



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
<b>Manar.M Alqabbani</b>	
<b>Dana.Dh Alyami</b>	
<b>Leen.S Alaboudi</b>	
<b>Nehal.S Almouies</b>	

**Instruction: Dr. Tahani Al-Balawi**



## Contents

1.Introduction .....	3
2.Overview of the project design .....	3
3. Approach and steps to implementation. ....	4
4.Code .....	9
5.Snapshots of the application.....	14
6. Challenges .....	18
7. References .....	18



## 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:

```

141     #create new user
142     def add_newclient(self):
143
144         self.random_num = random.randint(0,100) #salt number
145
146         #add salt with user input for password
147         self.password = str(self.user_pass.get()+str(self.random_num))
148
149         #hash the password
150         self.hash = hashlib.md5(self.password.encode())
151         self.hashed = self.hash.hexdigest()
152
153         # data format in json
154         self.data_format = [ { "password": self.hashed, "salt": self.random_num } ]
155
156         with open('log_info.json','r+') as file:
157             data = json.load(file) #load json python
158             data[self.user_name.get()]=self.data_format#insert the data into json file
159             file.seek(0)
160             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:

```

161
162     #create private key and public key
163     (self.user_public_key,self.user_private_key) = rsa.newkeys(1025)
164
165     #store private key
166     self.privatekey = open(f"./PrivateKey/{self.user_name.get()}PrivateKey.key", 'wb')
167     self.privatekey.write(self.user_private_key.save_pkcs1('PEM'))
168     self.privatekey.close()
169
170     #store public key
171     self.public_key = open(f"./PublicKey/{self.user_name.get()}PublicKey.key", 'wb')
172     self.public_key.write(self.user_public_key.save_pkcs1('PEM'))
173     self.public_key.close()
174
175     self.destroy()
176     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 try again')
    except:
        self.pass_label.config(text='Incorrect Email or Password,please try 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
encrypt_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(encrypt_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 compare 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 > ...
1 from atexit import register
2 from cProfile import label
3 from cgitb import text
4 import email
5 from ipaddress import ip_address
6 from logging import root
7 from tkinter import
8 from tkinter import ttk
9 from tkinter import font
10 from tkinter import filedialog
11 from turtle import bgcolor, left, right, width
12 import tkinter as tk
13 import json
14 from tkinter import messagebox
15 from nbformat import write
16 import hashlib
17 import random
18 import rsa
19 import socket
20 import os
21 import time
22 from cryptography.fernet import Fernet
23 from PIL import ImageTk, Image
24 from connection import port_num, IP_add
25
26
27
28 class login(Tk):
29     def __init__(self):
30         super().__init__() #inherent from Tk class
31
32         self.geometry("900x700") #size of the page
33         self.title('login')
34         self.config(bg='white')
35         self.resizable(False, False) # ability to change page size
36         titlee = Label(self, text="XSecure", fg="#EF7960", bg='white', font=('Arial', 22, 'bold'), pady=20)
37         titlee.place(x=210, y=50)
38         #for the photo
39         load = Image.open("./photo/logo.png")
40
41         #for the photo
42         load = Image.open("./photo/logo.png")
43         render = ImageTk.PhotoImage(load)
44         img = Label(self, image=render, bg='white')
45         img.image = render
46         img.place(x=50, y=130)
47
48         main_frame = Frame(self, bg='#6698BC')
49         main_frame.place(x=50, y=180, width=400, height=800)
50         user_txt = Label(main_frame, text='Username:', fg='white', bg='#6698BC', font=('Courier', 13), pady=20).place(x=50, y=180)
51         load2 = Image.open("./photo/user2.png")
52         render2 = ImageTk.PhotoImage(load2)
53         img2 = Label(main_frame, image=render2, bg='#6698BC')
54         img2.image = render2
55         img2.place(x=50, y=150)
56         self.user_name = Entry(main_frame, font=('Courier', 14)) #input from the user (user name)
