Al-Imam Mohammad ibn Saud Islamic University
College of Computer and Information Sciences
Computer Science Department

CS334: Information Security
2nd Semester 1443H / Spring 2022

Course Project

Name	ID	Email
Sadeem Faisal Alqahtani	440021429	sfaalqahtani29@sm.imamu.edu.sa
Sarah Khalid Alaradi	440023365	skmaloridi@sm.imamu.edu.sa
Shoug Ali Alsuhaibani	440022732	ssuhaibani@sm.imamu.edu.sa
Reem Abdulmohsen Alqahtani	440019278	rasalkahtani@sm.imamu.edu.sa

Section: 371

Supervised by: Dr. Tahani Albalawi

Date of Submission 24/4/2022

TABLE OF CONTENTS

INTRODUCTION

In this project, we designed a file sharing application with a usable GUI, that users can use to share files over the internet. When two parties can exchange files/messages in a secure way using both symmetric and asymmetric key approaches. The shared file is encrypted/decrypted with the AES algorithm generating symmetric. Then an RSA algorithm is implemented to encrypt/decrypt the symmetric key.

IMPLEMENTATION

To implement the application we preferred to use python programing language; for the simplicity and the richness of its libraries.

First, we implemented the AES algorithm:
We used AES library to encrypt and decrypt the messages. Cipher Block Chaining(CBC) AES mode was used in both encrypting/decrypting processes. The iv was generated using `token_bytes()` with the size of 16 bytes. Then the file was split into chunks and padded if needed then encrypt/decrypt using `AES.encrypt()` and `AES.decrypt()`. For security reasons we only kept one version of the file either encrypted or decrypted.

Second, we implemented the RSA algorithm:
We used the RSA library to encrypt/decrypt the keys. We generated keys and wrote them in a file by `generateKeys()` function. And return them from the file by `loadKeys()` function. And finally, we used `RSA.encrypt()` function to encrypt the key using the receiver public key and `RSA.decrypt()` function to decrypt the key using the receiver private key simply by using RSA library.

Finally, we implemented the GUI:
We used Tkinter which is a GUI toolkit for python. We used it to create all the interfaces from the login page to the inbox. When the user inputs its login details, the program checks if the user is already a registered user from a file database. On the other hand, when a new user signs up, the password is encrypted using an encryption method to the file database. Additionally, the username info is not case sensitive when saved into the database. When the user logins in, he has the option to send messages or view new messages which we implemented using the AES algorithm.

CODE

The Imports:

```
from tkinter import *
import os
from Crypto.Cipher import AES
import rsa
import os.path
import struct
from secrets import token_bytes
import sys
```

The AES and RSA encrypt/decrypt:

```
## AES file encryption
def encrypt_file(key, filename, chunk_size=64 * 1024):
    output_filename = filename + '.encrypted'
    iv = token_bytes(16)
    encryptor = AES.new(key, AES.MODE_CBC, iv)
    filesize = os.path.getsize(filename)
    with open(filename, 'rb') as inputfile:
        with open(output_filename, 'wb') as outputfile:
            outputfile.write(struct.pack('<Q', filesize))
            outputfile.write(iv)
            while True:
                chunk = inputfile.read(chunk_size)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    chunk += b"\0" * (16 - len(chunk) % 16)
                outputfile.write(encryptor.encrypt(chunk))
    os.remove(filename)

## AES file decryption
def decrypt_file(key, filename, chunk_size=24 * 1024):
    output_filename = os.path.splitext(filename)[0]
    with open(filename, 'rb') as infile:
        origsize = struct.unpack('<Q', infile.read(struct.calcsize('Q')))[0]
        iv = infile.read(16)
        decryptor = AES.new(key, AES.MODE_CBC, iv)
        with open(output_filename, 'wb') as outfile:
            while True:
                chunk = infile.read(chunk_size)
                if len(chunk) == 0:
                    break
                outfile.write(decryptor.decrypt(chunk))
            outfile.truncate(origsize)
    os.remove(filename)
```

```
## RSA symmetric key encryption
def generateKeys():
    NameOfFilep='keys/'+username_info.lower()+'public.pem'
    (publicKey, privateKey) = rsa.newkeys(1024)
    with open(NameOfFilep, 'wb') as p:
        p.write(publicKey.save_pkcs1('PEM'))
    NameOfFilev='keys/'+username_info.lower()+'private.pem'
    with open(NameOfFilev, 'wb') as p:
        p.write(privateKey.save_pkcs1('PEM'))

def loadKeys():
    NameOfFilep='keys/'+username1.lower()+'public.pem'
    with open(NameOfFilep, 'rb') as p:
        publicKey = rsa.PublicKey.load_pkcs1(p.read())
    NameOfFilev='keys/'+username1.lower()+'private.pem'
    with open(NameOfFilev, 'rb') as p:
        privateKey = rsa.PrivateKey.load_pkcs1(p.read())
    return privateKey, publicKey

def encrypt_key(message, key):
    return rsa.encrypt(message, key)

def decrypt_key(ciphertext, key):
    try:
        return rsa.decrypt(ciphertext, key)
    except:
        return False

def sign(message, key):
    return rsa.sign(message, key, 'SHA-1')

def verify(message, signature, key):
    try:
        return rsa.verify(message, signature, key) == 'SHA-1'
    except:
        return False
```

