public key are saved in pem format by using the save_pkcs1('PEM') function
                with open(priv, 'wb') as p:#open the private key file
                    p.write(privateKey.save_pkcs1('PEM'))#private are saved in pem format by using the save_pkcs1('PEM') function
                db = open("database.txt", "a")#open the database file
                db.write(" , "+pub+", "+priv+"\n")#write in the in the database file the public and private keys

                #---------------------------------------
                # Here is the page that inform the user that they have keys now and they can enter the system.
```



```
            #----------------------------------------

            label_pubkey=Label(root, text="Your public key file name is : " + pub ,font=("Arial", 10))#displays user public key
            label_pubkey.pack()
            label_pubkey.place(x=120, y=100)

            label_privkey=Label(root, text="Your private key file name is : " + priv ,font=("Arial",10))#displays user private key
            label_privkey.pack()
            label_privkey.place(x=120,y=160)

            button_pubkey=Button(root, text="Start",command=start ,height=2, width=20, bd=3)#button that runs start() method
            button_pubkey.pack()
            button_pubkey.place(x=240,y=300)

        #----------------------------------------
        # Here is the page that inform the user that they dont have keys and they should create one.
        #----------------------------------------

        label_pubkey=Label(root, text="Your public key file name is : NONE",font=("Arial", 10))#displays that there is no public key for you
        label_pubkey.pack()
        label_pubkey.place(x=120, y=100)

        label_privkey=Label(root, text="Your private key file name is : NONE",font=("Arial",10))#displays that there is no private key for yo
        label_privkey.pack()
        label_privkey.place(x=120,y=160)

        button_pubkey=Button(root, text="Generate your keys",command=Generate_key ,height=2, width=20, bd=3)#button that runs Generate_key()
        button_pubkey.pack()
        button_pubkey.place(x=240,y=300)

    root.title("Signup")#changing the title of the window
    root.geometry("600x450")#changing the size of the window
    def register():#button'Next' will run this method that will regist the user
        if entry_username.get() != "" and entry_password.get() != "" and entry_password2.get() != "":#if entry_username, entry_password, and
            Username=entry_username.get()#take the input username from the user
            Password1=entry_password.get()#take the input password from the user
            Password2=entry_password2.get()#take the input password again from the user
            db = open("database.txt", "r")#open the file database that saves all the users data for reading
            database = []#create new array
            for i in db:#in each line in this file
                a,b,y,z = i.split(",")#The split() function returns the strings as a list separted by comma
                database.append(a)#add the username in usernames list
            if not len(Password1)<=8:#the password from the user must be more than 8 character
                if Username in database:#if the username that the user picked in the file
                    messagebox.showinfo("","You already have an account")#show an information message that you already have an account
                    signup()#call method signup() to make the entrys clear
                else:
                    if Password1 == Password2:# if the two password form the user matched
                        Password1 = Password1.encode('utf-8')#encode() returns the byet of the password
                        Password1 = bcrypt.hashpw(Password1, bcrypt.gensalt())#hash the password by using method hashpw()
                        db = open("database.txt", "a")#open the file database that saves all the users data for writing
                        db.write(Username+", "+str(Password1))#write in the file database the username and the password as a string and separ
                        messagebox.showinfo("","User created successfully!")#show an informaton message that the user created successfully
                        generateKeys(Username)#after the creation the user must create the keys by using this method
                    else:#if does not match
                        messagebox.showerror("","Passwords do not match")#show an error message that passwords do not match
            else:#if the password less than 9
                messagebox.showerror("","Password too short, You must enter greater than 8 characters.")#show an error message that the passw
        else:#if entry_username, entry_password, and entry_password2 was empty
            messagebox.showerror("","Blank Not Allowed")#show an error message that the blank Not Allowed


    def back():# the button 'Back' will call this method that's call method all()
        all()#go back to login page

