An Online Verification Based on a Real-time Photo by Third Party Authentication

Xiaohong Zhang

Abstract

Currently, most social network users depend on passwords for authentication when logging into their accounts. But since the increasing growth of Internet Technology, we are registering more and more accounts than ever, as a result, we need to set a lot of passwords. To be a robust password against attacks, it should be long and random enough, but it is hard to memorize. So we are inspired to try a new idea. The third party authentication, which needs a user to take and send a real-time face image to online friends list asked for a authentication rather than memorizing and typing alphanumeric password strings, guarantees a safer and more convenient way of verification. In this report we show our idea about authentication protocol Frame and describe the communication process.

1. Introduction

Users frequently tend to reuse their passwords when registering various online services. It seems convenient but brings high risks to the security of the user's account information. And at many times, a very cautious user may forget the different passwords or be confused that which network or website a password is associated with.

In this paper, we describe a solution that allows a user log into a social network safely with real-time photo rather than the password. The innovation is to get authentication from third party. People are just required to input their constant username and take a face photo then send to the relevant social network online friends. We discuss the process of the communication system and do a simulation in the local area network.

The rest of the paper is organized as follows: section 2 gives a brief overview about relative works. In section 3, we propose our idea and give a protocol frame in section 4. In section 5 and 6, we mainly discuss the detailed skills of authentication. At last we analysis our simulation result and discuss the strengths and weaknesses.

2. Related works

- Token-based systems

Token is a piece of data that obtained from a physical device such as smartcards or electronic key for authentication purpose. Token-based system offers access to a special resource for a time of period in the remote site. A typical process would work as a communication between a remote site and a base site. A user wishes to provide the temporary authorization to the remote site to read his FOAF file at the base site. But it is vulnerable to intercept attacks that an intruder steals user's records by acting as a proxy between the user and the authentication device without the information of the user [1]. Thus as an alternative, a real-time face graph is introduced to solve the mentioned limitations.

- Recognition-based systems

In recognition-based systems, a group of images are presented in a graphical user interface and an accepted authentication works by a correct image being selected in a particular order. Some examples of recognition-based system are Awase-E system [2], AuthentiGraph [3], and Passfaces system[4]. A graphical password is easier than a texted password for most people to remember, but it is still not completely secure. It needs several rounds of image recognition for authentication to provide a reasonably large password space, which is tedious [5]. Also, it is obvious that recognition-based systems are vulnerable to replay attack and mouse tracking because of the use of a fixed image as a password.

- Third party – based systems

The external system performs the authentication and passes a token back to the web application through the request headers. The interaction is based on opening a log in window in our online application interface which is hosted by the 3rd party's servers such as Amazon, Facebook, Twitter, etc. Once a user logs in, an authentication token is sent back from the 3rd party, which is saved in our online site. For example, Facebook [6] requires users to register with a verification service by receiving information about the user. Many verification options can be selected such as mobile phone number or confirmation of a registered online email account. When a user input the PIN code received in the message or click a website in the email successfully, they can log in. But this method is actually a self authorization by the tool of a third party, not involving social network friends' authorization.

3. The problem and motivation

We mentioned the two main password problems are hard to remember and should be secure. Meeting both of these requirements is almost impossible for users. In order to memorize all kinds of social networks passwords users always choose the same password, which offers a great chance for hackers or some illegal websites to steal passwords. And once a password has been chosen and learned the user must be able to recall it to log in., but too long time offline usually makes the user forget the password.

If we can skip the password but still guarantee the security, which will be an ideal scenario. The real-time face image can do it. If someone steals your phone it still makes no sense because the verification system can only generate fresh image by taking photos. There is no way to get one from the gallery. While your other social network online friends receive your request, they will judge whether it is you or not. Thus, it makes sure the communication more time-effective.

