اونيۏرسيتي مليسيا ڤهڠ السلطان عبدالله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BCN3033

NETWORK PROGRAMMING

PROJECT 2 FINAL ASSESSMENT

PROJECT TITLE: CHAT APPLICATION (ENCRYPTED MESSAGE) OVER THE NETWORK

LECTURER NAME: DR AHMAD FIRDAUS BIN ZAINAL ABIDIN

NAME	MATRIC NUMBER
NUR ALIYA MAISARAH BINTI MOHD YUSOFF	CA22016

Contents

DOCUMENTATION

Develop a chat application that capable to send, receive messages and secure those messages.

```
import socket
import threading

import rsa

public_key, private_key = rsa.newkeys(1024)
public_partner = None

choice = input("Do you want to server (1) or to client (2)? ")

if choice == "1":
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("192.168.56.1", 9999))
    server.listen()

    client, _ =server.accept()
    client.send(public_key.save_pkcs1("PEM"))
    public_partner = rsa.PublicKey.load_pkcs1(client.recv(1024))

elif choice == "2":
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(("192.168.56.1", 9999))
    public_partner = rsa.PublicKey.load_pkcs1(client.recv(1024))
    client.send(public_key.save_pkcs1("PEM"))

else:
    exit()


def sending_message(c):
    while True:
        message = input("")
        c.send(rsa.encrypt(message.encode(), public_partner))
        print("You: " + message)

def receiving_message(c):
    while True:
        print("Partner: " + rsa.decrypt(c.recv(1024),
private_key).decode())


threading.Thread(target=sending_message,args=(client, )).start()
threading.Thread(target=receiving_message,args=(client, )).start()
```

i. Elaborate the usage of the codes in your application in the report

```
import socket
import threading
import rsa
```

The necessary modules are imported to

Socket module (‘import socket’):

The socket module provides low-level networking interface in Python. It will enable communication between processes across the network With socket, the connection can be established that could send and receive data, and handle errors related to network communication. In the provided code, the ‘socket’ module is used to create TCP socket to communication between the server and client.

Threading module (‘import threading’):

The threading module allows to create and manage thread in Python. Thread are lightweight processes within the process that can run concurrently. Threads also enable parallel execution, which is useful for tasks like handling multiple client connections simultaneously in networking applications. In the provided code, the ‘threading’ module is utilized to manage sending and receiving messages concurrently without blocking each other.

RSA module (‘import rsa’):

The RSA module provides functionalities for working with RSA encryption and decryption. RSA is a public key cryptosystem widely used for secure data transmission and encryption. In the provided code, the rsa module is used to generate RSA key pairs (rsa.newkeys()), encrypt and decrypt messages (rsa.encrypt() and rsa.decrypt()), and serialize/deserialize keys in PKCS#1 format (save_pkcs1() and load_pkcs1()).

```
public_key, private_key = rsa.newkeys(1024)
public_partner = None
```

This line generates a new RSA key pair of 1024 bits. public_key and private_key is then assigned to the generated public and private keys respectively. public_partner is initialized to ‘None’.

```
choice = input("Do you want to server (1) or to client
(2)? ")
```

This line prompt the user to choose between running the code as the server or as a client. The choice is then stored in the variable ‘choice’.

```
if choice == "1":
    server = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    server.bind(("192.168.56.1", 9999))
    server.listen()

    client, _ =server.accept()
    client.send(public_key.save_pkcs1("PEM"))
    public_partner
rsa.PublicKey.load_pkcs1(client.recv(1024))
```

If the user chooses to run the code as a server (1), a TCP socket is created. The server binds to the IP address “192.168.56.1” and port number 9999. It then starts listening for incoming connections. Once a client connects, it sends its public key to the client and receives the public key of the client. The received public key is stored in variable ‘public_partner.

```
elif choice == "2":
    client = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    client.connect(("192.168.56.1", 9999))
    public_partner =
rsa.PublicKey.load_pkcs1(client.recv(1024))
    client.send(public_key.save_pkcs1("PEM"))
```

If the user chooses to run the code as a client, a TCP socket is created. The client then connects to the server at IP address "192.168.56.1" and port 9999. It receives the server's public key and sends its own public key to the server. The received public key is stored in ‘public_partner’.

```
else:
    exit()
```