    #----------------------------------------
    # Here is the page that the user can create new account
    #----------------------------------------

    label_Username=Label(root, text="Username",font=("Arial", 10))#displays the what will be written in the entry_username
    label_Username.pack()
    label_Username.place(x=200, y=80)

    entry_username=Entry(root, bd=2 ,font=(25))#user input for his username
    entry_username.pack()
    entry_username.place(x=200, y=120)

    label_Password=Label(root, text="Password",font=("Arial",10))#displays the what will be written in the entry_password
    label_Password.pack()
    label_Password.place(x=200,y=160)

    entry_password=Entry(root, bd=2 ,font=(25),show = '*')#user input for his password
    entry_password.pack()
    entry_password.place(x=200, y=200)

    label_password2=Label(root, text="Enter your password again",font=("Arial",10))#displays the what will be written in the entry_password2
    label_password2.pack()
    label_password2.place(x=200,y=240)

    entry_password2=Entry(root, bd=2 ,font=(25),show = '*')#user input for his password again
```



```
        entry_password2.pack()
        entry_password2.place(x=200, y=280)

        button_back=Button(root, text="Back", command=back, height=2, width=15, bd=3)#button to go back to login bage
        button_back.pack()
        button_back.place(x= 320, y=340)

        button_next=Button(root, text="Next", command=register, height=2, width=15, bd=3)#button that runs Generate_key() method
        button_next.pack()
        button_next.place(x= 190, y=340)

    def gainAccess(Username=None, Password=None):#for login in the system
        if Username_entry.get() != "" and Password_entry.get() != "":#if the Username_entry and Password_entry was not empty
            Username=Username_entry.get()#take the input username from the user
            Password=Password_entry.get()#take the input password from the user
            db = open("database.txt", "r")# open the file database that saves all the users data
            Usernames = []#create a new array to save all the usernames on it
            Passwords = []#create a new array to save all the passwords on it
            for i in db:#return each line in the file
                a,b,y,z = i.split(",")#The split() function returns the strings as a list separted by comma
                b = b.strip()#the strip() method is to remove the whitespace from the beginning and at the end of the string
                Usernames.append(a)#add the username in usernames list
                Passwords.append(b)#add the username in passwords list
                #zip() function that will aggregate elements from two or more
                #dict() Dictionaries are used to store data values in key:value pairs.
                data = dict(zip(Usernames, Passwords))
            if Username in data:#if the user are already signup the database(our file)
                hashed = data[Username].strip('b')#take the second element in the list that have the same username
                hashed = hashed.replace("'", "")#delete backtick around the hashed password
                hashed = hashed.encode('utf-8')#conver it from string to byte
                Password = Password.encode('utf-8')#conver it from string to byte
                if bcrypt.checkpw(Password, hashed):#checkpw(passwd, hashedPasswd) Check that a unhashed password matches the hashed password.
                    messagebox.showinfo("","Hi : " + Username)#if matched show an information message theat welcoming for the user
                    operations(Username)#move the user to the oparetions page
                else:#if the passwords do not match
                    messagebox.showerror("","Wrong password")#show an error message that wrong password
            else:#If the user is not registere
                messagebox.showerror("","Username doesn't exist")#show an error message that username doesn't exist
                all()#go back to login again to clear the entries
        else:#if the Username_entry and Password_entry was empty
            messagebox.showerror("","Blank Not Allowed")#show an error message that the balnk not allowed

    #---------------------------------------
    # Here is the first page in the gui that asks the user to login or signup
    #---------------------------------------

    Username=Label(root, text="Username",font=("Arial", 12))#displays the what will be written in the Username_entry
    Username.pack()
    Username.place(x=200, y=100)

    Username_entry=Entry(root, bd=2 ,font=(25))#user input for his username
    Username_entry.pack()
    Username_entry.place(x=200, y=150)

    Password=Label(root, text="Password",font=("Arial",12))#displays the what will be written in the Password_entry
    Password.pack()
    Password.place(x=200,y=200)

    Password_entry=Entry(root, bd=2 ,font=(25),show = '*')#user input for his password
    Password_entry.pack()
    Password_entry.place(x=200, y=250)

