CHAT APPLICATION WITH END TO END ENCRYPTION

By

SHAIVAL RAJAN SHAH
15BCE110
KAUSHAL THAKKAR
15BCE125

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

CHAT APPLICATION WITH END TO END ENCRYPTION

MINI PROJECT - I

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Engineering

By

SHAIVAL RAJAN SHAH
15BCE110
KAUSHAL THAKKAR
15BCE125

Guided By
Dr. VIJAY UKANI
DEPARTMENT OF COMPUTER ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

CERTIFICATE

This is to certify that the project entitled "**CHAT APPLICATION WITH END TO END ENCRYPTION**" submitted by **SHAIVAL RAJAN SHAH (15BCE110), KAUSHAL THAKKAR (15BCE125)**, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. Vijay Ukani
Assistant professor
Department of Computer Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

Dr. Sanjay Garg
Dept. of Computer Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

On the very outset of this report, we take this opportunity to express our profound gratitude and heartfelt obligation towards our guide; **Dr. Vijay Ukani**, who helped us in this endeavor and took keen interest on our project topic “**Chat Application with end to end Encryption**”. Without his active guidance, constant support, and encouragement we would not able to complete this project. We extend our gratitude to Nirma University for giving us this Opportunity. At last but not the least we give special thanks to our friends who helped us in this project.

ABSTRACT

In this technologically advance world communication has become lot faster. Person can communication to another person who is living at another corner of earth within a matter of seconds. Social Networking sites have been used by millions of users to communicate with each other. Applications like WhatsApp, Hike etc. have been much successful in recent times. Chatting has become faster way of communication. Looking forward to this we have develop a chat application in which users can communicate with each other. This application can be used in our university for communication of all the people within the campus of our university.

CONTENTS

LIST OF FIGURES

Chapter 1: Introduction

1.1 General information about our application

Our Application is java based project that is it is developed in java language. Socket Programming in java is used to build this application. TCP (Transmission Control Protocol) is used for sending and receiving packet. GUI of client window is developed. Users have to give their name and ip address of server. Multiple Users can be connected and chat with each other. User have to select the name of user to send message to him/her. Client window is close to disconnect from the server. This application needs a LAN connection between 2 devices to communicate.

1.2 Objective of this project

In recent times chat applications are used more for faster communication. By developing this project, we learnt how to do socket programming in java. Exploring different classes of socket programming and implement them. This chat app can be integrated to the server of Nirma University and all the students and faculties can connect to it and can communicated with each other. Encryption techniques are studied to implement end to end encryption.

1.3 Scope of this work

There is lot of scope in Socket programming. Now, whole world is connected to network. Millions of GBs of data is send and received every minute. Chat Application is used immensely now a days and it the faster way of communication.

Chapter 2: Connection and Disconnection of Users

2.1 **Connection of User**

- Connection is done by socket and Server is created by ServerSocket. When new client is connected, his/her name, id and socket is transferred to the server and connection is done between server and client.
- We have created a class named ClientThread, that creates separate thread for all clients for sending and receiving message.
- There is method name connection which returns Boolean value of whether the user is connected or not.
- In this method socket is created using ip and port of a server.
- Uniqueid of client is generated.
- A packet is sent to the server giving information about user name and id and other Boolean variable which is true when user disconnects.
- If there is no problem in connecting this method returns true value otherwise false.

And at server side for accepting the socket and make new thread for separate client getclient() method is created which creates new thread of client connected. Different threads are created of different sockets of clients. Each client has unique socket and all the sockets are connected to ServerSocket. This thread runs till the user is disconnected. And when new client is connected then name and id is saved in the list at server side and send to all connected clients for adding in the online user list.

At clientThread:

There is separate condition if the information about new client is received. Here if the name and id of client is stored in clientList. Name and id of client is given to every user and that particular user gets all the name and id of clients already connected. This process is done sending packet to all the clients connected to server giving names of all users connected. Boolean value remove is set to false as new client is connected.

When a user is connected first it is checked that his/her name matches with a current user. If yes then samename value is set to true and user has to enter different name again. If no then name of that user is added in clientList and send to every other user through socket. samename is Boolean value use for checking if the name of new client is matching with the user already connected to server.

2.2 **Disconnection of User**

When user clicks on the close button we have to disconnect that user from the server. For disconnecting the user, First we have to send the name and id to all other user for removing that user from the list of online user and then we close the connection by closing the socket at both server and client side.

Here for doing this process we have add windowlistener to client window. When the windows is closing a packet is sent sending message “@EXIT” to server. Receive thread is stopped and socket of that client is closed.

At clientThread:

When server gets “@EXIT” message inputstream is closed of that clientThread. That client name is searched in the client list and remove from it. The name of client is send to all user by sending packet with Boolean value remove equal to true. ClientThread of that particular client is stopped and socket connection is closed.

Here user is disconnected from server and his/her name is removed from ClientList and his/her name is send to every user so that his/her name can be removed from the online user list.

Chapter 3: GUI for Login and Client Window

3.1 **For Login**

In the Login GUI there is a text field for name and other for IP Address of server. When user enters name and valid IP of the server then new client window opens. If name is same with the names among online users than user has to enter different name again.