The GUI implementation:

```
## Encryption function to encrypt password before we save it in our database.
def encryptPass ( pass1 ):
    newMessage=''
    for i in range(len(pass1)):
        if pass1[i] != ' ':
            n = ord(pass1[i]) + 2
            n = chr(n)
            newMessage = newMessage + n
        else:
            newMessage = newMessage + ' '
    return newMessage

## Function to take a username and password form user, encrypt password by using encryptPass
function and save it in database.
def register_user():
    print("User register")
    global username_info
    username_info = username.get()
    password_info = password.get()
    password_info=encryptPass(password_info)
    if Search_inDatabase(username_info.lower()):
        Label(screen1, text="This username is exist!", fg="red", font=("calibri", 9)).pack()
    else :
        file = open("DataBase.txt", "a")
        file.write(username_info.lower() + "\t")
        file.write(password_info+ "\n")
        generateKeys()
        file.close()
        Label(screen1, text=" Go back and Login!", fg="green", font=("calibri", 11)).pack()

    username_entry.delete(0, END)
    password_entry.delete(0, END)

## Search on database to check if user exist or no.
def Search_inDatabase(username):
    global Users
    with open('DataBase.txt') as f:
        if username in f.read():
            Users = open('DataBase.txt').read().split("\n")
            return True
        else:
            return False
```

```
## Function to check first if user exist or no, if exist it will check if the password is correct
or no, if correct the login is success and will open a home screen.
def login_verify():
    print("Verify Login")
    global username1
    global privateKey
    global publicKey
    username1 = username_verify.get().lower()
    password1 = password_verify.get()
    username_entry1.delete(0, END)
    password_entry1.delete(0, END)
    if Search_inDatabase(username1.lower()):
        password1 = username1 + "\t" + encryptPass(password1)
        if password1 in Users:
            privateKey, publicKey = loadKeys()
            SendOrCheck()
        else:
            label1= Label(screen2, text="Incorrect password",fg="red", font=("calibri", 9))
            label1.pack()
    else:
        label1=Label(screen2, text="User not Found",fg="red", font=("calibri", 9))
        label1.pack()

## Function register form if click on button register will run function register_user.
def register():
    print("register button")
    def go_back(): ##Back to main screen
        screen1.destroy()
    global screen1
    screen1 = Toplevel(screen)
    screen1.title("Register")
    screen1.geometry("450x450+600+250")
    global username
    global password
    global username_entry
    global password_entry
    username = StringVar()
    password = StringVar()
    Label(screen1, text="Please enter details below").pack()
    Label(screen1, text="").pack()
    Label(screen1, text="Username * ").pack()
    username_entry = Entry(screen1, textvariable=username)
    username_entry.pack()
    Label(screen1, text="Password * ").pack()
    password_entry = Entry(screen1, textvariable=password)
    password_entry.pack()
    Label(screen1, text="").pack()
    Button(screen1, text="Register", width=10, height=1, command=register_user).pack()
```

```
