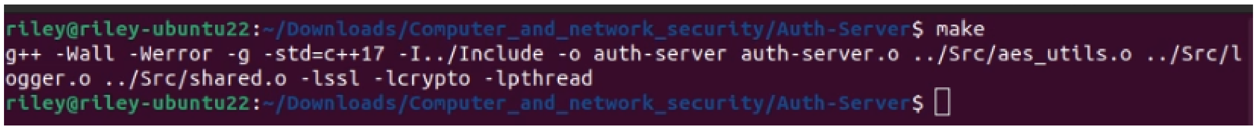
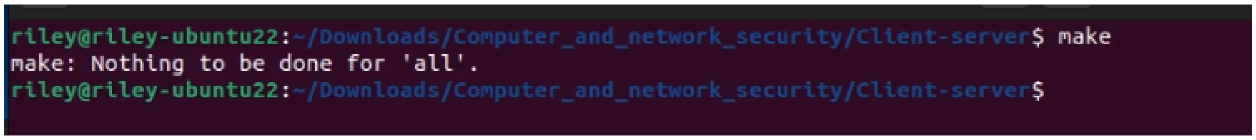
# Building and Running Instructions for KDC Chat Server

## Building the Executables:

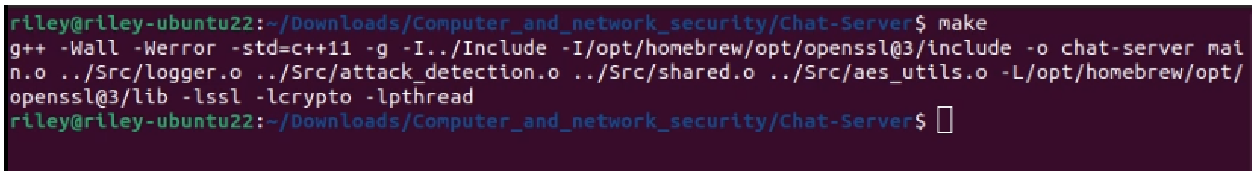
Within the code directory, go to the *Auth-Server, Client-Server,* and *Chat-Server* directories and run “make”.



```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Auth-Server$ make
g++ -Wall -Werror -g -std=c++17 -I../Include -o auth-server auth-server.o ../Src/aes_utils.o ../Src/l
ogger.o ../Src/shared.o -lssl -lcrypto -lpthread
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Auth-Server$
```

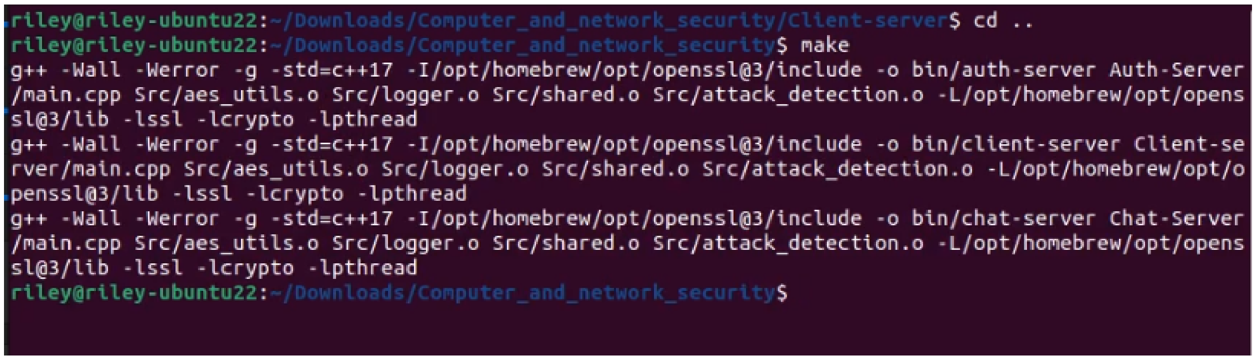


```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Client-server$ make
make: Nothing to be done for 'all'.
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Client-server$
```