4. Preliminaries

4.1 Auxiliary Tool

We used Cory Gross's chat room which shared in the GitHub as our original third party social network software. It contains the original PC server and PC client. The functions of the chatting room are very basic, and listed below:

- Store the username and password to the database.
- Multiple users chat to all via word message.
- Online Users list.

4.2 Authentication Protocol Frame

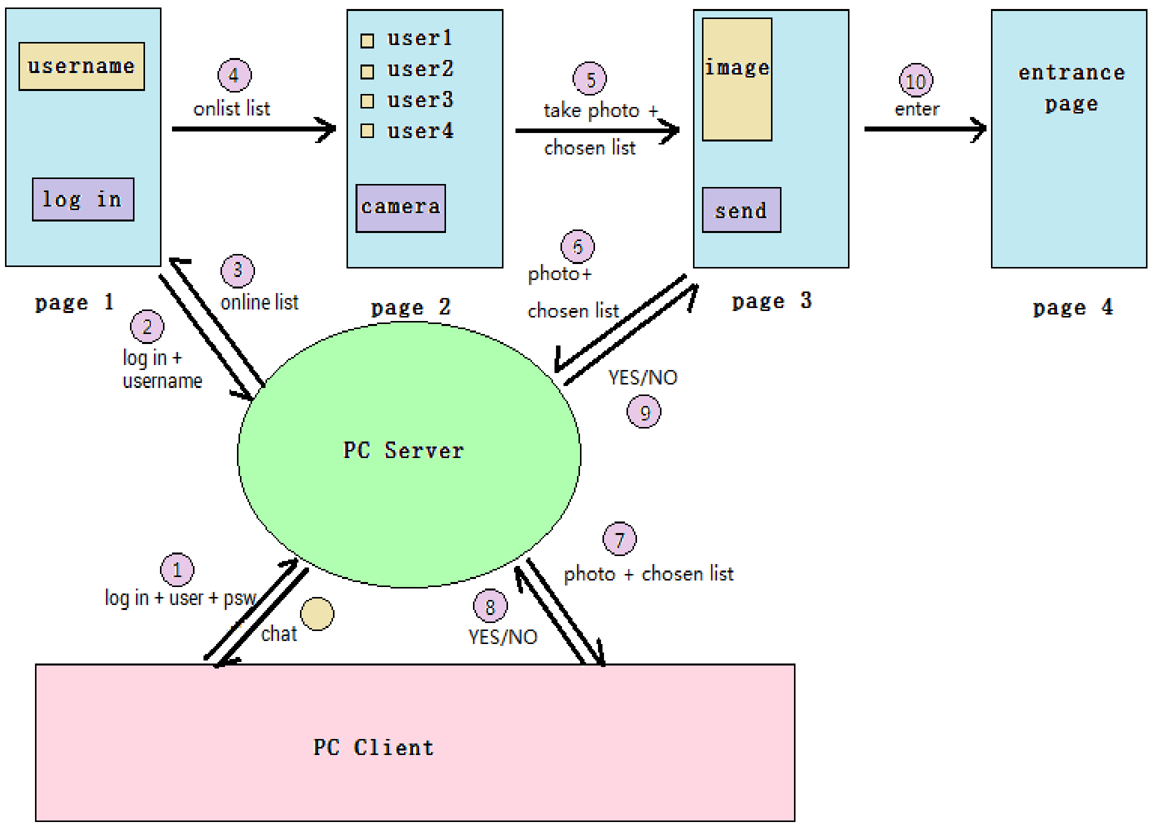


Figure 1. Authentication Protocol Frame

In the whole authentication system, three parties are involved. One is the common server (the Green block, we call it PC Server), another one is our verification system (Blue block, we call it Mobile Client/App), the last one is our authentication party or third party (the Pink block, we call it PC Client). The order of Arabic numerals in figure 1 shows the detailed steps. We will demonstrate the procedure by 4 steps of mobile app:

Step1:

A user inputs his/her username which has been signed in the PC client system. They don't need to type the password, since we made the it as a default one, which is "cybermed"(in real case, we can make it more complicated). The PC Server will check whether the password is the defaulted one or the registered one. If it is the former one, the online friends list will be sent to Mobile app.