    Button(screen1, text="Back", height=1, width=10, command=go_back).pack()

## Function login form if click on button login will run function login_verify to check.
def login():
    print("login button")
    def go_back(): ##Back to main screen
        screen2.destroy()
    global screen2
    screen2 = Toplevel(screen)
    screen2.title("Login")
    screen2.geometry("450x450+600+250")
    Label(screen2, text="Please enter details below to login").pack()
    Label(screen2, text="").pack()
    global username_verify
    global password_verify
    username_verify = StringVar()
    password_verify = StringVar()
    global username_entry1
    global password_entry1
    Label(screen2, text="Username * ").pack()
    username_entry1 = Entry(screen2, textvariable=username_verify)
    username_entry1.pack()
    Label(screen2, text="").pack()
    Label(screen2, text="Password * ").pack()
    password_entry1 = Entry(screen2, textvariable=password_verify,show='*')
    password_entry1.pack()
    Label(screen2, text="").pack()
    Button(screen2, text="Login", width=10, height=1, command=login_verify).pack()
    Label(screen2, text="").pack()
    Button(screen2,text="Back", height=1, width=10, command=go_back).pack()
    Label(screen2, text="").pack()
    Button(screen2,text="Cancel", height=1, width=10, command=Stop).pack()

## First screen that have three option 1. register 2. login 3. cancel
def main_screen():
    global screen
    screen = Tk()
    screen.geometry("450x450+600+250")
    screen.title("Sharing Application")
    Label(text="").pack()
    Label(text="Welcome to Sharing!", width="300", height="2", font=("Calibri", 13)).pack()
    Label(text="").pack()
    Button(text="Login", height="2", width="30", command=login).pack()
    Label(text="").pack()
    Button(text="Register", height="2", width="30", command=register).pack()
    Label(text="").pack()
    Button(text="Cancel", height="2", width="30", command=Stop).pack()
    screen.mainloop()
```

```
## Stop run if user want close application
def Stop():
    sys.exit()

## If user wants to send, it type the username to receiver and message. Message will be encrypted
by AES.
def Home_Screen_Sending():
    def clear():
        my_text.delete(1.0, END)
    def Send_message(): ## Send and encrypt message
        global signature , keyEnc , filename
        message=username1+"\n"+my_text.get(1.0,END)
        Rece=receiver.get(1.0,END).replace('\n','').lower()
        file = open(Rece, "w")
        file.write(message)
        file.close()
        key = token_bytes(16)
        print("key: ", key)
        filename = Rece
        encrypt_file(key, filename)
        with open('keys/'+Rece+'public.pem', 'rb') as p:
            ReceiverpublicKey = rsa.PublicKey.load_pkcs1(p.read())
        keyEnc = encrypt_key(key, ReceiverpublicKey)
        print("Encrypted key: ", keyEnc)
        signature = sign(key, privateKey)
        Kfile=open("EncKeys/"+filename+'.key1',"wb")
        Kfile.write(keyEnc)
        Kfile.close()
        K2file=open("EncKeys/"+filename+'.key2',"wb")
        K2file.write(signature)
        K2file.close()
        my_text.delete(1.0, END)
        receiver.delete(1.0, END)
    def go_back(): ##Back to home screen
        screen6.destroy()
    global screen6
    screen6 = Tk()
    screen6.geometry("450x450+600+250")
    screen6.title("Sharing")
    Label(screen6, text="").pack()