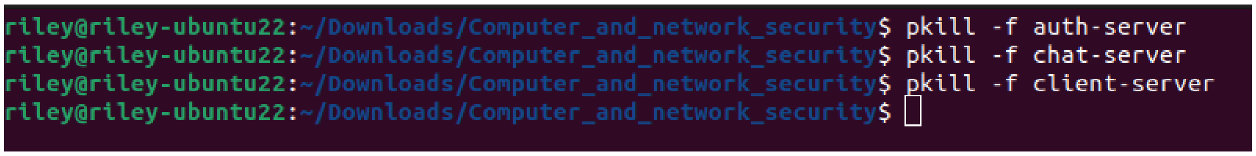
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Chat-Server$ make
g++ -Wall -Werror -std=c++11 -g -I../Include -I/opt/homebrew/opt/openssl@3/include -o chat-server mai
n.o ../Src/logger.o ../Src/attack_detection.o ../Src/shared.o ../Src/aes_utils.o -L/opt/homebrew/opt/
openssl@3/lib -lssl -lcrypto -lpthread
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Chat-Server$
```

Next go to the parent directory and run “make” on the parent directory



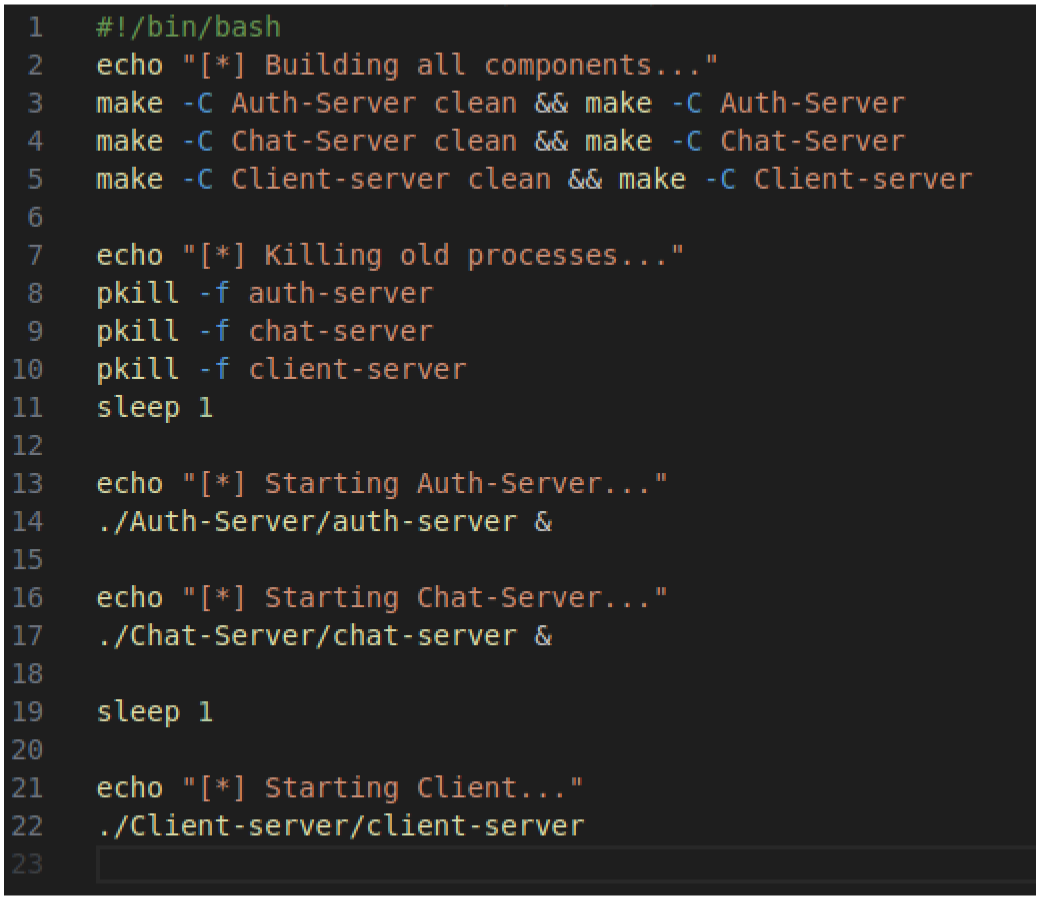
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Client-server$ cd ..
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$ make
g++ -Wall -Werror -g -std=c++17 -I/opt/homebrew/opt/openssl@3/include -o bin/auth-server Auth-Server
/main.cpp Src/aes_utils.o Src/logger.o Src/shared.o Src/attack_detection.o -L/opt/homebrew/opt/opens
sl@3/lib -lssl -lcrypto -lpthread
g++ -Wall -Werror -g -std=c++17 -I/opt/homebrew/opt/openssl@3/include -o bin/client-server Client-se
rver/main.cpp Src/aes_utils.o Src/logger.o Src/shared.o Src/attack_detection.o -L/opt/homebrew/opt/o
penssl@3/lib -lssl -lcrypto -lpthread
g++ -Wall -Werror -g -std=c++17 -I/opt/homebrew/opt/openssl@3/include -o bin/chat-server Chat-Server
/main.cpp Src/aes_utils.o Src/logger.o Src/shared.o Src/attack_detection.o -L/opt/homebrew/opt/opens
sl@3/lib -lssl -lcrypto -lpthread
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$
```

