Al Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences

Computer Science Department

CS332 Information Security

Report

(XSecure application)

First semester 1443/2022

Date : 8/5/2021

Section: 372

Group members' names	Students IDs
Manar.M Alqabbani	
Dana.Dh Alyami	
Leen.S Alaboudi	
Nehal.S Almouies	

Instruction: Dr. Tahani Al-Balawi

Contents

1.Introduction

In this project we will encrypt a file and send it over the internet using both symmetric and asymmetric key approaches, AES algorithm is to be used for encrypting data, and RSA algorithm is to be used to protect the AES key.

The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone. The AES algorithm (also known as the Rijndael algorithm) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits.

2.Overview of the project design

1- every user should have their own username and password.
2- Then it will automatically generate a public key and private key for the user.
3- Then it will save the user information in a file.

- **sender side:**

1- The sender needs to log into the application.
2- The sender chooses a any file to send
3- The sender chooses the public key of the receiver to encrypt the symmetric key with an RSA algorithm.
4- Then the file will be encrypting with AES algorithm.

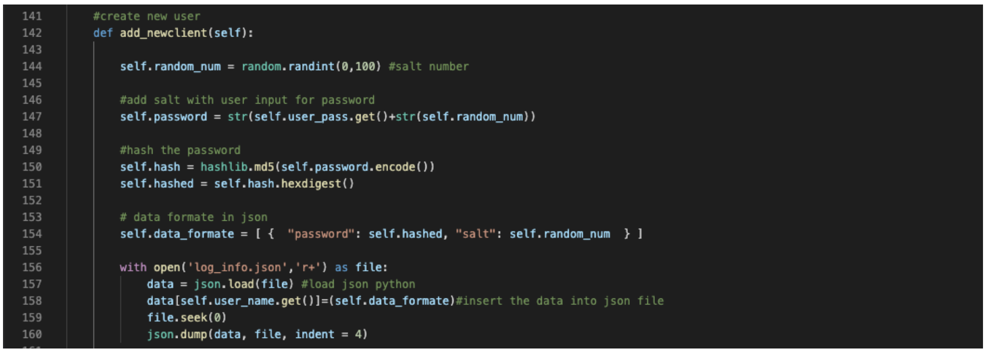
- **receiver side:**

1- The receiver needs to log into the application.
2- The receiver will use the stored private key to decrypt the symmetric key
3- The receiver uses the symmetric key to decrypt the file by AES algorithm
4- Then the receiver will be able to read the file.

3. Approach and steps to implementation.

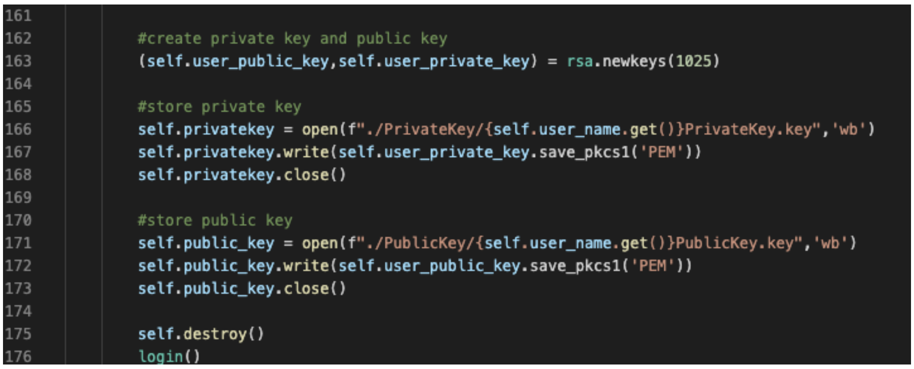
- We have 5 class: login, signup, homepage, send, receive.

  -The user must sign up to create a new user, the class sign up will create a username and password by (add_newclient) function. Then it will generate random number as a salt then it will add the password to the salt. Then it will be hashing the user password, then the information of the user will be stored in json file. We apply this by:



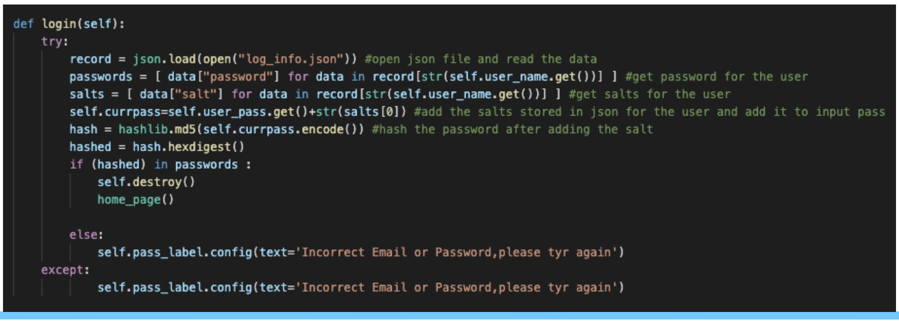
```
#create new user
def add_newclient(self):

    self.random_num = random.randint(0,100) #salt number

    #add salt with user input for password
    self.password = str(self.user_pass.get()+str(self.random_num))

    #hash the password
    self.hash = hashlib.md5(self.password.encode())
    self.hashed = self.hash.hexdigest()

    # data formate in json
    self.data_formate = [ {  "password": self.hashed, "salt": self.random_num  } ]

    with open('log_info.json','r+') as file:
        data = json.load(file) #load json python
        data[self.user_name.get()]=(self.data_formate)#insert the data into json file
        file.seek(0)
        json.dump(data, file, indent = 4)
```

- Then it will generate a public and private key for the user by RSA algorithm and will write those keys in a file. We apply this by:



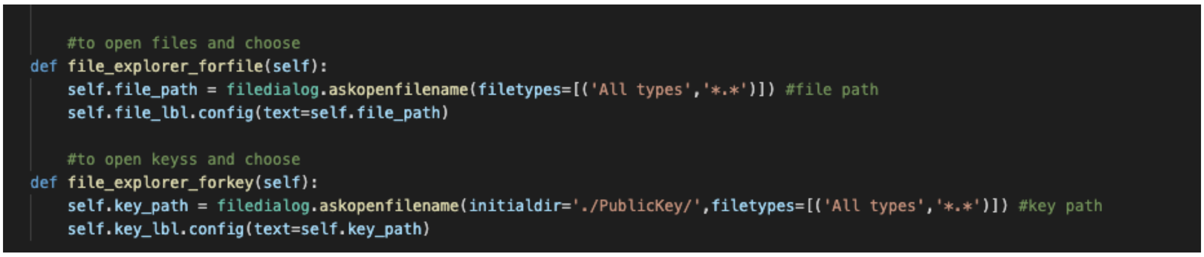
```

    #create private key and public key
    (self.user_public_key,self.user_private_key) = rsa.newkeys(1025)

    #store private key
    self.privatekey = open(f"./PrivateKey/{self.user_name.get()}PrivateKey.key",'wb')
    self.privatekey.write(self.user_private_key.save_pkcs1('PEM'))
    self.privatekey.close()

    #store public key
    self.public_key = open(f"./PublicKey/{self.user_name.get()}PublicKey.key",'wb')
    self.public_key.write(self.user_public_key.save_pkcs1('PEM'))
    self.public_key.close()

    self.destroy()
    login()
```

- If the sender wants to send a file securely or the receiver receive the file securely, they must log into the application with their username and password. In class login the user will write the username and password in the GUI, then the (login) function will compare the username and take the user password then hashed it and compare it with the user data stored before, if it is not equal to the user data an error message will appear: (Incorrect Email or Password, please try again). We apply this by:



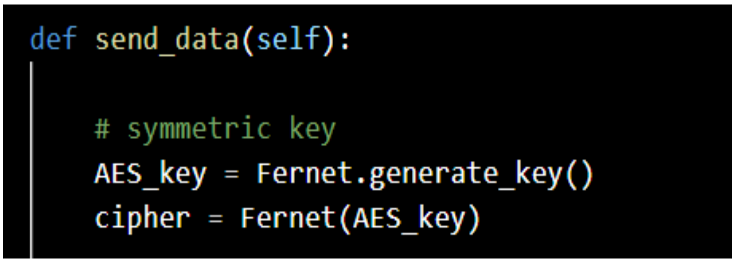
```
def login(self):
    try:
        record = json.load(open("log_info.json")) #open json file and read the data
        passwords = [ data["password"] for data in record[str(self.user_name.get())] ] #get password for the user
        salts = [ data["salt"] for data in record[str(self.user_name.get())] ] #get salts for the user
        self.currpass=self.user_pass.get()+str(salts[0]) #add the salts stored in json for the user and add it to input pass
        hash = hashlib.md5(self.currpass.encode()) #hash the password after adding the salt
        hashed = hash.hexdigest()
        if (hashed) in passwords :
            self.destroy()
            home_page()

        else:
            self.pass_label.config(text='Incorrect Email or Password,please tyr again')
    except:
            self.pass_label.config(text='Incorrect Email or Password,please tyr again')
```

- when the sender wants to choose a file to send, they can file explorer to choose it. This function (file_explorer_forfile) to get the file path, and (file_explorer_forkey) for the key file path. We apply this by:



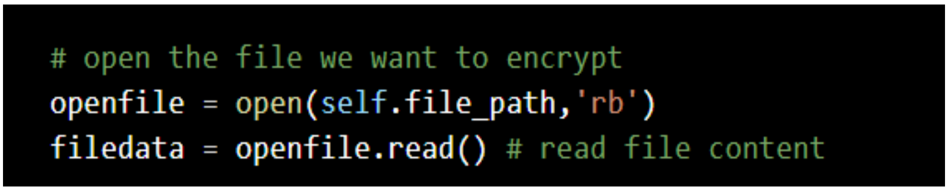
```
    #to open files and choose
def file_explorer_forfile(self):
    self.file_path = filedialog.askopenfilename(filetypes=[('All types','*.*')]) #file path
    self.file_lbl.config(text=self.file_path)

    #to open keyss and choose
def file_explorer_forkey(self):
    self.key_path = filedialog.askopenfilename(initialdir='./PublicKey/',filetypes=[('All types','*.*')]) #key path
    self.key_lbl.config(text=self.key_path)
```

- Now the file was selected by the user we will encrypting it with AES algorithm, function(send_data) will first generate the symmetric key (AES_key) by:



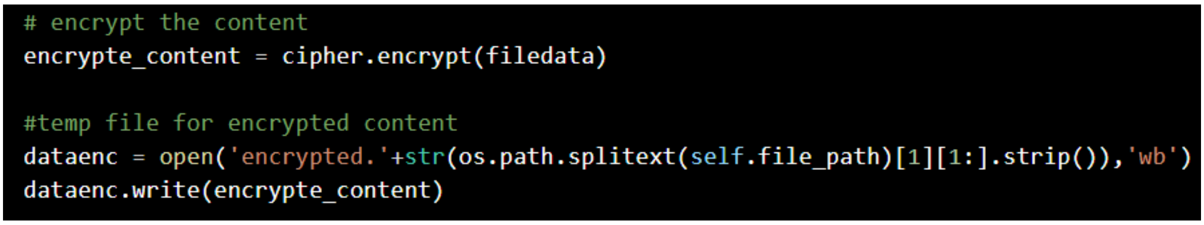
```
def send_data(self):

    # symmetric key
    AES_key = Fernet.generate_key()
    cipher = Fernet(AES_key)
```

