Network Security A3

Soumya Mohapatra (2021103), Harshit Pal (2021255)

Introduction:

The assignment consisted of building a certification system with a central certification authority and 2 clients 1 of which needs to send messages to others while the other sends replies. Certificates sent by the CA can have custom validity (this can be changed if wanted to a constant). Both clients ask for their own and each other's certificate. The certificate has the following fields: the client ID (port), public key of the requested client, time of issuing and duration of validity along with Certificate Authority id it is concatenated with its hash and encrypted by CAs private key for transfer.

Assumptions:

1. Each of the correspondents knows each other's public key. (in reality C1 and C2 may generate their own keys and enroll it to the CA)
2. They all are on the same machine i.e for the socket address the port number is different. This is for consistency since we use port number as their IDs not the entire socket
3. C1 is the establisher of connection. (For sake of convenience)

Key Components:

RSA Algorithm:

1. Provided in rsa.py which when run can generate key pairs for all three nodes. The length of the key can be set arbitrarily. We have set it to 8 digits for sake of readability. Generally it should be set to 100 digits with n being 300 digits
2. The module provides three utilities rsa_encrypt, rsa_decrypt for encryption and decryption (they are both the same, just different names for sake of readability). Takes in input bytes and returns encrypted bytes according to the key. We use block encryption by converting the bytes into integers and dividing it into segments according to N.
3. Rsa_generate_key_pair generate two key pairs and N for two given random_prime

Certificate Validity Checker:

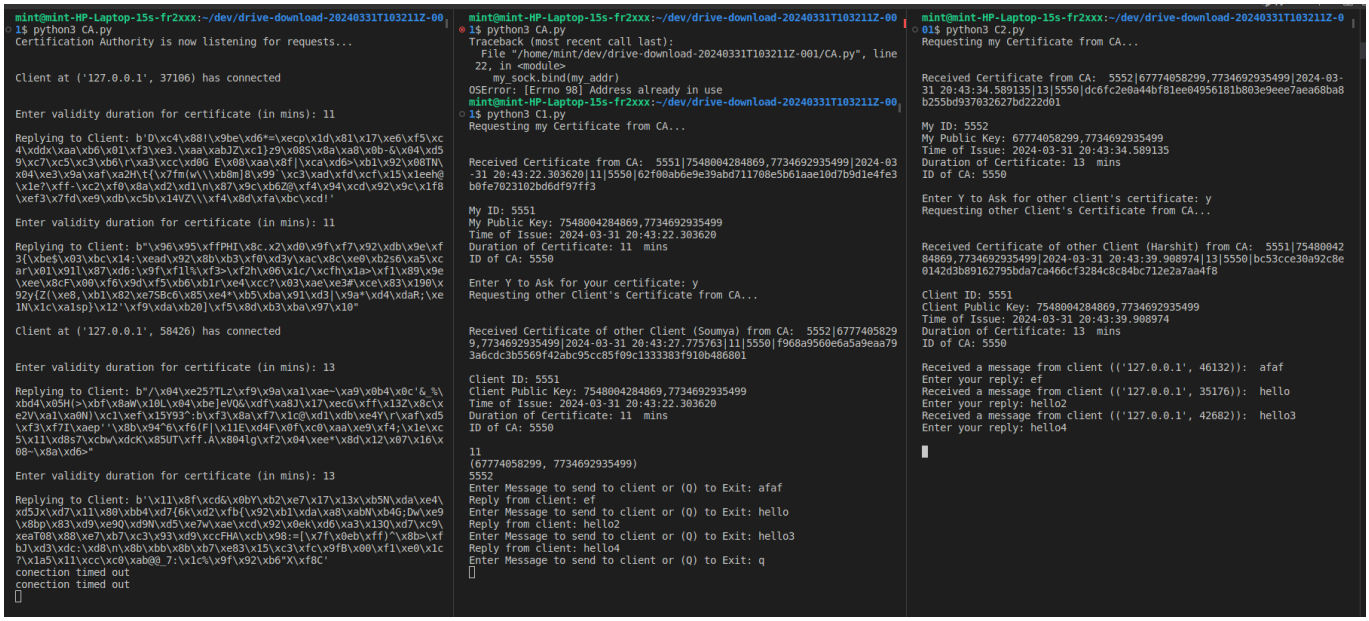
1. Checks the validity of the certificate by decrypting the entire message and then decrypting the sha256 hash encrypted using CAs private key. Comparing the digest tells

us about the integrity of the message. It is non-repudiable since we use asymmetric encryption over hashing

Encryption Between Clients:

1. After the clients have received their certificate they extract the public keys of the certificate. Anyone who sends a message first encrypts it with its private key and then with the other client's public key. Decryption happens in the reverse process. If the certificate expires the key information is deleted from the client side.

Sample output:



Conclusion:

The program successfully sends messages from one end to the other. A lot of decisions that are made are for readability hence the code can be made more secure by following suggestions above.