57         self.user_name.place(x=120, y=170, width=200, height=30)
58
59         pass_txt = Label(main_frame, text='Password:', fg='white', bg='#6698BC', font=('Courier', 13), pady=20).place(x=50, y=250)
60         load3 = Image.open("./photo/pass.png")
61         render3 = ImageTk.PhotoImage(load3)
62         img3 = Label(main_frame, image=render3, bg='#6698BC')
63         img3.image = render3
64         img3.place(x=50, y=300)
65         self.user_pass = Entry(main_frame, font=('Courier', 14)) #input from user (password)
66         self.user_pass.place(x=120, y=325, width=200, height=30)
67
68         #action if button is pressed
69         signup = Button(main_frame, text='Sign up', bg='#086600', bd=0, font=('Courier', 15), command=self.signup).place(x=170, y=470, width=100, height=40)
70         login_button = Button(main_frame, text='Login', bg='#086600', bd=0, font=('Courier', 15), command=self.login).place(x=180, y=400, width=100, height=40)
71         self.pass_label = Label(main_frame, text='new user?', fg='white', bg='#6698BC', font=('Courier', 13), pady=20).place(x=50, y=460)
72         self.pass_label.pack(pady=10)
73         self.pass_label.place(x=180, y=370)
74
75         def signup(self):
76             self.destroy()
77             signup()
78
79         def login(self):
80             try:
81                 record = json.load(open("log_info.json")) #open json file and read the data
82                 passwords = [data["password"] for data in record[str(self.user_name.get())]] #get password for the user
83                 salts = [data["salt"] for data in record[str(self.user_name.get())]] #get salts for the user
84                 self.currpass=self.user_pass.get()+str(salts[0]) #add the salts stored in json for the user and add it to input pass
85                 hash = hashlib.md5(self.currpass.encode()) #hash the password after adding the salt
86                 hashed = hash.hexdigest()
87                 if (hashed) in passwords:
88                     self.destroy()
89                     home_page()
90                 else:
91                     self.pass_label.config(text='Incorrect Email or Password, please try again')
92             except:
93                 self.pass_label.config(text='Incorrect Email or Password, please try again')
```



```
99
100 class signup(Tk):
101     def __init__(self):
102         super().__init__()
103
104         self.geometry("900x700")
105         self.title('signup')
106         self.config(bg='white')
107         self.resizable(False,False)
108         titlee= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
109         self.load = Image.open("./photo/logo.png")
110         self.render = ImageTk.PhotoImage(self.load)
111         self.img = Label(self, image=self.render,bg='white')
112         self.img.image = self.render
113         self.img.place(x=50, y=130)
114
115
116
117         main_frame = Frame(self,bg='#6698BC')
118         main_frame.place(x=500,width=400,height=800)
119         #user name
120         user_txt = Label(main_frame, text='Username:', fg='white',bg='#6698BC',font=('Courier',13),pady=20).place(x=50,y=100)
121         load2 = Image.open("./photo/user2.png")
122         render2 = ImageTk.PhotoImage(load2)
123         img2 = Label(main_frame, image=render2,bg='#6698BC')
124         img2.image = render2
125         img2.place(x=50, y=150)
126         self.user_name = Entry(main_frame,font=('Courier',14))
127         self.user_name.place(x=120,y=170,width=200,height=30)
128
129         #password
130         pass_txt= Label(main_frame, text='Password:', fg='white',bg='#6698BC',font=('Courier',13),pady=20).place(x=50,y=250)
131         load3 = Image.open("./photo/pass.png")
132         render3 = ImageTk.PhotoImage(load3)
133         img3 = Label(main_frame, image=render3,bg='#6698BC')
134         img3.image = render3
135         img3.place(x=50, y=300)
136         self.user_pass = Entry(main_frame,font=('Courier',14))
137         self.user_pass .place(x=120,y=325,width=200,height=30)
138
139
140         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)
141
142         #create new user
143         def add_newclient(self):
144
145             self.random_num = random.randint(0,100) #salt number
146
147             #add salt with user input for password