Then to make sure that no other instances of the servers are running on the ports, kill any lingering processes.



```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$ pkill -f auth-server
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$ pkill -f chat-server
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$ pkill -f client-server
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$
```

Similar functionality is shown in the provided “run_test.sh” file, but it does not allow you to run the servers in separate terminals.



```
#!/bin/bash
echo "[*] Building all components..."
make -C Auth-Server clean && make -C Auth-Server
make -C Chat-Server clean && make -C Chat-Server
make -C Client-server clean && make -C Client-server

echo "[*] Killing old processes..."
pkill -f auth-server
pkill -f chat-server
pkill -f client-server
sleep 1

echo "[*] Starting Auth-Server..."
./Auth-Server/auth-server &

echo "[*] Starting Chat-Server..."
./Chat-Server/chat-server &

sleep 1

echo "[*] Starting Client..."
./Client-server/client-server

```

## (Optional) Generating Keys:

As we are communicating over `localhost` the provided certificate is fine for demonstration purposes. However, you can regenerate one using the command below.

> [!NOTE]
> The certificate and key should be placed in the `./Certs` directory and named `as-c.pem` and `as-k.pem` respectively. You can modify the `./Auth-Server/auth-var.h` file to change where the keys are read from.

1. Run the following command

    ```
    openssl req -x509 -newkey rsa:4096 -keyout as-k.pem -out as-c.pem -nodes
    ```

    * Fill out information as desired, set common name to hostname/ip of the system we are using to resolve/connect to the server

You can generate a new chat server key (symmetric) by compiling and running the `./Util/keygen` program.

