Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Encrypted Group Chat

Advanced Computer Networks – CSU33032

Aislinn Addison-Smyth

Contents

1. Program Introduction

The aim of this assignment was to develop a secure social media application for our own social networking application. This social media application will be secure and will only allow users which are part of the secure group chat to decrypt each other’s messages.

I decided to implement my application through python with the use of Sockets. I created a server and a client class, where the client can take user input. I implemented the server class as a multithreaded socket server, my reason being that my server can connect with multiple clients at the same time in the same network. In my opinion, as I was creating a group chat if my server was not multithreaded, I would have to constantly reboot my server and previous messages would be lost.

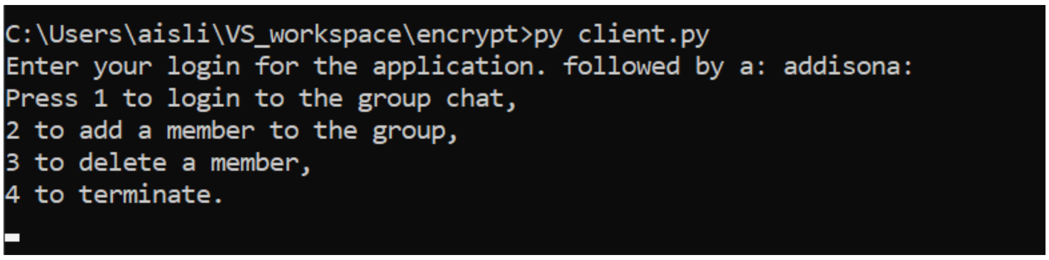
Throughout the course of this report, I will walk through, my code implementation, the reason behind my decisions and walk through the code itself.

2. Code Implementation

2.1Input Choices

I implemented my application to take user input, I find user input being the easiest way to accurately demonstrate the functionality of the application. See figure 1.1 for reference.

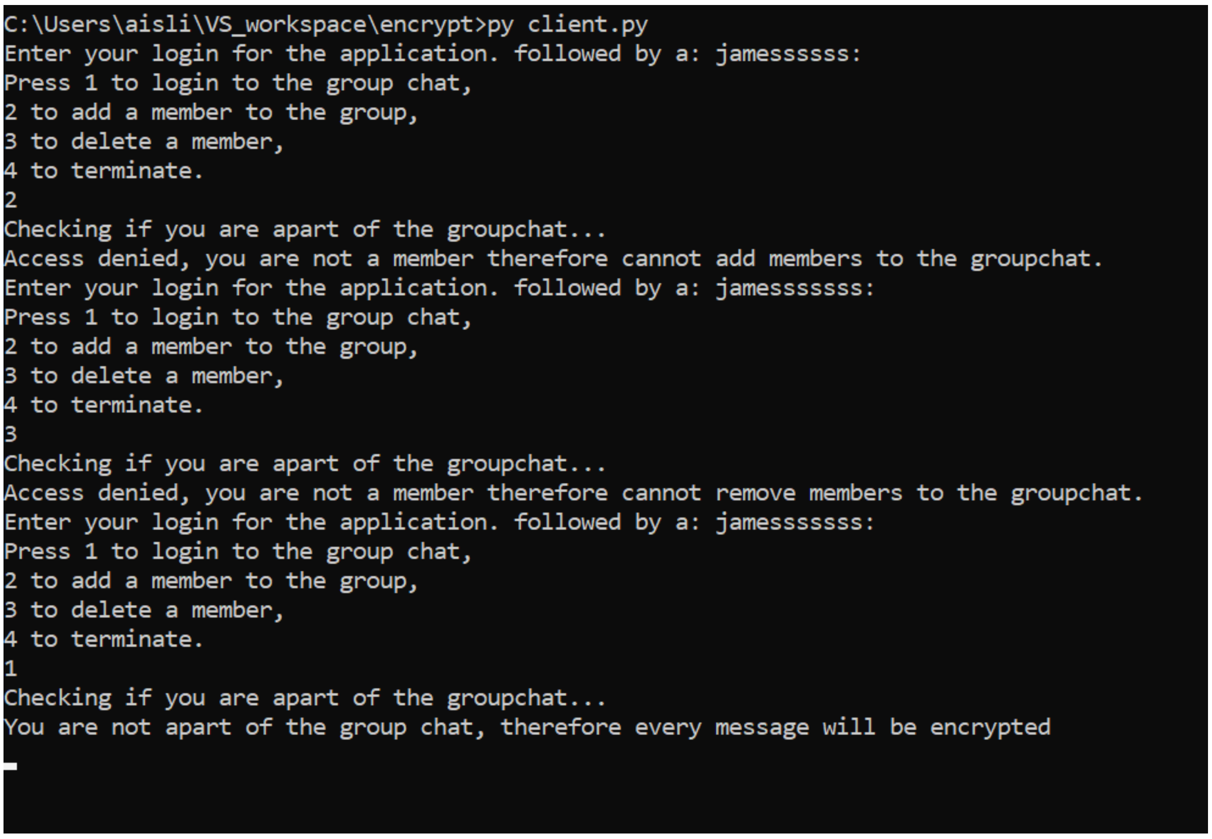
The program begins with asking the user for their login, it then refers to the group chat members to decide whether your login is a part of the group chat or not. The inputs seen in figure 1.1 would appear regardless of whether your login is in the group chat or not.



```
C:\Users\aisli\VS_workspace\encrypt>py client.py
Enter your login for the application. followed by a: addisona:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
```