148             self.password = str(self.user_pass.get()+str(self.random_num))
149
150             #hash the password
151             self.hash = hashlib.md5(self.password.encode())
152             self.hashd = self.hash.hexdigest()
153
154             # data formate in json
155             self.data_formate = [ { "password": self.hashd, "salt": self.random_num } ]
156
157             with open('log_info.json','r+') as file:
158                 data = json.load(file) #load json python
159                 data[self.user_name.get()]=self.data_formate#insert the data into json file
160                 file.seek(0)
161                 json.dump(data, file, indent = 4)
162
163             #create private key and public key
164             (self.user_public_key,self.user_private_key) = rsa.newkeys(1025)
165
166
167             #store private key
168             self.privatekey = open(f"./PrivateKey/{self.user_name.get()}PrivateKey.key",'wb')
169             self.privatekey.write(self.user_private_key.save_pkcs1('PEM'))
170             self.privatekey.close()
171
172             #store public key
173             self.public_key = open(f"./PublicKey/{self.user_name.get()}PublicKey.key",'wb')
174             self.public_key.write(self.user_public_key.save_pkcs1('PEM'))
175             self.public_key.close()
176
177             self.destroy()
178             login()
179
180
```



```
180
181
182 class home_page(Tk):
183     def __init__(self):
184
185         super().__init__()
186         self.geometry("900x700")
187         self.title('Home')
188         self.config(bg='white')
189         self.resizable(False,False)
190         title= Label(self,text="XSecure",fg="#EF7960",bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
191         self.load = Image.open("./photo/logo.png")
192         self.render = ImageTk.PhotoImage(self.load)
193         self.img = Label(self, image=self.render,bg='white')
194         self.img.image = self.render
195         self.img.place(x=50, y=130)
196
197         main_frame = Frame(self,bg='#6698BC')
198         main_frame.place(x=500,width=400,height=800)
199
200         option_txt = Label(main_frame, text='Please select option:', fg='white',bg='#6698BC',font=('Courier',15,'bold'),pady=20).place(x=90,y=100)
201         send_button = 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)
202         rec_button = 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)
203
204     def send(self):
205         self.destroy()
206         send()
207
208     def recieve(self):
209         self.destroy()
210         receiv()
211
212 class send(Tk):
213     def __init__(self):
214
215         super().__init__()
216         self.geometry("900x700")
217         self.title('Send')
218         self.config(bg='white')
219         self.resizable(False,False)
220         title= Label(self,text="XSecure",fg="#EF7960",bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
221         self.load = Image.open("./photo/logo.png")
222         self.render = ImageTk.PhotoImage(self.load)
223         self.img = Label(self, image=self.render,bg='white')
224         self.img.image = self.render
225         self.img.place(x=50, y=130)
226
227         main_frame = Frame(self,bg='#6698BC')
228         main_frame.place(x=500,width=400,height=800)
229
230         upload_txt = Label(main_frame, text='Upload File.', fg='white',bg='#6698BC',font=('Courier',25,'bold'),pady=0).place(x=100,y=50)
231         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,height=40)
232         upload_key = Label(main_frame, text='Public Key.', fg='white',bg='#6698BC',font=('Courier',25,'bold'),pady=0).place(x=100,y=230)
233         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=40)
234
235         self.file_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
236         self.file_lbl.pack(pady=110)
237         self.file_lbl.place(x=110,y=170)
238
239         self.key_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
240         self.key_lbl.pack(pady=110)
241         self.key_lbl.place(x=110,y=340)
242
243         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)
```



```
255 #to open files and choose
256 def file_explorer_forfile(self):
257     self.file_path = filedialog.askopenfilename(filetypes=[('All types','*.*)']) #file path
258     self.file_lbl.config(text=self.file_path)
