**Task**

The goal of the project is to securely connect the client to the server. When connecting the client and server, it is necessary to perform key exchange, authentication of both parties and communication encrypted with the exchanged keys, which uses message authentication.

**Server.py**

* Creates a socket by calling the socket() function. The socket() function returns a socket which is not in the state of accepting connections. The socket is also not specifically bound to an address and a port at the server.
* The socket() object obtained has to be bound to an address and a port. This is needed because the clients of this server program should know which IP address and port number at which they should connect to.
* Above step 2 is achieved by calling the bind() method by passing the IP address and port as a pair. Now having the socket created and bound to a specific IP address and Port at the server, the socket needs to be moved to a state where it waits for client connections.
* By calling the listen() method the socket enters into the TCP state WAIT. Remember listen() is not a blocking call it just makes a socket enter into listen() state so that the socket becomes a server socket.
* The accept()method is called once for each incoming connection and returns a pair containing a socket connecting to the client and the IP address of the client. It is written in a while loop.
* Using the socket for the client connection the server can receive data from the client using recv()method. The server can send data to a client by calling the send()method.

**Client.py**

* The client program creates a socket by calling the socket() function
* Calls connect() method, specifying the IP address and the port number of the server program
* It initiates sending messages to the server by calling send() with byte sequence(s). Through recv()calls the client receives any message sent from the server
* The connection is explicitly closed by calling the close() on the socket.

**Modules used**

[socket](https://docs.python.org/3/library/socket.html)

Used the object of socket to make server listen to incoming connections and for used in client.py to connect to server

[Fernet](https://cryptography.io/en/latest/fernet/)

Imported fernet from cryptography module. Used it for message encryption and decryption using the same fernet key for both server and client.

[Time](https://docs.python.org/3/library/time.html)

Used time.sleep() to pause code execution which takes seconds as a parameter in float.

[Subprocess](https://docs.python.org/3/library/subprocess.html)

The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. Used it to use the “clear” command of linux.

[Sys](https://docs.python.org/3/library/sys.html)

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available. Used it to exit code execution.

**Key Exchange**

The [Diffie-Hellman](https://medium.com/@sadatnazrul/diffie-hellman-key-exchange-explained-python-8d67c378701c) algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

* For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
* P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt

**Message encryption and authentication**

Cryptography is the practice of securing useful information while transmitting from one computer to another or storing data on a computer. Cryptography deals with the encryption of plaintext into ciphertext and decryption of ciphertext into plaintext.

Python supports a cryptography package that helps us encrypt and decrypt data. The fernet module of the cryptography package has inbuilt functions for the generation of the key, encryption of plaintext into ciphertext, and decryption of ciphertext into plaintext using the encrypt and decrypt methods respectively. The fernet module guarantees that data encrypted using it cannot be further manipulated or read without the key.