Step2:
The online friends list is shown on Mobile Client’s second page to be selected the ones who will authorize the user to log in. The user can choose many friends. But only the first respond makes sense. After Choosing friends, and press camera button, user will be lead into camera screen to take a real time photo of the user’s face (this Mobile Client cannot access to our photo library, this prevents the scenario that if the user's mobile device is stolen, other people can't use former pictures to pretend to be the user).
Step3:
The photo will be shown on the screen and the user decides to send or not. If the user doesn’t satisfy he can restart. The Mobile Client will send the selected friends list and the real time photo to PC Server by pressing the send button. When receives the selected clients and image on the PC server side, it will only send the real time photo to the selected ones.
Step4:
When PC Clients receives the photo, it will appear on PC Clients’ screen as a figure. It shows the photo and the user who wants to log in. After pressing YES or NO, the server will receive the request then let the Mobile client log in or declined.

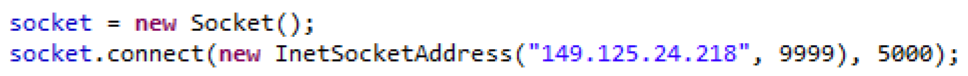
5. Fundamental techniques

5.1 TCP/IP Socket in Java

When two threads want to exchange message over the channel, each thread create an endpoint object that represents its end of the network channel. The OS manages the hardware and software that is used to transport messages across the channel (between the endpoints). Those endpoint objects are called sockets. The java.net class library provides classes Socket and ServerSocket for message passing for TCP/IP [7]. After android version 4.2.0, the socket method mustn’t be written in the main thread in order to keep the operating system from waiting connection for a long time.

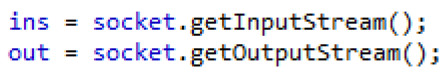
- The client's socket should specify a local I/O port for sending messages and specify the address of the destination machine
- The server's socket also should specify a local I/O port to receive messages. Messages can be received from any client, as far as the client machine knows the server's address and the port number which are bound to the server's socket.
- The client issues a request to the server to make a connection between the two sockets. Once the server accepts the connection request, messages can be passed in both directions across the channel.

The client assumes that the server is listening for connection request on portserverPort via TCP.



```
socket = new Socket();
socket.connect(new InetSocketAddress("149.125.24.218", 9999), 5000);
```

When the client's request is accepted, the client creates an input stream to receive data from its socket and an output stream to send data to the socket at the server's end of the channel. To the programmer, sending and receiving messages using sockets appears just like reading and writing data from files.



```
ins = socket.getInputStream();
out = socket.getOutputStream();
```

The server begins by creating a ServerSocket:

```
int serverPort = 9999;
ServerSocket serverSocket = new ServerSocket(serverPort);
```

After creating ServerSocket, the server calls accept() method of theServerSocket in order to listen for incoming connection requests from clients.

```
Socket server = serverSocket.accept();
```

Method accept () wait until a client requests a connection; then it returns a Socket that connects the client to the server. The server then gets input and output streams from the Socket and uses them to communicate with the client.

When the interaction ends, the client, server, or both, close the connection, and the server waits for a connection request from another client.