259
260 #to open keyss and choose
261 def file_explorer_forkey(self):
262     self.key_path = filedialog.askopenfilename(initialdir='./PublicKey/',filetypes=[('All types','*.*)']) #key path
263     self.key_lbl.config(text=self.key_path)
264
265 def send_data(self):
266
267     # symmetric key
268     AES_key = Fernet.generate_key()
269     cipher = Fernet(AES_key)
270
271     # open the file we want to encrypt
272     openfile = open(self.file_path,'rb')
273     filedata = openfile.read() # read file content
274
275     # encrypt the content
276     encrypt_content = cipher.encrypt(filedata)
277
278     #temp file for encrypted content
279     dataenc = open('encrypted.'+str(os.path.splitext(self.file_path)[1][1:].strip()),'wb')
280     dataenc.write(encrypt_content)
281
282     #create socket to send the file
283     mysocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
284     mysocket.bind((IP_add, port_num))
285     mysocket.listen(1)
286     receiver, addr = mysocket.accept()
287
288     # load user public key
289     user_public_key = open(self.key_path,'rb')
290     public_key_content = user_public_key.read()
291     public_key = rsa.PublicKey.load_pkcs1(public_key_content)
292
293     # encrypt symmetric with user public key
294     RSA_encrypting = rsa.encrypt(AES_key,public_key)
295     receiver.send(RSA_encrypting)
296
297     #file information
298     file_name = 'encrypted.'+str(os.path.splitext(self.file_path)[1][1:].strip())
299     file_size = os.path.getsize(file_name)
300
301     # Send file name and its size
302     receiver.send(file_name.encode())
303     receiver.send(str(file_size).encode())
304
305     # open encrypted file
306     with open(file_name, "rb") as file:
307         counter = 0
308         # send encrypted content to reciever
309         while counter <= file_size:
310             enc_data = file.read(1024)
311             if not (enc_data):
312                 break
313             receiver.sendall(enc_data)
314             counter += len(enc_data)
315
316 mysocket .close()
```

```
324
325 class receiv(Tk):
326     def __init__(self):
327
328         super().__init__()
329         self.geometry("900x700")
330         self.title('Recieve')
331         self.config(bg='white')
332         self.resizable(False,False)
333         title= Label(self,text="XSecure",fg='#EF7960',bg='white',font=('Arial',22,'bold'),pady=20).place(x=210,y=50)
334         self.load = Image.open("./photo/logo.png")
335         self.render = ImageTk.PhotoImage(self.load)
336         self.img = Label(self, image=self.render,bg='white')
337         self.img.image = self.render
338         self.img.place(x=50, y=130)
339
340
341         main_frame = Frame(self,bg='#6698BC')
342         main_frame.place(x=500,width=400,height=800)
343
344         select_txt = Label(main_frame, text='Select Folder.', fg='white',bg='#6698BC',font=('Courier',25,'bold'),pady=0).place(x=100,y=50)
345         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=40)
346         select_key = Label(main_frame, text='Private Key.', fg='white',bg='#6698BC',font=('Courier',25,'bold'),pady=0).place(x=100,y=230)
347         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=40)
348
349
350         self.folder_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
351         self.folder_lbl.pack(pady=110)
352         self.folder_lbl.place(x=110,y=170)
353
354         self.key_lbl = Label(main_frame, fg='#D86600',bg='white',relief=RAISED)
355         self.key_lbl.pack(pady=110)
356         self.key_lbl.place(x=110,y=340)
357
358         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)
359
```

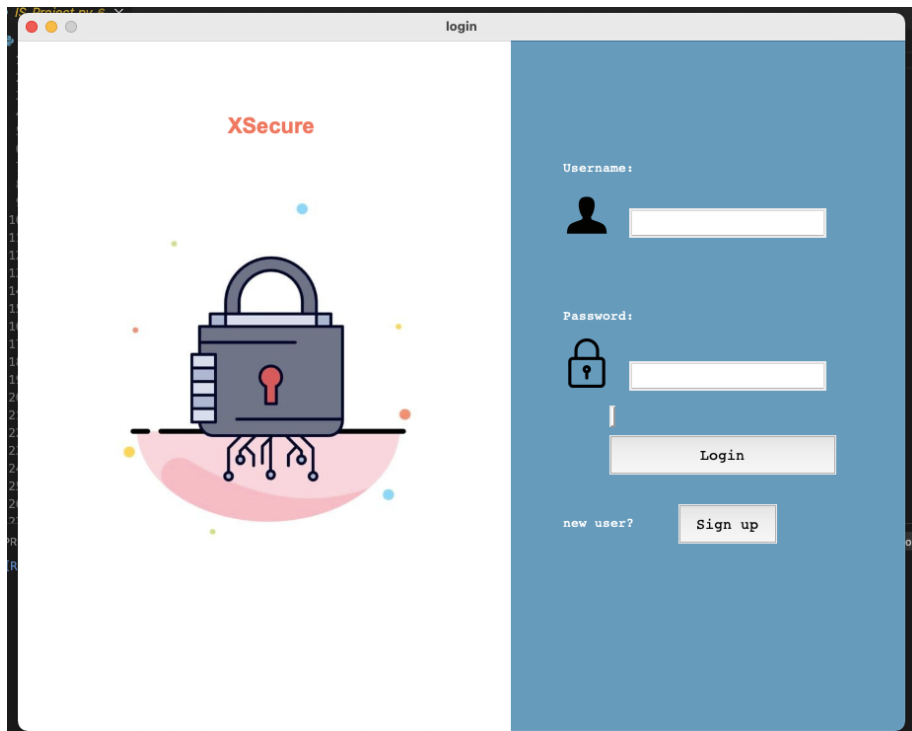


```
362
363     #select folder
364 def select_folder(self):
365     self.folder = filedialog.askdirectory() #folder path
366     self.folder_lbl.config(text=self.folder)
367     #select key
368 def file_explorer_forkey(self):
369     self.key_path = filedialog.askopenfilename(initialdir='./PrivateKey/',filetypes=[('All types','*.*')]) #key path
370     self.key_lbl.config(text=self.key_path)
371
372 def recieve_data(self):
373
374     connection_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
375     try:
376         connection_sock.connect((IP_add, port_num))
377     except:
378         print("connection failed.")
