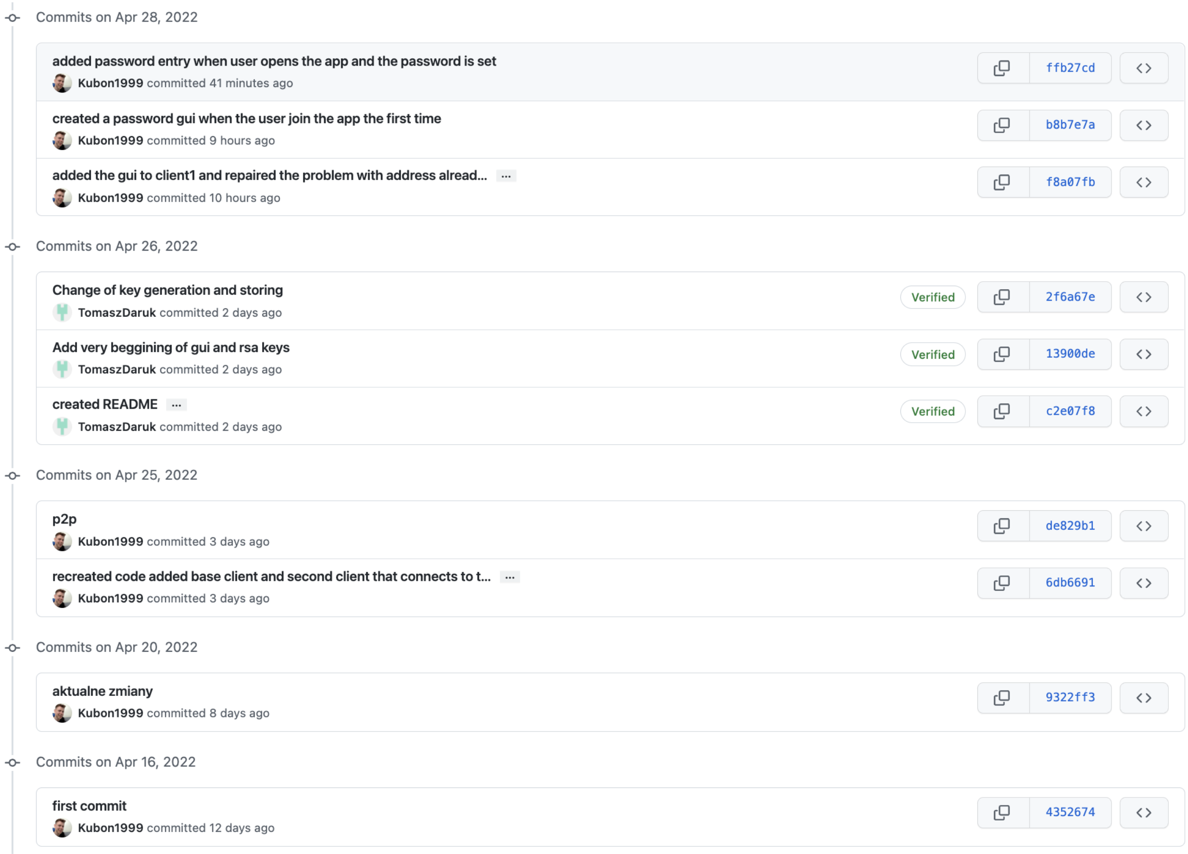
Security of Computer Systems

Project Report

Authors:
Jakub, Wołodźko, 175634
Tomasz, Daruk, 171974

Version: 1.0

Versions



Commits on Apr 28, 2022

added password entry when user opens the app and the password is set
Kubon1999 committed 41 minutes ago — ffb27cd

created a password gui when the user join the app the first time
Kubon1999 committed 9 hours ago — b8b7e7a

added the gui to client1 and repaired the problem with address alread… …
Kubon1999 committed 10 hours ago — f8a07fb

Commits on Apr 26, 2022

Change of key generation and storing
TomaszDaruk committed 2 days ago — Verified — 2f6a67e

Add very beggining of gui and rsa keys
TomaszDaruk committed 2 days ago — Verified — 13900de

created README …
TomaszDaruk committed 2 days ago — Verified — c2e07f8

Commits on Apr 25, 2022

p2p
Kubon1999 committed 3 days ago — de829b1

recreated code added base client and second client that connects to t… …
Kubon1999 committed 3 days ago — 6db6691

