SOFTENG 364 Assignment 2 Report

# Introduction

This report outlines the implementation of a secure chat program as required for Assignment 2 for SOFTENG 364. This secure chat program facilitates encrypted one-on-one communication between multiple clients and a central server. This is all done by utilizing the Transport Layer Security (TLS) Protocol to ensure confidentiality and data integrity. Contained in this report, there will be deeper discussions on the topics; System Architecture, Encryption, Wireshark Results and a professional summary.

# System Architecture

A diagram of a server

Description automatically generated

Figure 1: Client and Server Architecture of the Encrypted Chat Program

This system architecture depicts that the entirety of the system relies on the server. The Server is the centre of this architecture, and all connections are over a secure TLS connection.

The server is responsible for handling multiples clients simultaneously, managing authentication of clients and secure message exchange. Each client connection will be wrapped using TLS protocol ensuring all communication is encrypted. The server will handle socket-based communication, making use of *select* to handle multiple client connections concurrently.

Each client will in turn connect to the server over a secure TLS connection. Clients will send their login information, which is their username to register and be authenticated by the server. After being authenticated from the server, each client can talk to all other clients by exchanging messages to and from the server. This makes the server a medium for message and data exchange.

# Encryption

The secure chat program uses *TLSv1.2* as the encryption method between the server and all clients. The encryption algorithm used to secure the chat program is *AES-128*. This is used in accordance with the in built *ssl* library in Python to handle the encryption and decryption. Below are screenshots from the server side and client-side encryption

A screen shot of a computer code

Description automatically generated

Figure 2: Server-side encryption method screenshot

The above screenshot shows part of the server-side setup. *TLS* encryption is used along with *ssl* in Python. The first line demonstrates that the server will use the *TLS v1.2* protocol to ensure secure communication. Next the certificate and private key is loaded, and these are used during the TLS handshake for authentication so connecting clients can trust the server’s identity. The concurrent line will load the certificate authority which is used to verify client certificates.

The *AES-128* cipher is used as the encryption algorithm and *SHA* is used for message authentication. This is a symmetric encryption algorithm where a shared 128-bit key is used. Additionally, Secure Has Algorithm (*SHA*) is used for message authentication to ensure the encrypted message’s integrity.

A black background with colorful text

Description automatically generated

Figure 3: Client-side encryption method screenshot

On the client side, the client connects to the server via the encrypted *TLS* v1.2 encrypted socket connection. The client is also made to refer to the same *AES-128* symmetric encryption algorithm.

In general, all communications to and from the server to all clients will be secured using the *TLS v1.2* protocol with the *AES-128-SHA* encryption algorithm as this method is efficient and prevents unauthorised used or access. The Wireshark results section will confirm that traffic between server and clients is encrypted.

# Wireshark Results

Below are images of the Wireshark results of the traffic between the client and server.