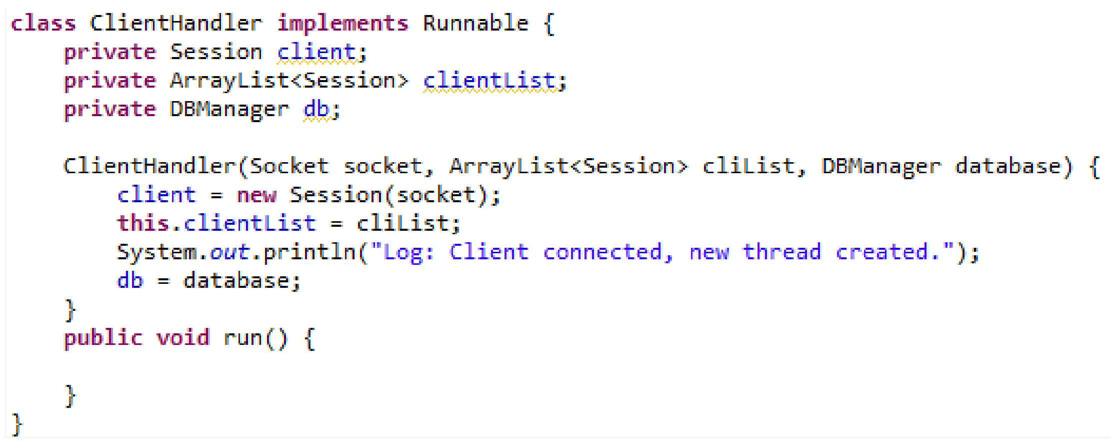
5.2 Multi-threading

Multithreading[8] is the ability of a program or an operating system process to manage more than one user at a time and to even manage multiple requests by the same user. Central processing units have hardware support to efficiently execute multiple threads. Every Java thread has a priority that helps the operating system determine the order in which threads are scheduled[10]. Threads with higher priority are more important to a program and should be allocated processor time before lower-priority threads. Create thread by implementing runnable interface.

Our case:

Server Uses ClientHandler class to manipulate mutiple client threads.

Code example:



```
class ClientHandler implements Runnable {
    private Session client;
    private ArrayList<Session> clientList;
    private DBManager db;

    ClientHandler(Socket socket, ArrayList<Session> cliList, DBManager database) {
        client = new Session(socket);
        this.clientList = cliList;
        System.out.println("Log: Client connected, new thread created.");
        db = database;
    }
    public void run() {

    }
}
```

At second step you will instantiate a thread object using the following constructor:

```
Thread(Runnable threadObj, String threadName);
```

Once Thread object is created, you can start it by start() method, which execute a cal to run() method:

void start();

5.3 I/O Streams

An I/O Stream represents an input source or an output destination. Streams support many different kinds of sequences of data, including simple bytes, primitive data types, localized characters, and objects. InputStream and OutputStream are the basic stream classes in Java, All the other streams just add capabilities to the basics, like the ability to read a whole chunk of data at once for performance reasons (BufferedInputStream) or convert from one kind of character set to Java's native unicode (Reader), or say where the data is coming from (FileInputStream, ServletInputStream, SocketInputStream, ByteArrayInputStream, etc.)

Our case:

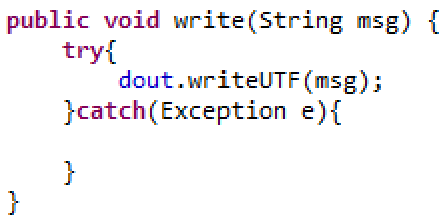
Use DataOutputStream, DataInputStream to transmit strings and pictures.

Use .writeUTF() and .readUTF() method to transmit strings only ;

And use .write() and .read() method to transmit pictures only;

Specially, in order to make sure an image has been received completely and skip out of the while loop, we send the size of image before itself been sent. When a part of size is received, the rest size will be refreshed. Until the rest size is 0, the loop will be broken.

