Secure Fast Chat: Client Program

Khushang Singla Mridul Agarwal Arhaan Ahmad

Nov 21, 2022

CONTENTS:

CHAPTER

ONE

CLIENT_SIDE

1.1 Message module

class Message.**Message**(*conn_socket*, *task*, *request*, *box*)

Bases: object

This is the class to handle Encryption of messages. The format in which the message is sent to server is determined in this class

Parameters

- **task** (*str*) – Task to be done. It can have the values signup, login, send_message
- **socket** (*socket.socket*) – The socket used for connection with Server
- **request_content** (*dict*) – Content to include in the request to send to server
- **_data_to_send** (*bytes*) – Contains the data to send to the server
- **_recvd_msg** (*bytes*) – Content recieved from server is stored here
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

Constructor Object

Parameters

- **conn_socket** (*socket.socket*) – Socket which has a connection with server
- **task** (*str*) – Task to do. It can have values: login, signup, send_message
- **request** (*str*) – Content to send to server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

_send_data_to_server()

Function to send the string to the server. It sends content of _send_data_to_server to the server

_recv_data_from_server(*size*, *encrypted=True*)

Function to recv data from server. Stores the bytes recieved in a variable named _recvd_msg.

Parameters

- **size** (*int*) – Length of content to recieve from server
- **encrypted** (*bool*) – (optional, True) Is the information being recieved encrypted

_json_encode(*obj*, *encoding='utf-8'*)

Function to encode dictionary to bytes object

Parameters

- **obj** (*dict*) – dictionary to encode
- **encoding** (*str*) – Encoding to use

Returns
Encoded obj

Return type
bytes

_json_decode(*obj*, *encoding='utf-8'*)

Function to decode bytes object to dictionary

Parameters

- **obj** (*bytes*) – Encoded json data
- **encoding** (*str*) – Encoding used

Returns
Decoded json object

Return type
json

_encrypt_server(*data*)

Encrypts the data

Parameters
data (*Any*) – the data to encrypt

Returns
Encrypted data

Return type
bytes

_hash_password(*passwd*)

Function to salt and hash the password before sending to server

Parameters
passwd (*str*) – Password to be hashed

Returns
Transformed Password

Return type
str

_encryptE2E(*msg*, *receiverPubkey: PublicKey*) → bytes

Encrypt the message to send to reciever

Parameters

- **msg** (*bytes*) – Message to encrypt
- **receiverPubkey** (*nacl.public.PublicKey*) – Public Key of the other user

Returns
Message encrypted using receiver key

Return type
bytes

`_decrypt`(*msg*, *senderPubKey: PublicKey*) → bytes

Function to decrypt the content accessible to users only

Parameters

- **msg** (*bytes*) – Content to decrypt
- **key** (*str*) – Key to use for decryption

Returns
Decrypted Message

Return type
bytes

`_create_login_request`()

The jsonheader has the following keys: | byteorder, request, content-encoding, username, password. The value for request is 'login' |

Returns
Message to send to server directly for login

Return type
bytes

`_create_signuppass_request`()

The jsonheader has the following keys: | byteorder, request, content-length, content-encoding. The value for request is 'signuppass' | The content has encoded password and e2e public Key

Returns
Message to send to server directly for login

Return type
bytes

`_create_signupuid_request`()

The jsonheader has the following keys: | byteorder, request, content-length, content-encoding. The value for request is 'signupuid' | The content has user id.

Returns
Message to send to server directly for signup request

Return type
bytes

`_login`()

Function to help login into the system. This function sends the login details to the server | The function expects to recieve a response of size 2 from server which gives 0 if success and 1 if wrong uid and 2 for wrong passwd

Returns
Response from server converted to int

Return type
int

`_signuppass`()

Function to save account password at server The function expects to recieve a response of size 2 from server which gives 0 if username already taken and 1 for success

Returns
Response from server converted to int

Return type
int

`_signupuid()`

Function to help signup to make new account. This function sends the new user userid to the server | The function expects to recieve a response of size 2 from server which gives 0 if username already taken and 1 if username is available

Returns
Response from server converted to int

Return type
int

`_create_group_key()`

Function to get the Private key of group to use it to encrypt the messages being sent in groups

Returns
private key

Return type
str

`_keyex()`

Send public key to server for encryption of server->client interaction

`_recvmsg()`

Recieves the information from server. It interprets this as a message from some user and returns the message recieved. The header of recieved request should at least have 'content','content-type','sender','content-length','byteorder','sender_e2e_public_key' as the keys

Returns
The dictionary containing details of message.