379         exit(0)
380
381     # recieve AES key
382     AES_key = connection_sock.recv(1024)
383
384     # recieve file information
385     file_name = connection_sock.recv(100).decode()
386     file_size = connection_sock.recv(100).decode()
387
388     # read file
389     with open(str(self.folder)+"-"+str(os.path.basename(file_name)), "wb") as file:
390         counter = 0
391         # recieve file content
392         while counter <= int(file_size):
393             data = connection_sock.recv(1024)
394             if not (data):
395                 break
396             file.write(data)
397             counter += len(data)
```

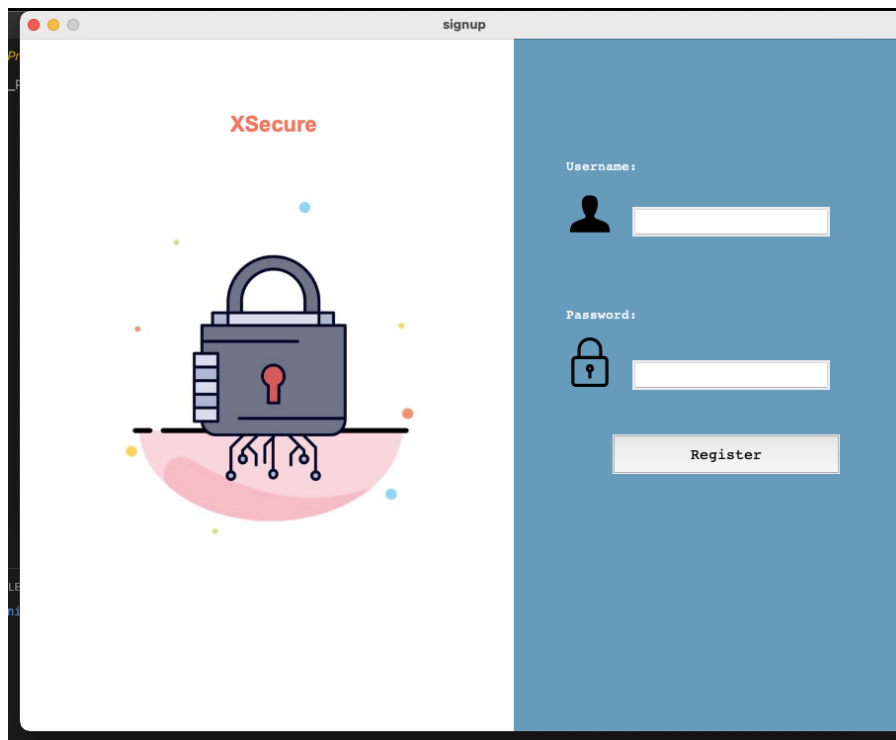
```
399
400     # Closing the socket.
401     connection_sock.close()
402
403
404     # rsa private key
405     user_privatekey_file = open(self.key_path,'rb')
406     privatekey = user_privatekey_file.read()
407     user_RSA_key = rsa.PrivateKey.load_pkcs1(privatekey)
408
409     #decrypt aes key
410     AES_key = rsa.decrypt(AES_key,user_RSA_key)
411     cipher = Fernet(AES_key)
412
413     #open file
414     encrypted_data = open(str(self.folder)+"-"+str(os.path.basename(file_name)),'rb')
415     encdata = encrypted_data.read()
416
