Aamleen Ahmad (2020002)
Mohammad Sufyan Azam (2020312)

Network Security Assignment 3

We had to implement Project 0: RSA-based Public-key Certification Authority (CA).

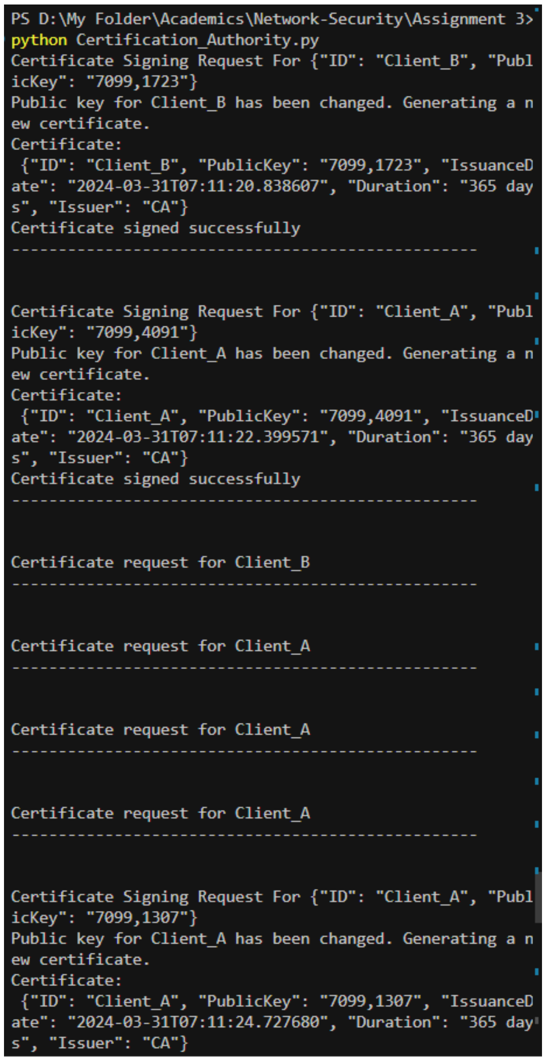
Background:
We created three files: Certification_Authority.py, Client.py, and RSA.py. Our code has the property that if the Certification_Authority server is not in use for more than 10 seconds, then it will timeout and exit. This was done to ensure that the Certification Authority server will never be unnecessarily hogging up the system resources when there isn't a need for it. Similarly, Client B will timeout after 7 seconds, and Client A will immediately timeout after sending and receiving the 3 test messages. Also, our code requires the input of a config.conf file, which contains the details of every client or CA present in the following format: "{ID} {IP_ADDRESS} {PORT}\n". The ID for the Certification Authority is CA, and for the clients, it is A and B. There is a batch file called delete.bat present which deletes the present certificates and keys so that the program generates them again.

Our code uses socket programming to connect with the different servers (i.e., CA and other clients). The program when initialized first checks whether an existing certificate and public key of CA is present or not. If it is valid and present then it loads the certificate and the key otherwise it generates new certificates and keys for smooth execution.

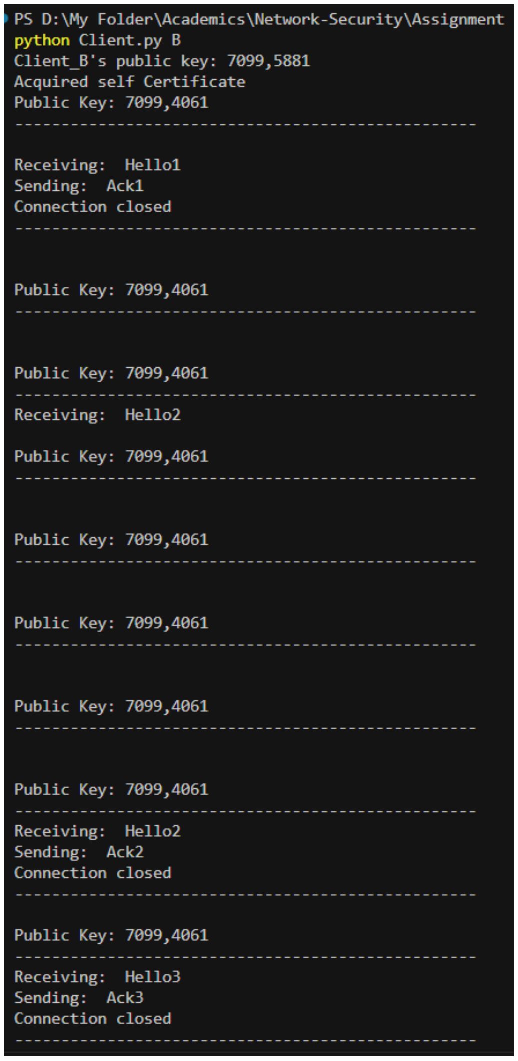
Usage:
Start three different terminals and run the Certificate_Authority.py file on one terminal. Then run the command 'python Client.py B' on the other terminal to start Client B. Then run the command 'python Client.py A' on the third terminal to start Client A.