- Then we will open the file the user have been choose and want to encrypt.
  We apply this by:



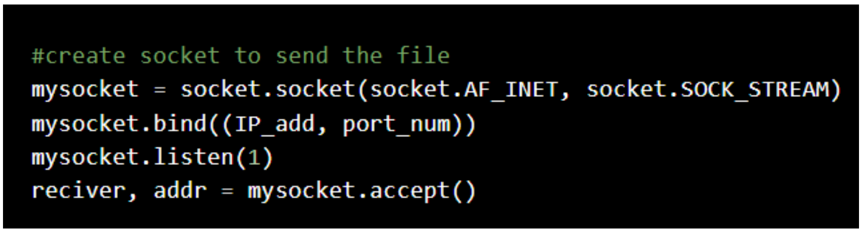
```
# open the file we want to encrypt
openfile = open(self.file_path,'rb')
filedata = openfile.read() # read file content
```

- After bring and open the file, now we will encrypt it content with an AES algorithm, Then we will save it in temporary file so we apply this by :



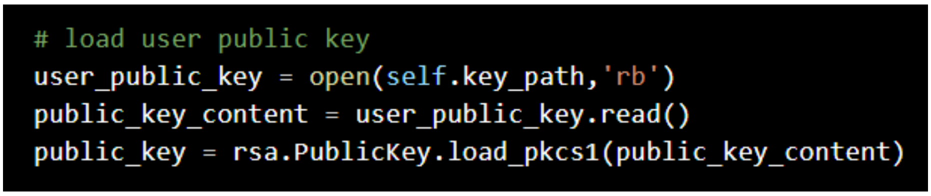
```
# encrypt the content
encrypte_content = cipher.encrypt(filedata)

#temp file for encrypted content
dataenc = open('encrypted.'+str(os.path.splitext(self.file_path)[1][1:].strip()),'wb')
dataenc.write(encrypte_content)
```

- Now we need to send the file and the AES key. But we need first to encrypt the AES key with an RSA algorithm using the receiver public key, So we need to create a socket to load the user public key and send the file so we apply this by :



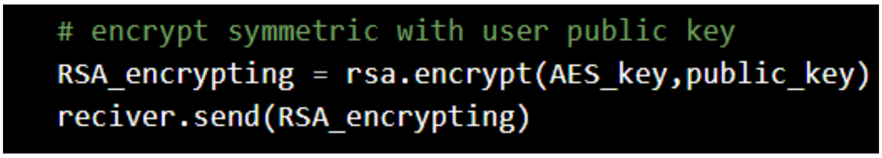
```
#create socket to send the file
mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysocket.bind((IP_add, port_num))
mysocket.listen(1)
reciver, addr = mysocket.accept()
```

- It will load the receiver public key to encrypt the symmetric key, so we apply this by:



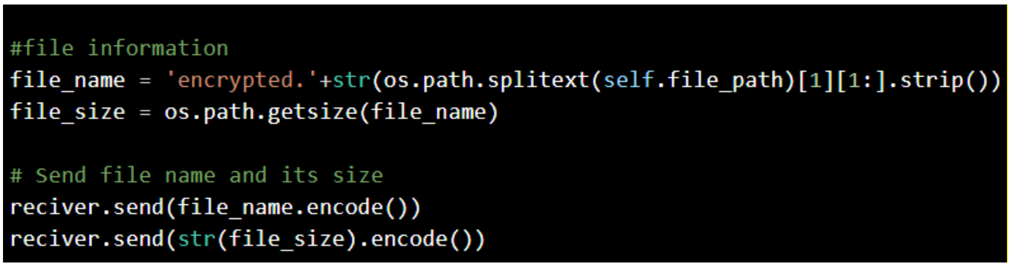
```
# load user public key
user_public_key = open(self.key_path,'rb')
public_key_content = user_public_key.read()
public_key = rsa.PublicKey.load_pkcs1(public_key_content)
```

- After loading the receiver public key, Now we will encrypt the symmetric key with the user public key using the RSA algorithm. We apply this by:



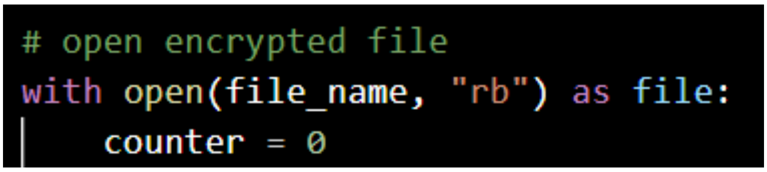
```
# encrypt symmetric with user public key
RSA_encrypting = rsa.encrypt(AES_key,public_key)
reciver.send(RSA_encrypting)
```

- To send the file for the receiver we will first bring the information of the file, to compere it later. We apply this by :



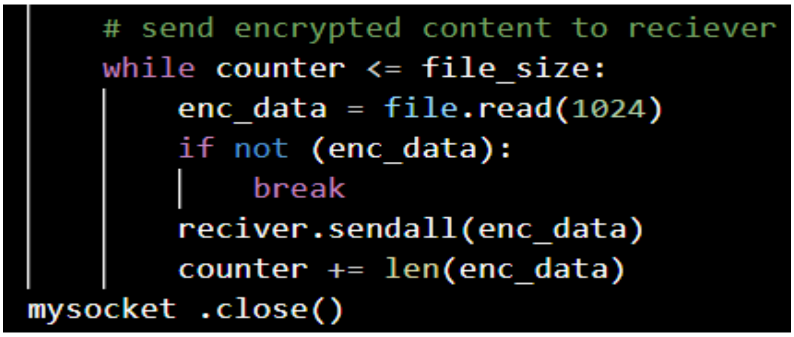
```
#file information
file_name = 'encrypted.'+str(os.path.splitext(self.file_path)[1][1:].strip())
file_size = os.path.getsize(file_name)

# Send file name and its size
reciver.send(file_name.encode())
reciver.send(str(file_size).encode())
```

- It will open the encrypted file that was save in the temporary file. We apply this by:



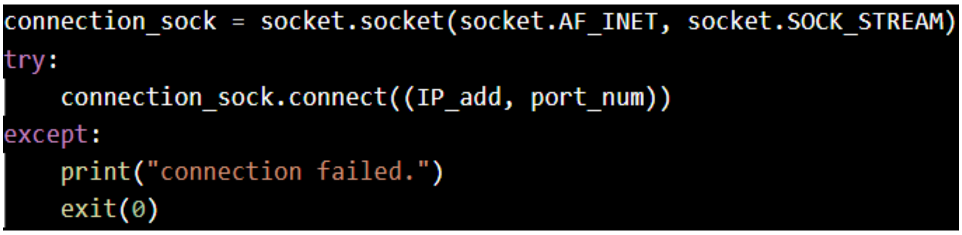
```
# open encrypted file
with open(file_name, "rb") as file:
    counter = 0
```

- Now we will send the encrypted file using this while loop and this loop will make it secure and no one can interrupt it and then we will close the socket.
  We apply this by:



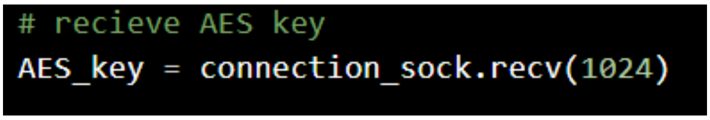
```
    # send encrypted content to reciever
    while counter <= file_size:
        enc_data = file.read(1024)
        if not (enc_data):
            break
        reciver.sendall(enc_data)
        counter += len(enc_data)
mysocket .close()
```

- In class **receive** we have function(recieve_data), this function will first let the receiver to connect to the socket and if the connection failed this message will appear (connection failed.). We apply this by:



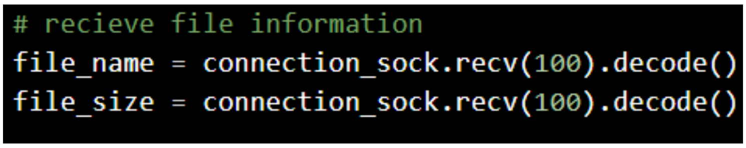
```
connection_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    connection_sock.connect((IP_add, port_num))
except:
    print("connection failed.")
    exit(0)
```

- If the connection succussed, The receiver will receive the AES key that was encrypted with RSA algorithm.
  We apply this by:



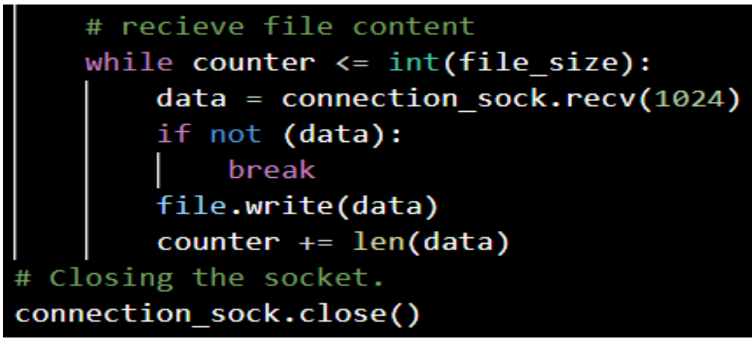
```
# recieve AES key
AES_key = connection_sock.recv(1024)
```

- Then he will receive the file information (file name and the file size).
  We apply this by:



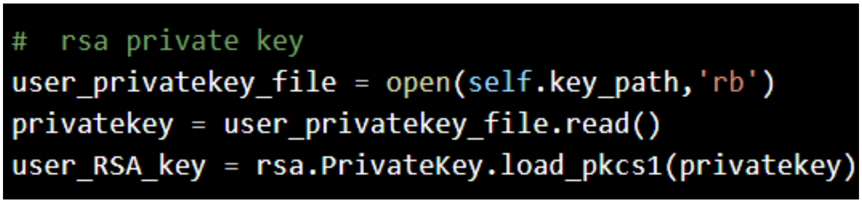
```
# recieve file information
file_name = connection_sock.recv(100).decode()
file_size = connection_sock.recv(100).decode()
```

- Now the receiver will receive the file content, after that the socket will close.
  We apply this by:



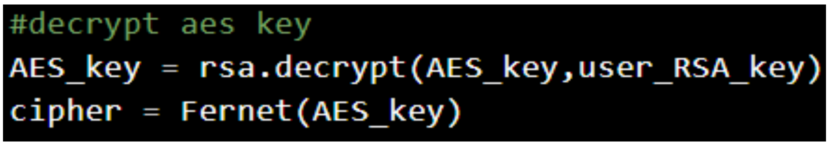
```
    # recieve file content
    while counter <= int(file_size):
        data = connection_sock.recv(1024)
        if not (data):
            break
        file.write(data)
        counter += len(data)
# Closing the socket.
connection_sock.close()
```

- Here we need to bring the receiver stored private key so we can use it to decrypt the RSA encrypted key and get the AES key.
  We apply this by:



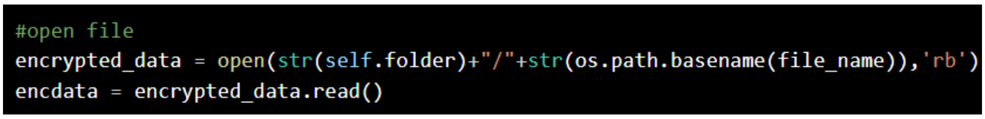
```
#  rsa private key
user_privatekey_file = open(self.key_path,'rb')
privatekey = user_privatekey_file.read()
user_RSA_key = rsa.PrivateKey.load_pkcs1(privatekey)
```