Figure 1.1: Input Functionality within Client class.

I have built in error-handling functionality within the code to prevent non-group members to dictate the group chat. Figure 1.2 show the functionality of the program when a non-member attempts to use the functionality of the program. The program will respond with an error message, ‘Access denied’ with a reason for the error.



```
C:\Users\aisli\VS_workspace\encrypt>py client.py
Enter your login for the application. followed by a: jamesssss:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
2
Checking if you are apart of the groupchat...
Access denied, you are not a member therefore cannot add members to the groupchat.
Enter your login for the application. followed by a: jamessssss:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
3
Checking if you are apart of the groupchat...
Access denied, you are not a member therefore cannot remove members to the groupchat.
Enter your login for the application. followed by a: jamessssss:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
1
Checking if you are apart of the groupchat...
You are not apart of the group chat, therefore every message will be encrypted
```

Figure 1.2: User Input when a non-member accesses group chat functionality.

If you are a designated member within the group chat, the program will allow you to send encrypted and decrypted messages, add members to the group and remove members within the group chat, see figure 1.3.

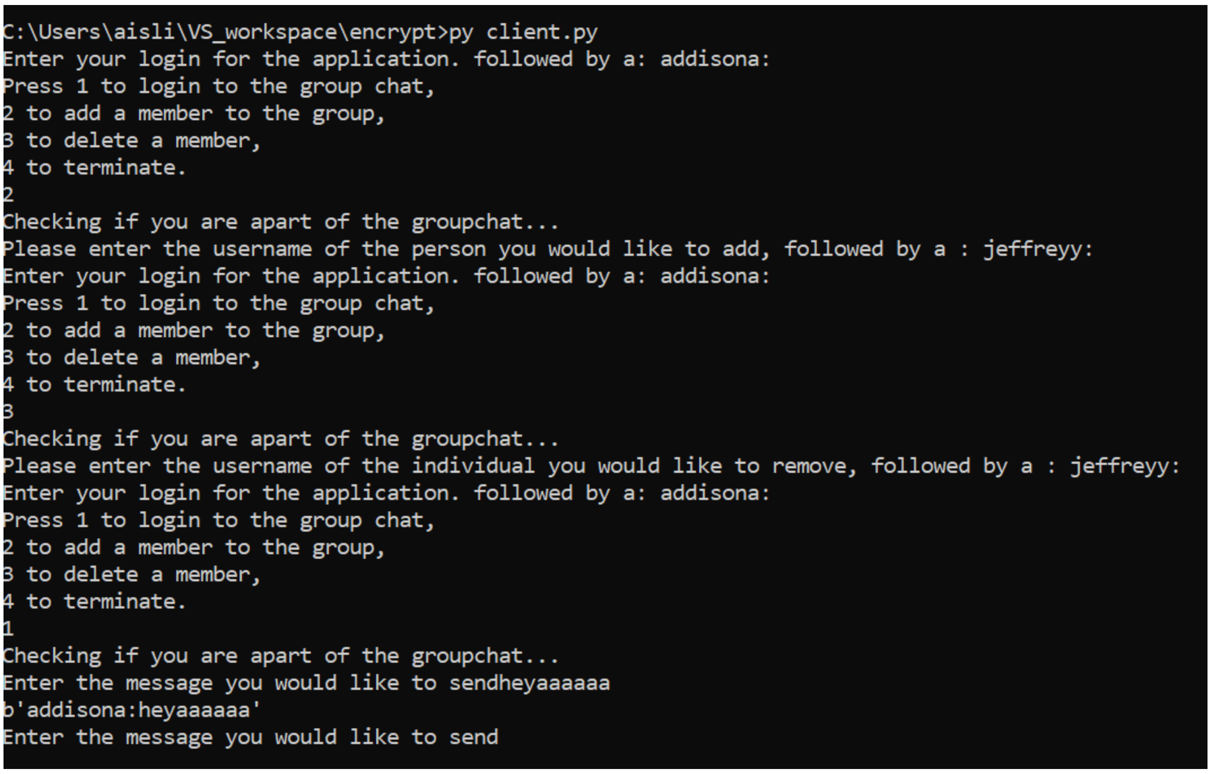

```
C:\Users\aisli\VS_workspace\encrypt>py client.py
Enter your login for the application. followed by a: addisona:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
2
Checking if you are apart of the groupchat...
Please enter the username of the person you would like to add, followed by a : jeffreyy:
Enter your login for the application. followed by a: addisona:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
3
Checking if you are apart of the groupchat...
Please enter the username of the individual you would like to remove, followed by a : jeffreyy:
Enter your login for the application. followed by a: addisona:
Press 1 to login to the group chat,
2 to add a member to the group,
3 to delete a member,
4 to terminate.
1
Checking if you are apart of the groupchat...
Enter the message you would like to sendheyaaaaaa
b'addisona:heyaaaaaa'
Enter the message you would like to send
```

Figure 1.3: Input Functionality for group chat members.

2.2System Implementation

