**A Multiple Secure Chat System**

Bhavik Shah (013672410)

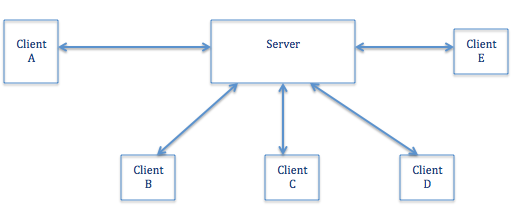
1. **What is the problem?**

Social networking has become a major part of people's lives. People of different ages use Facebook, Twitter to communicate and share private information like pictures and videos with others on a daily basis. But with this comes the risk of security. The consequences of sharing private information on a non-secure system can be severe. For example: private chat between two users containing useful information might be accessible to a non-trusted user.

People have become more concerned about their privacy with the high number of network attacks that have been occurring. It is important that the data needs to be communicated in a secure encrypted manner. So that other end of the receiving application can decrypt and use the information. This calls for a new service designed to enable users to post secure content across social media site. A secure chatting system that ensures safe communication by resisting to replay attacks by eavesdroppers.

1. **Methodology**

System Architecture



1. **General features**

Our application is built on a centralized distributed system. The server is at the center for distributing messages and managing various clients’ accounts. The server is scalable in nature and can address a number of clients. The secure chat system provides the feature of a chat room that allows various active clients to simultaneously connect to a chat room and interact with the other clients that are currently registered for the chat room.

In this project we have two kinds of chat rooms:

1. Public chat room
2. Private chat room

By default all clients can use the public chat room that is publicly available for the registered clients. Also, each client has the ability to create one or more private rooms. A client can invite one or more users to a private chat room. The client who creates the private chat room will be assigned the status of “room admin”. Being an admin for a private chat room will give you two major privileges:

1. The admin can invite more users
2. The admin can remove client from the room.

**b) Working**

The chat system provides secure communication by implementing shared key encryption. Firstly, each client has to register himself with the server. The registration is done using Triple Data Encryption standard. The server then generates a unique key for the client using client’s username and password that is going to be use to encrypt and decrypt messages exchanged between the server and the client. The advantage of using individual key encryption is to create a secure channel communication between the server and the client

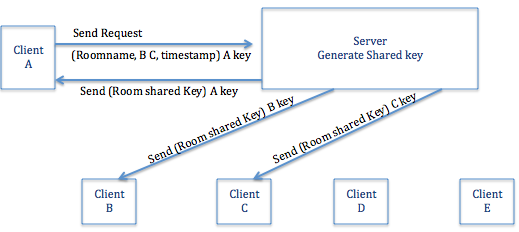
Once the client is registered, he can log into the system to join a chat room. By default, the client is directed to the public chat room where he can communicate with online users. Each client has the ability to create a private secure chat room. The client that creates a private chat room will then be elevated to the status of room admin and its client parameters will be stored at the server side. To create the chat room, the client sends a create room request which includes

CreatRoom Request Format

|  |  |  |
| --- | --- | --- |
| Room-name | Request (create room, users, time stamp) ClientKey | Clientname |

It uses Advanced Encryption Standard (AES) while creating the room. The client parameters of the room admin allow the server to recognize the chat room and its room admin. The request gets encrypted using a key that was generated using the user’s username and password. When the server receives the request, the server decrypts the request using the user’s username and password. Then, the server will check the timestamp for the request. If the time stamp greater (difference between sending time and the server time) than 30 seconds, the server will ignore the message. Otherwise, The server creates the room and generates a shared key for the room. The server sends the shared key to all the clients in the list and encrypts it with the unique key of each client. All invited users will get the shared key but with their individual encryption. The users will use the shared key to encrypt and decrypt the communicated messages.

Create Room Request



As mentioned above, the room admin has the privilege to add or remove a client from a private chat. Suppose the room admin decides to add a client to the existing client list of a private chat room. The admin creates a string consisting of a request message, client parameters and a time-stamp and sends this to the server.

AddClient Request Format

|  |  |
| --- | --- |
| Request (Addclients, Userlist, Timestamp) ClientKey | Room Name |

The server on receiving the message understands that the string message contains a request to add a client. To verify the authenticity of the sender of the request, the server tallies the room admin parameters of the concerned chat room with the sender of the request message. If there is a match then the client is added to the clients list, and the server will send the room shared key to the new user.