If the user enters input other than ‘1’ and ‘2’, the program exits.

```
def sending_message(c):
    while True:
        message = input("")
        c.send(rsa.encrypt(message.encode(),
public_partner))
        print("You: " + message)
```

This function is responsible for sending messages. It continuously prompts the user for input, encrypts the input message using the RSA algorithm with the public key of the receipent (‘public_partner’), sends the encrypted message over the socket ‘c’, and prints the message on the sender’s screen.

```
def receiving_message(c):
    while True:
        print("Partner: " + rsa.decrypt(c.recv(1024),
private_key).decode())
```

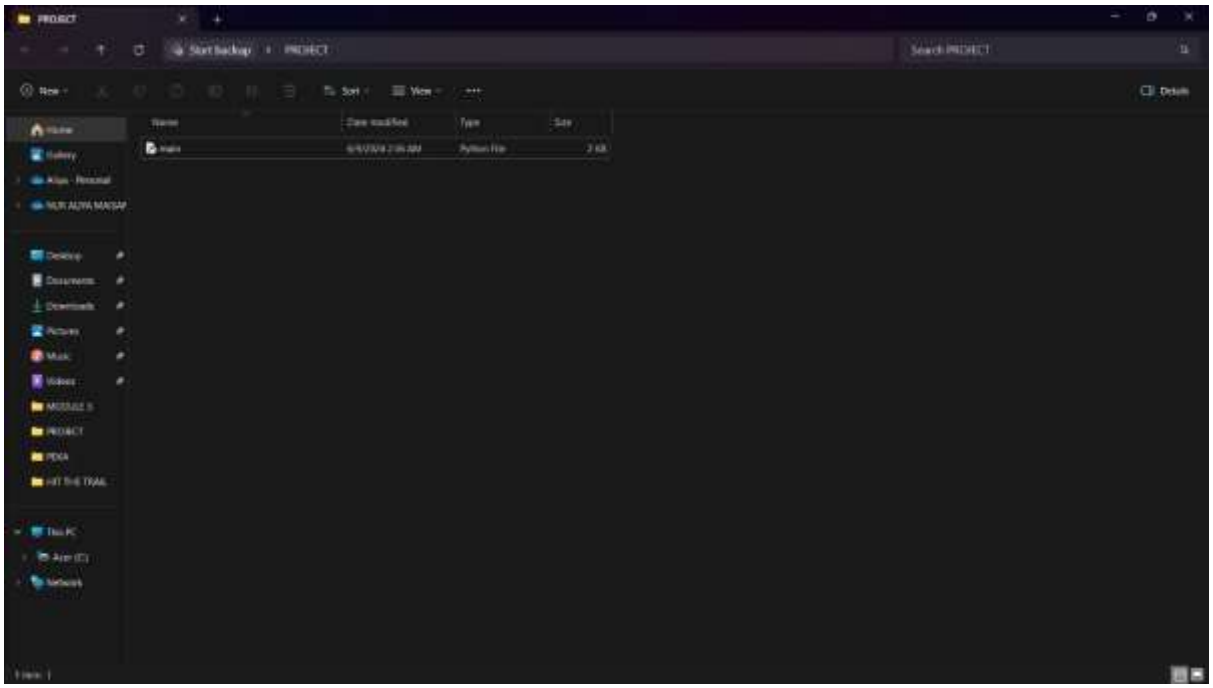
This function is responsible for receiving messages. It continuously listens for incoming messages on the socket ‘c’, decrypts the received messages using the RSA algorithm with the recipient’s private key (‘private_key’), and prints the decrypted message in the recipient’s screen.

```
threading.Thread(target=sending_message,args=(client, )).s
tart()
threading.Thread(target=receiving_message,args=(client, ))
.start()
```

