ROHIT GARG
2018A7PS0193G

COMPUTER NETWORKS
CS F303
COMPRE LAB README.PDF

PREREQUISITE
openssl library (sudo apt-get install libssl-dev)

ZIP CONTENTS
* client.c - client program
* server.c - server program
* private1.pem - file containing private key for client 1
* public1.pem - file containing corresponding public key for client 1
* private2.pem - file containing private key for client 2
* public2.pem - file containing corresponding public key for client 2
* README.txt - readme
* README.pdf - pdf containing instructions to run and compile program and showing sample run for a chat session

COMPILATION
gcc -o c client.c -lssl -lcrypto
gcc -o s server.c

EXECUTION
[1] Run Server (**./s 8080**) - [Replace 8080 with port number]

[2] In a separate terminal, Run first client (**./c 127.0.0.1 8080 private1.pem public2.pem**)

->[Replace 8080 with port number given to server. Replace private1.pem and public2.pem with keys for private key 1 and public key 2 respectively]

[3] In a third terminal, Run second client (**./c 127.0.0.1 8080 private2.pem public1.pem**)

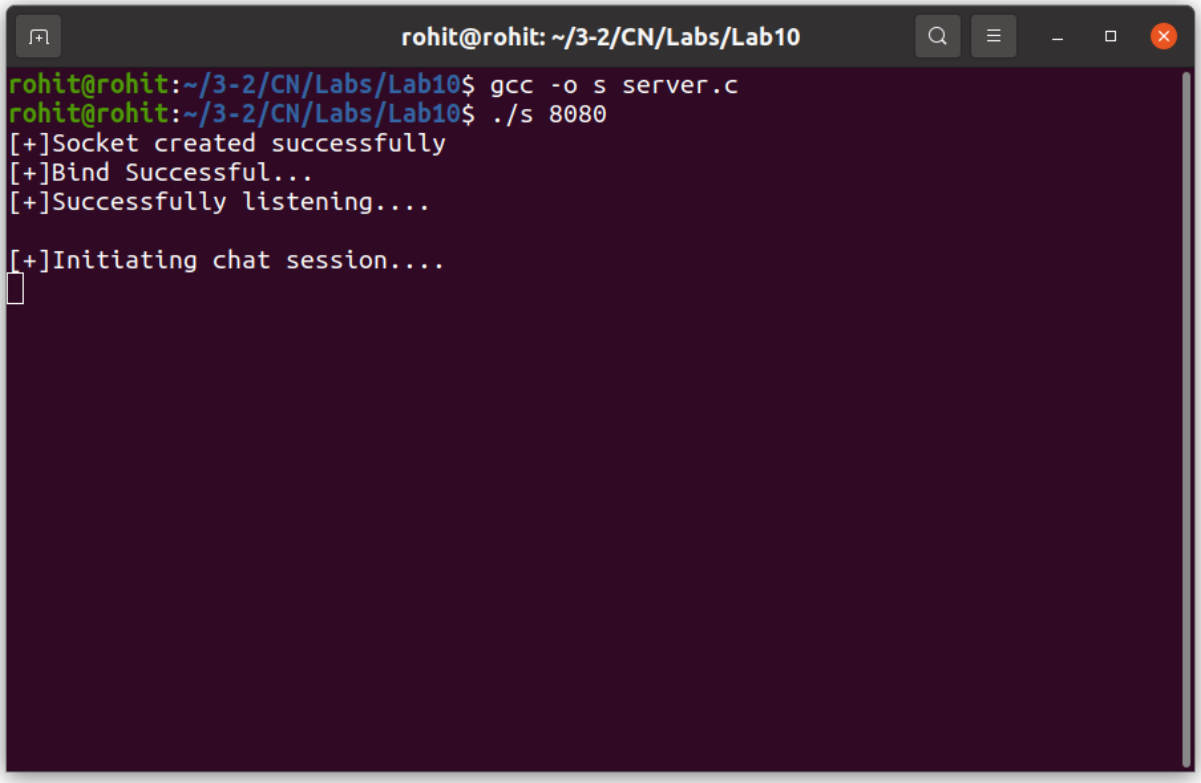
->[Replace 8080 with port number given to server. Replace private2.pem and public1.pem with keys for private key 2 and public key 1 respectively]

Note: This PDF is meant to show the sample run of the program along with commands issued at the terminal windows. Inputs at terminal are highlighted in bold.

