

# **Security Report**

## **Steganography Chatting App**

**Team Number: 14**

**By:**

**Nourhan Khaled 34-3767**

**Radwa Sherif 34-14042**

**Habiiba ElGhamry 34-567**

**Yasmine Amr 34-8861**

**Farida Shafik 34-3537**

Github repo link: <https://github.com/NourhanKhaled/steganography-chat>

Video link: [https://drive.google.com/file/d/1p1NGd5XKN1NgdJSW7RiEM0tt\\_W1uOT6G/view](https://drive.google.com/file/d/1p1NGd5XKN1NgdJSW7RiEM0tt_W1uOT6G/view)

## **Summary:**

For our CSEN 1001: Computer and Network Security course project, we applied steganography to a client/server chat system.

### *What is Steganography?*

Steganography the art and method of concealing a secret message in a carrier file in order for the presence of the secret messages to remain unknown. The word is of Greek origin, being the combination of two words:

1. *Steganos* ([στεγανός](#)) which means "covered, concealed, or protected", and
2. *graphein* ([γράφειν](#)) which means "writing".

This technique of information hiding dates back several years, and has been frequently researched. Steganography can be applied to several media, such as:

1. Text
2. Images
3. Audio
4. Video

An important point to make is steganography's difference from cryptography. Both methods aim to conceal data, however, while cryptography changes the data's structure, steganography tries to make it invisible.

### *How is it applied to chatting?*

The medium chosen for this project was images. In our system, a user could send a message to another online user by typing in his/her message, choosing the destination, then choosing a JPEG image from his/her local system, and have the message hidden

in the image, which would then be sent to the user at the destination and revealed there.

### **Motivation:**

With the volume of media being exchanged nowadays, steganography presents a very convenient method of information hiding. Our choice of images as a medium was influenced by the fact that, first, text steganography is quite complex, as a text file lacks a large scale redundancy of information in comparison to other digital medium like image, audio and video. On the other hand, audio was not suitable either, as the Human Auditory System is more sensitive than the Human Visual Systems, which would make it more easily broken. As for videos, their size would be too large to make information sending convenient.

Image steganography can be applied using several methods, such as spatial domain transformation, transformation domain transformation, and the distortion technique. The method used in our project is spatial domain transformation, where bits are changes to include information. The specific spatial method used is called Least Significant Bit, which, as its name implies, changes the least significant bit.

### **Implementation:**

The project was built on top of a chat system built two years ago for the Networks course. It is client/server, with users signing up initially on the server, and then signing in. This required authentication, which will be described in detail below when discussing cryptographic algorithms. As mentioned above, users are required to upload an image from their system before sending a message. A steganography library referenced below was used. After uploading an image, creating a message and setting the destination, the following steps were done to apply steganography:

1. To load the image into the user space, a new image is created. Java can the directly change it. Attempting to make changes directly to the original file causes issues, as the changes are not always applied.
2. The image is converted into a byte array, and looped over. The least significant byte is then altered to contain information about the image.

Steganography has now been applied at the sender's side, and the destination is appended to the message and sent to the server. The server then reads the destination and forwards the image's byte array to it. The destination receives the byte array, creates an image out of it, then decodes it by reading the least significant bit of every byte. The message is then shown to the receiver to read.

### **Cryptographic Algorithms Use and Justification:**

When a user signs up for the first time, their credentials are sent to the server. The username is sent in the clear and the password is encrypted using AES with a 128-bit key. At the server, the password is decrypted and hashed using SHA-256 and a salt. If the username does not previously exist, the new password, salt and username are then saved to a config file, and the user is logged in successfully. On subsequent login attempts, the password is also sent encrypted using AES-128. At the server side, the received password is hashed once again using the stored salt and checked against the values in the config file. If they match, the user is successfully logged in.

When sending a message, the user is asked to choose a picture in which our application will hide the message. The message is first encrypted using AES, to add an extra layer of security. Then it is hidden in the chosen image using steganography. Our steganography algorithm of choice works as follows: it changes the value of the last bit of a certain byte in the image to match the value of one bit in the message. Therefore, the chosen image must contain at least as many bytes as there are bits in the message to be encoded. This is known as the **capacity** of the image. For example, a 128x128

RGBA image contains  $128 \times 128 \times 4 = 65,536$  bytes. It can be used to hide a message encoded in 8192 bytes.