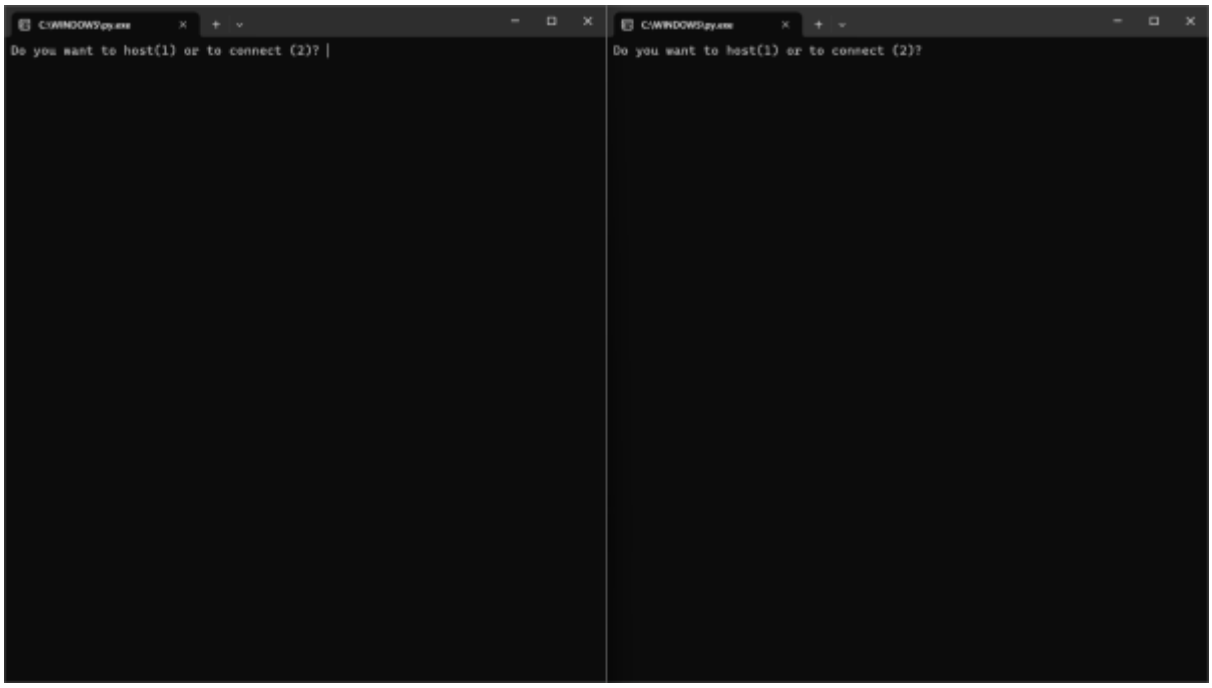
These lines create two separate threads to handle sending and receiving messages concurrently. The ‘sending_message’ function runs in one thread while the ‘receiving_message' function runs in another thread. Each function is passed to the ‘client’ socket as an argument.

ii. Demonstrate the steps to use the chat application and securing message facility

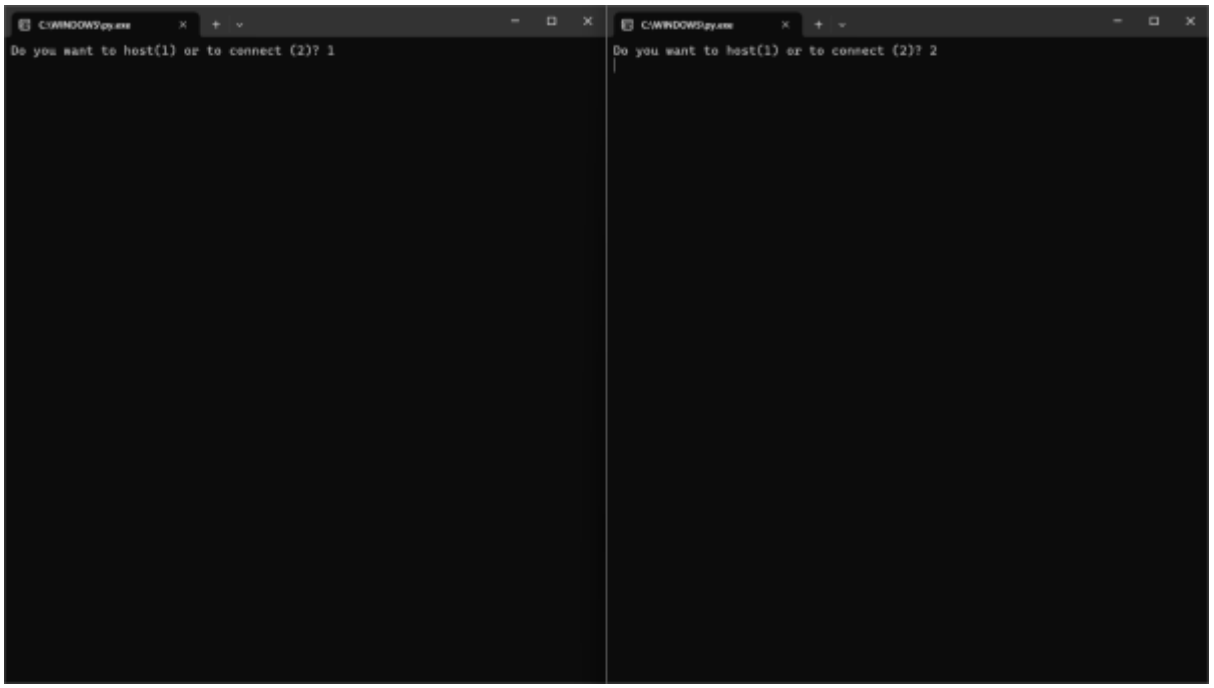
1. Open the file location where the code is saved.