    Label(screen6, text="to:").pack()
    receiver = Text(screen6, width=9, height="1")
    receiver.pack()
    my_text = Text(screen6, width=30,height= "7")
    my_text.pack(pady=10)
    button_frame = Frame(screen6)
    button_frame.pack()
    clear_button = Button(button_frame,text="Clear",command=clear)
```

```
    clear_button.grid(row=0,column=0)
    submit_button = Button(button_frame,text="Submit",command=Send_message)
    submit_button.grid(row=0,column=1)
    Button(screen6,text="Back", height=1, width=10, command=go_back).pack()
    Label(screen6,text='').pack(pady=20)

## After user login successfully will open a screen that contains three options 1. Send message
2. Inbox 3. logout
def SendOrCheck():
    def go_back(): ##Back to main screen
        screen7.destroy()
    global screen7
    screen7 = Tk()
    screen7.geometry("450x450+600+250")
    screen7.title("Sharing Application")
    Label(screen7, text="").pack()
    Label(screen7, text="Hi! " + username1, width="300", height="2", font=("Calibri", 13)).pack()
    Button(screen7, text="Send Message", height="2", width="30", command=Home_Screen_Sending).pack()
    Button(screen7, text="Inbox", height="2", width="30", command=Home_Screen_Check).pack()
    Label(screen7, text="").pack()
    Button(screen7, text="logOut", height=1, width=10, command=go_back).pack()
    screen.mainloop()

## Check if the user has a new message or no, if it has it will open, decrypt message and display
in screen.
def Home_Screen_Check():
    def go_back():
        screen8.destroy()
    global screen8
    screen8 = Tk()
    screen8.geometry("450x450+600+250")
    screen8.title("Inbox")
    Label(screen8,text="",height="3").pack()
    list_of_files = os.listdir()
    if username1+ '.encrypted' in list_of_files:
        keyEnc=open("EncKeys/"+username1+'.key1', 'rb').read()
        signature=open("EncKeys/"+username1+'.key2', 'rb').read()
        keyDec = decrypt_key(keyEnc, privateKey)
        os.remove("EncKeys/"+username1.lower()+'.key1')
        os.remove("EncKeys/"+username1.lower()+'.key2')
        if keyDec:
            print("Decrypted key: ", keyDec)
            decrypt_file(keyDec, username1+'.encrypted')
            print('File Decrypted')
            file1 = open(username1, "r")
            verify1 = file1.readlines()
            from_user = verify1[0].replace('\n','').lower()
```

```
            message = ""
            for lines in range(1,len(verify1)):
                message = message + verify1[lines]
            with open('keys/'+from_user+'public.pem', 'rb') as p:
                SenderPublicKey =rsa.PublicKey.load_pkcs1(p.read())
            Label(screen8, text="From: " + from_user, justify=LEFT, bg="ivory3").pack()
            Label(screen8, text="").pack()
            Label(screen8, text=message).pack()
            file1.close()
            os.remove(username1.lower())
        else:
            print('Unable to decrypt the message')
        if verify(keyDec, signature, SenderPublicKey):
            print('Successfully verified signature')
        else:
            print('The message signature could not be verified')
    else:
        Label(screen8, text="No new messages.").pack()
    Label(screen8,text='').pack(pady=20)
    Button(screen8,text="Back", height=1, width=10, command=go_back).pack()
    Label(screen8,text='').pack(pady=20)
```