1. Enter into the `./Util` directory and run make

    ```
    cd ./Util && make
    ```

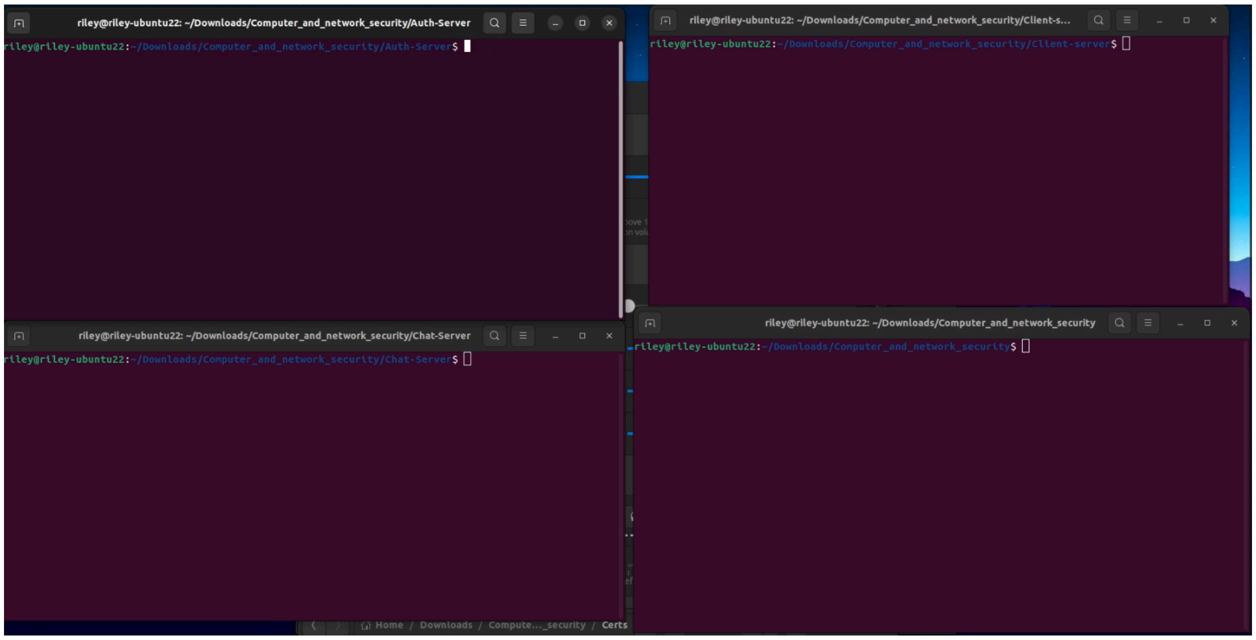
2. Run the `keygen` program

    ```
    ./keygen
    ```

3. Copy the resulting key to a file named `./Certs/chat_server_key.bin`

## Running the Program:

To run the program with each server output on a separate terminal, first open 4 terminal windows. One should be under the *Auth-Server* Directory, another under the *Chat-Server* Directory, and two in the *Client-Server* Directory.


riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security/Auth-Server
riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security/Client-s...
riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security/Chat-Server
riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security
Home / Downloads / Compute..._security / Certs

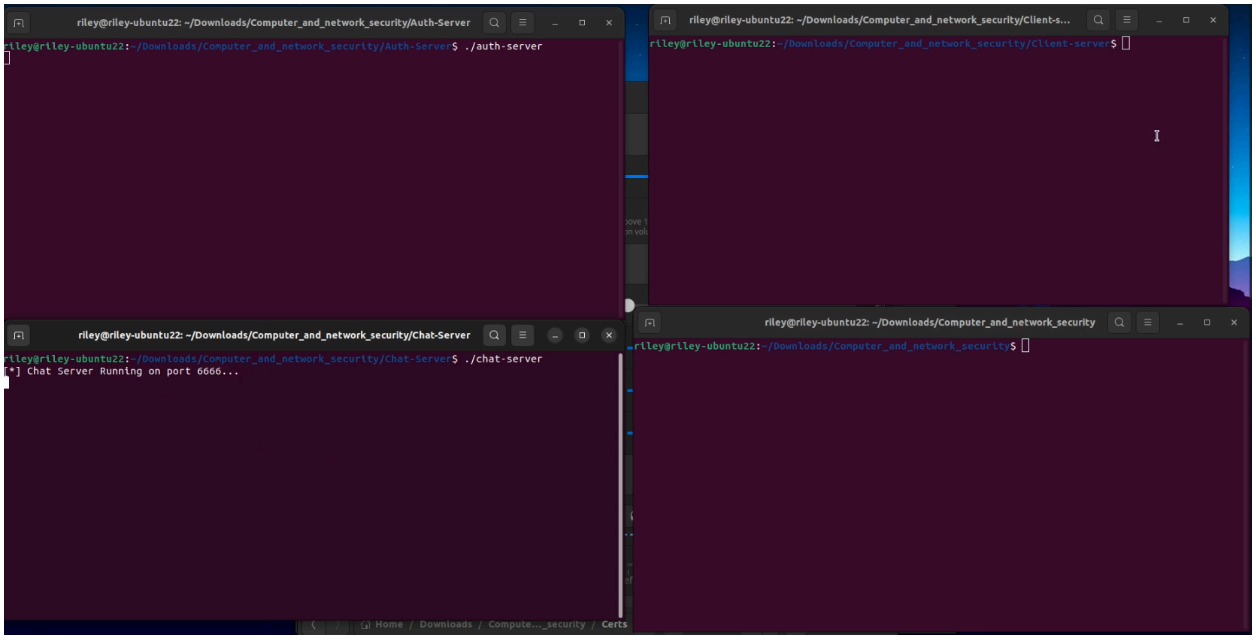
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Auth-Server$
```