My code is implemented with the use of a client and a server class, where the client class takes use input and implements code functionality. Server class being a multithreaded socket server which can listen to multiple clients, have multiple connections, and send and receive data.

I designed the code in my client class to be separated by functions, each ‘choice’ from figure 1.1 sends the user to a different function. I have 5 functions altogether in the client class, my inputs() function which take user input and sends the code to whatever function Is assigned to their specified number. I have a chat() function and encryptedChat() function, my chat() function is accessed when a user within the group chat wants to login and access the group chat. EncryptedChat() is accessed when a non-member attempts to access the group chat, when the data is sent it is already encrypted as in the case the data gets intercepted it is in ciphertext and cannot be understood. This function simply explains to the user that all messages they can view will be encrypted and therefore receives and prints the ciphered text. The add() function is used to when a member would like to add a member to the group chat, add() accesses the text file groupMembers.txt which is where all group member logins are stored, and is written to once a login is added. Finally, remove() is the remove function which a member can use to remove an already member of the group chat. The client class also has a main which automatically defaults directly to the inputs() function. Additionally, at the end of each function there is a call back to the inputs() class to prevent the code from exiting and leaving the existing connection open.

In my server class, as the code was quite short I had two functions, create_socket() and getData with the main method calling the create_socket() function. The create_socket() function creted a socket, set the socket to be reused (as the server is threaded the sockets must be released and reused), it then binded the socket to a port and host I specify, and finally sets the sockets into listen mode. The second half of this function accepts a connection from a specified address, and then sends the threads into listen mode, to keep on listening with the same socket on the same port number. The getData() function is a function which is referred to in the create_socket() function, this function receives the data, prints the data which is in encrypted form and sends all the data on the connection.