- Now we will decrypt the AES key and get the original one.
  We apply this by:



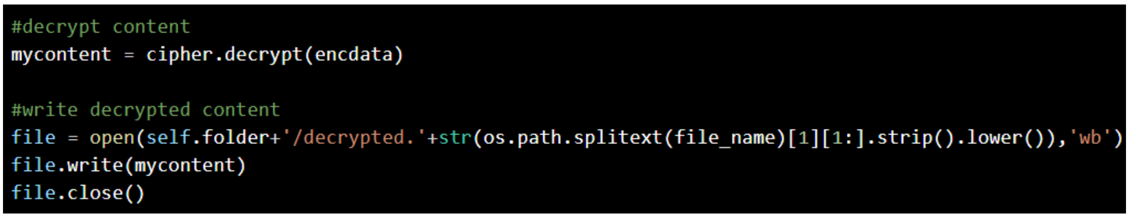
```
#decrypt aes key
AES_key = rsa.decrypt(AES_key,user_RSA_key)
cipher = Fernet(AES_key)
```

- Now we will open the file that we get from the sender to read the encrypted data.
  We apply this by:



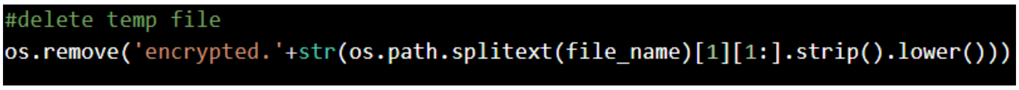
```
#open file
encrypted_data = open(str(self.folder)+"/"+str(os.path.basename(file_name)),'rb')
encdata = encrypted_data.read()
```

- Here we will decrypt the content,
  We apply this by:



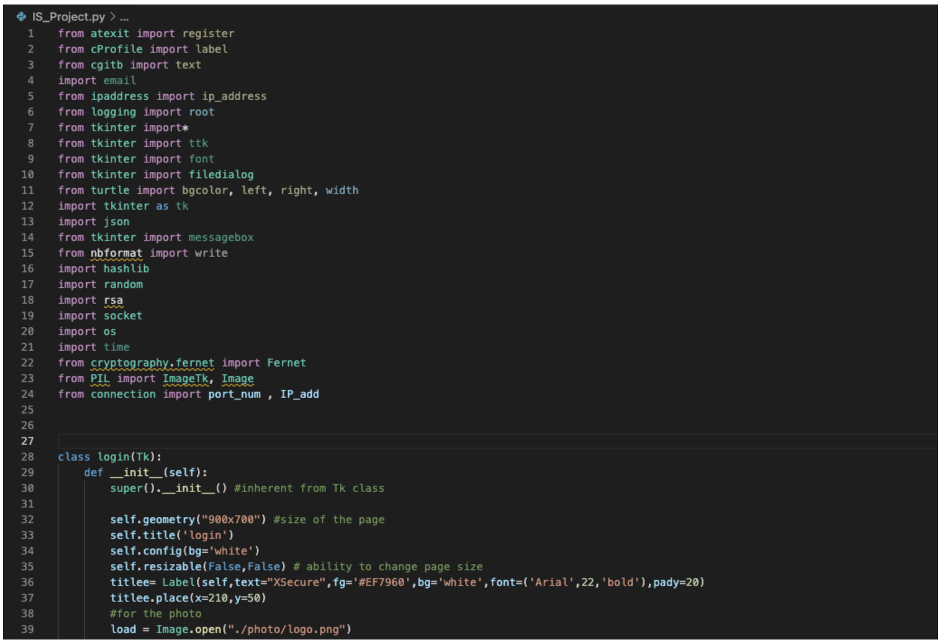
```
#decrypt content
mycontent = cipher.decrypt(encdata)

#write decrypted content
file = open(self.folder+'/decrypted.'+str(os.path.splitext(file_name)[1][1:].strip().lower()),'wb')
file.write(mycontent)
file.close()
```

- Then we will delete the temporary file that we get it from the sender.
  We apply this by:



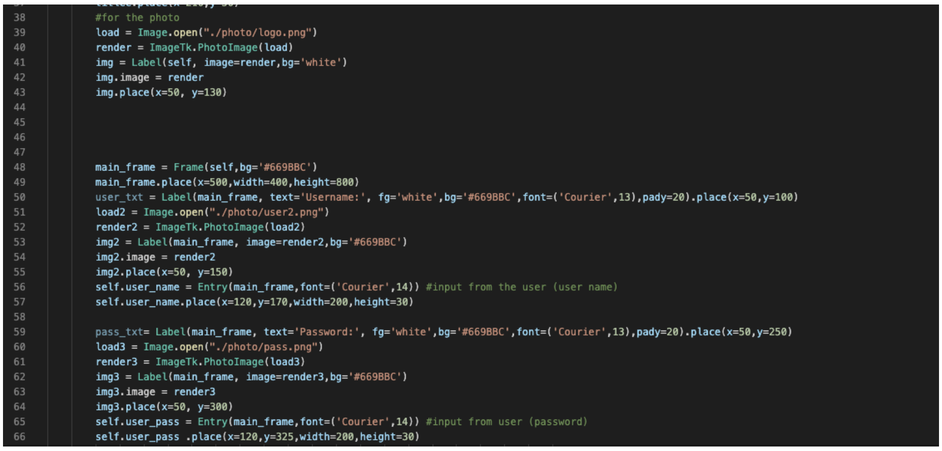
```
#delete temp file
os.remove('encrypted.'+str(os.path.splitext(file_name)[1][1:].strip().lower()))
```

4.Code

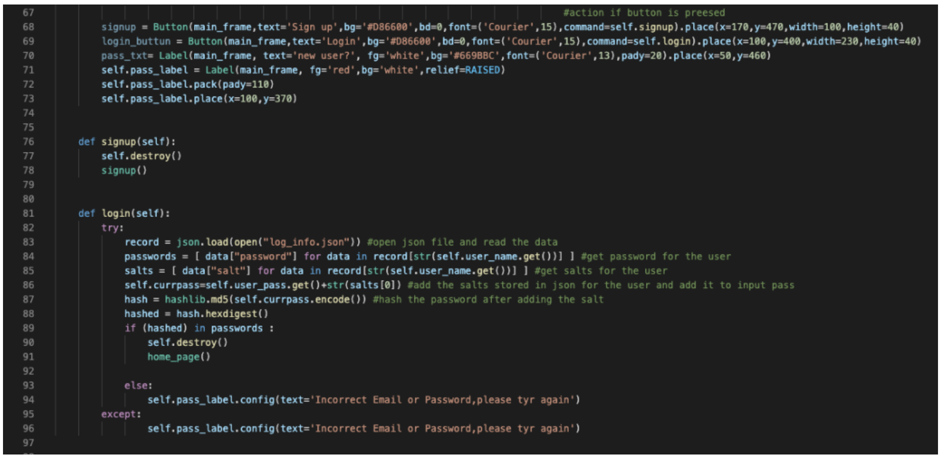


IS_Project.py > ...

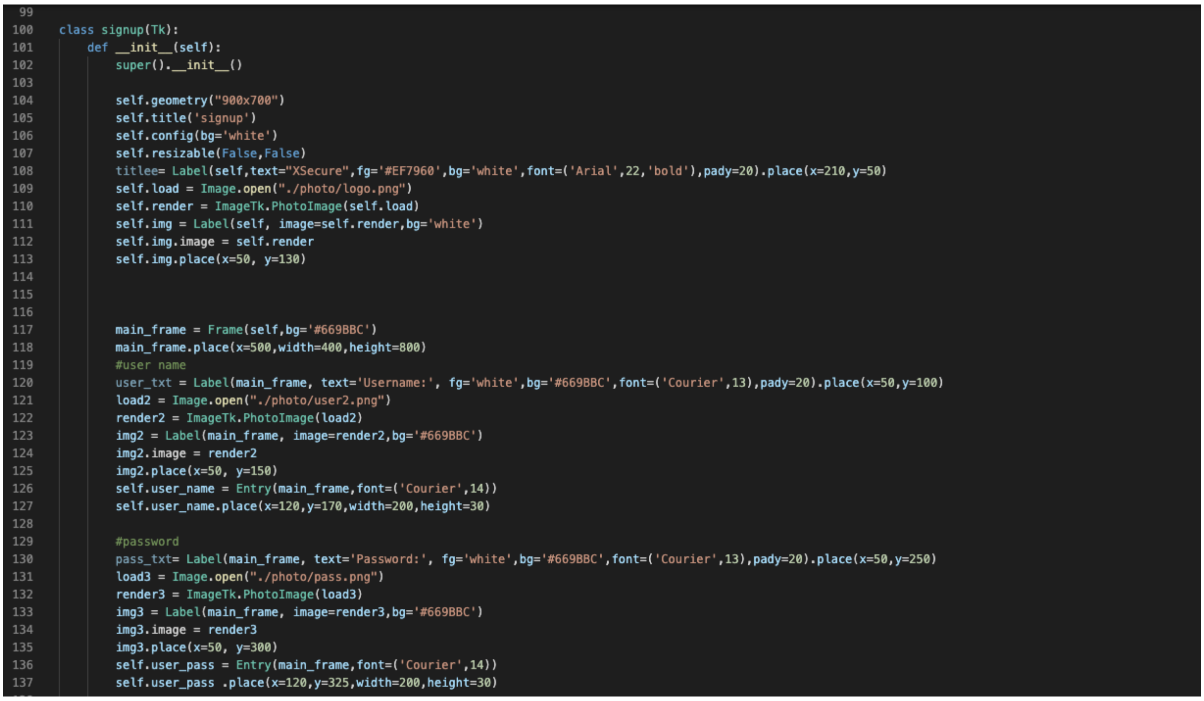
```python
from atexit import register
from cProfile import label
from cgitb import text
import email
from ipaddress import ip_address
from logging import root
from tkinter import*
from tkinter import ttk
from tkinter import font
from tkinter import filedialog
from turtle import bgcolor, left, right, width
import tkinter as tk
import json
from tkinter import messagebox
from nbformat import write
import hashlib
import random
import rsa
import socket
import os
import time
from cryptography.fernet import Fernet
from PIL import ImageTk, Image
from connection import port_num , IP_add



class login(Tk):
    def __init__(self):
        super().__init__() #inherent from Tk class

        self.geometry("900x700") #size of the page
        self.title('login')
        self.config(bg='white')
        self.resizable(False,False) # ability to change page size
        titlee= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20)
        titlee.place(x=210,y=50)
        #for the photo
        load = Image.open("./photo/logo.png")
```