Login Window

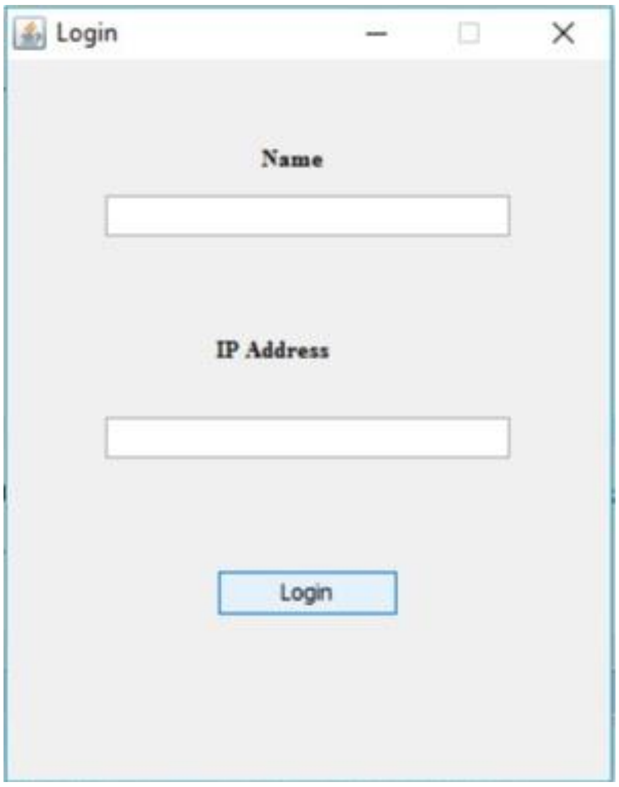

Fig. 3.1

3.2 **For Client**

Client Window

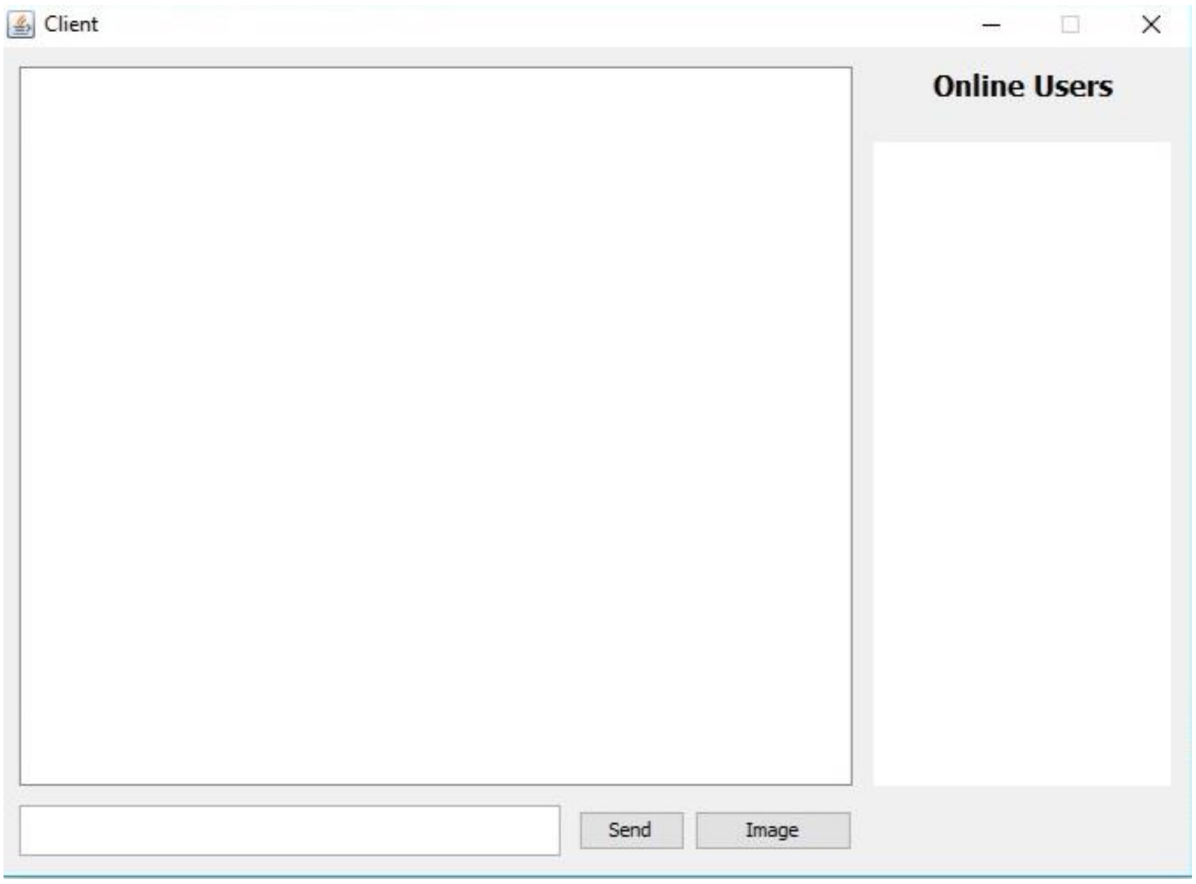

Fig. 3.2

In the client window there is a text field for typing the message and a JTextPane for display the sent and received message. There is a JList for online users. JTextPane can display message as well as image. 2 buttons namely send and image which are used to send message and image to other client online respectively. Jlist shows all the names of online users and user have to select one of those to send message to him/her.