The server as a command line argument accepts the port number to which it should bind:-

<<Terminal Window 1>> **gcc -o s server.c**
<<Terminal Window 1>> **./s 8080**

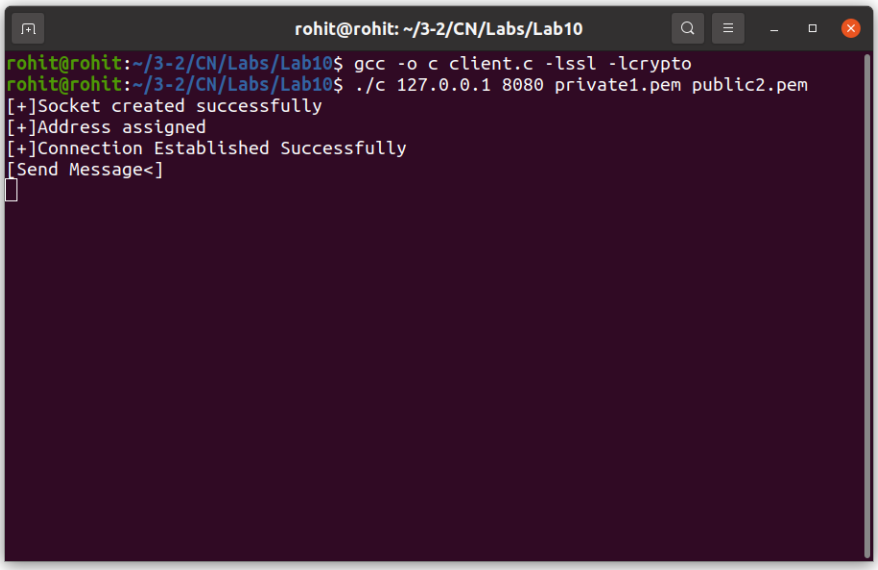
Terminal Window 1: server program execution



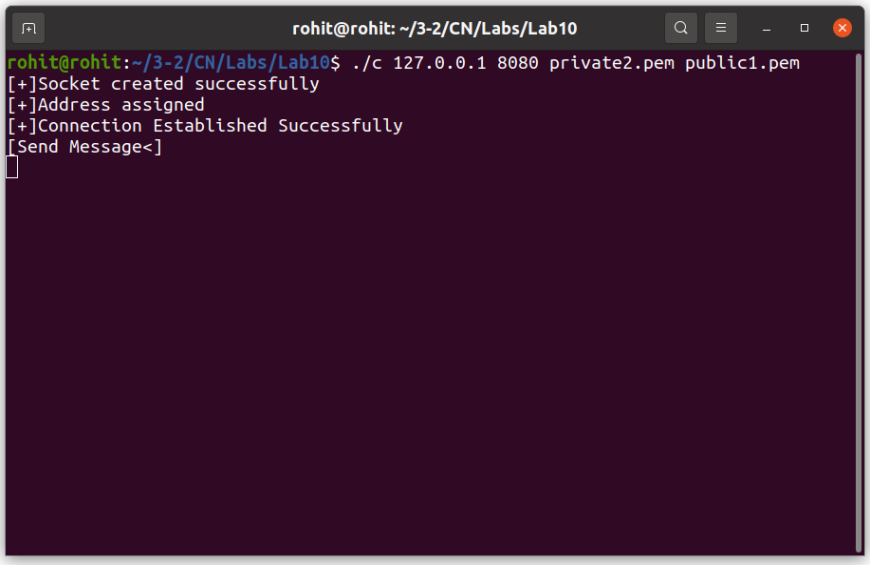
Each client, as command line arguments, accepts the IP address at which it will find the server, the port number of the server at that IP address, the filename containing its private key, and the filename containing the other client's public key.