Another important measure of the quality of a steganographic techniques is **fidelity**. This is the measure of the distortion that happens to the image after hiding the message in it. In our case, we use the peak signal-to-noise ratio in order to determine this value. This can be calculated using the following equation:  $20 \times \log_{10}(255 / \sqrt{MSE})$ , where MSE is the mean squared error between the original image and the image after applying steganography. Without encrypting the message before hiding it in the image, it was evident how the size of the message and the size of the image affected the fidelity of the algorithm (i.e., the bigger the message for the image the less the fidelity and the larger the image for a given message the larger the fidelity). After encrypting the message, however, the fidelity for a given image is the same for messages of variable lengths as long as they are less than 64 bits, this is due to the fact that AES outputs similar-sized blocks.

The reason behind our choice of SHA-256 to hash password is that it is deterministic, fast to compute, and collision resistant. This made it a perfect candidate for the hashing in our chatting application. There was the option of using the larger SHA-512, upon research, we realized that practically, SHA-256 is just as secure as SHA-512, due to the fact that they are all collision resistant with current or foreseeable tech. SHA-256 is also more practical as it is smaller and requires less bandwidth which is essential for a chatting application. A salt was essential, as it is common knowledge that many users use similar passwords, and therefore, to protect our users' passwords they were used. Salts help protect against dictionary attacks and rainbow table attacks.

The reason for choosing AES is that it's one of the most commonly used and secure end-to-end encryption techniques. For example, it's used in HTTPS and FTPS. It combines security with the speed needed in a chatting application. We work under the

simplifying assumption that the keys have been exchanged in advance between the user and the server.

### **Attack Scenarios:**

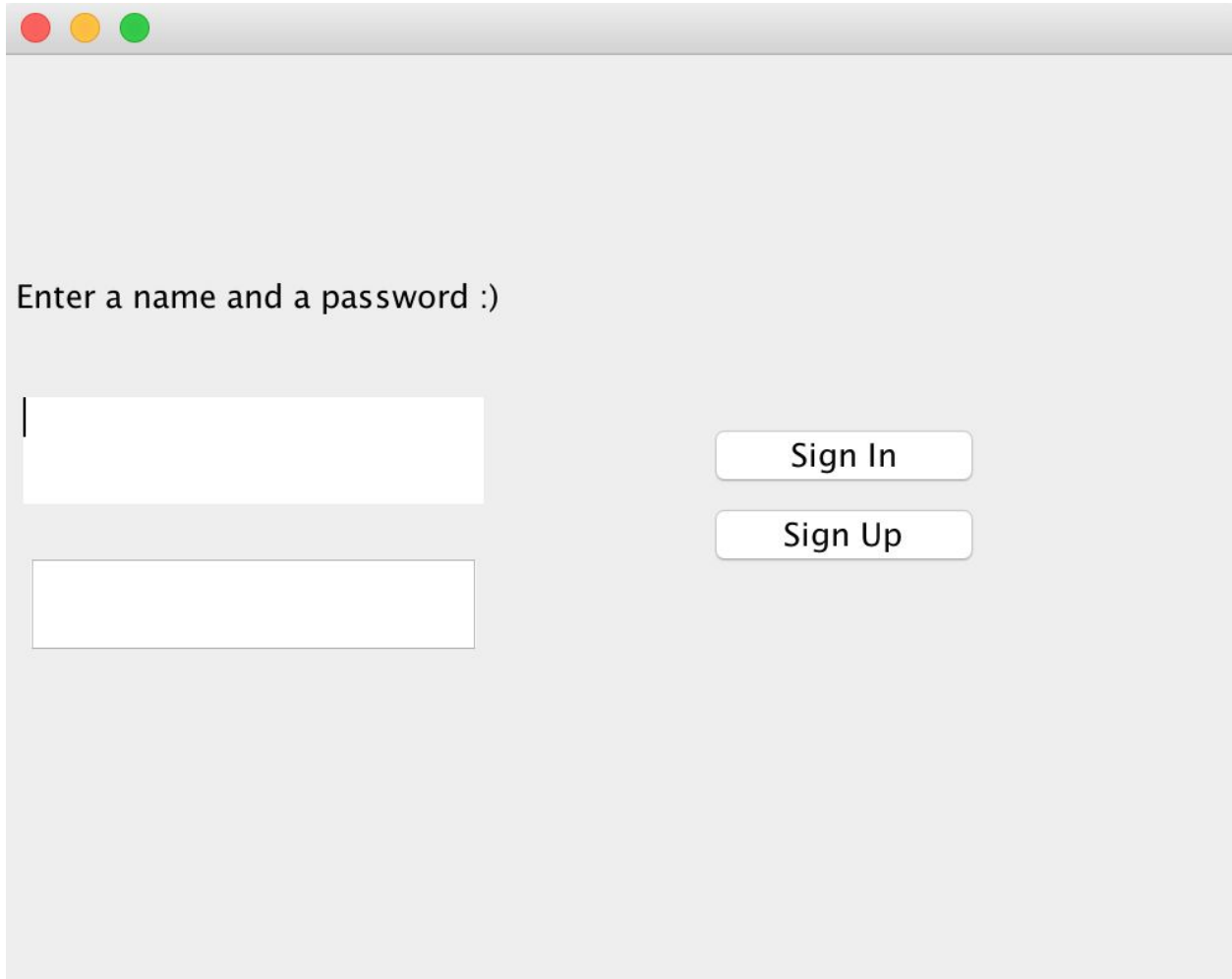
Several attacks were considered while creating this project.

