NSSII Assignment 2

Manavjeet Singh, 2018295

Part 1, IRC client

Dependencies

The application is dependent on following python modules

- socket
- pickle
- cryptography
- threading
- os
- time
- random
- base64

Running

- Use "make server" to run the server
- Use "make c0" to run the client 0
- Use "make c1" to run the client 1
- Use "make c2" to run the client 2
- Use "make c3" to run the client 3

Configure

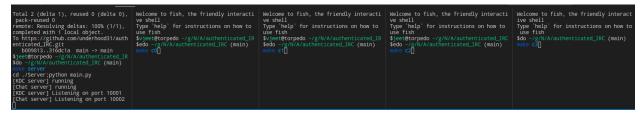
The server and clients are preconfigured.

But if you want to configure again, run "python configure.py" and enter the relevant details and follow the following steps:

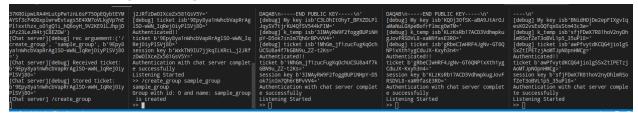
- Move "all_client_listening.info" and "all_client_secrets.info" files generated in root folder to the Server folder.
- Move the respective client config info files to the client folders.

Test Run

Run the programs as instructed in Running section and do the following:



 Run "/create_group sample_group" on client0. A group id will be returned. Following steps are assuming that the group id is 0.



• Enter "/group_invite 0 1" to invite client 1 to group 0. Similarly for client2, and client3



Run "/init_group_dhxchg 0 1" to do a DH key exchange with client3 for group0. Repeat
this for client2 and client3 to update the group key for their DH keys.



Run "/write_group 0 hello" to write the message to the group0, encrypted by their DH key.



Run "/who" to see all those on the server



Run "/write_all message" to broadcast message to all the clients



Run "/request_public_key 3" to get the public key of the client 3



Commands and assumptions

- "/who": Who all are logged in to the chat server, along with a user IDs.
- "/write all": Write message which gets broadcasted to all users.
- "/create_group <grp_name>": Create a group to which users may be added. A group ID and name is returned.
- "/group invite <grp id> <cli>to id>": Send an invite to individual users IDs."
- "/group_invite_accept": Convert acceptance variable to true, all requests will be accepted
- "/group invite decline": Convert acceptance variable to false, all requests will be denied
- "/request public key": Send request for public key to a specific users.
- "/send_public_key": Send back public key back as a response to the above request. This command works internally, the user cannot fill it.
- "/init_group_dhxchg": This process initiates a DH exchange first with any two users and then adds more users to the set..

- "/write_group <grp_id> message": Write messages to a group specifying its group ID.
- "/list_user_files <ip addr> <port>": list the files in the client directory
- "/request_file <ip_addr> <port> <file_name>": loads the file into local client directory

Documentation and Code

Some highlights of the documentation is given below:

Client to KDC server

KDC side:

Client Side:

Session key is the function of K_c, TS and nonce.

Client to Chat server authentication

```
The client should have a valid ticket and session_key before contacting the chat server.

This can be done by calling the authenticate function of this class
```

```
Client

Chat Server

Chat Serve
```

The chat server matches the ticket and authenticates the client.

There is a shared database data structure that keeps track of client session_keys, ports, ip addresses and tickets.

For detailed information on diffie hellman key exchange, read documentation of dh_key_xchange_request, df_xchg_handler and start_listening from client, server_chat and again client documentation from docs folder or the following links.

For detailed documentation and code open HTML files in the docs folder in the submissions or the following links. (Ps, if some comment is not clear, click on the expand code button under to see the raw text that would be clear).

- client.py: https://underhood31.github.io/authenticated IRC/client.html
- Server, main.py: https://underhood31.github.io/authenticated IRC/server main
- Server, kdc.py: https://underhood31.github.io/authenticated IRC/server kdc
- Server, chat.py: https://underhood31.github.io/authenticated IRC/server chat

References

https://nitratine.net/blog/post/asymmetric-encryption-and-decryption-in-python/#:~:text=Asymmetric%20encryption%20uses%20two%20keys,key%20can%20decrypt%20the%20message. https://cryptographv.io/en/latest/hazmat/primitives/asymmetric/rsa/

 $\frac{\text{https://www.studytonight.com/python/python-threading-lock-object\#:} \sim :text=Lock\%20Object\%3A}{\%20Python\%20Multithreading\&text=This\%20lock\%20helps\%20us\%20in,we\%20initialize\%20th}{e\%20Lock\%20object}$

https://cryptography.io/en/latest/fernet/#using-passwords-with-fernet

https://devga.io/encrypt-decrypt-data-python/

https://cryptography.io/en/latest/hazmat/primitives/cryptographic-hashes/