Code example for transmitting strings:



```
public void write(String msg) {
    try{
        dout.writeUTF(msg);
    }catch(Exception e){

    }
}
```

Code example for transmitting pictures:

```
public void writeIMG(){
    try{
        FileInputStream fis = new FileInputStream("F:/YES.jpg");
        BufferedInputStream bis = new BufferedInputStream(fis);
        int size =bis.available();
        dout.writeInt(size);
        byte[] buffer = new byte[8192];
        int len;
        while (true){
            if(size==0){
                dout.flush();
                bis.close();
                //dout.close();
                break;
            }
            else{
                len = bis.read(buffer);
                dout.write(buffer, 0, len);
                dout.flush();
                size=size-len;
            }
        }


    }catch(Exception e){

    }
}
```

5.4 Head Tag

Here the head tag has a special meaning that refers to we add some information before a string or image so that receivers can tell the different requests.

```
String usernametrue = usernameinput.getText().toString();
dout.writeUTF("LOGIN: "+usernametrue+","+"cybermed");//username and fixed password

while(true){
    response = dis.readUTF();
    if(response.startsWith("USERLIST:")){
        usernames = response.substring(response.indexOf(' ')).split(" ");
        //usernames = response.split(" ")[1].split(" ");
        dis.close();
        dout.close();
        socket.close();
        intent.putExtra("usernametrue", usernametrue);
        intent.putExtra("userlist", usernames);
        intent.setClass(MainActivity.this, FriendList.class);
        MainActivity.this.startActivity(intent);
    }
}
```

In the above code, the capitalized words LOGIN is the head tag. When the receiver read the line starting with LOGIN, it will know the following information is about log in. The capitalized words USERLIST tells the receiver the following strings are online friends list.

To send an image, we need to send a string alone firstly but not to add the capitalized words in the head.

6. Important skills to deal with problems

6.1 Database

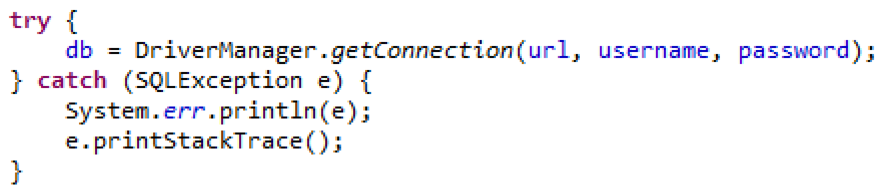
A database[9] is a means of storing information in such a way that information can be retrieved from it. It presents information in tables with rows and columns. A table is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts. The ability to retrieve related data from a table is the basis for the term relational database. The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a relational database. It helps you to write Java applications that manage these three programming activities:

1. Connect to a data source, like a database
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query

In this project we use MySql as our database too.

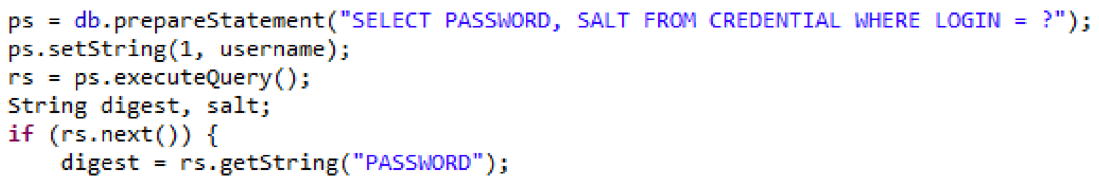
Our case:

connect the server to its database:



```
try {
    db = DriverManager.getConnection(url, username, password);
} catch (SQLException e) {
    System.err.println(e);
    e.printStackTrace();
}
```

Look up and update the database:



```
ps = db.prepareStatement("SELECT PASSWORD, SALT FROM CREDENTIAL WHERE LOGIN = ?");
ps.setString(1, username);
rs = ps.executeQuery();
String digest, salt;
if (rs.next()) {
    digest = rs.getString("PASSWORD");
```

6.2 PC Server Recognizes the Android APP

As we have signed in the chat room system on PC, the username is being existed in its database. Then log in on the android app with one username, but the problem is that there is no access for password. In order to enter the chat room system and get online friends list, the PC server should be able to recognize the username and know it is not a PC client. We write another method to deal with such a specific request. That is, we set a fixed password in the android app program. When press the log in button, it will send the typed username and the internal fixed password to PC server. When receiving this information on the PC server side, it will compare the fixed password firstly and then the username with the database rather than compare them at the same time. Because for normal PC clients, they have the access to input username and password so the PC server can compare them with database together. If both are true, the android app will be added to the online user threads list so that have the right to be broadcasted all online users list.