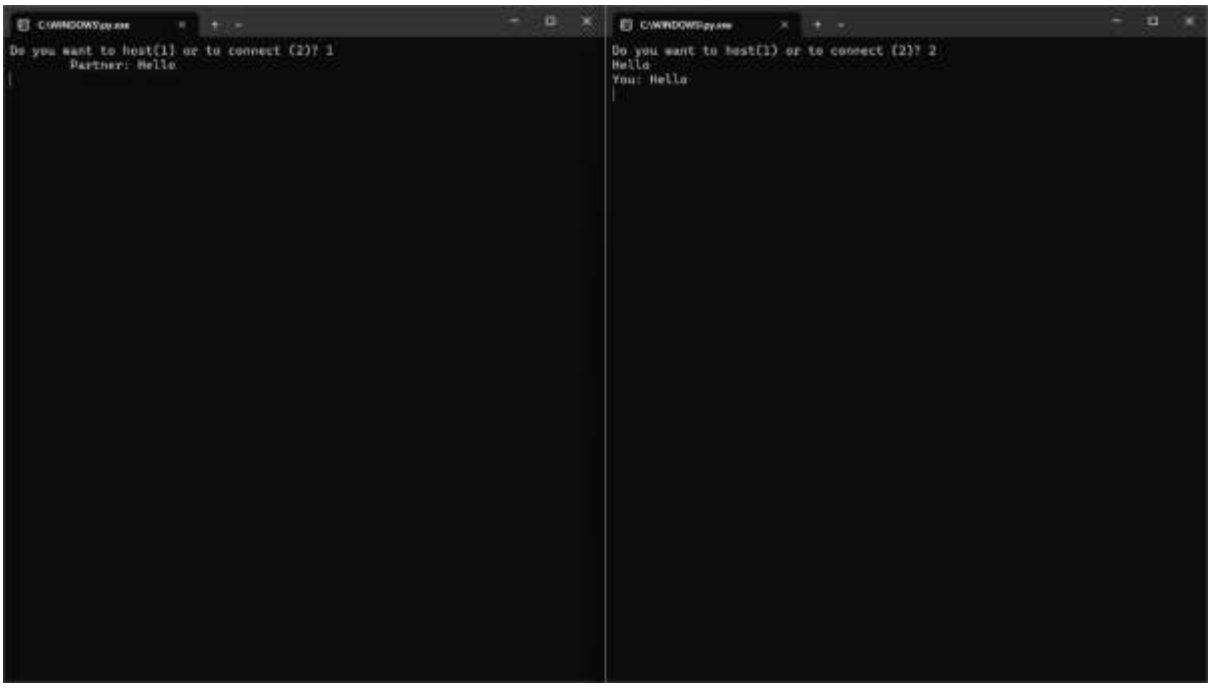
2. Run two command prompt as we use the same network devices