417     #decrypt content
418     mycontent = cipher.decrypt(encdata)
419
420     #write decrypted content
421     file = open(self.folder+'decrypted.'+str(os.path.splitext(file_name)[1][1:].strip().lower()),'wb')
422     file.write(mycontent)
423     file.close()
424
425     #delete temp file
426     os.remove('encrypted.'+str(os.path.splitext(file_name)[1][1:].strip().lower()))
427
428
429 if __name__ == "__main__":
430     obj = login()
431     obj.mainloop()
```

## 5. Snapshots of the application

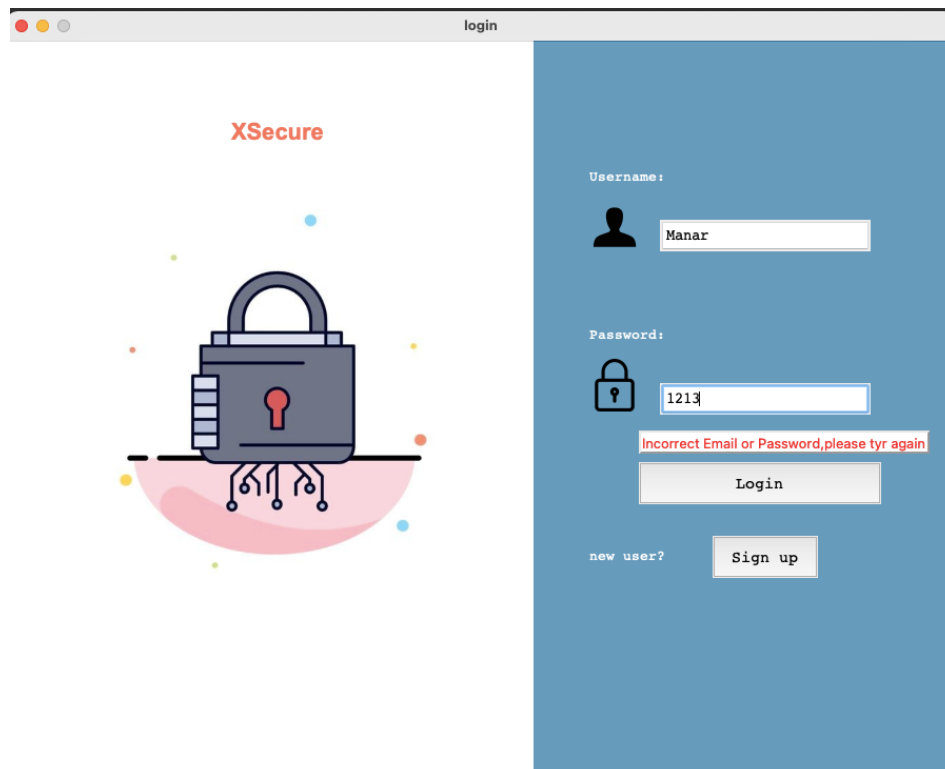
- Login page



- Signup page

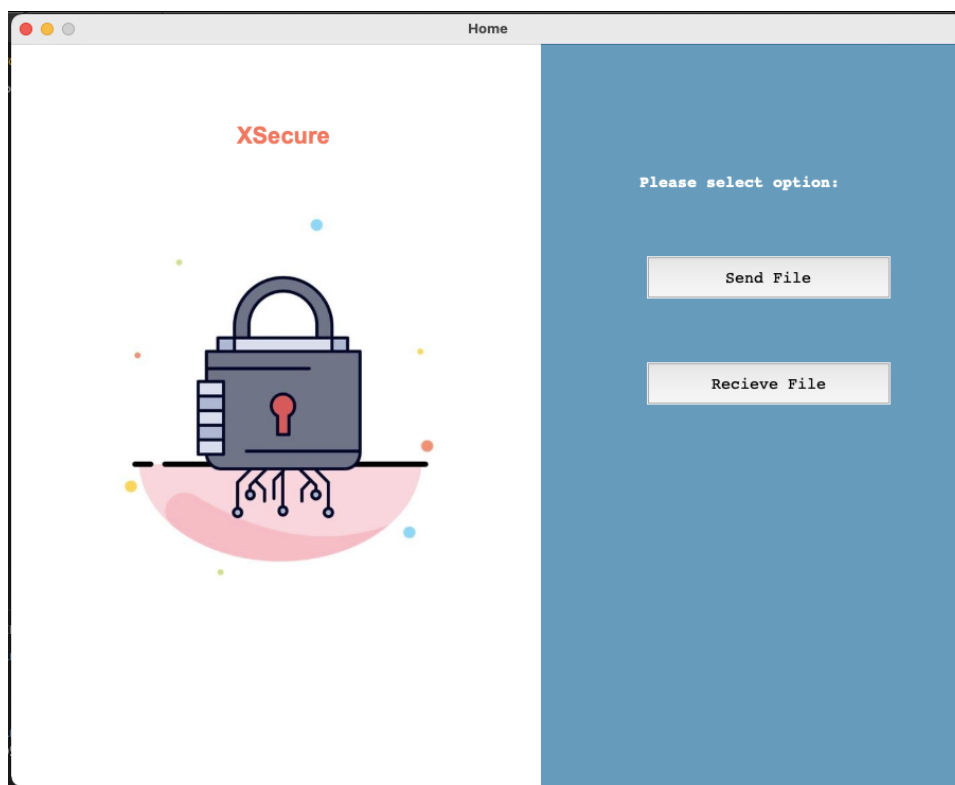


- in case of incorrect password



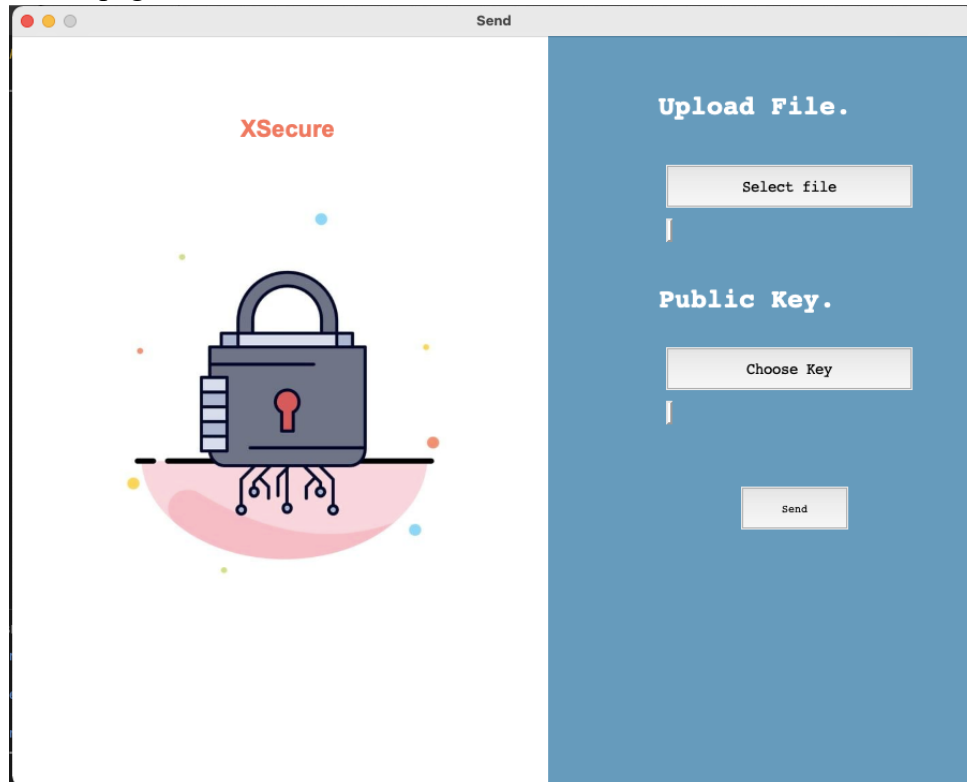
The screenshot shows a web browser window titled "login". The page has a blue background. On the left, there is a logo for "XSecure" featuring a padlock with circuit lines. On the right, there is a login form. The "Username:" field contains "Manar". The "Password:" field contains "1213". Below the password field, there is a red error message: "Incorrect Email or Password, please try again". Below the error message, there is a "Login" button. At the bottom left of the form, there is a link "new user?" and a "Sign up" button.