1. An attacker accessing the config file. Security measures:
  - a. Hashes.
    - i. Advantages:
      1. used to encrypt the users' passwords
      2. Adds a layer of security
    - ii. Disadvantages:
      1. Increased computation
      2. Increased delay
      3. Vulnerable to some password and dictionary attacks
  - b. Salts.
    - i. Advantages:
      1. Provided resistance to same password and dictionary attacks.
    - ii. Disadvantages:
      1. Dictionary attacks can still be applied in the case of a specific user being targeted.
2. A Man-in-the-Middle Attack.

The security measure here is the combination of encryption and steganography. As the message is hidden in the image, a man-in-the-middle attacker sees an image being sent between the chatting parties. If they do not know about steganography, they would not suspect that any important information is being exchanged. A more experience attacker would know about steganographic techniques and may try a few of them to try to extract the

message. Even if they are successful in decoding the message from the image, the content of the message is encrypted so they still cannot read it.

**Demo:**

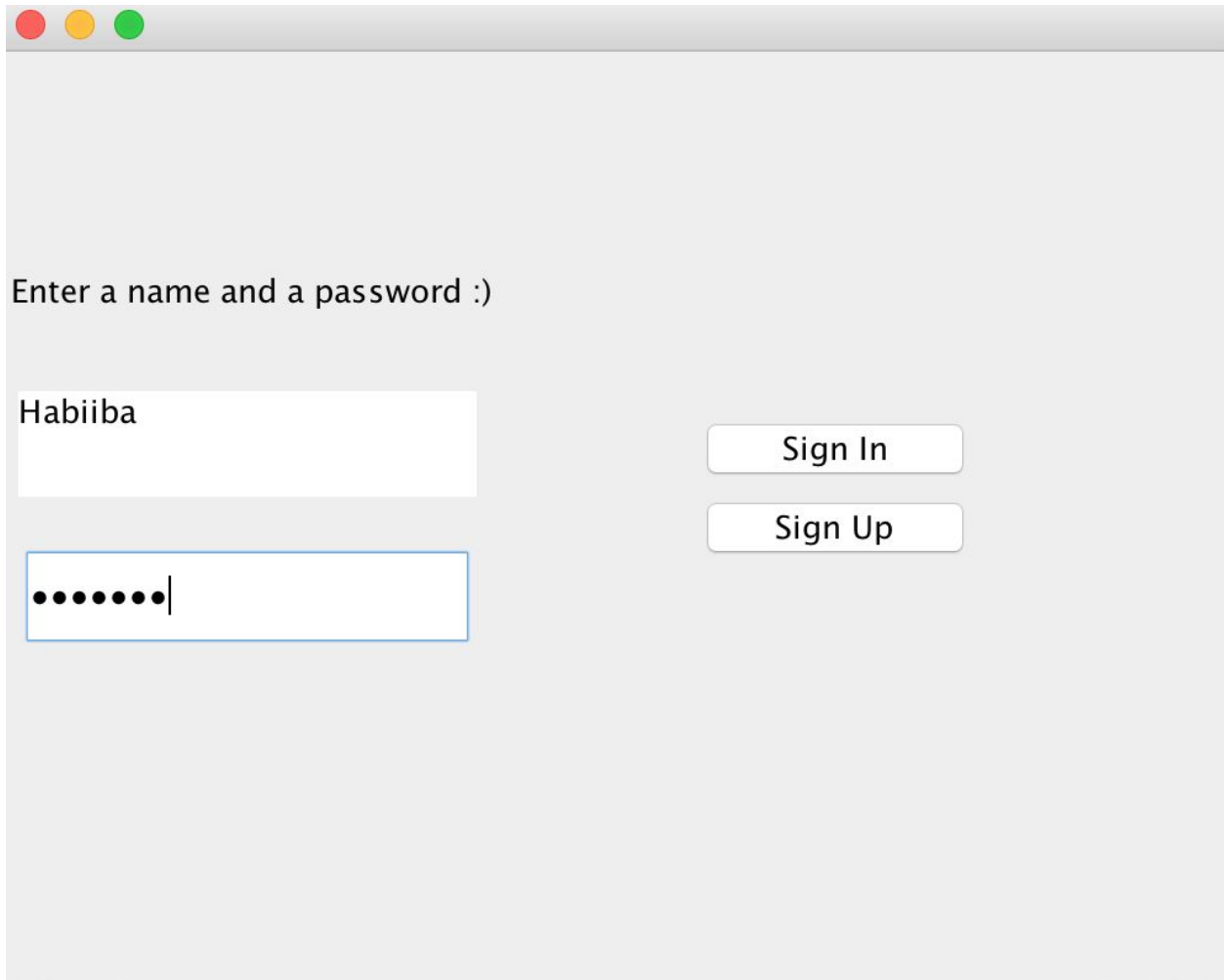


Enter a name and a password :)

Sign In

Sign Up

Starting up the app, the sign up/ sign in window appears.



Enter a name and a password :)