Sample Screenshot of the working -

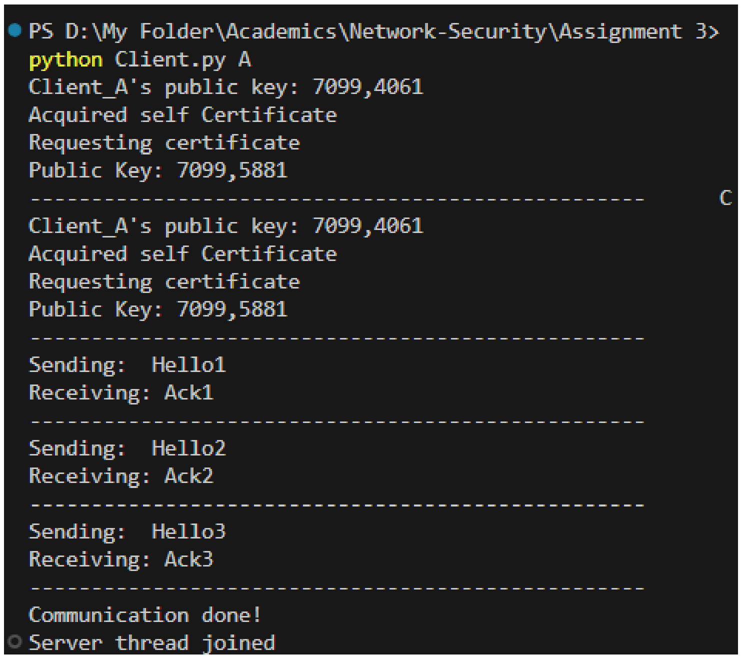
Aamleen Ahmad (2020002)
Mohammad Sufyan Azam (2020312)



```
PS D:\My Folder\Academics\Network-Security\Assignment 3>
python Certification_Authority.py
Certificate Signing Request For {"ID": "Client_B", "Publ
icKey": "7099,1723"}
Public key for Client_B has been changed. Generating a n
ew certificate.
Certificate:
 {"ID": "Client_B", "PublicKey": "7099,1723", "IssuanceD
ate": "2024-03-31T07:11:20.838607", "Duration": "365 day
s", "Issuer": "CA"}
Certificate signed successfully
--------------------------------------------------


Certificate Signing Request For {"ID": "Client_A", "Publ
icKey": "7099,4091"}
Public key for Client_A has been changed. Generating a n
ew certificate.
Certificate:
 {"ID": "Client_A", "PublicKey": "7099,4091", "IssuanceD
ate": "2024-03-31T07:11:22.399571", "Duration": "365 day
s", "Issuer": "CA"}
Certificate signed successfully
--------------------------------------------------


Certificate request for Client_B
--------------------------------------------------


Certificate request for Client_A
--------------------------------------------------


Certificate request for Client_A
--------------------------------------------------


Certificate request for Client_A
--------------------------------------------------


Certificate Signing Request For {"ID": "Client_A", "Publ
icKey": "7099,1307"}
Public key for Client_A has been changed. Generating a n
ew certificate.
Certificate:
 {"ID": "Client_A", "PublicKey": "7099,1307", "IssuanceD
ate": "2024-03-31T07:11:24.727680", "Duration": "365 day
s", "Issuer": "CA"}
```



```
PS D:\My Folder\Academics\Network-Security\Assignment
python Client.py B
Client_B's public key: 7099,5881
Acquired self Certificate
Public Key: 7099,4061
--------------------------------------------------

Receiving:  Hello1
Sending:  Ack1
Connection closed
--------------------------------------------------


Public Key: 7099,4061
--------------------------------------------------


Public Key: 7099,4061
--------------------------------------------------
Receiving:  Hello2

Public Key: 7099,4061
--------------------------------------------------


Public Key: 7099,4061
--------------------------------------------------


Public Key: 7099,4061
--------------------------------------------------


Public Key: 7099,4061
--------------------------------------------------


Public Key: 7099,4061
--------------------------------------------------
Receiving:  Hello2
Sending:  Ack2
Connection closed
--------------------------------------------------

Public Key: 7099,4061
--------------------------------------------------
Receiving:  Hello3
Sending:  Ack3
Connection closed
--------------------------------------------------
```



```
PS D:\My Folder\Academics\Network-Security\Assignment 3>
python Client.py A
Client_A's public key: 7099,4061
Acquired self Certificate
Requesting certificate
Public Key: 7099,5881
--------------------------------------------------      C
Client_A's public key: 7099,4061
Acquired self Certificate
Requesting certificate
Public Key: 7099,5881
--------------------------------------------------
Sending:  Hello1
Receiving: Ack1
--------------------------------------------------
Sending:  Hello2
Receiving: Ack2
--------------------------------------------------
Sending:  Hello3
Receiving: Ack3
--------------------------------------------------
Communication done!
Server thread joined
```