<<Terminal Window 2>> **gcc -o c client.c -lssl -lcrypto**
<<Terminal Window 2>> **./c 127.0.0.1 8080 private1.pem public2.pem**
<<Terminal Window 3>> **./c 127.0.0.1 8080 private2.pem public1.pem**

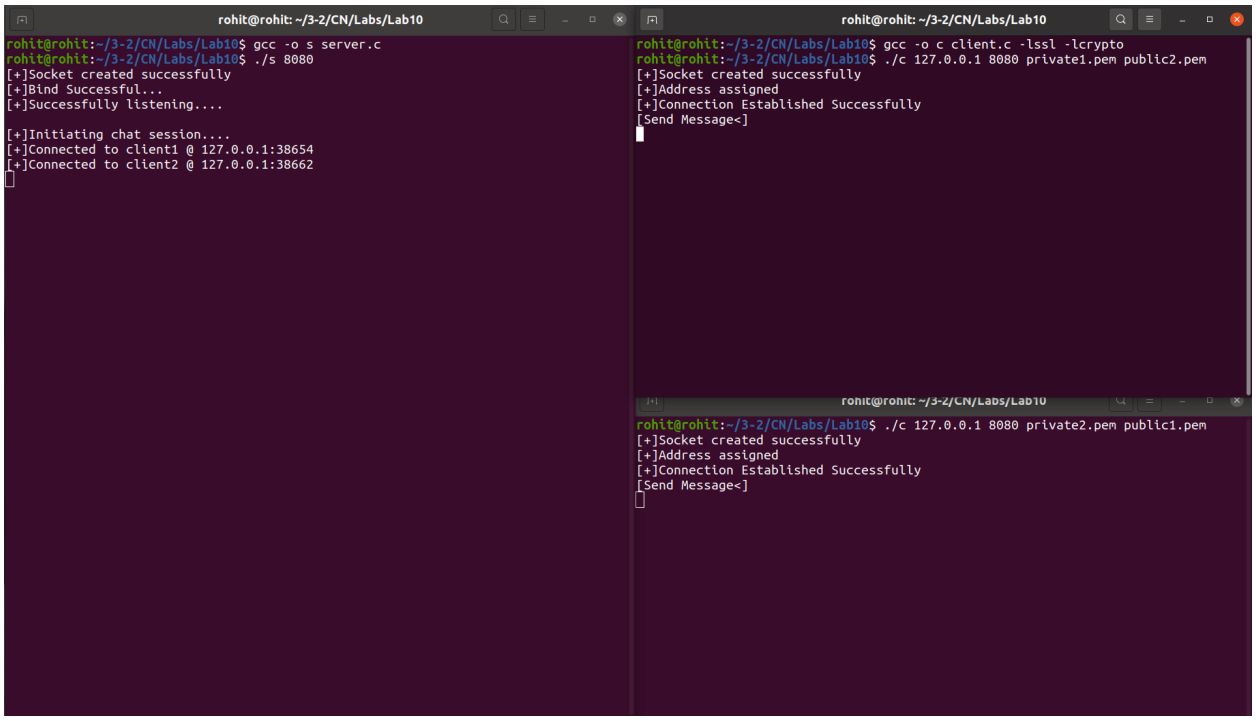
Terminal Window 2: client 1 execution



Terminal Window 3: client 2 execution

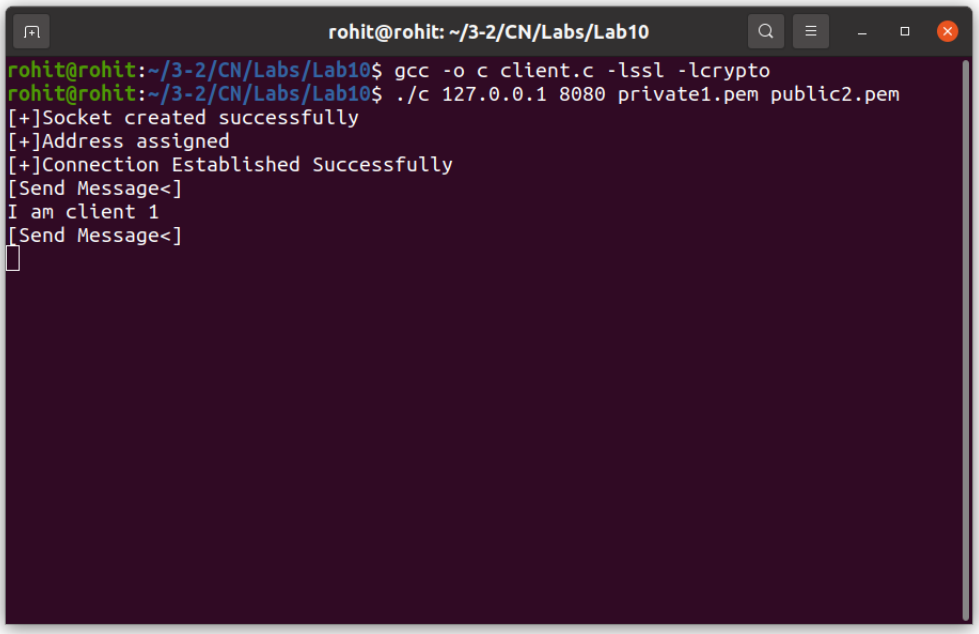


Both the clients are now connected to the server and ready to chat. The terminal windows look like this:-