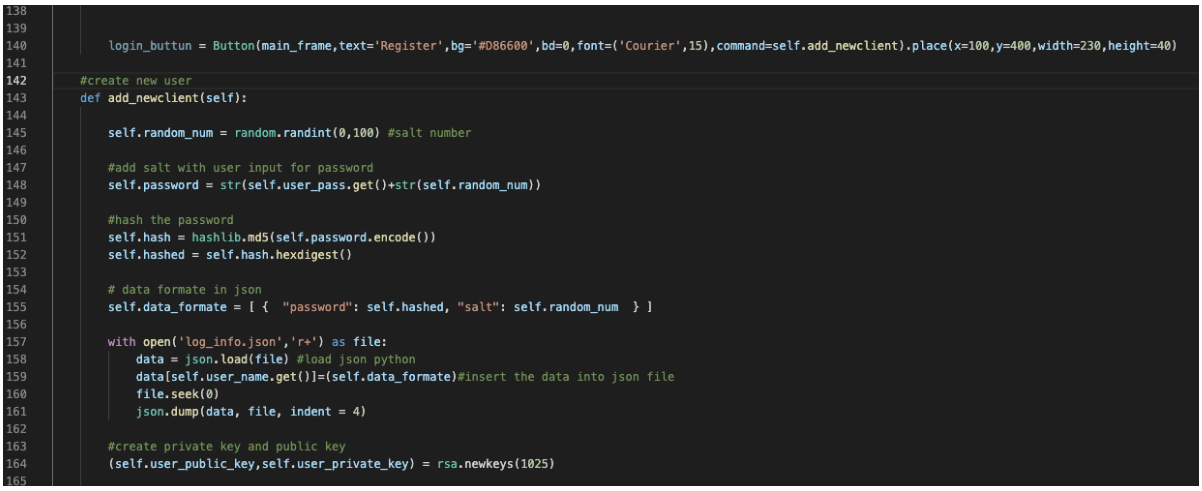
```python
        #for the photo
        load = Image.open("./photo/logo.png")
        render = ImageTk.PhotoImage(load)
        img = Label(self, image=render,bg='white')
        img.image = render
        img.place(x=50, y=130)




        main_frame = Frame(self,bg='#669BBC')
        main_frame.place(x=500,width=400,height=800)
        user_txt = Label(main_frame, text='Username:', fg='white',bg='#669BBC',font=('Courier',13),pady=20).place(x=50,y=100)
        load2 = Image.open("./photo/user2.png")
        render2 = ImageTk.PhotoImage(load2)
        img2 = Label(main_frame, image=render2,bg='#669BBC')
        img2.image = render2
        img2.place(x=50, y=150)
        self.user_name = Entry(main_frame,font=('Courier',14)) #input from the user (user name)
        self.user_name.place(x=120,y=170,width=200,height=30)

        pass_txt= Label(main_frame, text='Password:', fg='white',bg='#669BBC',font=('Courier',13),pady=20).place(x=50,y=250)
        load3 = Image.open("./photo/pass.png")
        render3 = ImageTk.PhotoImage(load3)
        img3 = Label(main_frame, image=render3,bg='#669BBC')
        img3.image = render3
        img3.place(x=50, y=300)
        self.user_pass = Entry(main_frame,font=('Courier',14)) #input from user (password)
        self.user_pass .place(x=120,y=325,width=200,height=30)
```



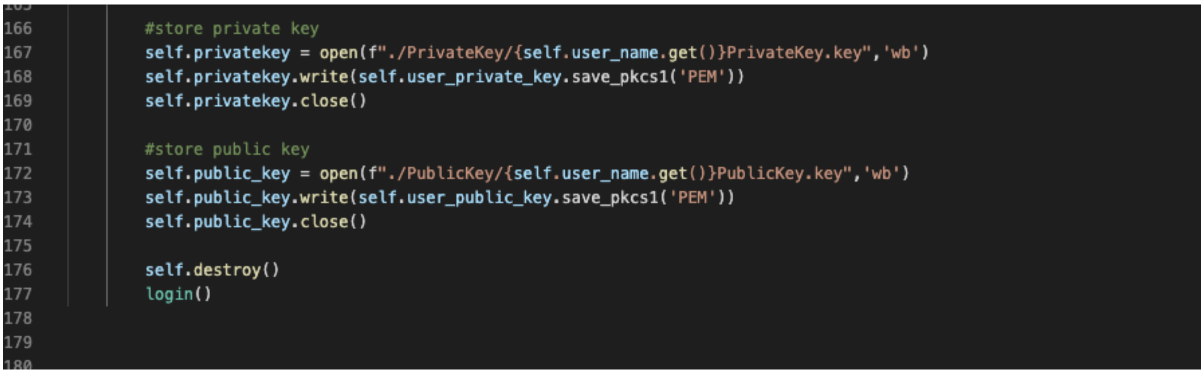
```python
                                                                         #action if button is preesed
        signup = Button(main_frame,text='Sign up',bg='#D86600',bd=0,font=('Courier',15),command=self.signup).place(x=170,y=470,width=100,height=40)
        login_buttun = Button(main_frame,text='Login',bg='#D86600',bd=0,font=('Courier',15),command=self.login).place(x=100,y=400,width=230,height=40)
        pass_txt= Label(main_frame, text='new user?', fg='white',bg='#669BBC',font=('Courier',13),pady=20).place(x=50,y=460)
        self.pass_label = Label(main_frame, fg='red',bg='white',relief=RAISED)
        self.pass_label.pack(pady=110)
        self.pass_label.place(x=100,y=370)


    def signup(self):
        self.destroy()
        signup()


    def login(self):
        try:
            record = json.load(open("log_info.json")) #open json file and read the data
            passwords = [ data["password"] for data in record[str(self.user_name.get())] ] #get password for the user
            salts = [ data["salt"] for data in record[str(self.user_name.get())] ] #get salts for the user
            self.currpass=self.user_pass.get()+str(salts[0]) #add the salts stored in json for the user and add it to input pass
            hash = hashlib.md5(self.currpass.encode()) #hash the password after adding the salt
            hashed = hash.hexdigest()
            if (hashed) in passwords :
                self.destroy()
                home_page()

            else:
                self.pass_label.config(text='Incorrect Email or Password,please tyr again')
        except:
                self.pass_label.config(text='Incorrect Email or Password,please tyr again')
```



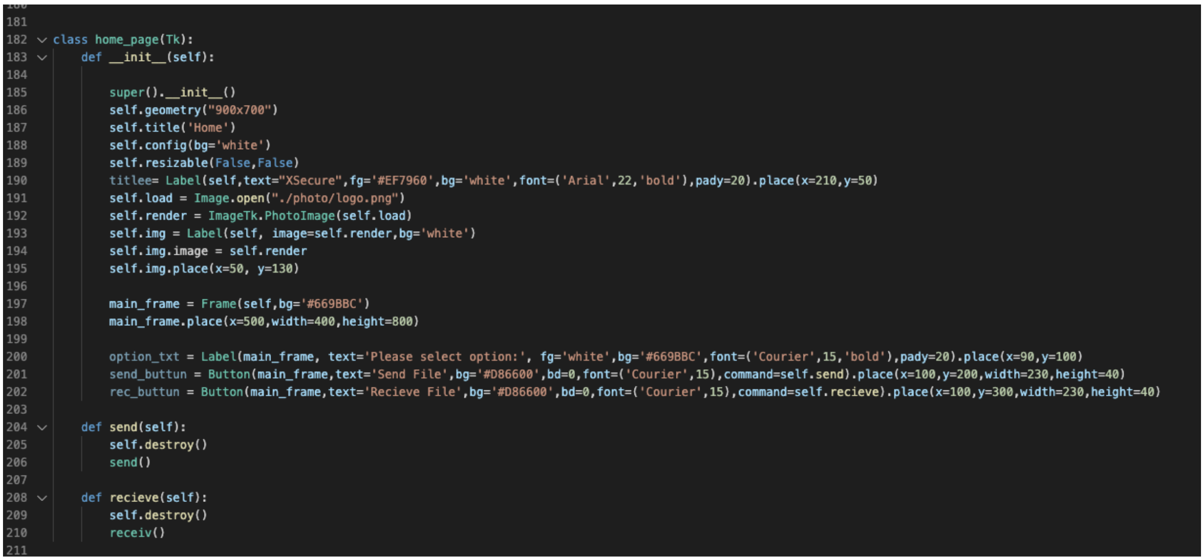
```python

class signup(Tk):
    def __init__(self):
        super().__init__()

        self.geometry("900x700")
        self.title('signup')
        self.config(bg='white')
        self.resizable(False,False)
        titlee= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
        self.load = Image.open("./photo/logo.png")
        self.render = ImageTk.PhotoImage(self.load)
        self.img = Label(self, image=self.render,bg='white')
        self.img.image = self.render
        self.img.place(x=50, y=130)



        main_frame = Frame(self,bg='#669BBC')
        main_frame.place(x=500,width=400,height=800)
        #user name
        user_txt = Label(main_frame, text='Username:', fg='white',bg='#669BBC',font=('Courier',13),pady=20).place(x=50,y=100)
        load2 = Image.open("./photo/user2.png")
        render2 = ImageTk.PhotoImage(load2)
        img2 = Label(main_frame, image=render2,bg='#669BBC')
        img2.image = render2
        img2.place(x=50, y=150)
        self.user_name = Entry(main_frame,font=('Courier',14))
        self.user_name.place(x=120,y=170,width=200,height=30)

        #password
        pass_txt= Label(main_frame, text='Password:', fg='white',bg='#669BBC',font=('Courier',13),pady=20).place(x=50,y=250)
        load3 = Image.open("./photo/pass.png")
        render3 = ImageTk.PhotoImage(load3)
        img3 = Label(main_frame, image=render3,bg='#669BBC')
        img3.image = render3
        img3.place(x=50, y=300)
        self.user_pass = Entry(main_frame,font=('Courier',14))
        self.user_pass .place(x=120,y=325,width=200,height=30)
```



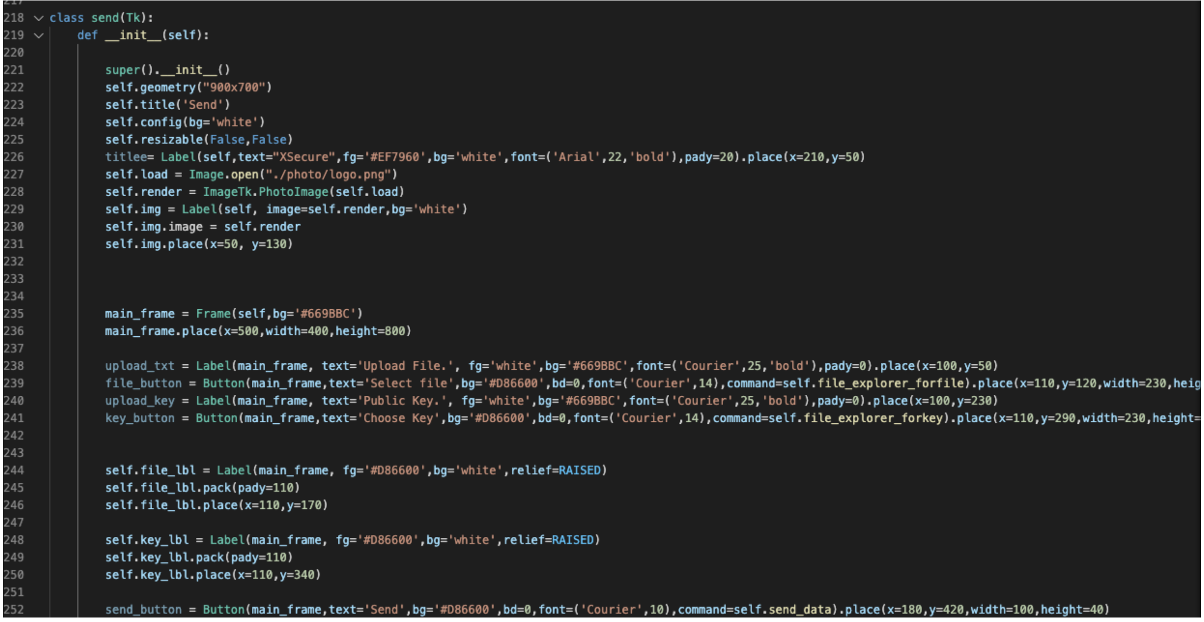
```python


        login_buttun = Button(main_frame,text='Register',bg='#D86600',bd=0,font=('Courier',15),command=self.add_newclient).place(x=100,y=400,width=230,height=40)

    #create new user
    def add_newclient(self):

        self.random_num = random.randint(0,100) #salt number

        #add salt with user input for password
        self.password = str(self.user_pass.get()+str(self.random_num))

        #hash the password
        self.hash = hashlib.md5(self.password.encode())
        self.hashed = self.hash.hexdigest()

        # data formate in json
        self.data_formate = [ {  "password": self.hashed, "salt": self.random_num  } ]

        with open('log_info.json','r+') as file:
            data = json.load(file) #load json python
            data[self.user_name.get()]=(self.data_formate)#insert the data into json file
            file.seek(0)
            json.dump(data, file, indent = 4)

        #create private key and public key
        (self.user_public_key,self.user_private_key) = rsa.newkeys(1025)