```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Client-server$
```

```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Chat-Server$
```

```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$
```

Start by running "./auth-server" in the corresponding terminal window, then run "./chat-server" in its corresponding window. Output should be similar to the following image:


riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security/Auth-Server
riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security/Client-s...
riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security/Chat-Server
riley@riley-ubuntu22: ~/Downloads/Computer_and_network_security
Home / Downloads / Compute..._security / Certs

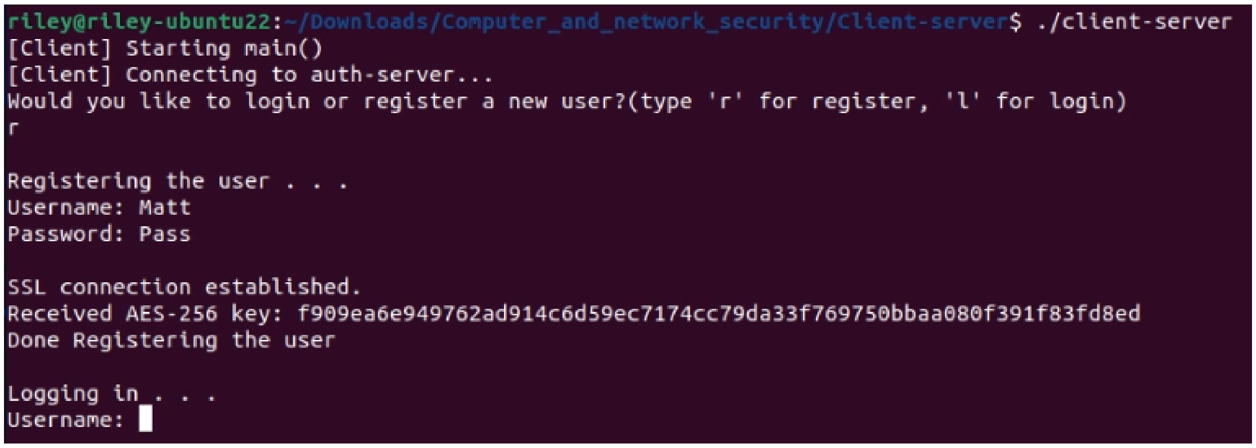
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Auth-Server$ ./auth-server
```

```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Client-server$
```

```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Chat-Server$ ./chat-server
[*] Chat Server Running on port 6666...
```