Habiiba

.....|

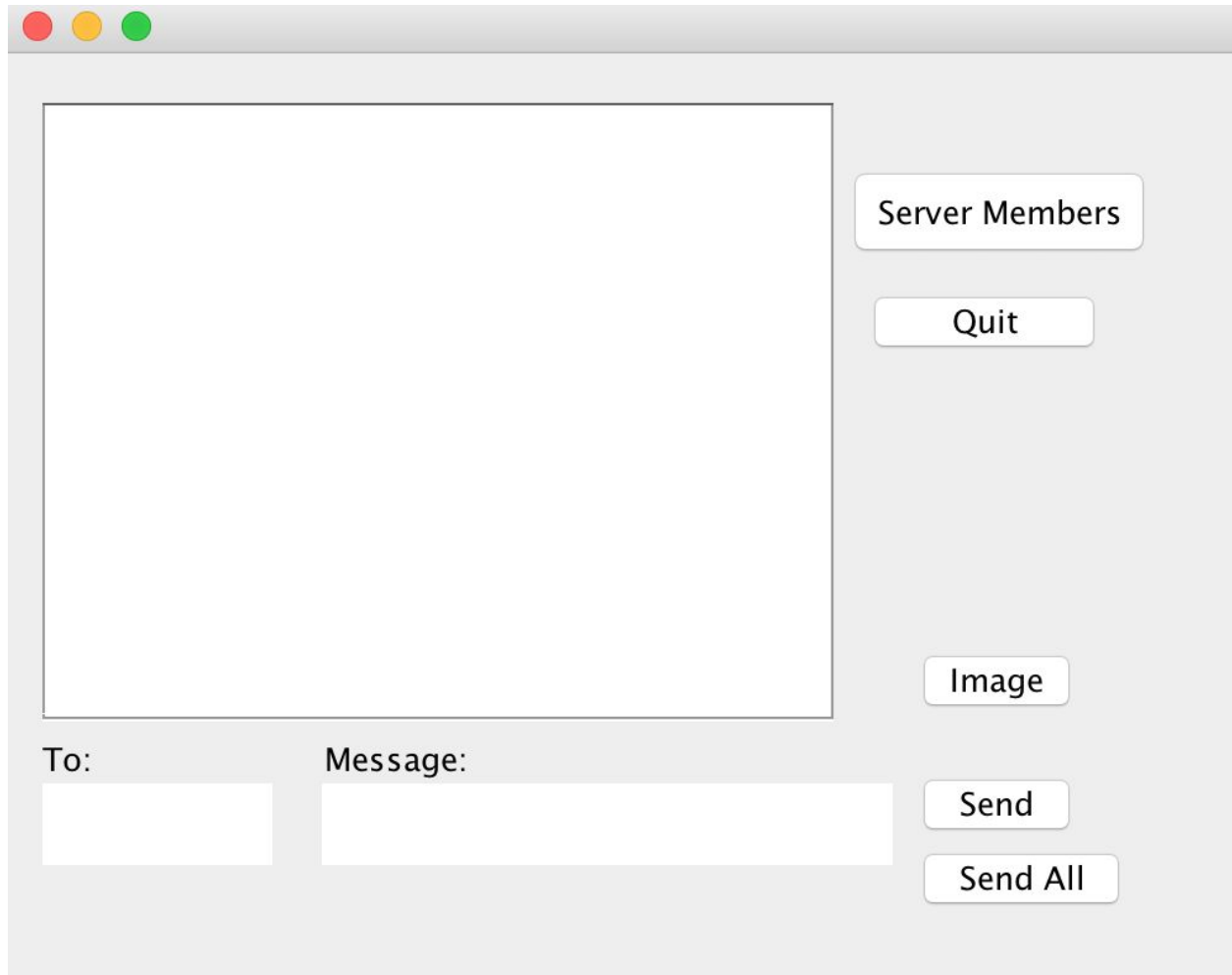
Sign In

Sign Up

The image shows a web form for signing up. It has a light gray background and a title bar with three colored buttons (red, yellow, green). The form contains a text input field with the name 'Habiiba', a password