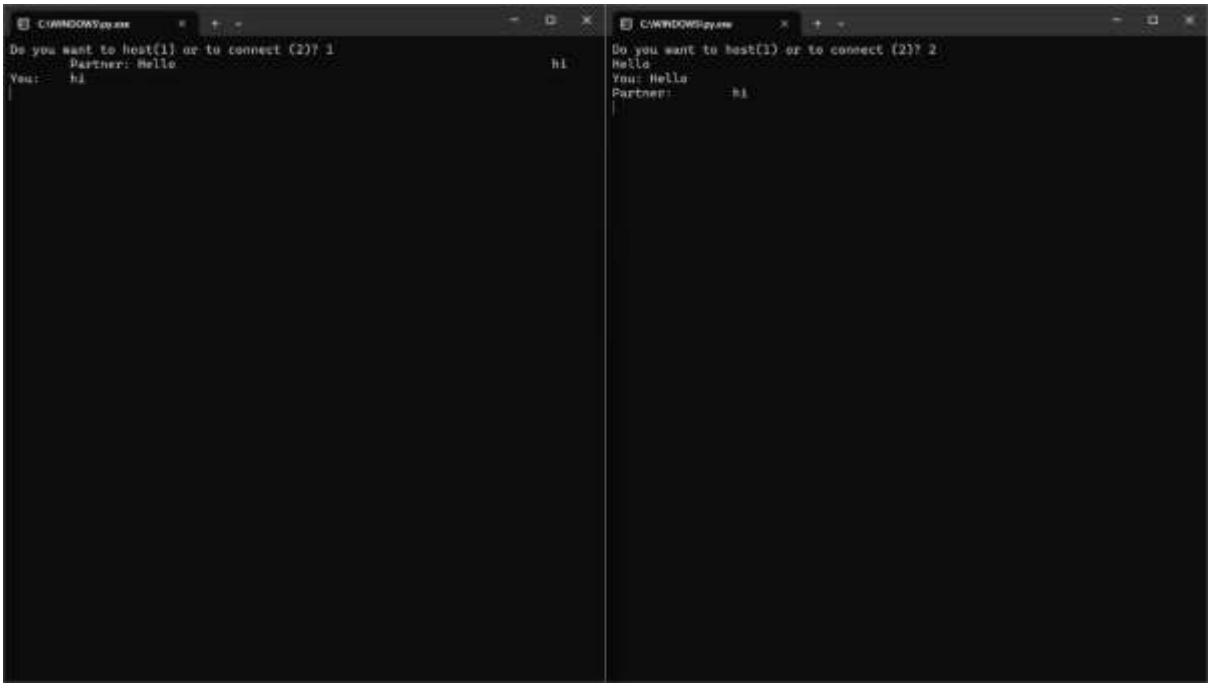
3. One is going to be the host while the other one is going to be the client. The host are going to start the socket as it act as the server.



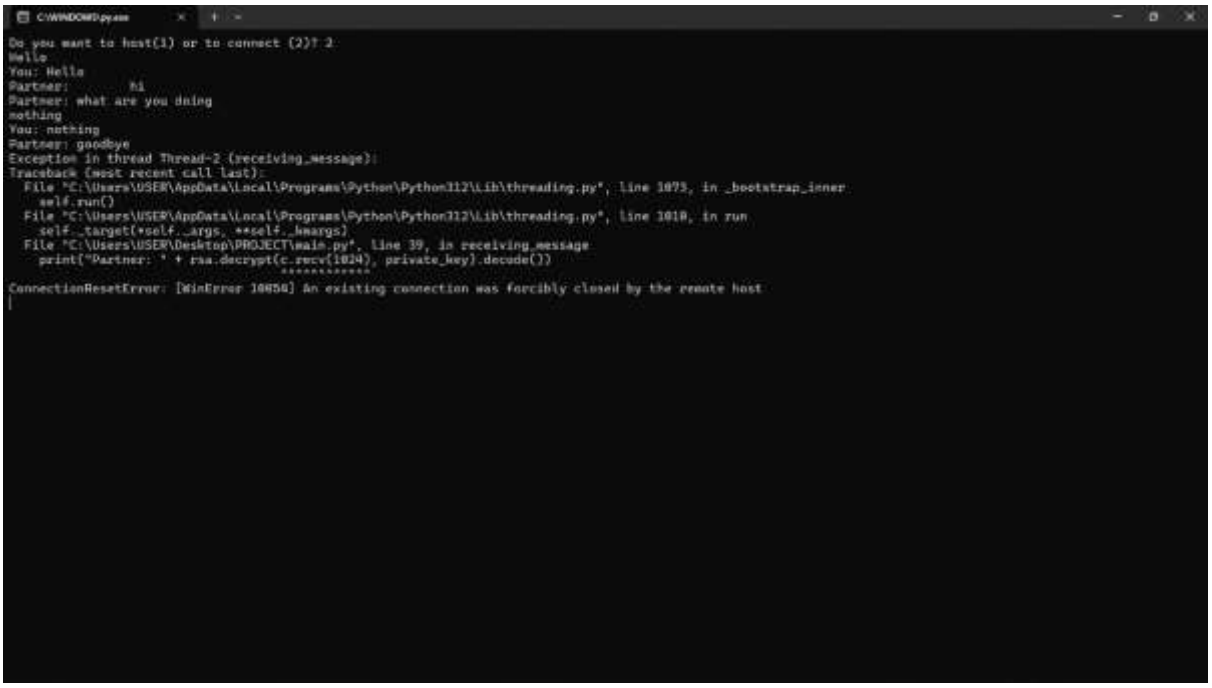
4. When client input the message, and send it to the server side, server will print out the message received from the client side.