For a clear look at the code check **this GitHub repository**.

EXECUTION

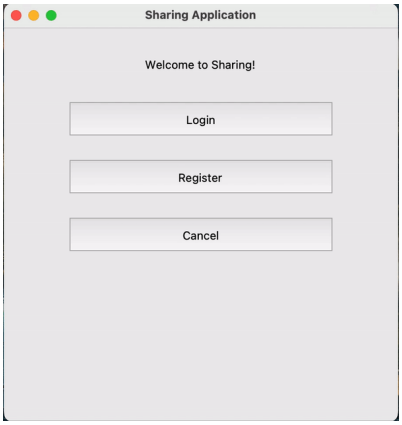

Figure 2. Register

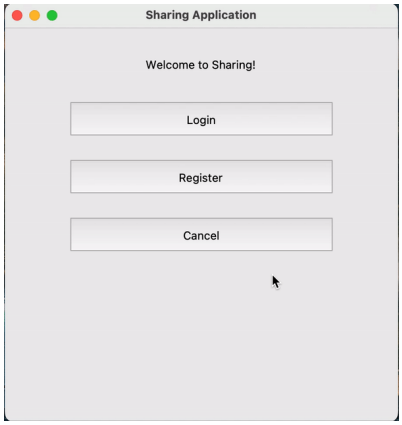

Figure 1. Register with existing user name

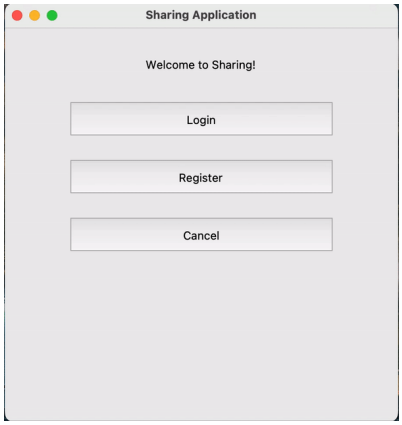

Figure 4. Log in

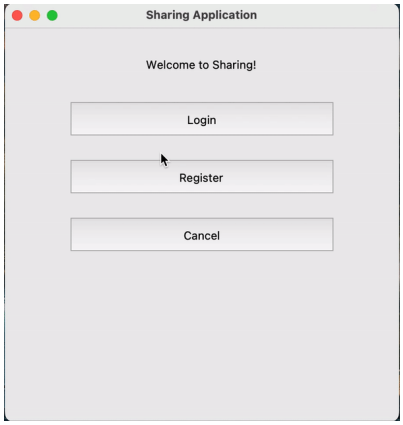

Figure 3. Log in with wrong password

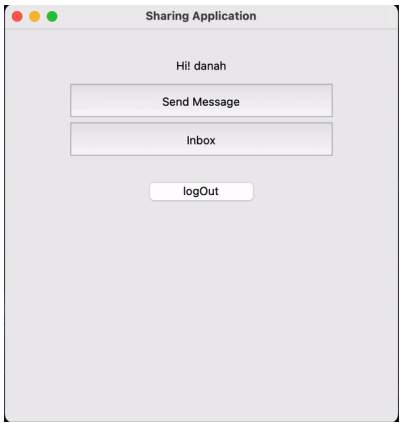

Figure 5. send message

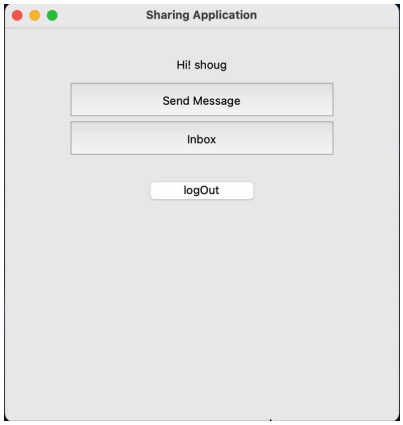

Figure 6. check the inbox

To see the execution open the file in **.docx** format.

CHALLENGES

1. Learning about python Crypto library and the features it has.
2. Solving the issues encountered during RSA installation.
3. Encrypting/decrypting files using AES.
4. Encrypting the symmetric key used in AES with RSA then attaches it to the message.
5. Learning to use and understand the Tkinter toolkit for the GUI.
6. Connecting the GUI message to encrypt using AES, in other words connecting the interface to the actual implementation.
7. Checking whether the user is an existing user or a new user.
8. Extracting the users' message and encrypting it.

REFERENCE

"AES File Encryption in Python," *AES file encryption in Python*. [Online]. Available: https://jonlabelle.com/snippets/view/python/aes-file-encryption-in-python. [Accessed: 02-Apr-2022].

D. Masika, "Implementing RSA encryption and decryption in Python," *Section*, 28-Jan-2022. [Online]. Available: https://www.section.io/engineering-education/rsa-encryption-and-decryption-in-python/. [Accessed: 04-Apr-2022].

J.Godinho, "How to Create a Graphical Register and Login System in Python Using Tkinter Part 2." *Www.youtube.com*, 27 Sept. 2018, www.youtube.com/watch?v=Z-deSpgtIG0. Accessed 29 Mar. 2022.