input field with seven dots and a cursor, and two buttons labeled 'Sign In' and 'Sign Up'.

Signing up





The message window. All chats appear in the same window with the sender name.

Members: Habiiba, Farida,

Server Members

Quit

Image

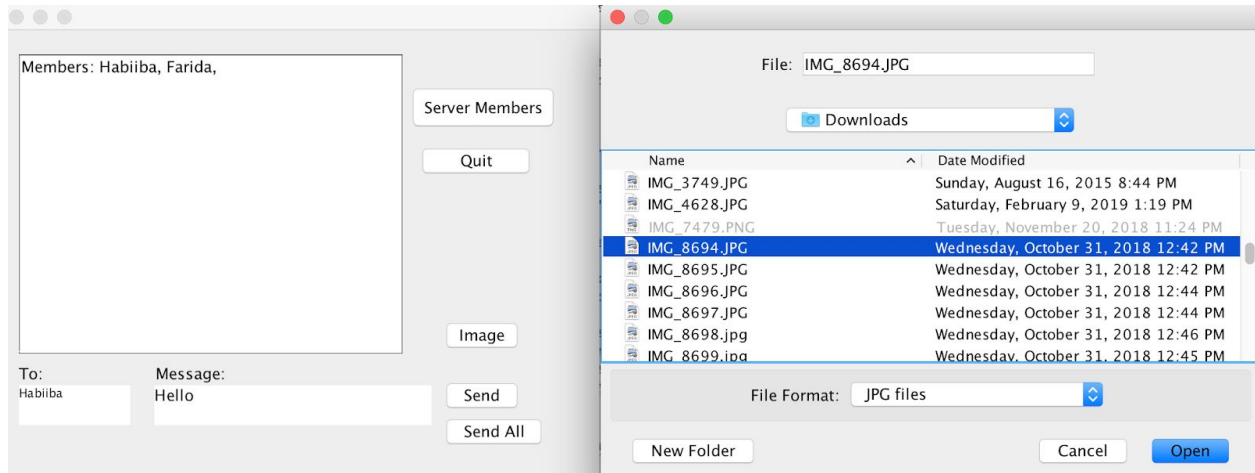
To:

Message:

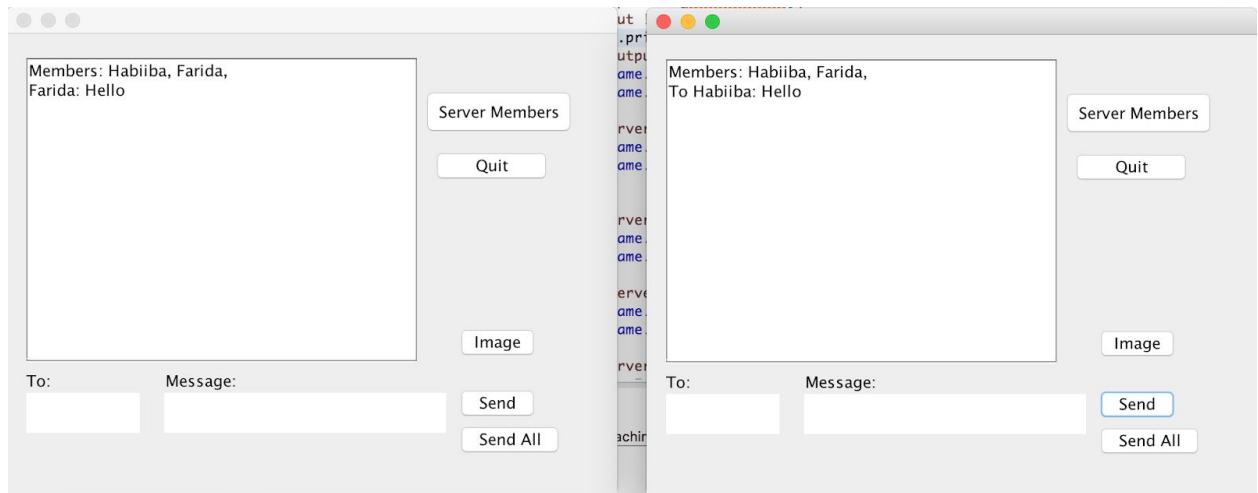
Send

Send All

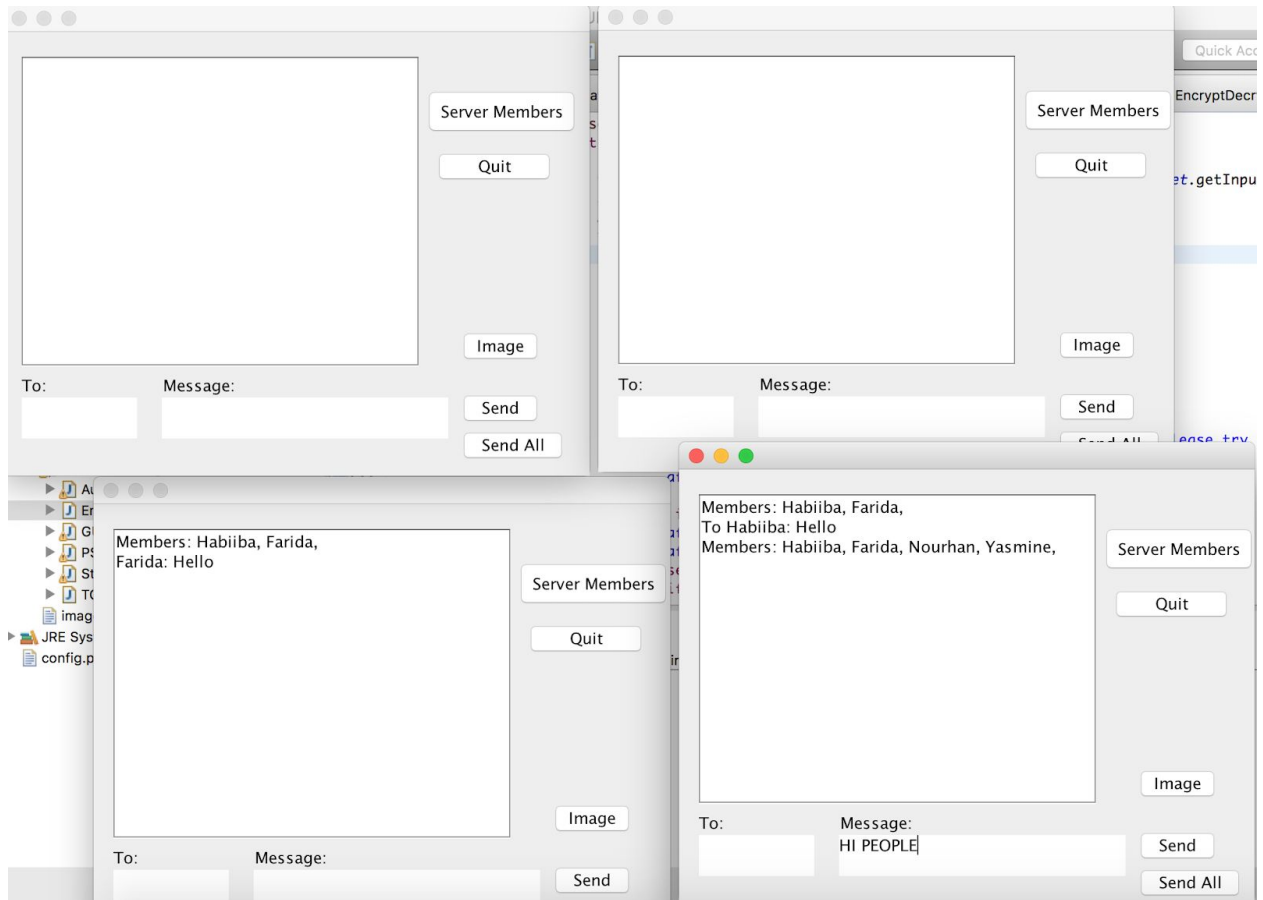
Displaying active members.



