Part 3 report

Environment Setup

Prerequisites:

1. **Clang++ (C++ Compiler)**: The program uses the Clang compiler to compile C++ code. Ensure it's installed.
2. **C++14 Support**: The program is written in C++14, so the compiler needs to support C++14 (Clang++ does by default).
3. OpenSSL install: You need to install OpenSSL to run the compile the code.
4. The directory is divide into client and server, it looks like this.

```
./client
├── make                # Makefile for building the client program
├── client.cpp          # Client source code
└── ./serverkey         # Directory for client-side server keys
        └── public.pem      # Public key for server communication
./server
├── make                # Makefile for building the server program
├── server.cpp          # Server source code
└── ./serverkey         # Directory for server-side keys
        ├── public.pem      # Public key for server
        └── private.pem     # Private key for server
```

```
./b11705032_part3
├── server          # The directory of server
└── client          # The directory of client
```

Compiling the Client Program

The program is written in C++ and consists of a file(client.cpp). To compile it with Clang++, run the following command in your terminal:

```
cd client
```

First ensure your make file is correct, it must contain something like the following to compile openssl library.

```
CXXFLAGS = -std=c++14 -I/opt/homebrew/Cellar/openssl@3/3.4.0/include/openssl -I/opt/homebrew/Ce
LDFLAGS = -L/opt/homebrew/Cellar/openssl@3/3.4.0/lib -lssl -lcrypto
```

and run make to compile if you really want to.

```
make
```

Compiling the Server Program

The program is written in C++ and consists of a file(client.cpp). To compile it with Clang++, run the following command in your terminal:

```
cd server
```

First ensure your make file is correct, it must contain something like the following to compile openssl library.

```
CXXFLAGS = -std=c++14 -I/opt/homebrew/Cellar/openssl@3/3.4.0/include/openssl -I/opt/homebrew/Ce
LDFLAGS = -L/opt/homebrew/Cellar/openssl@3/3.4.0/lib -lssl -lcrypto
```

and run make to compile if you really want to.

```
make
```

3. Executing the Program

Once compiled, you can run the executable by executing the following command:

Make sure you're under the correct directory ./client.

```
./client <ip> <portnum>
```

Make sure you're under the correct directory, ./server.

```
./server <portnum> -mode
```

Program Flow

1. Run the server
2. Run the client
3. Once the client connected, it will read the initial public key of server already written in file, so the Register and Login message can also be encrypted.
4. Client send message to server with encrypted message via the same socket.
5. Server decrypt the message and response to client.
6. Unfortunately, I wasn't able to handle server-to-client and client-to-client encryption in time. Hence, the rest of the program remain the same as part 1 and part 2 while all the functionality still works normally.

Error Handling

Any encryption or decryption fail will be response to client with warning so it can resend the message.