```



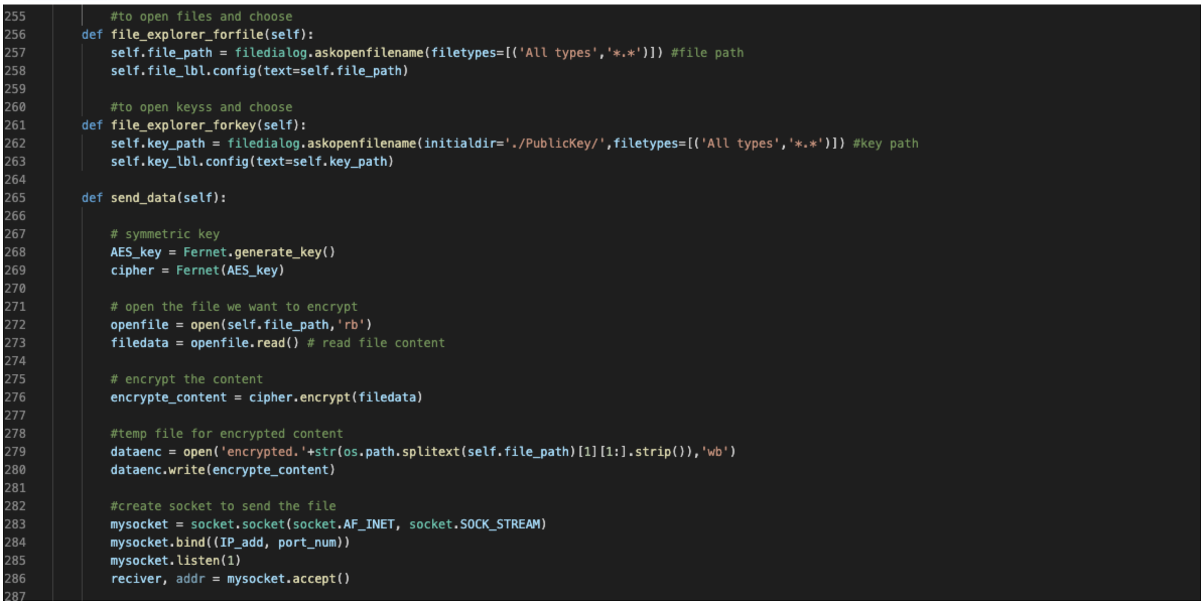
```python
        #store private key
        self.privatekey = open(f"./PrivateKey/{self.user_name.get()}PrivateKey.key",'wb')
        self.privatekey.write(self.user_private_key.save_pkcs1('PEM'))
        self.privatekey.close()

        #store public key
        self.public_key = open(f"./PublicKey/{self.user_name.get()}PublicKey.key",'wb')
        self.public_key.write(self.user_public_key.save_pkcs1('PEM'))
        self.public_key.close()

        self.destroy()
        login()



```



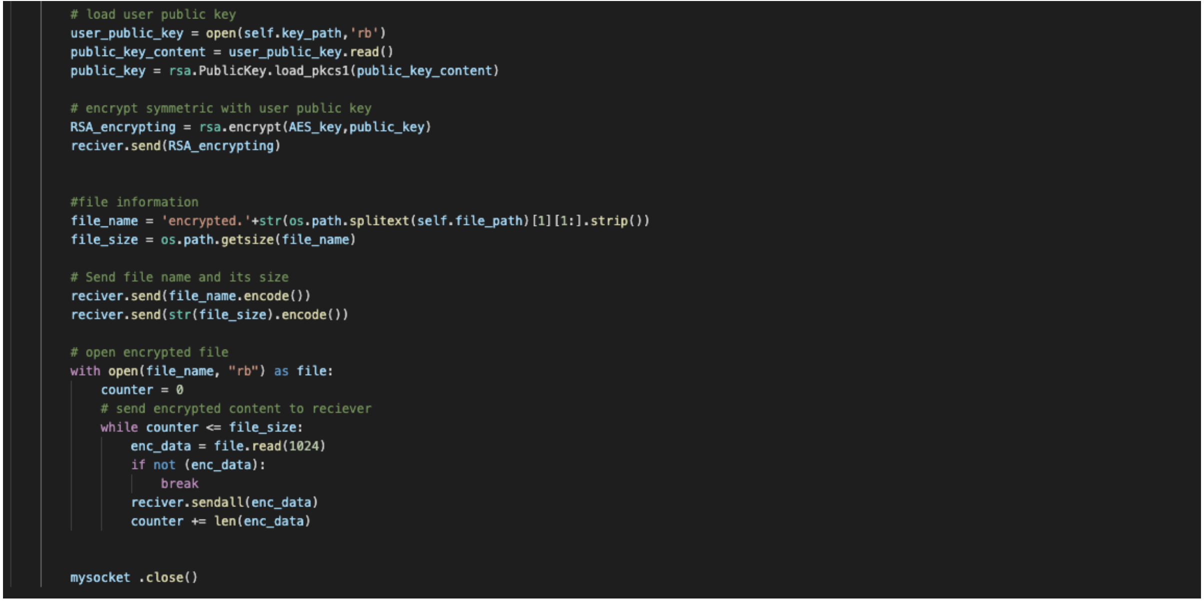
```python
class home_page(Tk):
    def __init__(self):

        super().__init__()
        self.geometry("900x700")
        self.title('Home')
        self.config(bg='white')
        self.resizable(False,False)
        titlee= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
        self.load = Image.open("./photo/logo.png")
        self.render = ImageTk.PhotoImage(self.load)
        self.img = Label(self, image=self.render,bg='white')
        self.img.image = self.render
        self.img.place(x=50, y=130)

        main_frame = Frame(self,bg='#669BBC')
        main_frame.place(x=500,width=400,height=800)

        option_txt = Label(main_frame, text='Please select option:', fg='white',bg='#669BBC',font=('Courier',15,'bold'),pady=20).place(x=90,y=100)
        send_buttun = Button(main_frame,text='Send File',bg='#D86600',bd=0,font=('Courier',15),command=self.send).place(x=100,y=200,width=230,height=40)
        rec_buttun = Button(main_frame,text='Recieve File',bg='#D86600',bd=0,font=('Courier',15),command=self.recieve).place(x=100,y=300,width=230,height=40)

    def send(self):
        self.destroy()
        send()

    def recieve(self):
        self.destroy()
        receiv()

```



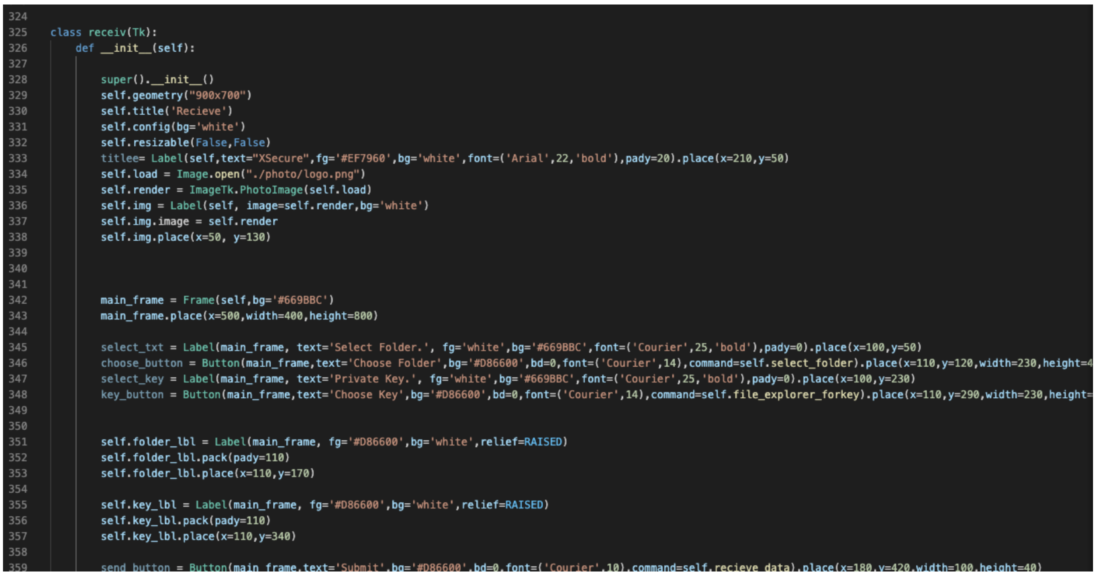
```python
class send(Tk):
    def __init__(self):

        super().__init__()
        self.geometry("900x700")
        self.title('Send')
        self.config(bg='white')
        self.resizable(False,False)
        titlee= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
        self.load = Image.open("./photo/logo.png")
        self.render = ImageTk.PhotoImage(self.load)
        self.img = Label(self, image=self.render,bg='white')
        self.img.image = self.render
        self.img.place(x=50, y=130)



        main_frame = Frame(self,bg='#669BBC')
        main_frame.place(x=500,width=400,height=800)

        upload_txt = Label(main_frame, text='Upload File.', fg='white',bg='#669BBC',font=('Courier',25,'bold'),pady=0).place(x=100,y=50)
        file_button = Button(main_frame,text='Select file',bg='#D86600',bd=0,font=('Courier',14),command=self.file_explorer_forfile).place(x=110,y=120,width=230,heig
        upload_key = Label(main_frame, text='Public Key.', fg='white',bg='#669BBC',font=('Courier',25,'bold'),pady=0).place(x=100,y=230)
        key_button = Button(main_frame,text='Choose Key',bg='#D86600',bd=0,font=('Courier',14),command=self.file_explorer_forkey).place(x=110,y=290,width=230,height=


        self.file_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
        self.file_lbl.pack(pady=110)
        self.file_lbl.place(x=110,y=170)

        self.key_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
        self.key_lbl.pack(pady=110)
        self.key_lbl.place(x=110,y=340)

        send_button = Button(main_frame,text='Send',bg='#D86600',bd=0,font=('Courier',10),command=self.send_data).place(x=180,y=420,width=100,height=40)
```