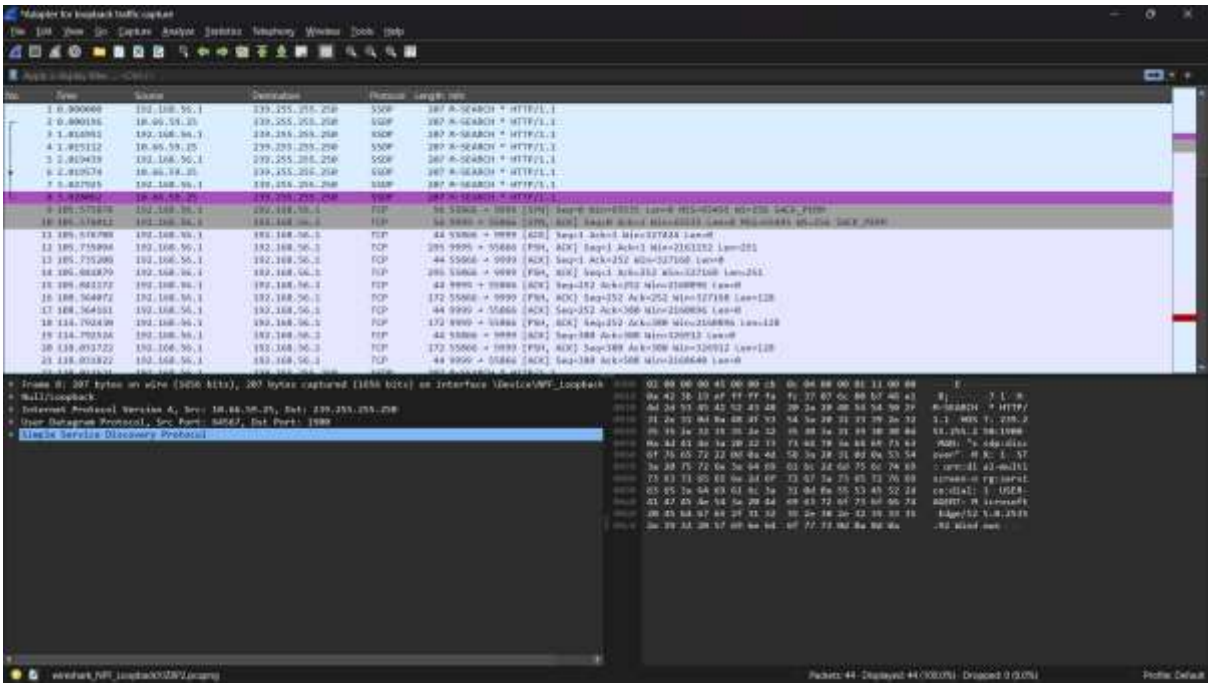
5. Then, the server can also send messages to the client side.