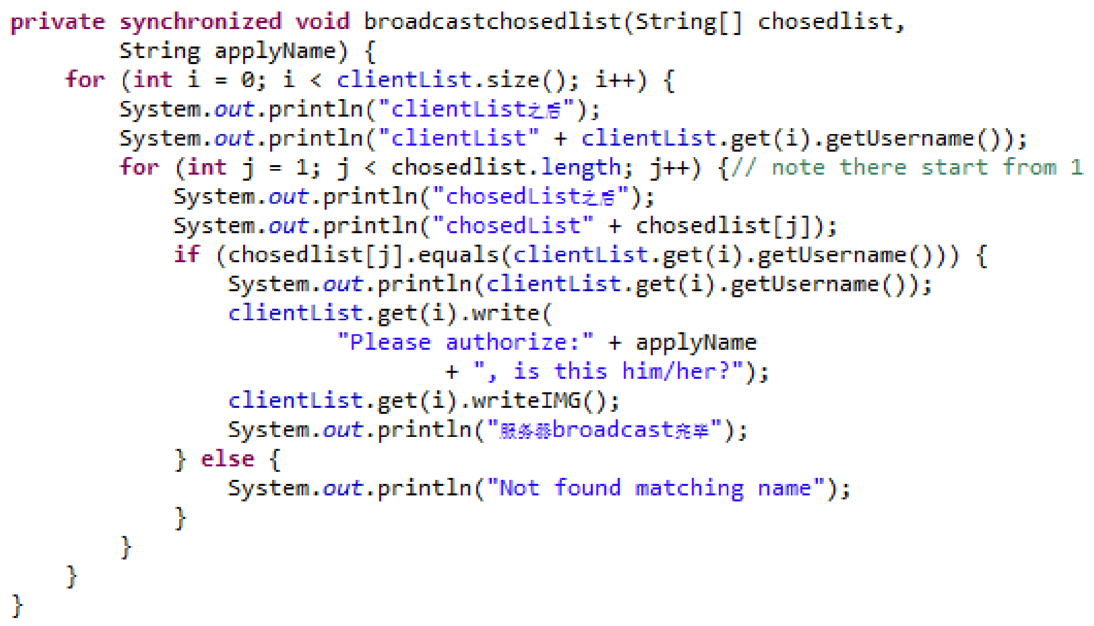
The code to realize this function is like the following, and word of cybermed is the so called fixed password, of course you can set any other fixed password you like.

```
        if (password.equals("cybermed")) {
                if (db.userExists(username)) {
                    accepted = true;

                    client.setUsername(username);
                    client.write("ACCEPTED");
                    clientList.add(client);

                        updateClientUserList();
             }
```

6.3 Get Online Friends List from PC Server

The PC Server broadcasts online list as soon as a user logs in. The mobile app keeps waiting until it receives a string starts with”"USERLIST:".Then split the string to an array, which contains online user’s name. In order to only send the list to the specially selected clients, the PC server need to make a comparison with the array.

On the PC server side:



```
private synchronized void broadcastchosedlist(String[] chosedlist,
        String applyName) {
    for (int i = 0; i < clientList.size(); i++) {
        System.out.println("clientList之后");
        System.out.println("clientList" + clientList.get(i).getUsername());
        for (int j = 1; j < chosedlist.length; j++) {// note there start from 1
            System.out.println("chosedList之后");
            System.out.println("chosedList" + chosedlist[j]);
            if (chosedlist[j].equals(clientList.get(i).getUsername())) {
                System.out.println(clientList.get(i).getUsername());
                clientList.get(i).write(
                        "Please authorize:" + applyName
                                + ", is this him/her?");
                clientList.get(i).writeIMG();
                System.out.println("服务器broadcast完毕");
            } else {
                System.out.println("Not found matching name");
            }
        }
    }
}
```