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



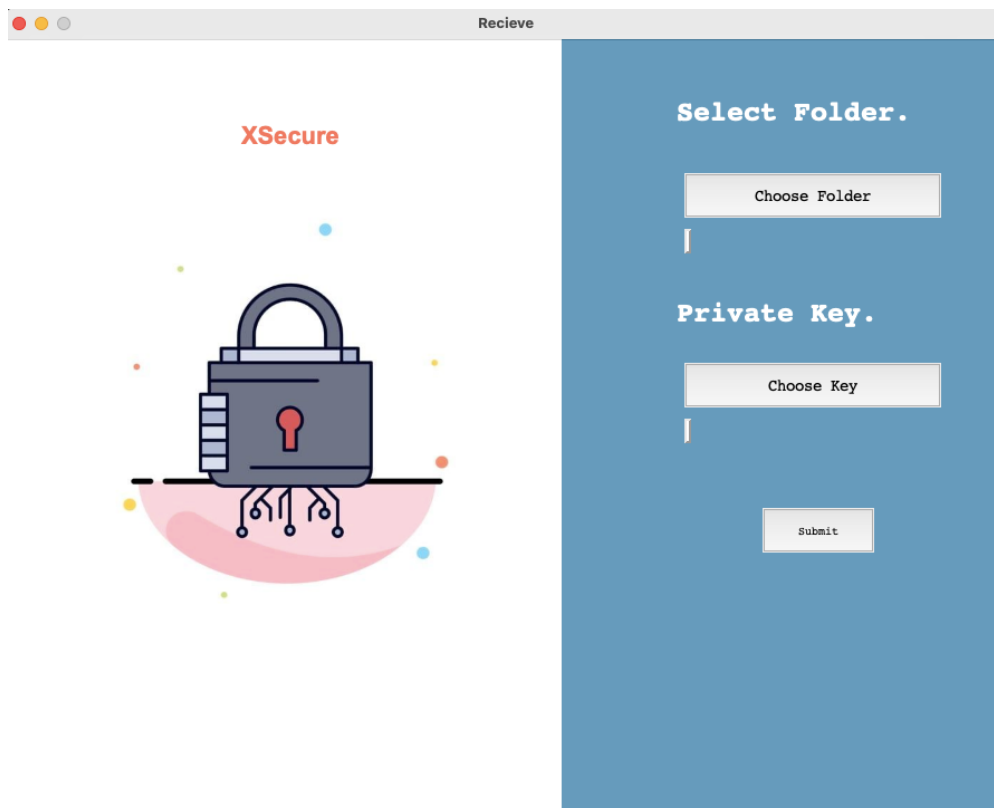
The screenshot shows a web browser window titled "Home". The page has a blue background. On the left, there is a logo for "XSecure" featuring a padlock with circuit lines. On the right, there is a form with the text "Please select option:". Below this text, there are two buttons: "Send File" and "Recieve File".

- Sender page



The Sender page is a web application window titled "Send". It features a light blue header bar with the title. The main content area is split into two panels. The left panel has a white background and displays the "XSecure" logo in red text above a stylized illustration of a padlock with circuit lines extending from its base, set against a pink semi-circle. The right panel has a blue background and contains three sections: "Upload File." with a "Select file" button, "Public Key." with a "Choose Key" button, and a "Send" button at the bottom.

- Receiver page



The Receiver page is a web application window titled "Recieve". It features a light blue header bar with the title. The main content area is split into two panels. The left panel has a white background and displays the "XSecure" logo in red text above a stylized illustration of a padlock with circuit lines extending from its base, set against a pink semi-circle. The right panel has a blue background and contains three sections: "Select Folder." with a "Choose Folder" button, "Private Key." with a "Choose Key" button, and a "Submit" button at the bottom.





