

# Encrypted Chat App

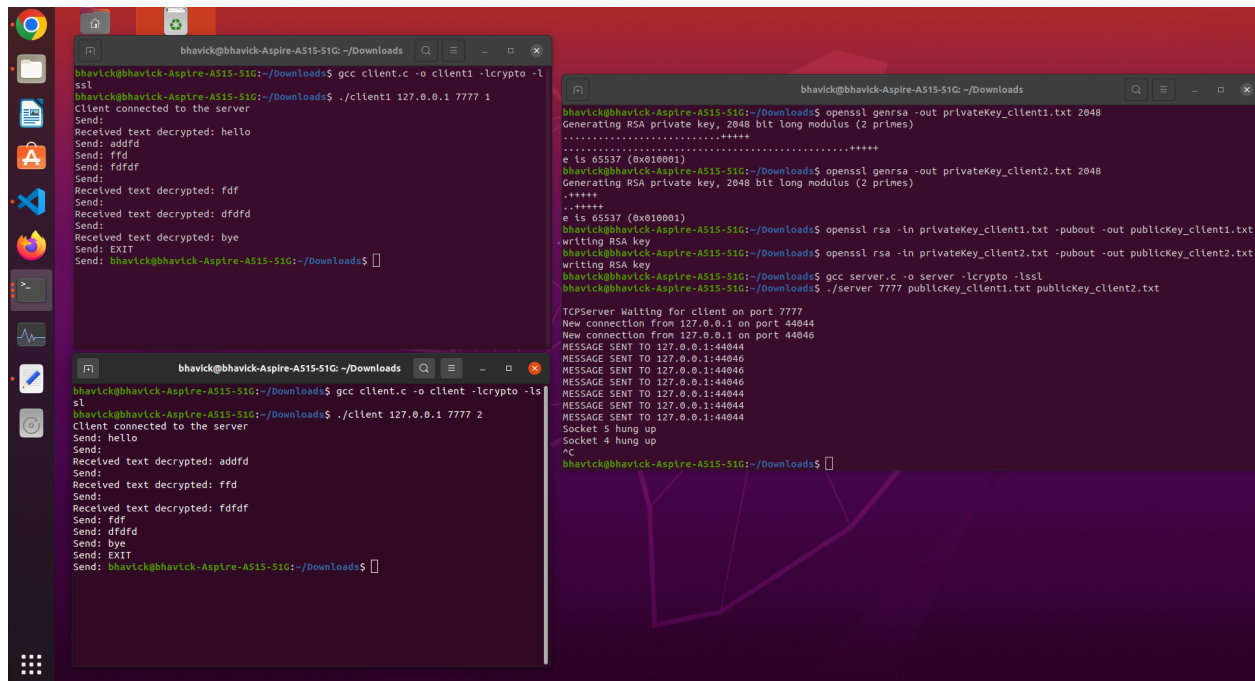
I have developed a multithreaded TCP-based chat program written in C that allows numerous users to connect to a centralized server and transmit messages to one another. Messages are encrypted by the server using the public key of the receiver client and decrypted by the receiver client using its private key.

Working :

There are two clients and a server in the application. The two clients communicate with each other via the server. When one client sends a message to the server, the server stores the message as a text file in the server, where the filename is `clientid_timestamp.txt`. The server then encrypts and sends the file to the other client that decrypts the file and displays its content on its terminal.

- The server, as a command-line argument, accepts the binding port number, and the public key of each of the two clients.
- The two clients accept the server's IP address and port number as command-line arguments.
- After connecting to the server, each of the clients keeps on reading a line from the standard input.
- The first client sends the line read from the standard input to the server.
- The server writes a new line it receives from the client in a new text file (Filename: `<clientid>_<timestamp>.txt`). It then encrypts the file and sends the encrypted text file to the second client.
- The server displays "MESSAGE SENT TO <IP ADDRESS>:<PORT NO>" on its terminal.
- The second client, on receiving the encrypted file, decrypts the file using the private key that you can put directly in the program, reads the content of the file, and displays the content on the terminal.
- Both clients exit when either client sends "EXIT" from the terminal.

## Demonstration :



```
bhavick@bhavick-Aspire-A515-S1G: ~/Downloads$ gcc client.c -o client1 -lcrypto -lssl
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ ./client1 127.0.0.1 7777 1
Client connected to the server
Send:
Received text decrypted: hello
Send: addfd
Send: ffd
Send: fdffd
Send:
Received text decrypted: fdff
Send:
Received text decrypted: fdffd
Send:
Received text decrypted: bye
Send: EXIT
Send: bhavick@bhavick-Aspire-A515-S1G:~/Downloads$

bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ gcc client.c -o client -lcrypto -lssl
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ ./client 127.0.0.1 7777 2
Client connected to the server
Send: hello
Send:
Received text decrypted: addfd
Send:
Received text decrypted: ffd
Send:
Received text decrypted: fdffd
Send: fdf
Send: fdffd
Send: bye
Send: EXIT
Send: bhavick@bhavick-Aspire-A515-S1G:~/Downloads$

bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ openssl genrsa -out privateKey_client1.txt 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ openssl genrsa -out privateKey_client2.txt 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ openssl rsa -in privateKey_client1.txt -pubout -out publicKey_client1.txt
writing RSA key
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ openssl rsa -in privateKey_client2.txt -pubout -out publicKey_client2.txt
writing RSA key
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ gcc server.c -o server -lcrypto -lssl
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$ ./server 7777 publicKey_client1.txt publicKey_client2.txt

TCPServer Waiting for client on port 7777
New connection from 127.0.0.1 on port 44044
New connection from 127.0.0.1 on port 44046
MESSAGE SENT TO 127.0.0.1:44044
MESSAGE SENT TO 127.0.0.1:44046
MESSAGE SENT TO 127.0.0.1:44046
MESSAGE SENT TO 127.0.0.1:44046
MESSAGE SENT TO 127.0.0.1:44044
MESSAGE SENT TO 127.0.0.1:44044
Socket 5 hung up
Socket 4 hung up
^C
bhavick@bhavick-Aspire-A515-S1G:~/Downloads$
```