Commits on Apr 20, 2022

aktualne zmiany
Kubon1999 committed 8 days ago — 9322ff3

Commits on Apr 16, 2022

first commit
Kubon1999 committed 12 days ago — 4352674

1. Project – control term

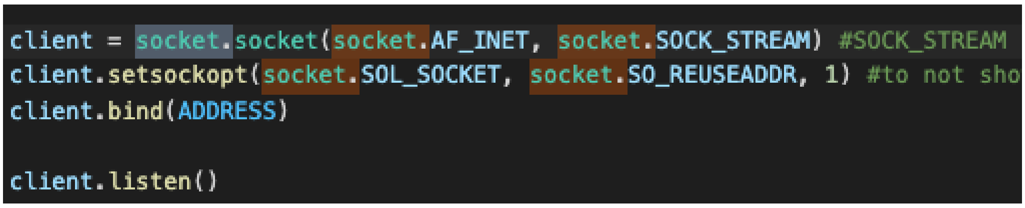
1.1 Description

The project in control term already has a lot of features. The core feature is to send messages in a p2p connection – which works excellent. The connection is transferred using a socket, it is a TCP connection on the grounds that we use *socket.SOCK_STREAM* argument in the sockets library properties. Connection should be secure for that we have a public and private key on each site of the client application. The user on his first start of the program is asked to set a password which hash is used to encrypt the private and public keys. The program in the next start will ask for the password entered earlier. If the password is entered correctly the program should encrypt the messages successfully otherwise the program should return fake data. This feature will be done until the final term. We have practiced the encryption library, now in the next stage we plan to implement it to the application.

1.2 Results

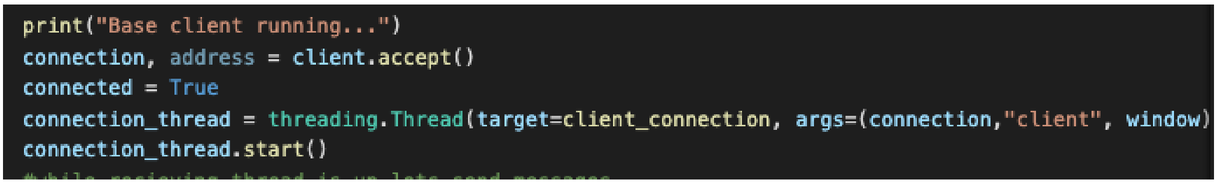
CODE:

Here we set the client socket properties:



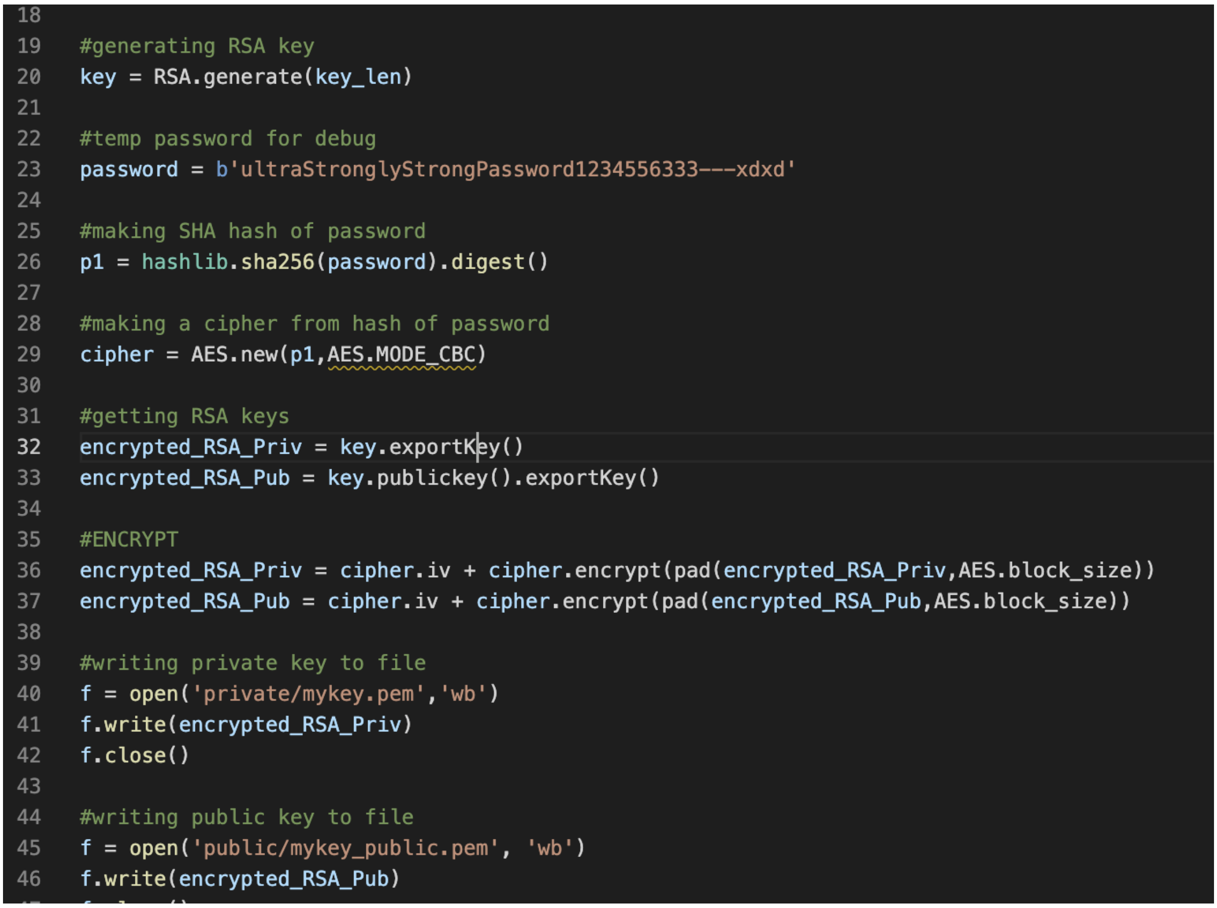
```
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #SOCK_STREAM
client.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) #to not sho
client.bind(ADDRESS)

client.listen()
```