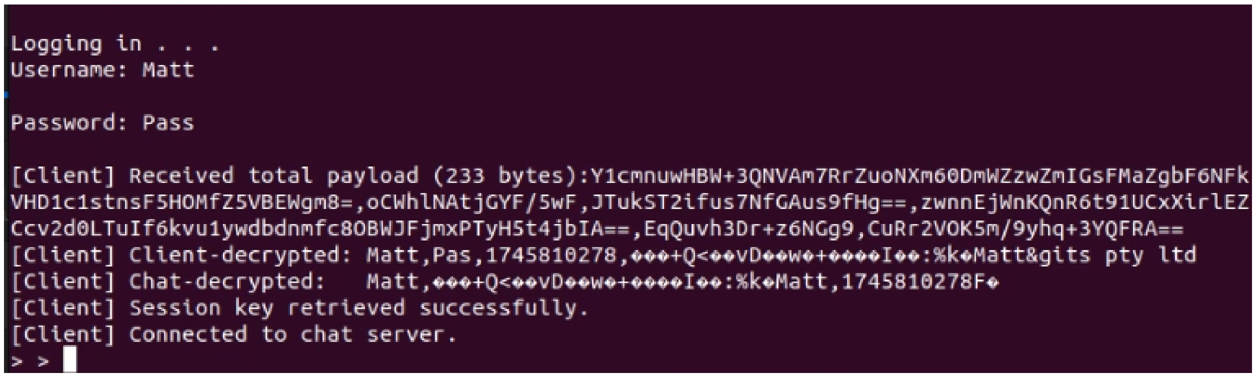
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security$
```

Then run “./client-server” in one of the client terminals. You will then be asked to register or login to the auth server. Select “r” to register and provide a name and password.



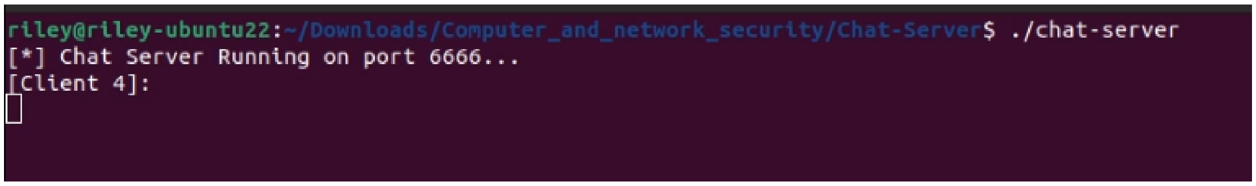
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Client-server$ ./client-server
[Client] Starting main()
[Client] Connecting to auth-server...
Would you like to login or register a new user?(type 'r' for register, 'l' for login)
r

Registering the user . . .
Username: Matt
Password: Pass

SSL connection established.
Received AES-256 key: f909ea6e949762ad914c6d59ec7174cc79da33f769750bbaa080f391f83fd8ed
Done Registering the user

Logging in . . .
Username: 
```

You will then be asked to log in. Do so with the username and password you registered with. This will show that a payload was received and that a connection to the chat server was established.