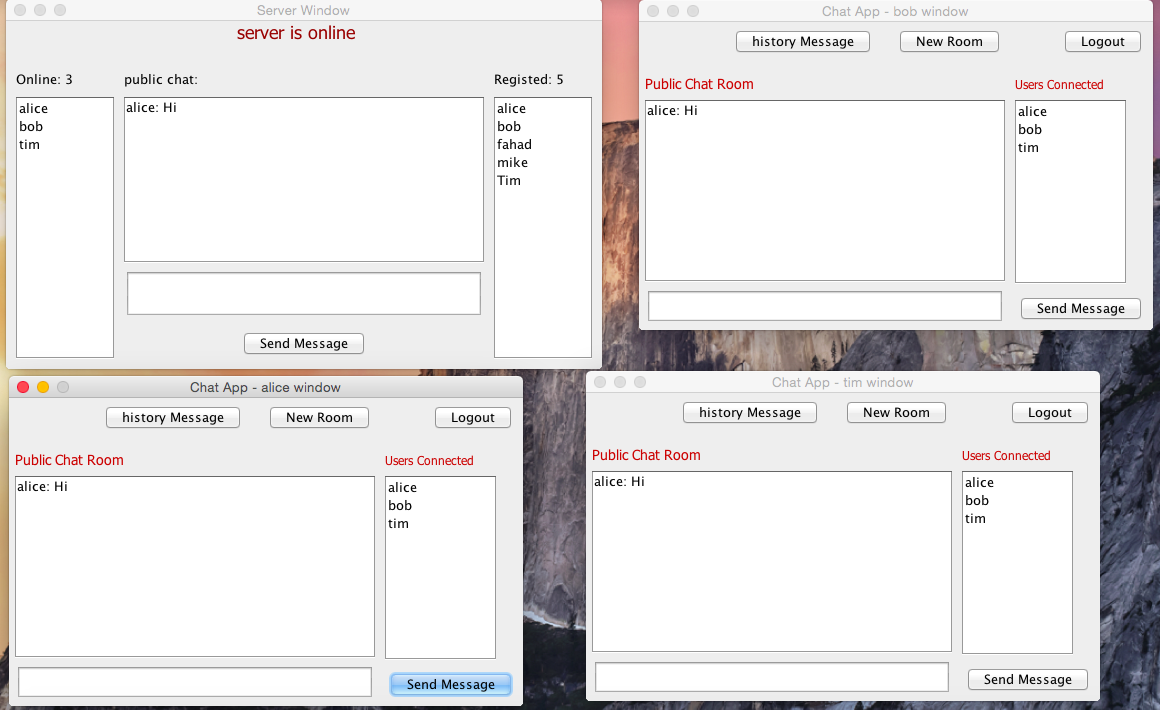
For deletion of a client from the client list of the chat room, the procedure is similar, the only difference is that the request section of the string will contain delete instead of add. Once a client is removed from the list, the shared key of the room get updated and the server redistribute the updated shared key to each of the clients present in the private-room’s client list.

As we mentioned above clients in the same private room can share messages using a shared key. Before client A can send a message to other clients, he has to encrypt the message using the shared key and send the message to the server. The server will multi-cast the message to the clients. When a client receives the message, he has to decrypt the message using the room shared key to read the message content.

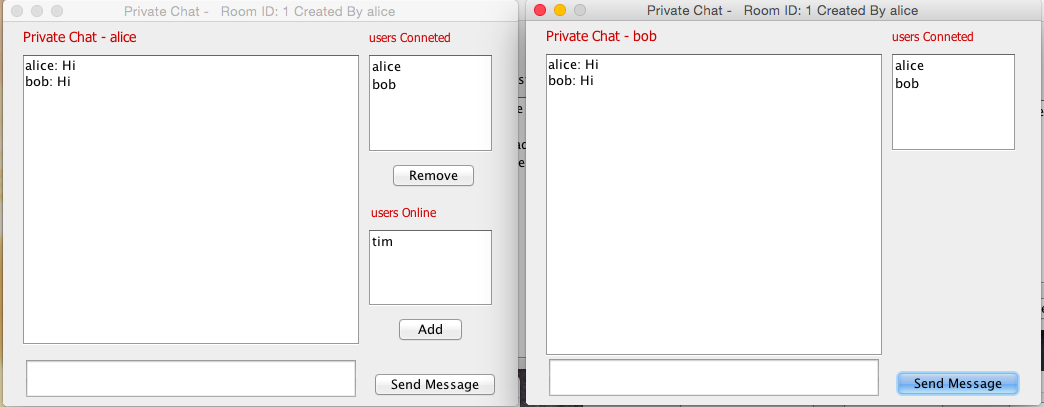
**Outcomes:**

The outcome of the above implementation is a secure distributed chat system that follows a client server protocol. The chat system follows shared key encryption to verify the authenticity of the client. The shared key obtained from the following process is constantly updated for each deletion of a client from the client list which makes it secure. Thus we obtain a secure chat room that allows users to choose a private chat room they wish to be in and communicate safely with its members.

User Public-Room Interface



User's Private-Room Interface



**Lesson learned:**

1. Understandings for security systems and issues have been improved.
2. I learned how to implement a scalable distributed system. The system can handle many clients and multiple private chat rooms at the same time.
3. Using check sum method “MD5” to fulfill the purpose of validating correctness of the delivery of the message and to verify that the message was not altered or t**a**mpered.
4. I learned how to use RMI in building distributed system.

**Conclusion:**

Secure distributed chat system that follows a client server protocol is being created. The chat system follows shared key encryption to verify the authenticity of the client which makes it secure. There are various security issues in a chat system which can cause loss of private user information. The implemented solution provides a secure system that overcomes the security issues encountered in exchange of information and provides safe communication.