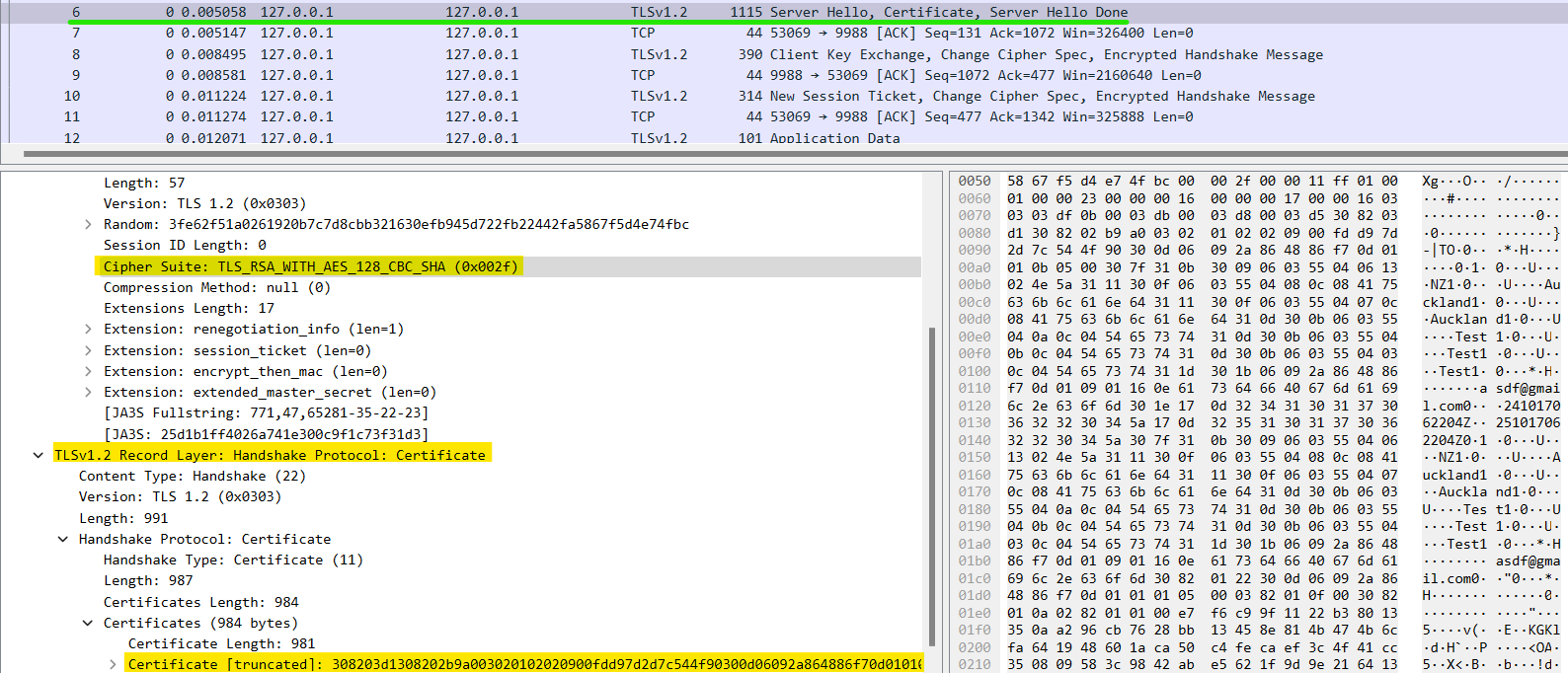


Figure 4: Wireshark analysis of the secure TCP handshake

Figure 4 shows the secure TCP handshake happening between the server an a client. The client firstly sends a hello packet with a random byte string which is used to generate the key for encryption. The cipher spec is also shown to be *AES\_128\_CBC\_SHA* as highlighted.

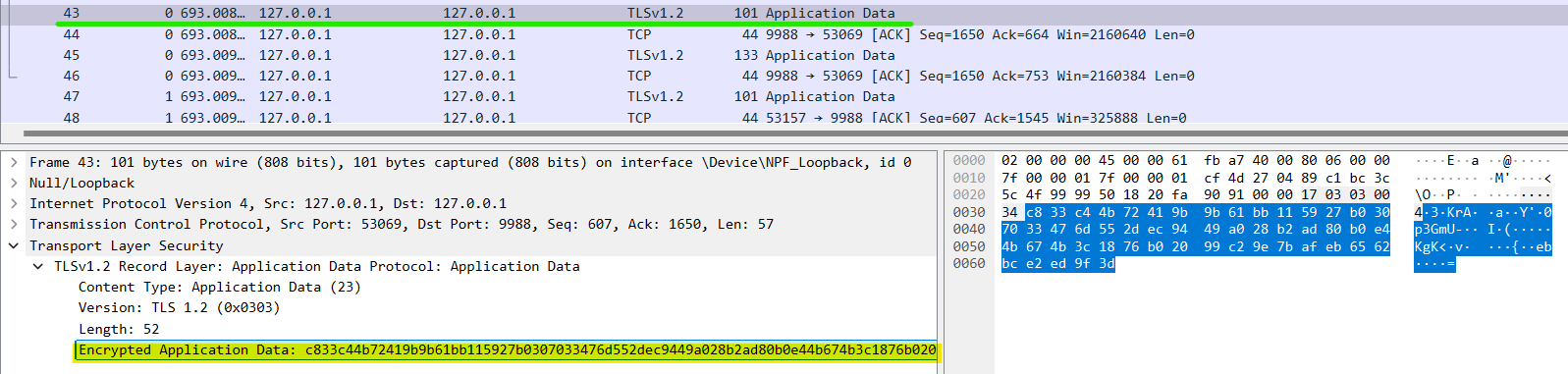


Figure 5: Wireshark analysis of a client sending an encrypted message to another client via the server.

The Wireshark analysis on figure 5 shows a message being sent from a client as a payload application data. The message is encrypted as we can see on the highlighted section. This ensures that all traffic between serves and clients is encrypted ensuring data integrity.

# Summary

In conclusion, the secure chat program successfully implements the encrypted one-on-one communication requirements using the *TLS v1.2* protocol and the *AES-128* encryption algorithm along with *SHA* for hashing and message authentication. This also utilizes the *ssl* library in Python to ensure all clients maintain a secure communication channel with the server.

The system architecture also effectively can manage multiple clients all talking with other clients via interacting with the central server. Due to the server being the main channel data is sent to and from, this *AES-128* encryption algorithm makes a great choice due to its efficiencies to minimize overhead on the server whilst also maintain a secure communication channel.

The Wireshark analysis confirms that the traffic between the server and clients are all encrypted and any 3rd party listening in to the communication will not be able to make out any sense of the encrypted data presented. Overall, this implementation ensures secure data transmission.