Chapter 4: The Packet Class

4.1 **Introduction**

Only and only packet class is responsible for sending and receiving content from client to server or server to client. Object of packet class is send and receive at both sides. Information about connection, disconnection of users or sending message or image for all this object of packet is send at both sides. Packet class is serializable class. Only object of serializable class can be send through socket and the content inside packet class must be serializable content. Flow of packet send is from client to server and back to client from server.

4.2 **Contents of Packet Class**

- String msg = **null**;
  msg is used to store message send by sender
- int id;
  id is the uniqueId generated for user.
- String rname=**null**;
  rname is the name of receiver.
- String sname=**null**;
  sname is the name of sender.
- String imgname=**null**;
  imgname is the name of image send by the sender.
- byte[] img=**null**;
  img is the byte array of image send. Image is converted into byte array then send to receiver.
- String imgtype=**null**;
  Imgtype is the type of image send. For example png, jpeg, etc.
- boolean remove;
  remove is boolean value which becomes true when user is disconnected.
- boolean samename = **false**;

samename is boolean value which becomes true when name of user connected is already connected to server.

4.3 **Constructors of Packet Class**

```
Packet(byte[] img,String imgname,String imgtype,String rname,String sname){
            this.img = img;
            this.imgname = imgname;
            this.imgtype=imgtype;
            this.rname=rname;
            this.sname=sname;
}
```

The above constructor is used when image send by the sender.

```
Packet(String msg){
            this.msg = msg;
}
```

The above constructor is used when message is send by sender.

```
Packet(int id,String name,boolean remove){
            this.id=id;
            this.rname=name;
            this.remove = remove;
}
```

The above constructor is used when any user disconnects from server.

```
Packet(boolean samename){
            this.samename = samename;
}
```

The above constructor is used when same named user is connected.

Chapter 5: The ClientThread Class

ClientThread Class is used to create threads of different users when connected at server side. Thread created keeps running until the user is disconnected. This thread is responsible of sending message or image to receiver and sending the names of all clients connected to the server. An static ArrayList of clients is created to keep track of all the clients

5.1 **Receiving Message**

If message received is different from “@EXIT” then message is received and if the user is explicitly writes this message. When message is received name of user is appended to it. When server receives the message it first stores the receivers name which is appended with the message in a string. New socket is created. Name of the receiver is searched in the clientList and socket of that client is fetched. Now new outputstream is created and packet is sent to that socket which is created before so that message is sent to receiver only. If message received is different from “@EXIT” then message is send to receiver through socket.

5.2 **Receiving Image**

Image is received in the form of byte array. When packet is received containing byte array new byte array is created in which received array is stored. Inputstream is created and name of receiver is also received. Name of receiver is searched in the clientList and socket is created and socket of that client is stores in this socket. Now packet is sent to receiver containing byte array of image through socket. Image is received in byte array and send in byte array only selecting socket of receiver and send to it.

Chapter 6: Encryption and decryption of message

6.1 **Encryption**

Here AES encryption is used. Java has predefined class and method for AES encryption. Cipher class encrypts the message. Here 128 bit key is selected and message is encrypted using it.

Key: Ba212345Ba212345

Above given key is use to encrypt message.

Cipher gets the method of AES encryption and encrypts the message.

Byte array of encrypted message is created.

Encrypted byte array is stored in StringBuilder and final encrypted message is returned.

6.2 **Decryption**

Decryption is done using same 128 key. Cipher class is used to decrypt the message. Byte array is created of message received and is decrypt using decrypt method of AES and final decrypted message is returned.

Chapter 7: Summary and Conclusion

7.1 **Summary**

In this report, we have explained everything about our application. Source code and explanation of it is provided. Different methods are shown. Sending and Receiving message from particular client and sending image is done in our project. Track of all users have been taken whether they connect or disconnect. Only one class object is send and received through socket. So this application can easily add new features to it as only packet class needs to change and some minor changes to other class.

7.2 **Conclusion**

- By doing this project we have learnt socket programming.
- Different methods are learnt in socket programming.
- Serialization of class has been studied.
- Encryption technique is studied.

REFERENCES

- **Java Network Programming, 4th Edition**

  By: Elliotte Rusty Harold
  Publisher: O'Reilly Media

- **TCP/IP Sockets in Java: Practical Guide for Programmers**

  By: Kenneth L. Calvert
  Michael J. Donahoo

Appendix A – list of useful websites

- https://www.tutorialspoint.com/java/java_networking.htm
- https://www.javatpoint.com/socket-programming
- http://java2all.com/technology/network-programming/socket-programming/socket-programming-in-java