We run a thread with a function that listens to the incoming messages while allowing the user to send messages from the client application.



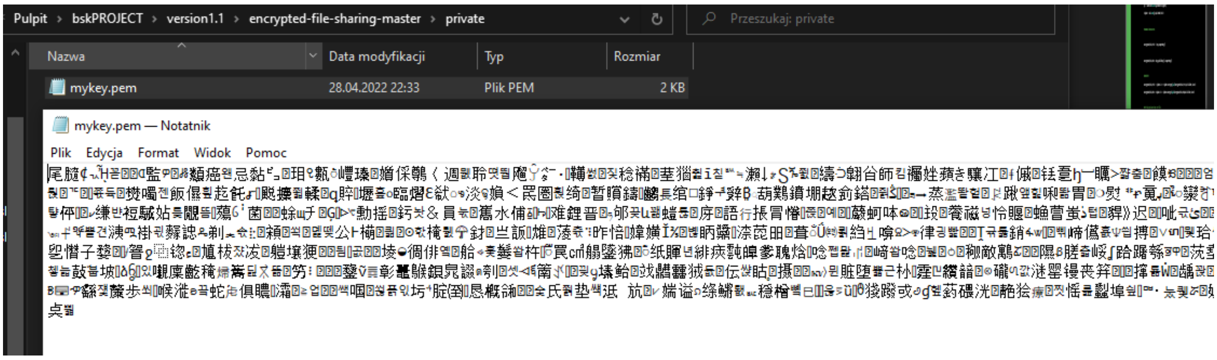
```
print("Base client running...")
connection, address = client.accept()
connected = True
connection_thread = threading.Thread(target=client_connection, args=(connection,"client", window))
connection_thread.start()
```

This is how our encoding code looks like now, it is yet not implemented into the final product, but we have practiced the encryption library.



```

#generating RSA key
key = RSA.generate(key_len)

#temp password for debug
password = b'ultraStronglyStrongPassword1234556333---xdxd'

#making SHA hash of password
p1 = hashlib.sha256(password).digest()

#making a cipher from hash of password
cipher = AES.new(p1,AES.MODE_CBC)

#getting RSA keys
encrypted_RSA_Priv = key.exportKey()
encrypted_RSA_Pub = key.publickey().exportKey()

#ENCRYPT
encrypted_RSA_Priv = cipher.iv + cipher.encrypt(pad(encrypted_RSA_Priv,AES.block_size))
encrypted_RSA_Pub = cipher.iv + cipher.encrypt(pad(encrypted_RSA_Pub,AES.block_size))

#writing private key to file
f = open('private/mykey.pem','wb')
f.write(encrypted_RSA_Priv)
f.close()

#writing public key to file
f = open('public/mykey_public.pem', 'wb')
f.write(encrypted_RSA_Pub)
```

This is how the private key encrypted with AES looks like:



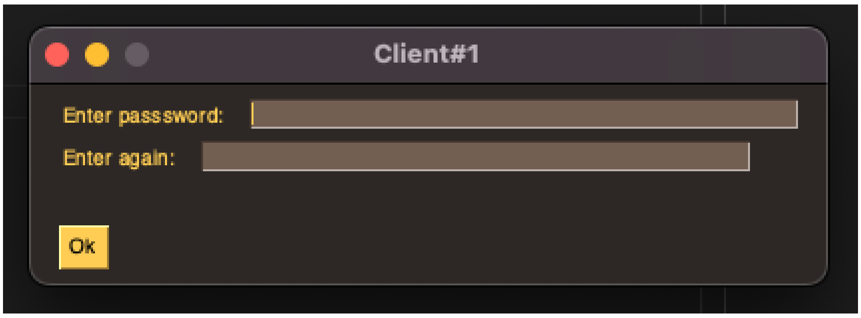
Pulpit › bskPROJECT › version1.1 › encrypted-file-sharing-master › private
Przeszukaj: private