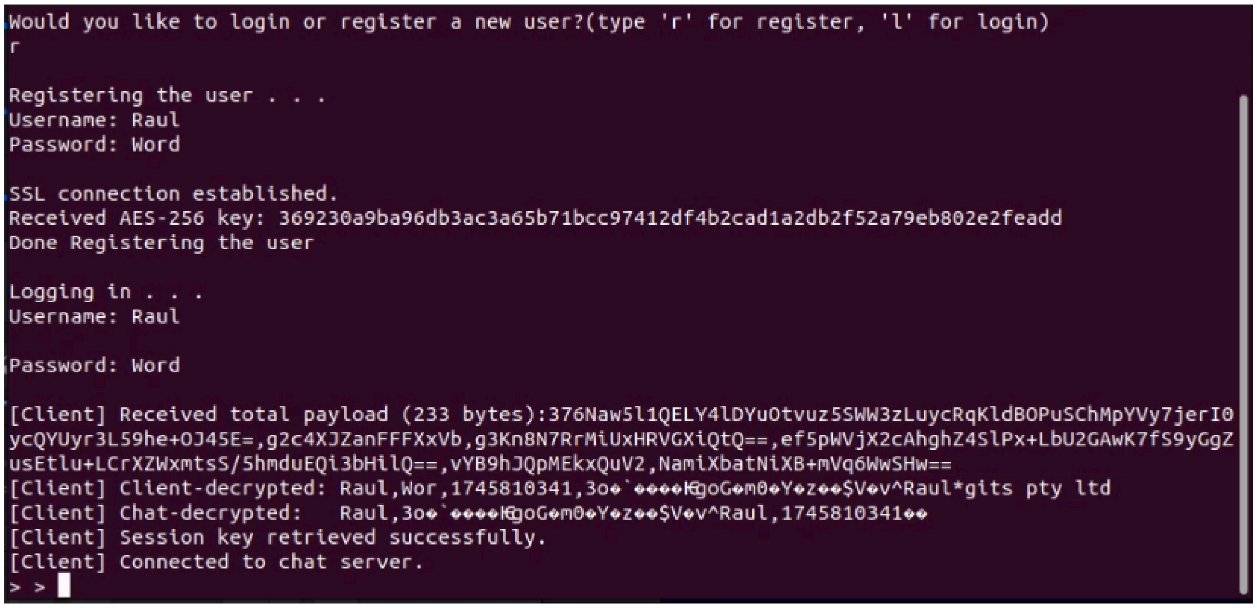
```
Logging in . . .
Username: Matt

Password: Pass

[Client] Received total payload (233 bytes):Y1cmnuwHBW+3QNVAm7RrZuoNXm60DmWZzwZmIGsFMaZgbF6NFk
VHD1c1stnsF5HOMfZ5VBEWgm8=,oCWhlNAtjGYF/5wF,JTukST2ifus7NfGAus9fHg==,zwnnEjWnKQnR6t91UCxXirlEZ
Ccv2d0LTuIf6kvu1ywdbdnmfc8OBWJFjmxPTyH5t4jbIA==,EqQuvh3Dr+z6NGg9,CuRr2VOK5m/9yhq+3YQFRA==
[Client] Client-decrypted: Matt,Pas,1745810278,���+Q<��vD��w�+����I��:%k�Matt&gits pty ltd
[Client] Chat-decrypted:   Matt,���+Q<��vD��w�+����I��:%k�Matt,1745810278F�
[Client] Session key retrieved successfully.
[Client] Connected to chat server.
> > 
```

On the chat-server terminal, a client message should appear.



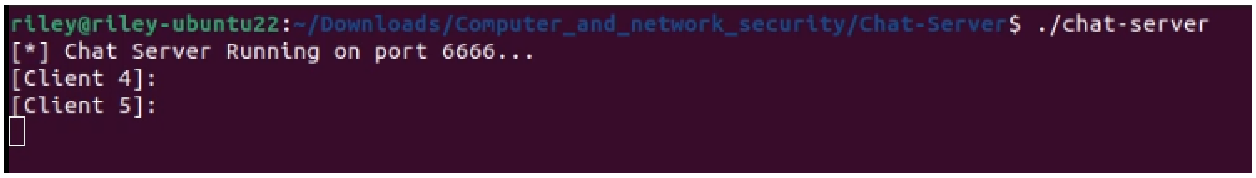
```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Chat-Server$ ./chat-server
[*] Chat Server Running on port 6666...
[Client 4]:
```

Do the same thing in the second client terminal window to register and log in as a second client.



