Assignment 3

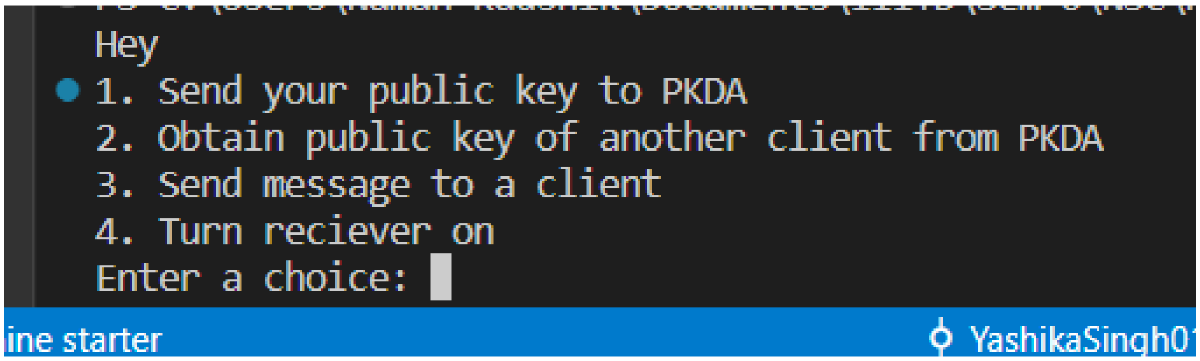
Assumptions:
The public key of PKDA is known to the clients (hence an import statement is used) and those given in the doc.

CA.py:
This file is responsible for providing the keys to different clients It accept keys from clients when they send them encrypted with its own private keyIt also send keys by encrypting with its own private keys

Initially the public and private keys are generated by the clients and the PKDA.

Clients:

When the clients are started they are presented with 4 options:



```
Hey
1. Send your public key to PKDA
2. Obtain public key of another client from PKDA
3. Send message to a client
4. Turn reciever on
Enter a choice:
```

When Option 1 is selected:

The public key generated by the client is sent to the PKDA after encryption with the public key of the PKDA along with the an ID of the client.

Then PKDA decrypts the message with its own private key and stores the key of the client in a mapping

When Option 2 is selected:

The program asks the client for whose key is needed. Upon entering, the request is encrypted and sent to PKDA. The PKDA sends the required public key in the encrypted format which the client can decrypt. This key is now stored with the client

When Option 3 is selected:

The client is asked who to send the message to. Then it is checked if the client has the required public key for the same. If not, the client is prompted to ask for it form the pkd. Else, the message is asked for which is encrypted along with a nonce, timestamp added and sent along with the hashed version of the string.

When Option 4 is selected:

The receiver is turned on and the client listens for messages. The client on receiving the message decrypts the message with its own private key and check the timestamp and get the difference between now and the timestamp. If the difference is more than a threshold, the request may be old and is not responded to. The decrypted message is hashed and compared with the hash received to check if the message has been tampered with.

Helpers.py

This file contains some of our utility functions like encrypt, decrypt, and the generate keys functions. Since R.S.A encrypts numbers, to send text, we also made a function called convertToNum which converts the string to the corresponding ASCII characters and a convertToStr which does the reverse.

This file also contains the mapping of the clients along with the port number where their sockets listen to.

Implementation of R.S.A

The public and private keys are generated using R.S.A. One way that we considered was using loops to generate the keys. However it was taking a lot of time and was not feasible to test our assignment each time since the prime numbers we used were large (so the message could be encrypted).

Hence, we used the concept of Fermat numbers to generate the values of e and d

Reference followed -
https://www.brainkart.com/article/Distribution-of-Public-Keys_8469/#:~:text=On%20the%20face%20of%20it,at%20large%20(Figure%2014.9).