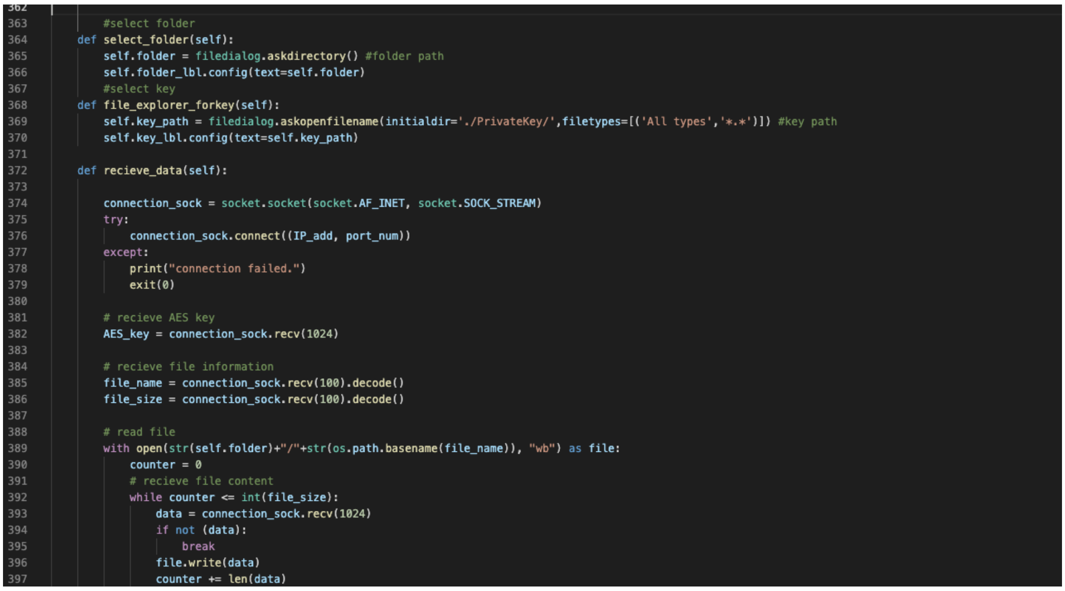
```python
    #to open files and choose
    def file_explorer_forfile(self):
        self.file_path = filedialog.askopenfilename(filetypes=[('All types','*.*')]) #file path
        self.file_lbl.config(text=self.file_path)

    #to open keyss and choose
    def file_explorer_forkey(self):
        self.key_path = filedialog.askopenfilename(initialdir='./PublicKey/',filetypes=[('All types','*.*')]) #key path
        self.key_lbl.config(text=self.key_path)

    def send_data(self):

        # symmetric key
        AES_key = Fernet.generate_key()
        cipher = Fernet(AES_key)

        # open the file we want to encrypt
        openfile = open(self.file_path,'rb')
        filedata = openfile.read() # read file content

        # encrypt the content
        encrypte_content = cipher.encrypt(filedata)

        #temp file for encrypted content
        dataenc = open('encrypted.'+str(os.path.splitext(self.file_path)[1][1:].strip()),'wb')
        dataenc.write(encrypte_content)

        #create socket to send the file
        mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        mysocket.bind((IP_add, port_num))
        mysocket.listen(1)
        reciver, addr = mysocket.accept()
```



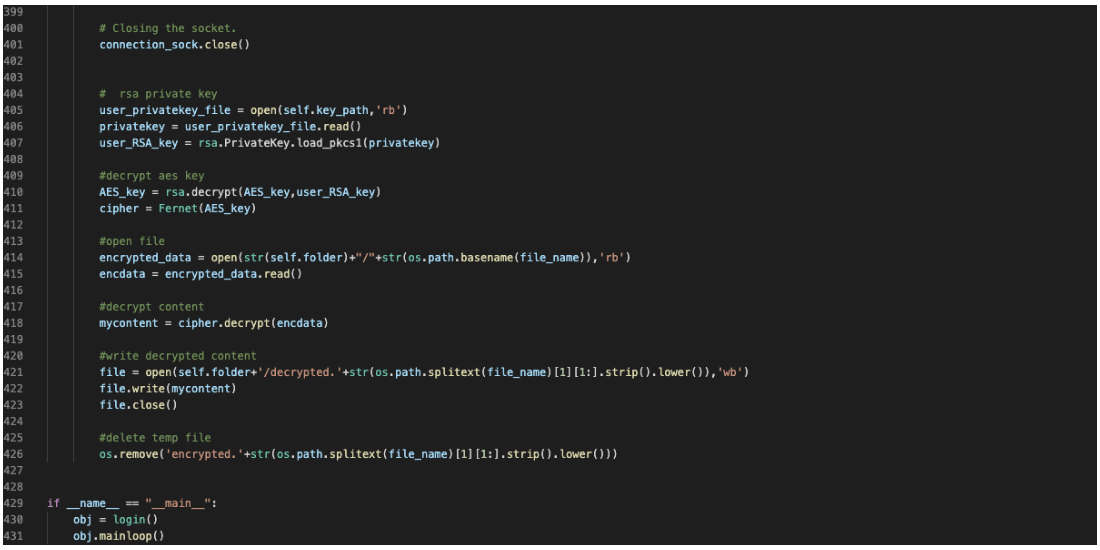
```python
        # load user public key
        user_public_key = open(self.key_path,'rb')
        public_key_content = user_public_key.read()
        public_key = rsa.PublicKey.load_pkcs1(public_key_content)

        # encrypt symmetric with user public key
        RSA_encrypting = rsa.encrypt(AES_key,public_key)
        reciver.send(RSA_encrypting)


        #file information
        file_name = 'encrypted.'+str(os.path.splitext(self.file_path)[1][1:].strip())
        file_size = os.path.getsize(file_name)

        # Send file name and its size
        reciver.send(file_name.encode())
        reciver.send(str(file_size).encode())

        # open encrypted file
        with open(file_name, "rb") as file:
            counter = 0
            # send encrypted content to reciever
            while counter <= file_size:
                enc_data = file.read(1024)
                if not (enc_data):
                    break
                reciver.sendall(enc_data)
                counter += len(enc_data)


        mysocket .close()
```



```python

class receiv(Tk):
    def __init__(self):

        super().__init__()
        self.geometry("900x700")
        self.title('Recieve')
        self.config(bg='white')
        self.resizable(False,False)
        titlee= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
        self.load = Image.open("./photo/logo.png")
        self.render = ImageTk.PhotoImage(self.load)
        self.img = Label(self, image=self.render,bg='white')
        self.img.image = self.render
        self.img.place(x=50, y=130)



        main_frame = Frame(self,bg='#669BBC')
        main_frame.place(x=500,width=400,height=800)

        select_txt = Label(main_frame, text='Select Folder.', fg='white',bg='#669BBC',font=('Courier',25,'bold'),pady=0).place(x=100,y=50)
        choose_button = Button(main_frame,text='Choose Folder',bg='#D86600',bd=0,font=('Courier',14),command=self.select_folder).place(x=110,y=120,width=230,height=4
        select_key = Label(main_frame, text='Private Key.', fg='white',bg='#669BBC',font=('Courier',25,'bold'),pady=0).place(x=100,y=230)
        key_button = Button(main_frame,text='Choose Key',bg='#D86600',bd=0,font=('Courier',14),command=self.file_explorer_forkey).place(x=110,y=290,width=230,height=


        self.folder_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
        self.folder_lbl.pack(pady=110)
        self.folder_lbl.place(x=110,y=170)

        self.key_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
        self.key_lbl.pack(pady=110)
        self.key_lbl.place(x=110,y=340)

        send_button = Button(main_frame,text='Submit',bg='#D86600',bd=0,font=('Courier',10),command=self.recieve_data).place(x=180,y=420,width=100,height=40)
```



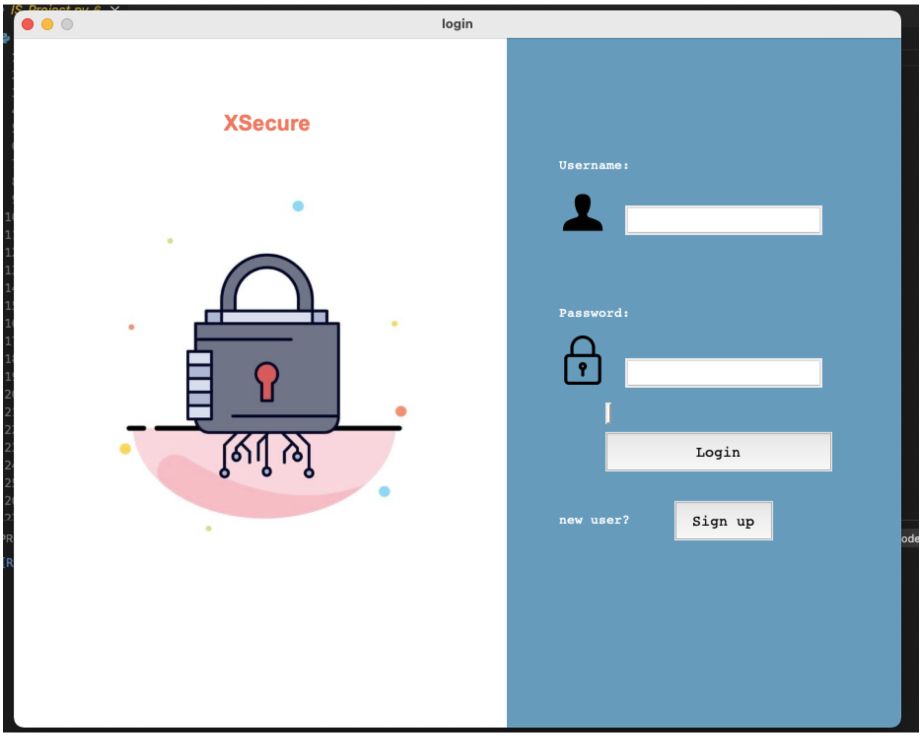
```python

        #select folder
    def select_folder(self):
        self.folder = filedialog.askdirectory() #folder path
        self.folder_lbl.config(text=self.folder)
        #select key
    def file_explorer_forkey(self):
        self.key_path = filedialog.askopenfilename(initialdir='./PrivateKey/',filetypes=[('All types','*.*')]) #key path
        self.key_lbl.config(text=self.key_path)

    def recieve_data(self):

        connection_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            connection_sock.connect((IP_add, port_num))
        except:
            print("connection failed.")
            exit(0)

        # recieve AES key
        AES_key = connection_sock.recv(1024)

        # recieve file information
        file_name = connection_sock.recv(100).decode()
        file_size = connection_sock.recv(100).decode()

        # read file
        with open(str(self.folder)+"/"+str(os.path.basename(file_name)), "wb") as file:
            counter = 0
            # recieve file content
            while counter <= int(file_size):
                data = connection_sock.recv(1024)
                if not (data):
                    break
                file.write(data)
                counter += len(data)
```