After connecting to the server, the client keeps on reading a line from the standard input. We send the message “I am client 1” from client1’s side. It encrypts the line and sends it to the server. The server forwards the message received from one client to another.

<<Terminal Window 2>> [Send Message<]
<<Terminal Window 2>> **I am client 1**

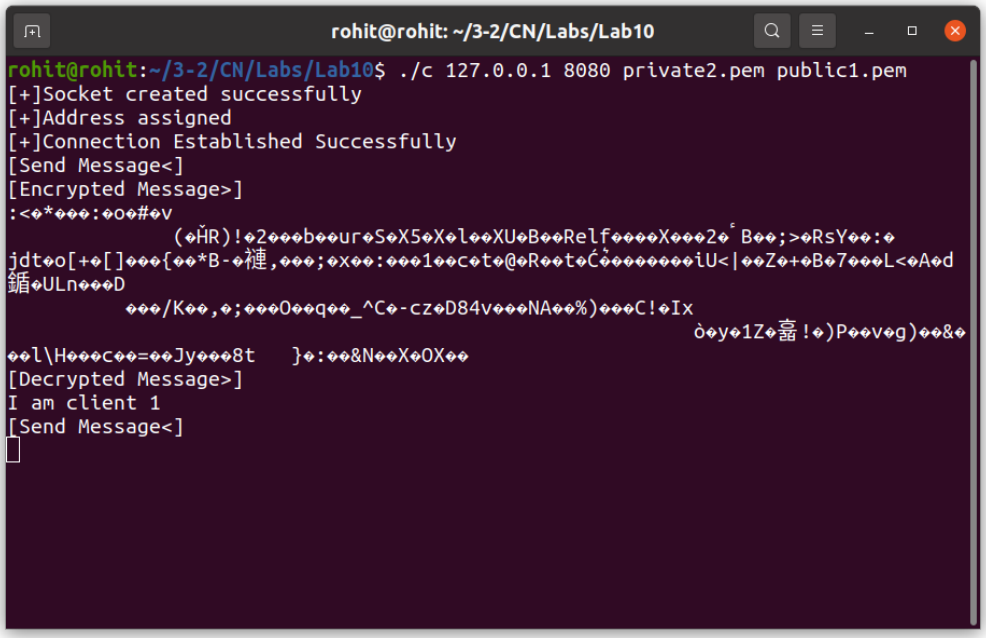


When a client receives a message from the other client, it decrypts and displays both ciphertext and plaintext. Output at terminal 3 for client 2 looks like this:

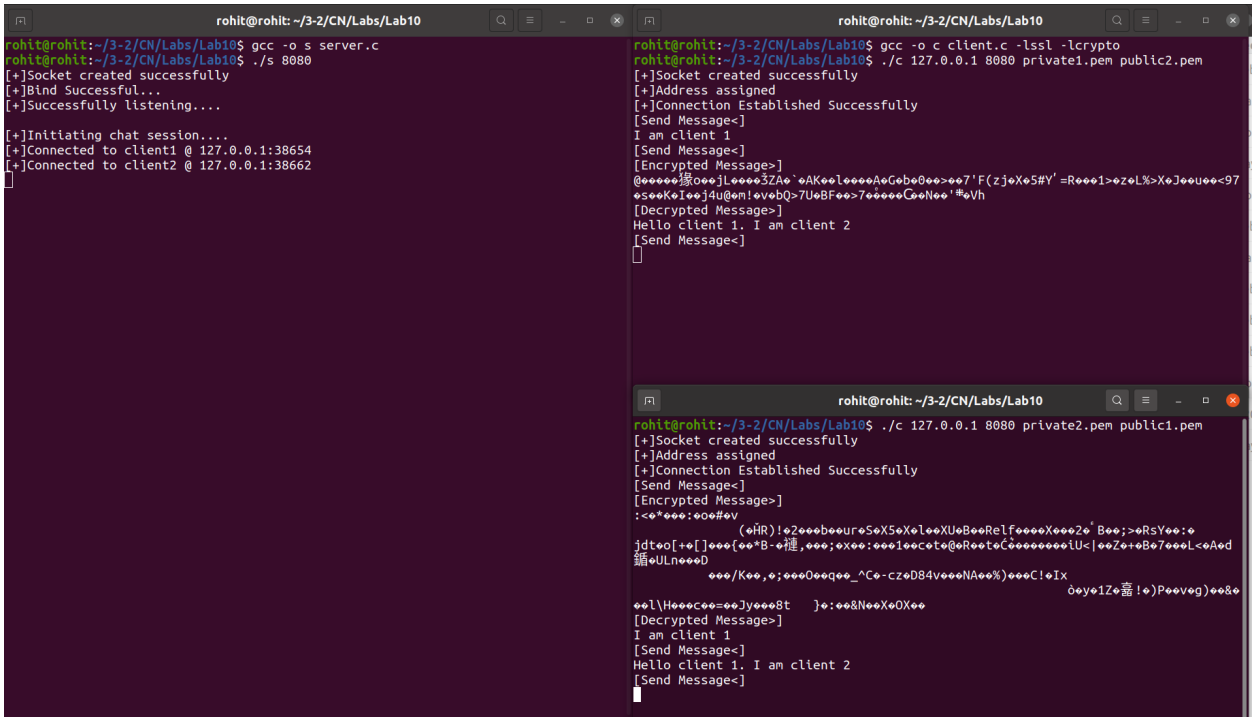
<<Terminal Window 3>> [Encrypted Message>]
<<Terminal Window 3>> :<*:o#v

(ĤR)!2b ur S X5 X l XU B Relf X 2 B ;> RsY
:
jdt o[+ [] { *B- 褳,
;x :1 c t @ R t Ć iU<| Z + B 7 L< A d鍎 ULn D
/K , ; O q _^C -cz D84v NA %) C! Ix

ò y 1Z 嚞!)P v g) & l\H c = Jy 8t
}: &N X OX
<<Terminal Window 3>> [Decrypted Message>]
<<Terminal Window 3>> I am client 1



In this way, client to client communication via server can take place. Here is the screenshot of the same.



Both the clients exit if the user types "exit" in any client. Server concludes the previous chat session and now stays around for two new connections by two clients.

<<Terminal Window 3>> **exit**
or
<<Terminal Window 2>> **exit**