On the mobile app side:

```
while(true){
    response = dis.readUTF();
    if(response.startsWith("USERLIST:")){
        usernames = response.substring(response.indexOf(' ')).split(" ");
        dis.close();
        dout.close();
        socket.close();
        intent.putExtra("usernametrue", usernametrue);
        intent.putExtra("userlist", usernames);
        intent.setClass(MainActivity.this, FriendList.class);
        MainActivity.this.startActivity(intent);
    }
}
```

6.4 Dynamic Checkbox layout on Mobile App

The column number of the Checkbox is according to the number of online clients. The for loop and addView() method make it dynamic, so that avoid tedious and complex codes.

```
for(int i=1;i<rowCount;i++){
    checkBox=new CheckBox(this);
    checkBox.setId(i);
    checkBox.setText(userlist[i]);
    listLayout.addView(checkBox);
```

6.5 Take and Show Picture

To generate the real-time face photo, the mobile app must only have the function to take picture rather than choose one exited picture from the photo gallery. To realize the function, we call the mobile system camera device.

6.6 Popup Authorization Window on PC Client Side

After receiving the request information, the PC client should popup a visual interface for the requested user to make a corresponding reply. But how to make sure the chosen online user can send the reply information to server successfully? The most important point is the AuthorizeFrame class should have a member variable of client socket thread, so that the ChatFrame class can pass parameter to AuthorizeFrame class to make a connection and respond the PC server. The following sample is sample of authorization window.

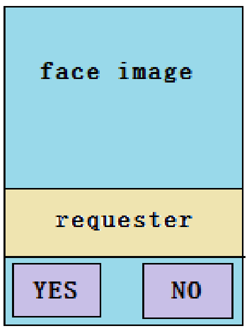


Figure 2. Authentication Frame

The main code inside ChatFrame class is:

```
 AuthorizeFrame authorizeFrame = new AuthorizeFrame(client,line);
  authorizeFrame.setLocationRelativeTo(null);
  authorizeFrame.setVisible(true);
```

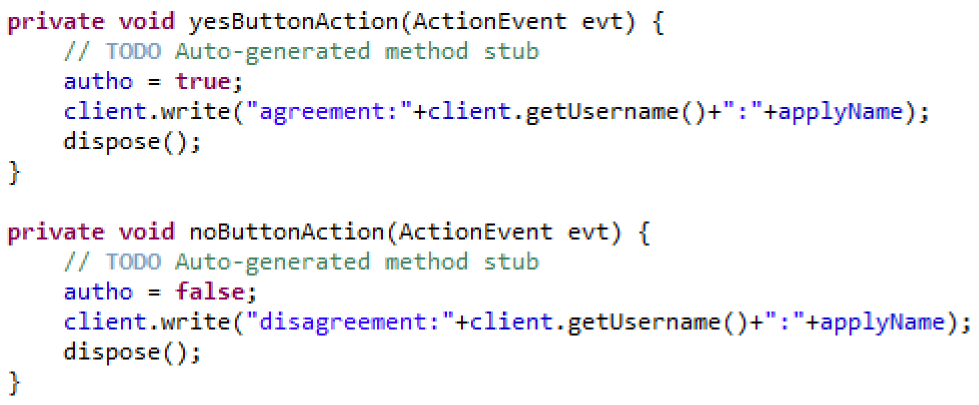
the main code inside AuthorizeFrame class is:

```
  public class AuthorizeFrame extends javax.swing.JDialog {
  private Client client;
   private String line;
  public AuthorizeFrame(Client cli, String li) {
  client = cli;
    line= li;
    … …
 }
   … …
}
```

By the above frame, the authorization frame can connect to PC server via java TCP socket.