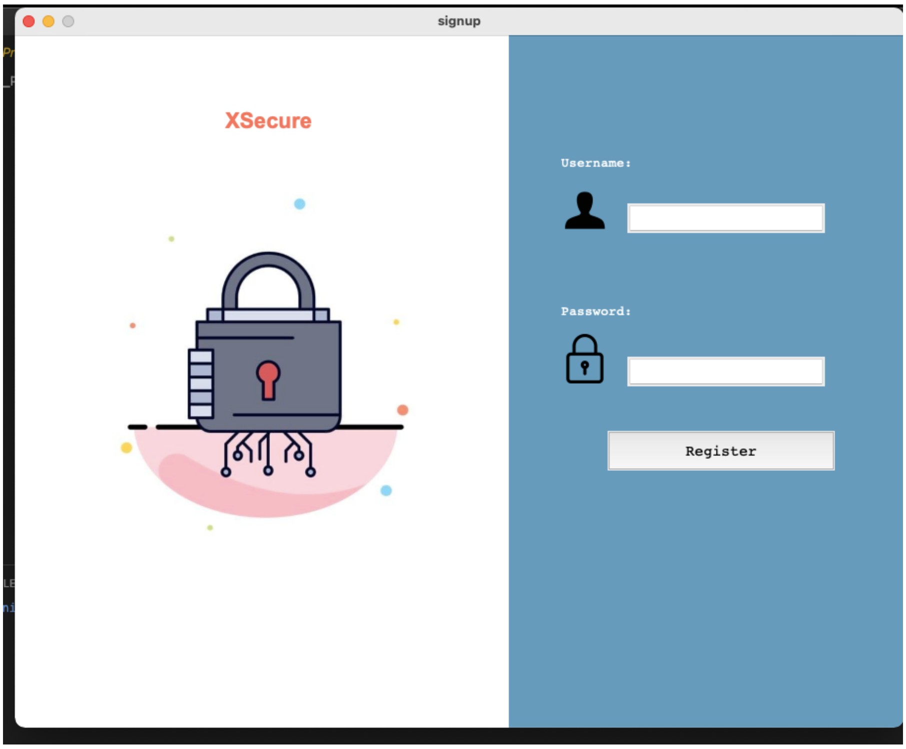
```python

        # Closing the socket.
        connection_sock.close()


        #  rsa private key
        user_privatekey_file = open(self.key_path,'rb')
        privatekey = user_privatekey_file.read()
        user_RSA_key = rsa.PrivateKey.load_pkcs1(privatekey)

        #decrypt aes key
        AES_key = rsa.decrypt(AES_key,user_RSA_key)
        cipher = Fernet(AES_key)

        #open file
        encrypted_data = open(str(self.folder)+"/"+str(os.path.basename(file_name)),'rb')
        encdata = encrypted_data.read()

        #decrypt content
        mycontent = cipher.decrypt(encdata)

        #write decrypted content
        file = open(self.folder+'/decrypted.'+str(os.path.splitext(file_name)[1][1:].strip().lower()),'wb')
        file.write(mycontent)
        file.close()

        #delete temp file
        os.remove('encrypted.'+str(os.path.splitext(file_name)[1][1:].strip().lower()))


if __name__ == "__main__":
    obj = login()
    obj.mainloop()
```

5.Snapshots of the application

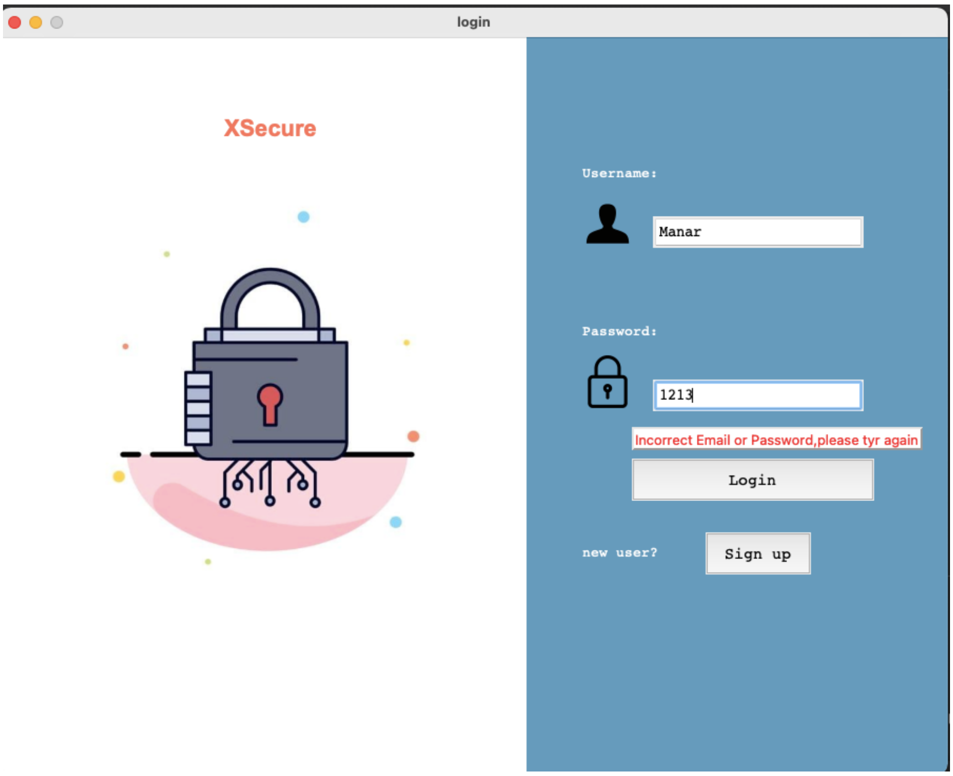
- Login page



- Signup page



- in case of incorrect password

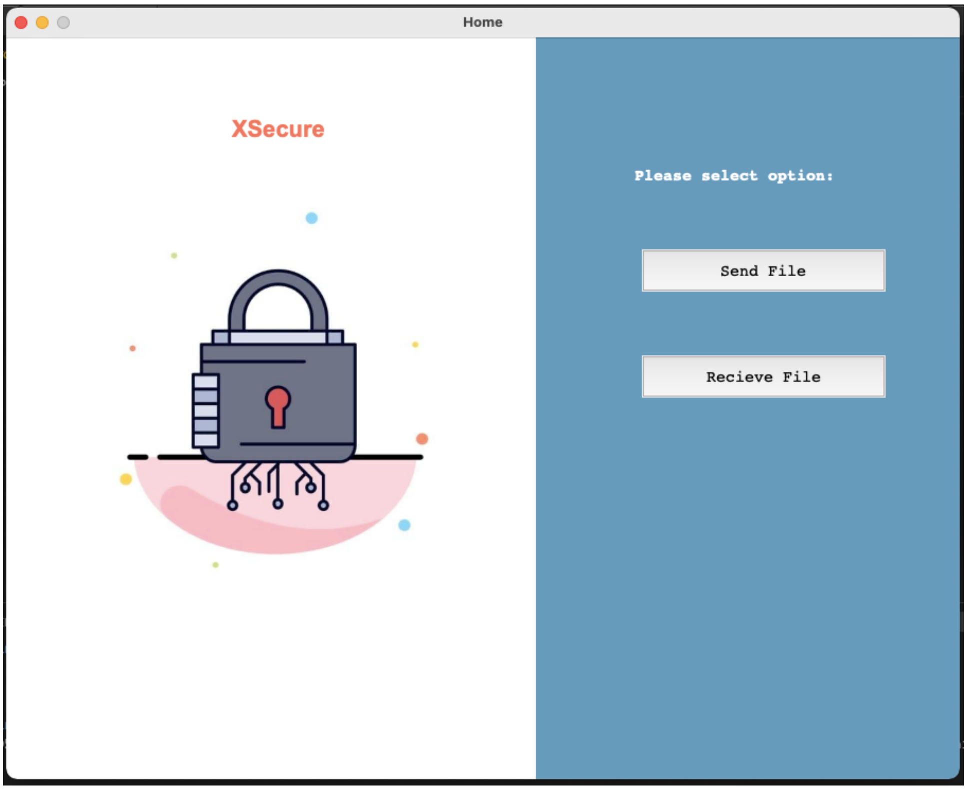


Manar

1213

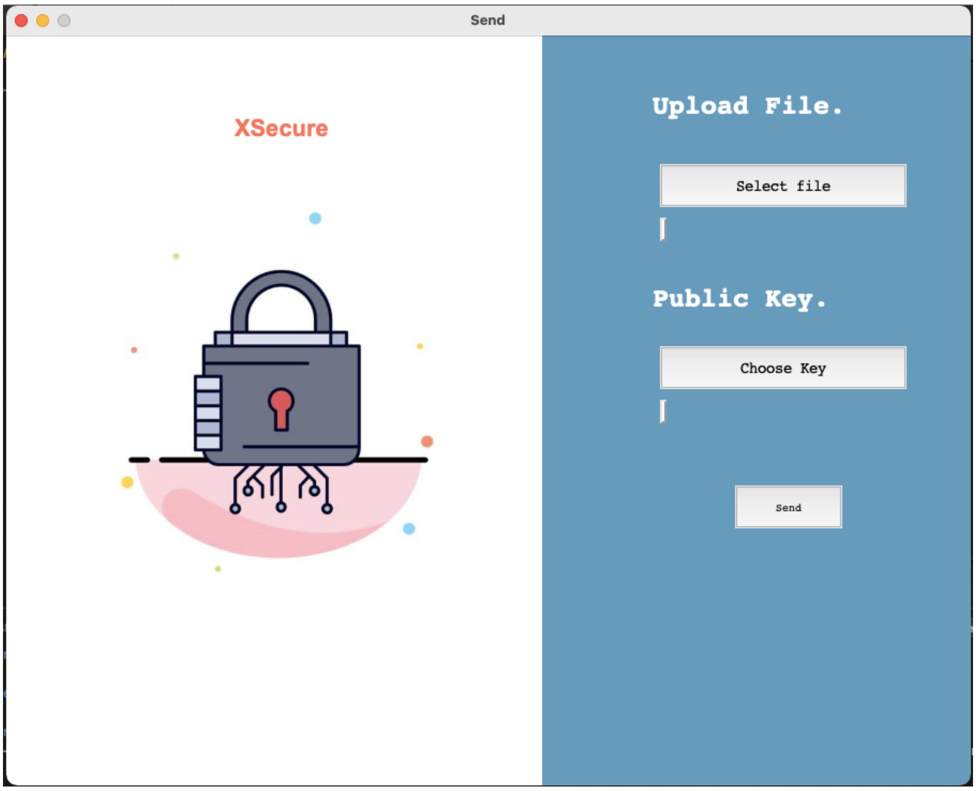
Incorrect Email or Password,please tyr again