    Login=Button(root, text="Login", command=gainAccess, height=2, width=15, bd=3)#button that runs gainAccess() method
    Login.pack()
    Login.place(x= 190, y=300)

    Signup=Button(root, text="Signup", command=signup, height=2, width=15, bd=3)#button that runs signup() method
    Signup.pack()
    Signup.place(x= 320, y=300)

all()#to run the program

#mainloop() tells Python to run the Tkinter event loop.
#This method listens for events, such as button clicks or keypresses
#and blocks any code that comes after it from running until you close the window
root.mainloop()

```

Figure55 :A screenshot of the entire code

○ Challenges.

1. Dealing with files in encryption and decryption operations.
2. Byte handling when writing and reading data from files.
3. Combine the two algorithms into the system.
4. Connect each user to his private and public keys.
5. Dealing with the GUI.
6. Re-testing the system several times to discover weaknesses and errors.
7. Since Python is the second programming language that we learned, we found some difficulties in the implementation.
8. Conflict of ideas for how to save information in the database and what is the correct way.
9. Because the hash was used, it was difficult to verify that the password given by the user was identical to the one in the database.

○ Conclusion

Encryption techniques are used in many applications to protect data confidentiality and authentication between communicating parties. One of the basic encryption techniques is the AES algorithm and the RSA algorithm, which were applied programmatically in this project to create a file encryption application.

We learned a lot from this project, and in the future, we may be able to apply and combine more effective encryption algorithms to create more robust and secure software than we used in this project.

o References

[1] SearchSecurity. 2022. What is the Advanced Encryption Standard (AES)? Definition from SearchSecurity. [online]Available at: <https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard> [Accessed 8 May 2022].

[2] Medium. 2022. Everything You Want To Know About AES, But Were Afraid To Ask …. [online] Available at: <https://medium.com/asecuritysite-when-bob-met-alice/everything-you-want-to-know-about-aes-but-were-afraid-to-ask-b46165410ea8> [Accessed 8 May 2022].

[3] Encryption Consulting | Encryption Consulting. 2022. RSA | What is RSA? | Encryption Consulting. [online]Available at: <https://www.encryptionconsulting.com/education-center/what-is-rsa/> [Accessed 8 May 2022].

[4] Youtube.com. 2022. [online] Available at: <https://www.youtube.com/watch?v=kNgaO1etjXM> [Accessed 8 May 2022].

[5] Youtube.com. 2022.[online] Available at: <https://www.youtube.com/watch?v=g81l3y2i-so> [Accessed 8 May 2022].

[6] [duplicate], H., 2022. How to hide password in Tkinter. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/42722129/how-to-hide-password-in-tkinter> [Accessed 8 May 2022].

[7] Tutorialspoint.com. 2022. *Python - Tkinter tkMessageBox*. [online] Available at: <https://www.tutorialspoint.com/python/tk_messagebox.htm> [Accessed 8 May 2022].

[8] Tutorialspoint.com. 2022. *How to create a Tkinter error message box?*. [online] Available at: <https://www.tutorialspoint.com/how-to-create-a-tkinter-error-message-box> [Accessed 8 May 2022].

[9] LaBelle, J., 2022. AES File Encryption in Python. [online] Jonlabelle.com. Available at: <https://jonlabelle.com/snippets/view/python/aes-file-encryption-in-python> [Accessed 8 May 2022].

[10] Novixys Software Dev Blog. 2022. Using AES for Encryption and Decryption in Python Pycrypto | Novixys Software Dev Blog. [online] Available at: <https://www.novixys.com/blog/using-aes-encryption-decryption-python-pycrypto/> [Accessed 8 May 2022].

[11] Engineering Education (EngEd) Program | Section. 2022. Implementing RSA Encryption and Decryption in Python. [online] Available at: <https://www.section.io/engineering-education/rsa-encryption-and-decryption-in-python/> [Accessed 8 May 2022].

Appendixes:

We have attached the application code and some code related files with this report in the same zip folder.