The RSA algorithm was used to encrypt/decrypt to communicate effectively with either the CA to get the certificates or the Client to send/receive messages.

RSA.py:
This file was used to implement the RSA algorithm. A class called RSA was defined in it which had all the required functions to implement the RSA algorithm. This class was imported in other files for use. Its essential functions were -
generate_keys(): It was used to create n, d, e, phi using p and q class variables which were two prime numbers.
get_public_key(): It was used to get the public key pair (n, d) for the RSA class.
save_public_key(): It was used to store the public key in a separate json file.
load_public_key(): It was used to load the public key from the json file and store the relevant variables (i.e, n,d) into the class variables.
encrypt_using_public_key(): It returned encrypted bytes of data using the public key (n,d).
decrypt_using_private_key(): It returned decrypted bytes of data using the private key (n,e).
encrypt(): It returned encrypted bytes of data using the private key (n,e). This function is used by the Certification Authority.
decrypt(): It returned decrypted bytes of data using the public key (n,d). This function is used by the Certification Authority.

Client.py:
This file was used to handle all the client functions that were needed for successfully testing the Certification Authority. It starts a client on the terminal that it is run. Its essential functions were -
get_certificate(): It returns its own certificate signed by the Certificate Authority.
handle_connection_request(): It handles all the connection requests including requests sent from the other client.
server(): It starts a server that will listen to requests made to the client.
request_certificate_of_client(): It requests certification of the other client from the Certification Authority.
communicate(): Initiates communication with the other client

Aamleen Ahmad (2020002)
Mohammad Sufyan Azam (2020312)

Certification_Authority.py:

This file handles all the requests made by the client to give or sign the certificates of either the client that has made the request or the client that it needs the certificate to.

Its essential functions were -

__create_certificate__(): It creates a new certificate and returns it in a dictionary format.

__verify_certificate_validity__(): It verifies that the certificate has not expired yet.

get_certificate(): Checks for the certificate validity (expiry date check), public key match (of Client), and generates or retrieves the certificate as deemed appropriate.

server(): It starts a server that will listen to requests made to it by the clients.

handle_connection_request(): It handles all the connection requests sent from the client. It can either be to sign the certificate or to retrieve the certificate.

sign_certificate_request(): This function is called whenever a certificate needs to be signed.