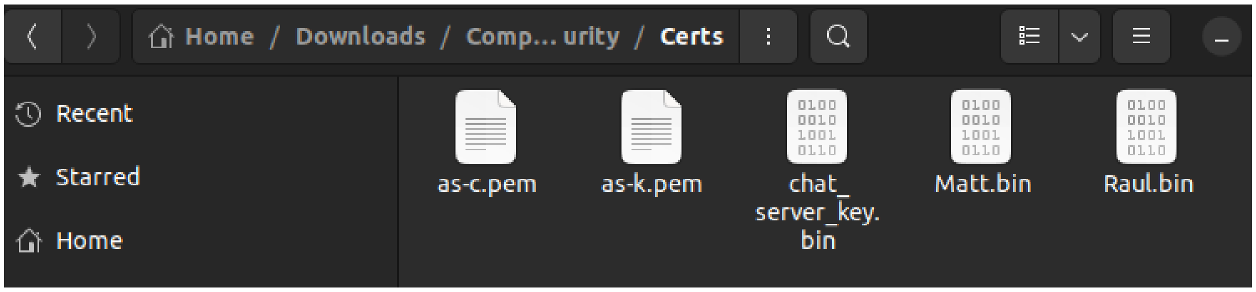
```
Would you like to login or register a new user?(type 'r' for register, 'l' for login)
r

Registering the user . . .
Username: Raul
Password: Word

SSL connection established.
Received AES-256 key: 369230a9ba96db3ac3a65b71bcc97412df4b2cad1a2db2f52a79eb802e2feadd
Done Registering the user

Logging in . . .
Username: Raul

Password: Word

[Client] Received total payload (233 bytes):376Naw5l1QELY4lDYuOtvuz5SWW3zLuycRqKldBOPuSChMpYVy7jerI0
ycQYUyr3L59he+OJ45E=,g2c4XJZanFFFXxVb,g3Kn8N7RrMiUxHRVGXiQtQ==,ef5pWVjX2cAhghZ4SlPx+LbU2GAwK7fS9yGgZ
usEtlu+LCrXZWxmtsS/5hmduEQi3bHilQ==,vYB9hJQpMEkxQuV2,NamiXbatNiXB+mVq6WwSHw==
[Client] Client-decrypted: Raul,Wor,1745810341,3o◆`◆◆◆◆K̶goG◆m0◆Y◆z◆◆$V◆v^Raul*gits pty ltd
[Client] Chat-decrypted:   Raul,3o◆`◆◆◆◆K̶goG◆m0◆Y◆z◆◆$V◆v^Raul,1745810341◆◆
[Client] Session key retrieved successfully.
[Client] Connected to chat server.
> > 
```

The client should now have two client prompts.

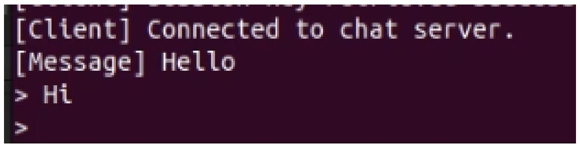


```
riley@riley-ubuntu22:~/Downloads/Computer_and_network_security/Chat-Server$ ./chat-server
[*] Chat Server Running on port 6666...
[Client 4]:
[Client 5]:
```

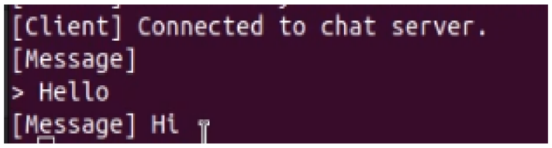
If you look at the certs directory, two new keys will appear for the two new users. Both of these were stored by the authentication server.



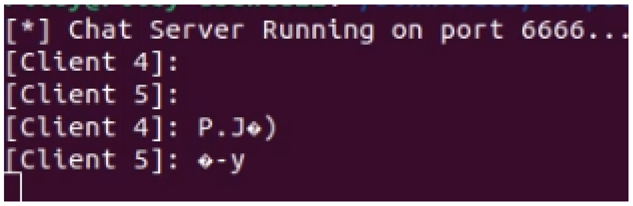
Both clients can now type messages to each other. They will appear in plaintext for the clients and show up encrypted at the chat server.



```
[Client] Connected to chat server.
[Message] Hello
> Hi
>
```



```
[Client] Connected to chat server.
[Message]
> Hello
[Message] Hi
```



```
[*] Chat Server Running on port 6666...
[Client 4]:
[Client 5]:
[Client 4]: P.J�)
[Client 5]: �-y
```

**Congrats!** You have now used the KDC and chat server to have encrypted communications.