Return type
dict

`_get_user_public_key(uid)`

Function to get public key of a user

Parameters
uid (*str*) – uid of user

Returns
key of user if found, None otherwise

Return type
nacl.public.PublicKey

`_sendmsg()`

This function sends the message to the server

Returns
Exit status, 0 for success, 1 if no such uid

Return type
int

_create_grp()

Function to send a request to create a group

Returns
exit status. 0 for success, 1 if group name already exists

Return type
int

_get_group_key(*guid*)

Function to get the encrypted group private key from server.

Parameters
guid (*str*) – Group Name

Returns
This function returns key if found, else None if User not in group

Return type
str or None

_add_member_in_group()

Function to add Member in a group

Returns
Exit status to tell the status

Return type
int

_send_message_in_group()

Function to send message in a group

Returns
0 for success, 1 for failure, 2 if not in group

Return type
int

processTask()

Processes the task to do

Returns
Returns int to represent result of the process. The details of return values are given in the corresponding functions handling the actions.

Return type
int

1.2 app module

app.connectToServer()

Function to connect to server and exchange keys for encrypted connection

Returns
Connection Socket and box with keys for decryption and encryption

Return type
socket.socket,nacl.public.Box

app.**login**(*sock*, *box*)

Function to help user log in to the app

Parameters

- **sock** (*socket.socket*) – Socket used for connection to server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

Returns
Socket with which user is connected to server

Return type
socket.socket

app.**signup**(*sock*, *box*)

Function to help user make new account

Parameters

- **sock** (*socket.socket*) – Socket used for connection to server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

Returns
Socket with which user is connected to server

Return type
socket.socket

app.**handleMessageFromServer**(*socket*, *box*)

This function is called when there is a message from server....

Parameters

- **sock** (*socket.socket*) – Socket used for connection to server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

1.3 userInputHandler module

userInputHandler.**checkValidityOfUID**(*uid*)

Function to check if the uid is valid. A valid uid is one which has only a-z,A-Z,0-9,_ characters

Parameters
uid (*str*) – User id to check for

Returns
Return True if valid

Return type
bool

userInputHandler.**sendMessage**(*cmd*, *content_type*, *socket*, *box*)

Parse the messsage to send to the required user

Parameters

- **cmd** (*str*) – The cmd written after "send "
- **socket** (*socket.socket*) – Connection socket

- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

userInputHandler.**sendGroupMessage**(*cmd*, *content_type*, *socket*, *box*)

Parse the message to send to everyone in the group

Parameters

- **cmd** (*str*) – The cmd written after "sendgrp "
- **content_type** (*str*) – Type of message content, i.e. file or text. It is 'file' or 'text'
- **socket** (*socket.socket*) – Connection socket
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

userInputHandler.**createGroup**(*cmd*, *socket*, *box*)

Create a group with the name cmd

Parameters

- **cmd** (*str*) – Group name
- **socket** (*socket.socket*) – Socket which is connected to server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

userInputHandler.**addMemberInGroup**(*cmd*, *socket*, *box*)

Function to add member in a group

Parameters

- **cmd** (*str*) – The part of command containing group name and new member userid
- **socket** (*socket.socket*) – Socket with active authorized connection to server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

userInputHandler.**handleUserInput**(*socket*, *box*)

This function is called when the user sends some input. This function does the work asked by user

Parameters

- **socket** (*socket.socket*) – Socket for connection with server
- **box** (*nacl.public.Box*) – Server Public Key and User Private Key

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

C

H

L

M

P

S

U