- Example of public key file

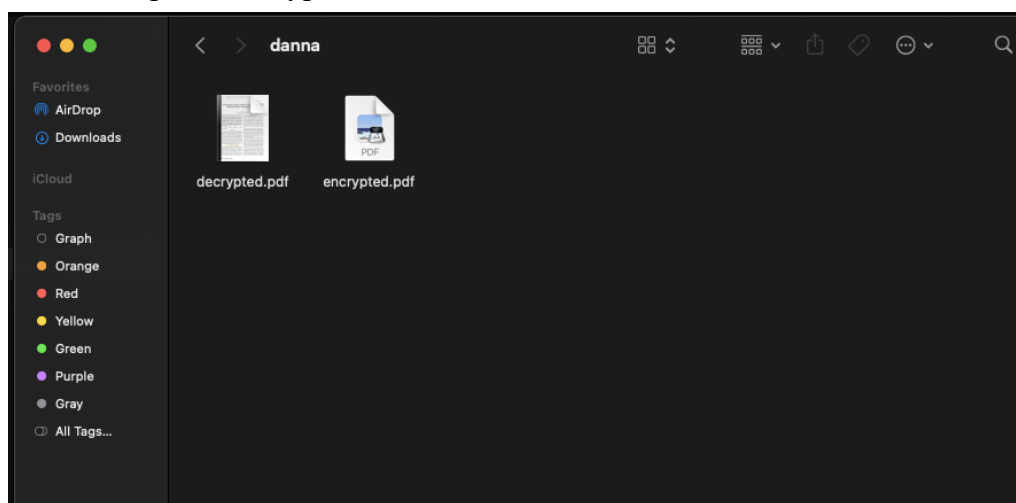
```
ManarPublicKey.key — IS Project

ManarPublicKey.key x
PublicKey > ManarPublicKey.key
1 -----BEGIN RSA PUBLIC KEY-----
2 MIGJAoGBAJIHLw/NadST6/2ufcPnqQBUNpbE2tn5NrWMTp3ZD1THMGnAHNlkZv
3 Qfd/8MnY5laplTeGDN/3bUBjDn1NNHT80F7+vIL0YPr4dtJHFm8Jjz1lQmF0+lk+
4 +YyINym6KSbU7VqG47KM7WkD0IO6reUFsX+oky3EFbnQRU9i49ybAgMBAE=
5 -----END RSA PUBLIC KEY-----
6
```

- Example of private key file

```
ManarPrivateKey.key x
PrivateKey > ManarPrivateKey.key
1 -----BEGIN RSA PRIVATE KEY-----
2 MIICYAIBAAKBgQCSBzC8PzWnUk+v9rn3D56kAVDawXNrZ+Ta1jLT92Q9UxzBhpwB
3 zZZGb0H3f/DJ20ZwqZU3hgzf921AYw59TTR0/Dhe/ryC9GD6+HbSRxZvCY89ZUJh
4 TvpZPvmMpTWJuiKm101ahu0yj01pAziDuq3lBbF/qJMTx8W50EVPYUpcmwIDAQAB
5 AoGBAIR8rwUtQHR+G/yGX9/oDP1ah0MdVnFvFGK5Zu1a1RyNQ5xq64fHzpVBhc71
6 BhCIEkw/4ldTjm0JU4HqBYWt8UrpPh2qrdBdWgzDX4SmjHvzlCs/bkdaITjw+pmd
7 D8sBK9wb/b4jRSS+q8KdGYu6jNFaw71I3egW5mpVEYisBIBhAkUAvwKSh/49Qmdn
8 VELpA/KVwDNJ0dAg6c7sUg4rjeBPFbda1xKNTkaD5LWhkKsyZRXt0sncwP9eEVRr
9 8HGKbCbMAKjC8EsCPQDDtpUoMH2ZGfcRgo+xreH30Y0Ac054cym4Aot503nfJ00a
10 NsfNkZaCnNwC5nu9MteWr7ijBsdMecSzMvECRE/vIacrDM31rQadC0kvap7S++X
11 u85cu+Ty7SModbAe1TMgHmZdeCCkKgLCwgovj0GHTIyv0Yvq/4dnTUJJIiMfdFv
12 AjxNkbUme6mch81r9DptqsRQC47+KMRrmwFi/Cc5vtaj2aQYADRKxrRHoRc1G6jZ
13 gSgvgjAE9W/SYMTUXZECRE4hhzeRkKm/0qQpnenT/nUP8xnS6cT0/p4K1zShU2wI
14 EUMX3cSmGwcx63ZKFll13t+NrwN8/86p/LEIiq93addA3GSf
15 -----END RSA PRIVATE KEY-----
16
```

- 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/encrypt\_aes: implementation of RSA and (Advanced Encryption Standard) AES*. [online] Available at: <[https://github.com/mukeshkdangi/encrypt\\_aes](https://github.com/mukeshkdangi/encrypt_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].