2.3 Encryption/Decryption

To implement the encryption and decryption aspect of my code, I imported a cryptography library titled:

```
from simplecrypt import encrypt, decrypt
```

I found this library to be straight forward to use, in order to encrypt the data, our information had to be in byte-like form, therefore it could not send strings, so I converted any strings I had sent into a byte:

```
byteUsername = bytes(username,"utf-8")
```

I set my data to be sent in encrypted form as I had specified above and only used the decrypt function to decrypt messages once a group member wanted to login to the chat.

```
s.send(encrypt(username, byteMsg))
```

Above is the encrypted data being sent.

```
decryptedData = decrypt(username,data)
```

To decrypt the information, the decrypt case had to be used with two parameters in byte like form.

Overall, I found this library useful and very efficient to use.

3. Code

3.1 Client

```python
import socket
from simplecrypt import encrypt, decrypt
#import threading

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432  # The port used by the server

def inputs():
    username = input("Enter your login for the application. followed by a: ")
    choice = int(input("Press 1 to login to the group chat, 2 to add a member to the group, 3 to delete a member, or 4 to terminate"))
    if(choice == 1):
        print("Checking if you are apart of the groupchat...")
        lines = open("chatMembers.txt").read().splitlines()
        for line in lines:
            if(str(username) in lines):
                msg = input("Enter the message you would like to send")
                chat(username,msg)
            else:
                encryptedChat()
    if(choice == 2):
        print("Checking if you are apart of the groupchat...")
        lines = open("chatMembers.txt").read().splitlines()
        for line in lines:
            if(str(username) in lines):
                username = input("Please enter the username of the person you would like to add, followed by a : ")
                add(username)
            else:
                print("Access denied, you are not a member therefore cannot add members to the groupchat.")
                inputs()
    if(choice == 3):
        print("Checking if you are apart of the groupchat...")
        lines = open("chatMembers.txt").read().splitlines()
        for line in lines:
            if(str(username) in lines):
                username = input("Please enter the username of the individual you would like to remove, followed by a : ")
                remove(username)
            else:
                print("Access denied, you are not a member therefore cannot remove members to the groupchat.")
```

```
                  inputs()
    if(choice == 4):
        exit

def chat(username,msg):
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((HOST, PORT))
        byteMsg = bytes(msg,"utf-8")
        s.send(encrypt(username, byteMsg))
        data = s.recv(1024)
        decryptedData = decrypt(username,data)
        byteUsername = bytes(username,"utf-8")
        print(byteUsername + decryptedData)

def encryptedChat():
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((HOST, PORT))
        print("You are not apart of the group chat, therefore every message will be encrypted")
        data = s.recv(1024)
        print(f"Received {data!r}")
        print(data)
#        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 0) #Re-use the socket
#        s.bind((HOST, PORT))
#        s.listen(10)
#        conn, addr = s.accept()
#        threadListener = threading.Thread(target = getData(s,conn), args=(conn,addr)) #Ready to create a new thread for any request made and run requestReceived
#        threadListener.daemon = True
#        threadListener.start()


def add(username):
    lines = [username]
    with open('chatMembers.txt', 'a') as f:
        for line in lines:
            f.write(line)
            f.write('\n')
    inputs()


def remove(username):
    with open('chatMembers.txt') as fin, open('wordsCleaned.txt', 'wt') as fout:
        list(fout.write(line) for line in fin if line.rstrip() != username)
         inputs()

```

```


if __name__ == '__main__':
    inputs()
```

3.2 Server

```
import socket
import threading
from simplecrypt import encrypt, decrypt

HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432      # Port to listen on (non-privileged ports are > 1023)

def create_socket():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 0) #Re-use the socket
    s.bind((HOST, PORT))
    s.listen(10)

    while True:
        conn, addr = s.accept()
        print ('Got connection from', addr)
        threadListener = threading.Thread(target = getData(s,conn),
args=(conn,addr)) #Ready to create a new thread for any request made and run
requestReceived
        threadListener.daemon = True
        threadListener.start()

def getData(s,conn):
    data = conn.recv(1024)
    print(f"Received {data!r}")
    conn.sendall(data)

if __name__ == '__main__':
    create_socket()
```