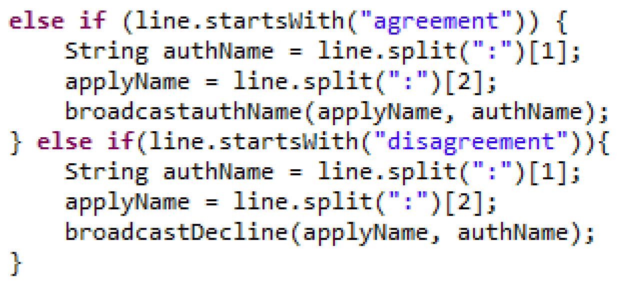
6.7 Authentication Process

The YES button means agreement and NO means disagreement. When press either one, the PC client will send its username and getter’s name to PC server. The difference is the former head tag is agreement and later disagreement so that the PC server can make judgment and send client’s respond information to android app. When app receives yes information it will skip to entrance page otherwise show failed notation. The main code on the mobile app side is:



```
private void yesButtonAction(ActionEvent evt) {
    // TODO Auto-generated method stub
    autho = true;
    client.write("agreement:"+client.getUsername()+":"+applyName);
    dispose();
}

private void noButtonAction(ActionEvent evt) {
    // TODO Auto-generated method stub
    autho = false;
    client.write("disagreement:"+client.getUsername()+":"+applyName);
    dispose();
}
```

On the PC server side:



```
else if (line.startsWith("agreement")) {
    String authName = line.split(":")[1];
    applyName = line.split(":")[2];
    broadcastauthName(applyName, authName);
} else if(line.startsWith("disagreement")){
    String authName = line.split(":")[1];
    applyName = line.split(":")[2];
    broadcastDecline(applyName, authName);
}
```

7. Limits and future:

- To enhance security, the fixed password should be more complicated.
- A real-time photo may not be secure enough. Adding real-time sound to photo may be better.
- The chat room does not have the function to send or receive images from other PC clients. At present, clients can only chat with each other via strings. But this point is not necessary for the project.
- The secure degree can be defined by the user selves. For example,if the user is logging in a bank system,he can set the authorization condition is more than 5 of 10 friends allow him/her to log in.If,the user is just logging in a game,he can set the authorization condition is 1 of 10 friends allows then OK,log in.

References

[1] J.D. Pierce, Jason G. Wells, Matthew J. Warren, David R. Mackay, A conceptual model for graphical authentication, in: 1st Australian Information security Management Conference, 24 Sept. Perth, Western Australia, November 2003, paper 16.
[2] T. Takada, H. Koike, Awase-E: image-based authentication for mobile phones using user's favorite images, in: Human–Computer Interaction with Mobile Devices and Services, vol. 2795, Springer, Berlin / Heidelberg, 2003, pp. 347–351.
[3] Jason Wells, Damien Hutchinson, Justin Pierce, Enhanced security for preventing man-in-the-middle attacks in authentication, data entry and transaction verification, in: Australian Information Security Management Conference, 2008. Paper 58.
[4] S. Xiaoyuan, Z. Ying, et al. Graphical passwords: a survey, in: 21st Annual Computer Security Applications Conference, 2005, pp. 463–472.
[5] A.E. Dirik, N. Memon, et al. Modeling user choice in the PassPoints graphical password scheme, in: ACM Proceedings of the 3rd symposium on Usable privacy and security, Pittsburgh, Pennsylvania, 2007, pp. 20–28.
[6] Adam Stass, Dustin Yu. 2014. Systems and methods for peer-to-peer online verification using third party authentication. United States Patent US20140115683 A1.
[7] http://www.bogotobogo.com/Java/tutorial/tcp_socket_server_client.php
[8] http://en.wikipedia.org/wiki/Multithreading_(computer_architecture)
[9] http://docs.oracle.com/javase/tutorial/jdbc/overview/index.html
[10] http://www.tutorialspoint.com/java/java_multithreading.htm