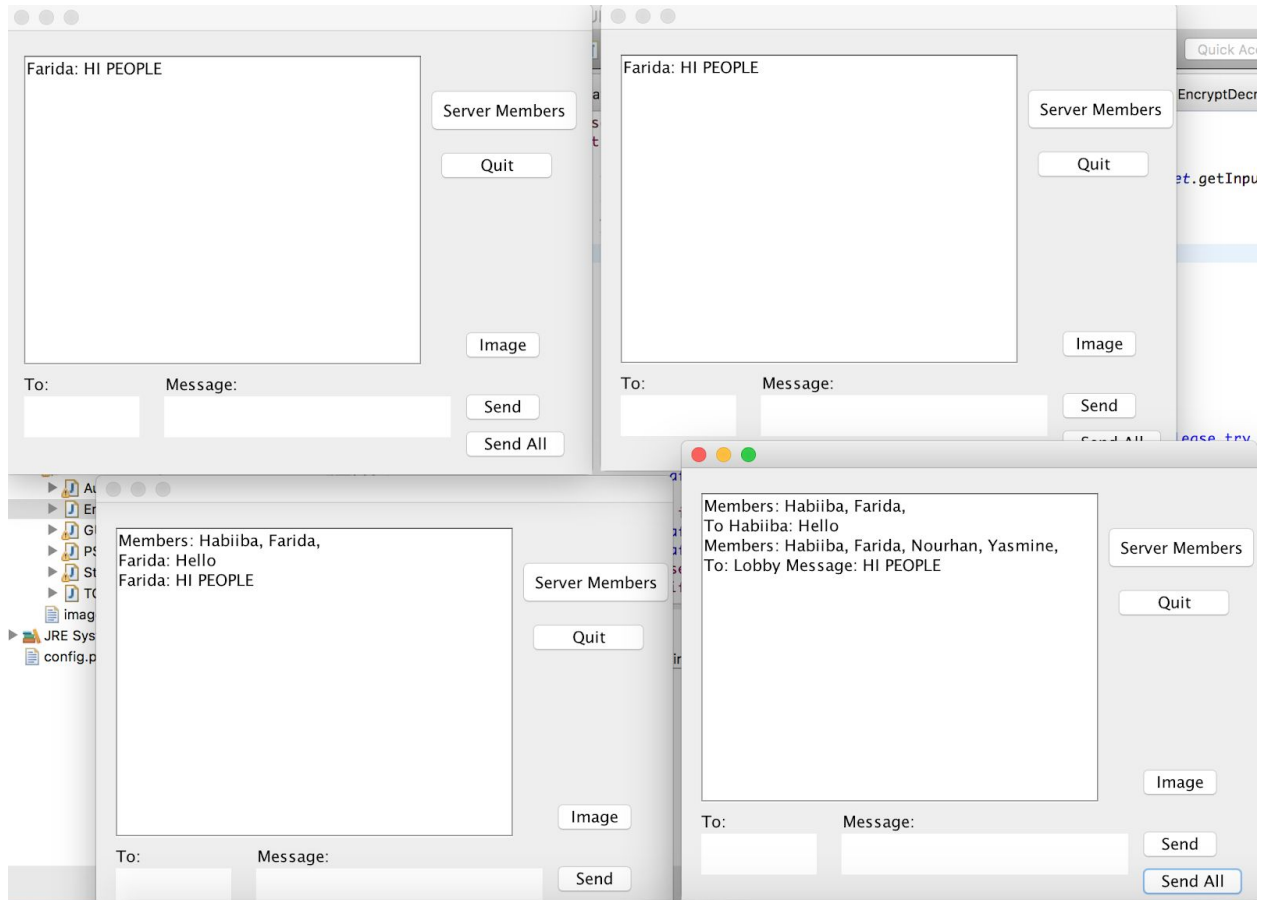
Attaching image to hide message.