- Select (if you want to be sender or receiver) page

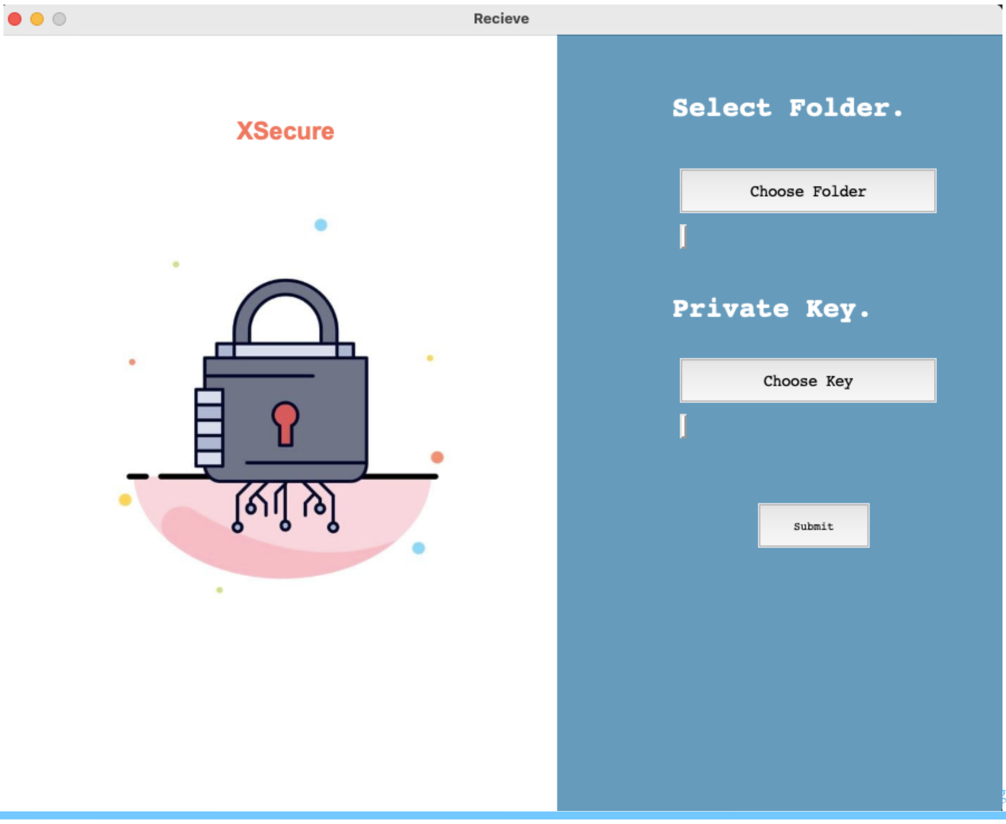


Please select option:

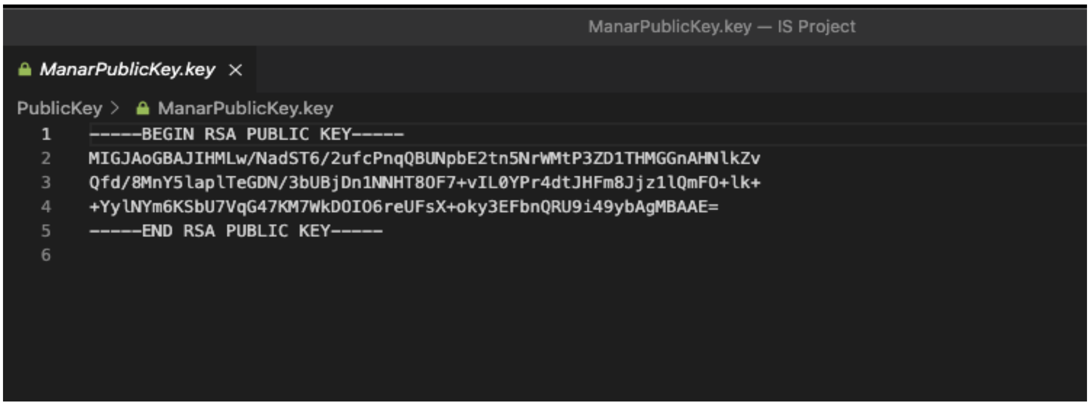
- Sender page



- Receiver page

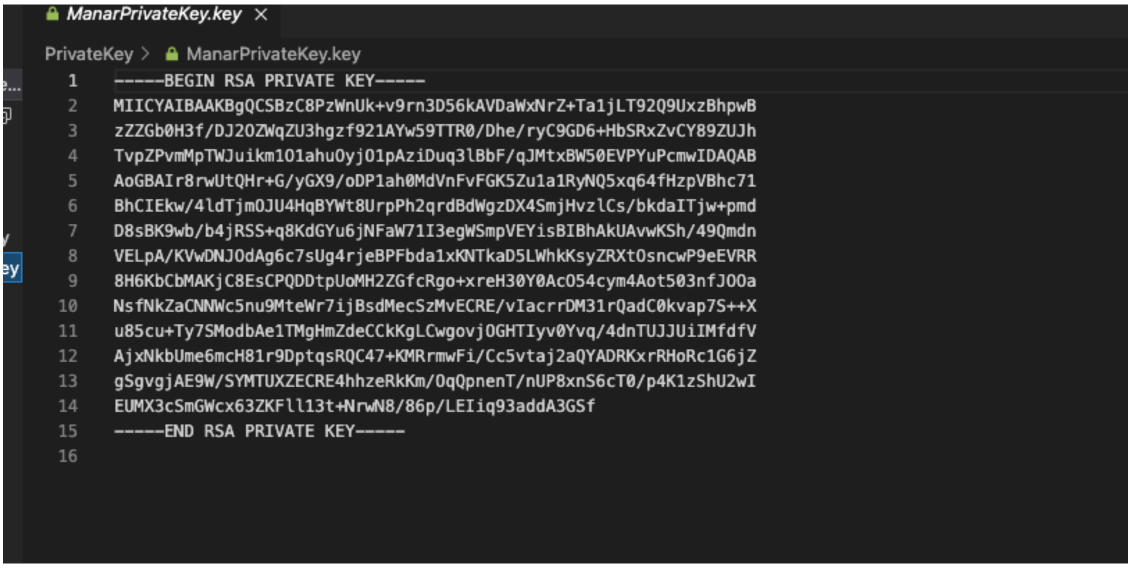


- Example of public key file


ManarPublicKey.key — IS Project
ManarPublicKey.key
PublicKey > ManarPublicKey.key

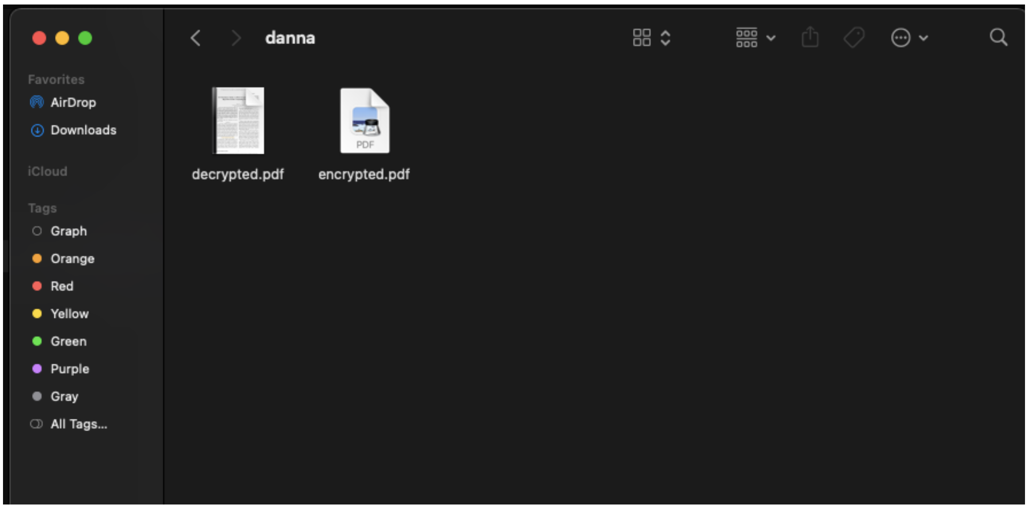
```
-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBAJIHMLw/NadST6/2ufcPnqQBUNpbE2tn5NrWMtP3ZD1THMGGnAHNlkZv
Qfd/8MnY5laplTeGDN/3bUBjDn1NNHT8OF7+vIL0YPr4dtJHFm8Jjz1lQmFO+lk+
+YylNYm6KSbU7VqG47KM7WkDOIO6reUFsX+oky3EFbnQRU9i49ybAgMBAAE=
-----END RSA PUBLIC KEY-----
```

- Example of private key file



```
-----BEGIN RSA PRIVATE KEY-----
MIICYAIBAAKBgQCSBzC8PzWnUk+v9rn3D56kAVDaWxNrZ+Ta1jLT92Q9UxzBhpwB
zZZGb0H3f/DJ2OZWqZU3hgzf921AYw59TTR0/Dhe/ryC9GD6+HbSRxZvCY89ZUJh
TvpZPvmMpTWJuikm1O1ahuOyjO1pAziDuq3lBbF/qJMtxBW50EVPYuPcmwIDAQAB
AoGBAIr8rwUtQHr+G/yGX9/oDP1ah0MdVnFvFGK5Zu1a1RyNQ5xq64fHzpVBhc71
BhCIEkw/4ldTjmOJU4HqBYWt8UrpPh2qrdBdWgzDX4SmjHvzlCs/bkdaITjw+pmd
D8sBK9wb/b4jRSS+q8KdGYu6jNFaW71I3egWSmpVEYisBIBhAkUAvwKSh/49Qmdn
VELpA/KVwDNJOdAg6c7sUg4rjeBPFbda1xKNTkaD5LWhkKsyZRXtOsncwP9eEVRR
8H6KbCbMAKjC8EsCPQDDtpUoMH2ZGfcRgo+xreH30Y0AcO54cym4Aot503nfJOOa
NsfNkZaCNNWc5nu9MteWr7ijBsdMecSzMvECRE/vIacrrDM31rQadC0kvap7S++X
u85cu+Ty7SModbAe1TMgHmZdeCCkKgLCwgovjOGHTIyv0Yvq/4dnTUJJUiIMfdfV
AjxNkbUme6mcH81r9DptqsRQC47+KMRrmwFi/Cc5vtaj2aQYADRKxrRHoRc1G6jZ
gSgvgjAE9W/SYMTUXZECRE4hhzeRkKm/OqQpnenT/nUP8xnS6cT0/p4K1zShU2wI
EUMX3cSmGWcx63ZKFll13t+NrwN8/86p/LEIiq93addA3GSf
-----END RSA PRIVATE KEY-----
```

- Example of encrypted file



6. Challenges

- Less knowledge with library that work with the program.
- When we trying to decrypt the file, we was facing an error for the file path.
- We was typing the content of the file and the file name and size on the same file which was wrong because we need to send the file name and size on different file to let the receiver check it.
- The Socket function take a lot of time because its new way to us
- Less of references

7. References

Cryptography.io. 2022. *Fernet (symmetric encryption) — Cryptography 38.0.0.dev1 documentation*. [online] Available at: <https://cryptography.io/en/latest/fernet/> [Accessed 8 May 2022].

Docs.python.org. 2022. *tkinter — Python interface to Tcl/Tk — Python 3.10.4 documentation*. [online] Available at: <https://docs.python.org/3/library/tkinter.html> [Accessed 8 May 2022].

GeeksforGeeks. 2022. *Encrypt and Decrypt Files using Python - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/encrypt-and-decrypt-files-using-python/> [Accessed 8 May 2022].

GeeksforGeeks. 2022. *MD5 hash in Python - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/md5-hash-python/> [Accessed 8 May 2022].

GitHub. 2022. *GitHub - mukeshkdangi/encypt_aes: implementation of RSA and (Advanced Encryption Standard) AES*. [online] Available at: <https://github.com/mukeshkdangi/encypt_aes> [Accessed 8 May 2022].

Programcreek.com. 2022. *Python Examples of rsa.newkeys*. [online] Available at: <https://www.programcreek.com/python/example/104537/rsa.newkeys> [Accessed 8 May 2022].

Python, R., 2022. *Socket Programming in Python (Guide) – Real Python*. [online] Realpython.com. Available at: <https://realpython.com/python-sockets/> [Accessed 8 May 2022].

Stack Abuse. 2022. *Reading and Writing JSON to a File in Python*. [online] Available at: <https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/> [Accessed 8 May 2022].

Stack Overflow. 2022. *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Available at: <https://stackoverflow.com> [Accessed 8 May 2022].