Secure Fast Chat: Client Program

Khushang Singla Mridul Agarwal Arhaan Ahmad

Nov 16, 2022

CONTENTS:

1 Client_Side **1**

1.1 Message module 1

1.2 app module 4

1.3 userInputHandler module 5

2 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER

ONE

CLIENT_SIDE

1.1 Message module

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

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

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

_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*, *authenticated=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

_json_encode(*obj*, *encoding*)

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*)
Function to decode bytes object to dictionary

Parameters

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

Returns Decoded json object

Return type json

`_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 string

`_encode`(*text*, *key*)
Function to encode the text using key from server

Parameters

- **`text`** (*`str`*) – string to encode
- **`key`** (*`str`*) – key for encryption

Returns Encoded text

Return type str

`_encrypt`(*msg*, *key*)
Encrypt the message to send to reciever

Parameters

- **`msg`** (*`bytes`*) – Message to encrypt
- **`key`** (*`str`*) – Key to encrypt the message

`_decrypt`(*msg*, *key*)
Function to decrypt the content accessible to users only

Parameters

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

`_create_loginpass_request`()
The jsonheader has the following keys: | byteorder, request, content-length, content-encoding.

Returns Message to send to server directly for login

Return type bytes

`_create_loginuid_request`()
The jsonheader has the following keys: | byteorder, request, content-length, content-encoding. The value for request is 'loginuid' | The content has user id.

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

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 login

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 bytes

_keyex()

_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-len','byteorder' as the keys

_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 bytes

_sendmsg()
This function sends the message to the server

_create_grp()
Function to send a request to create a group

_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

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**(*sock*)
Function to connect to server and exchange the key for encryption

Parameters **sock** (*socket.socket*) – Socket variable to use for connection

app.**getUserSecretFromPassword**(*passwd*)
Function to convert password to 'User Secret'

Parameters **passwd** (*str*) – Password to convert

app.**login**(*sock=None*)
Function to help user log in to the app

Returns Socket with which user is connected to server

Return type socket.socket

app.**signup**(*sock=None*)
Function to help user make new account

Returns Socket with which user is connected to server

Return type socket.socket

app.**handleMessageFromServer**(*socket*)
This function is called when there is a message from server. . . .

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*`)`

Parse the messsage to send to the required user

Parameters

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

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

Parse the message to send to everyone in the group

Parameters

- **`cmd`** (*`str`*) – The cmd written after “sendgrp “
- **`socket`** (*`socket.socket`*) – Connection socket

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

Create a group with the name cmd

Parameters

- **`cmd`** (*`str`*) – Group name
- **`socket`** (*`socket.socket`*) – Socket which is connected to server

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

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

`userInputHandler.`**`handleUserInput`**`(`*socket*`)`

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

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

app, 4

m

Message, 1

u

userInputHandler, 5

INDEX

Symbols

_add_member_in_group() (*Message.Message method*), 4
_create_group_key() (*Message.Message method*), 3
_create_grp() (*Message.Message method*), 4
_create_loginpass_request() (*Message.Message method*), 2
_create_loginuid_request() (*Message.Message method*), 2
_create_signuppass_request() (*Message.Message method*), 2
_create_signupuid_request() (*Message.Message method*), 3
_decrypt() (*Message.Message method*), 2
_encode() (*Message.Message method*), 2
_encrypt() (*Message.Message method*), 2
_get_group_key() (*Message.Message method*), 4
_get_user_public_key() (*Message.Message method*), 3
_hash_password() (*Message.Message method*), 2
_json_decode() (*Message.Message method*), 1
_json_encode() (*Message.Message method*), 1
_keyex() (*Message.Message method*), 3
_login() (*Message.Message method*), 3
_recv_data_from_server() (*Message.Message method*), 1
_recvmsg() (*Message.Message method*), 3
_send_data_to_server() (*Message.Message method*), 1
_send_message_in_group() (*Message.Message method*), 4
_sendmsg() (*Message.Message method*), 3
_signuppass() (*Message.Message method*), 3
_signupuid() (*Message.Message method*), 3

A

addMemberInGroup() (*in module userInputHandler*), 5
app
    module, 4

C

checkValidityOfUID() (*in module userInputHandler*), 5
connectToServer() (*in module app*), 4
createGroup() (*in module userInputHandler*), 5

G

getUserSecretFromPassword() (*in module app*), 4

H

handleMessageFromServer() (*in module app*), 4
handleUserInput() (*in module userInputHandler*), 5

L

login() (*in module app*), 4

M

Message
    module, 1
Message (*class in Message*), 1
module
    app, 4
    Message, 1
    userInputHandler, 5

P

processTask() (*Message.Message method*), 4

S

sendGroupMessage() (*in module userInputHandler*), 5
sendMessage() (*in module userInputHandler*), 5
signup() (*in module app*), 4

U

userInputHandler
    module, 5