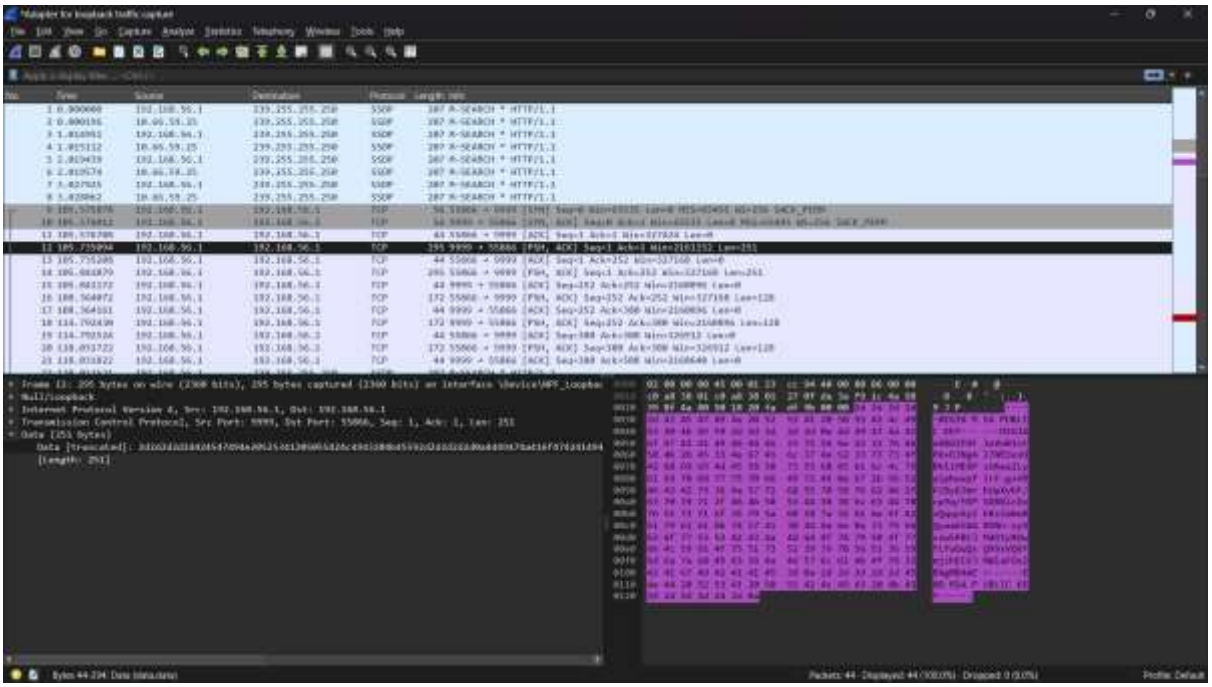
6. Server and client can continue to send messages to each other. To stop messaging, server side just needs to stop the connection by pressing [ctrl + c].



7. To check whether the message is encrypted, we can use Wireshark application to capture traffic.



8. Check the traffic with the same ip address that we set in the code. Then, we could see that the messages were encrypted.



VIDEOS

Youtube link: https://youtu.be/D8-SD9aP-4E?si=boe3MXe-XsbSKUSc

REFERENCES

Aaron S. (2024,Jan 01). Python vs C++: Let's Compare. BitDegree.org. https://www.bitdegree.org/tutorials/python-vs-c-plus-plus

Arpit B. (2021, Apr 6). End-to-end Message Encryption [video]. Youtube. https://www.youtube.com/watch?v=m_7xSIhxZL8

Geeksforgeeks.org. (2023, Jul 14). Differences between Python and C++. https://www.geeksforgeeks.org/difference-between-python-and-c/

Issa A. (2022, Sep 2). Real-Time-Secure-Chat-Application. Github.com. https://github.com/AhmedIssa11/Real-Time-Secure-Chat-Application

Julfikar M.. (2018, Dec 11). Secure Socket programming in Python. Medium.com

https://medium.com/@md.julfikar.mahmud/secure-socket-programming-in-python-d37b93233c69

Keerti P. (2024, Jan 15). Code an End to End Chat Application with Socket Programming | MUST KNOW | C++ [video]. Youtube. https://www.youtube.com/watch?v=okzEZmnVWnM

Leonardo F. (2021, Jan 17). pyChat. Github.com. https://github.com/f0lg0/pyChat

Learn Microsoft. (2021, March 08). Comment pragma. https://learn.microsoft.com/en-us/cpp/preprocessor/comment-c-cpp?view=msvc-170

NeuralNine. (2022, Nov 11). Coding Encrypted Chat in Python [video]. Youtube. https://www.youtube.com/watch?v=U_Q1vqaJi34

Sabah, Noor & Kadhim, Jamal & Dhannoon, Ban. (2017). Developing an End-to-End Secure Chat Application. 17. https://www.researchgate.net/publication/322509087_Developing_an_End-to-End_Secure_Chat_Application