Nazwa	Data modyfikacji	Typ	Rozmiar
mykey.pem	28.04.2022 22:33	Plik PEM	2 KB

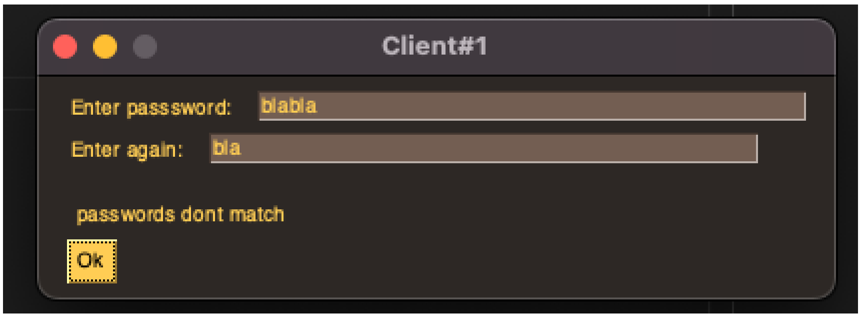
mykey.pem — Notatnik
Plik Edycja Format Widok Pomoc

GUI:

Here we see the application asking to set the password:



Passwords do not match:

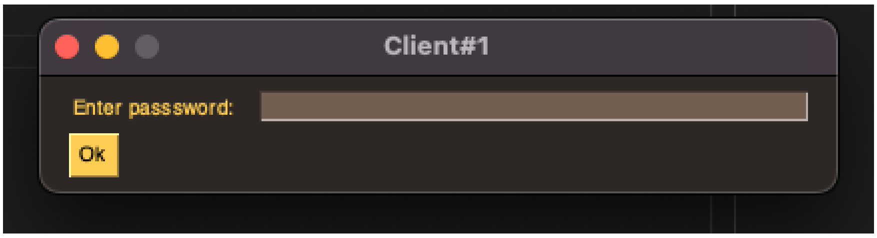


Enter password: blabla

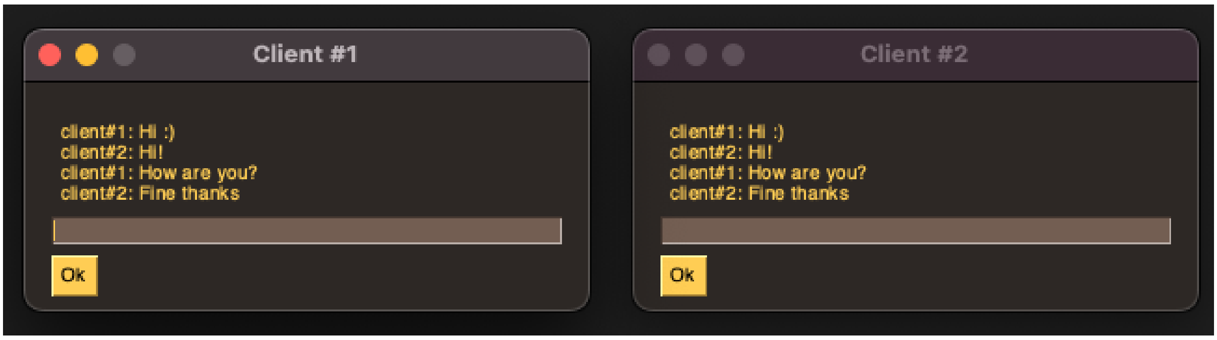
Enter again: bla

passwords dont match

Client asks for the password that was previously set:



Clients exchanging messages:



```
client#1: Hi :)
client#2: Hi!
client#1: How are you?
client#2: Fine thanks
```

```
client#1: Hi :)
client#2: Hi!
client#1: How are you?
client#2: Fine thanks
```

1.3 Summary

The project moved a lot towards the final product, the most valuable features that are missing: the file upload & message encryption. We have a solid base applications with many features described above, now we just need to create some additional functionalities.

2. Project – Final term

2.1 Description

Content

2.2 Description

Content

2.3 Description

Content

2.4 Results

Content

2.5 Summary

Content

3. Literature

[1] Article.

[2] Website, (access date).

[3] Book.