Message successfully sent.



Lobby chat, press send all.



Lobby chat.

Enter a name and a password :)

Habiiba

Sign In

Sign Up

.....

Username exists.

Error message on existing usernames.



Enter a name and a password :)

Man

Sign In

Sign Up

**Username does not exist.**

Enter a name and a password :)

Habiiba

Sign In

Sign Up

Incorrect username or password. Please try again!

```
#Thu May 02 12:07:17 EET 2019
Habiiba=LgwNMYaj0Dptu3Zvm8gNzQ\=\#\#\#0feef7f7edf4f5f99367b0f10fbbb579a208e3be9873b2978a1c149268914497
Farida=YjmnP7nrewpsYGf5Fi+P4A\=\#\#\#2d27770aed7c963d8a5de932f7246a12fdc9ca49c23acbed154b7f6967f3d274
Nourhan=eHJtZjS6XfXobue0rFvm0g\=\#\#\#b98d80d593c16f3f082b34aff24b234dc150022a448a2fbab7346989c27e6d2c
Yasmine=AIpyCrDm7R2godH6dLlQ\=\#\#\#0584a105b3d4b888fc4dc25a1318e441e4c1bcde2b89cb5e1e1131e04effb45d
```

```
AlpyCrDm7R2godH6dLlQ==##0584a105b3d4b888fc4dc25a1318e441e4c1bcde2b89cb5e1e1131e04effb45d
```

Press 'F2' for focus

config.properties storing usernames, salts and hashed passwords.



### **Libraries and Frameworks:**

- For SHA-256 hashing we used the implementation in this link:  
<https://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/>
- For AES, we used the built-in javax.crypto.Cipher library.
- For Steganography, we used the implementation in this link:  
<https://www.dreamincode.net/forums/topic/27950-steganography/>
- To calculate the MSE and the peak signal-to-noise ratio:  
<https://www.programcreek.com/java-api-examples/?code=Theldus/PSE/PSE-master/src/sample/nodes/MSE.java>

### **References:**

1. Petitcolas, FAP; Anderson RJ; Kuhn MG (1999). "Information Hiding: A survey"(PDF). *Proceedings of the IEEE (special Issue)*. **87** (7): 1062–78. CiteSeerX 10.1.1.333.9397. doi:10.1109/5.771065.
2. <https://crypto.stackexchange.com/questions/43990/what-are-advantages-and-disadvantages-of-sha-256>
3. Febryan, Aryfandy, Tito Waluyo Purboyo, and Randy Erfa Saputra. "Steganography Methods on Text, Audio, Image and Video: A Survey." *International Journal of Applied Engineering Research* 12.21 (2017): 10485-10490.
4. <https://security.stackexchange.com/questions/165559/why-would-i-choose-sha-256-over-sha-512-for-a-ssl-tls-certificate>
5. <https://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/>