Security Project Report

Mirna El Gazzar	34-1138
Kareem Shaker	34-1181
Ramy El Zahaby	34-490
Mohamed Sherif	34-1160
Ahmed Walid	34-2270

1. Summary of the project, motivation and features:
   - We chose to implement the steganography chat, which is a way of hiding data shared between users to be used in chat rooms by users who want to share private messages back and forth, while preventing other users in the room from interpreting the message.

   - Our application has the following features:
     Authentication
     Creating a chat room
     Broadcasting a message to all the members of the chat room
     Sending a private message to a specific user (By encoding a private message using an image to be received by the other users)

2. Design choices of cryptographic algorithms and protocols for each functionality & justification for choosing it:
   - *Design choice of cryptography:*
     Using steganography, when two users want to send private messages back and forth to each other without letting the other users to interpret that message, the message sent will be received by the intended users without any manipulation, while an encoding of the same message is sent to the rst of the users so that they don’t understand it.

     The idea behind steganography is dependent on the secrecy of the algorithm, in other words, the sender and the receiver are the only ones entitled to understand the message, as only them know how to encode and decode the message using the image.

     That methodology was preferred because it is unpredictable, hence improving the secrecy needed, as no one can expect that the message is encoded in an image.

- *Protocol for each functionality:*
  a) For authentication:
     Instead of using a database that has the usernames and the passwords of all the signed in users, we injected 4 users credentials in the source code, and we have to use them in order to access the application.

  b) For broadcasting a message:
     When a message is sent by a user to all the members of the chat room, first it is encoded at the end of the sender and travels in the form of an image and is received and decoded at the end of the receivers. It goes through that process to account for the possibility that there might be a man in the middle that wants to intercept the message, so if he succeeds and gets the message he wouldn't be able to interpret it as it is encoded in the image.

  c) For sending a private message:
     The message is encoded at the end of the sender, and sent in the image to all the users in the chat room. At the end of the selected user, the message is decoded and interpreted, while the rest of the users who are not selected the image is received (encoding of the message) and not interpreted.

  d) The chat room:
     We used an already implemented chat room. However, since it lacked many of the features of our project, we of course added our features to it.

3. A comparison between our security methods, protocols and algorithm and the others & why our approach is the best:
   There were several implementations of steganography, our implementation had the following protocol:
   When a message is sent…
   - Each character is converted to Ascii character.
   - The sequence of Ascii character is converted to binary (every Ascii character is 8 bits).
   - The sequence of bits is split into pairs of bits.
   - Every pair (2 bits) is converted to a decimal number.
   - Now, we have a 1 d array of decimal numbers (every letter of the original message now takes 4 decimal numbers), we convert that array into a 2d array.
   - Then we use the pixels of the image to hide those numbers. Each pixel has 4 channels(RGBA), 8 bits each, and each letter of the original message is now represented by 4 numbers. We start by choosing the position of the starting pixel (where we start to use pixels to hide the numbers. Then, in order to hide the message, we first reset the least 2 significant bits of EACH channel of each pixel

used, then we add every one of the 4 numbers (the representation of each letter) to each channel of the pixels. Changing the least 2 significant bits of each pixel to a part of the decimal number.

Example:

H A C K

72 65 67 75

01 00 10 00 01 00 00 01 01 00 00 11 01 00 10 11

[1,0,2,0] [1,0,0,1] [1,0,0,3] [1,0,2,3]

[[[1,0,2,0],[1,0,0,1]],

[[1,0,0,3],[1,0,2,3]]]

key -> [RandomY, RandomX, Height, Width]

[0, 0, 2, 2]

B ,G ,R , A

RandomY, RandomX -> [15 ,25 ,40 ,255]

0000 1111 0001 1001 0010 1000 1111 1111

Set 2 least significant bits to 0

0000 1100 0001 1000 0010 1000 1111 1100

OR

1 0 2 0

0000 1101 0001 1000 0010 1010 1111 1100

13 24 42 252

Pixel before: [15,25,40,255]

Pixel after: [13,24,42,252]

Instead of fixing the starting pixel to be used to hide the message characters, we randomize the starting position.

The key is sent to the entitled user to be used to decrypt the message, the key is 4 bits that have the starting position of the used pixels and the portion of the image used to hide the message.

Other implementations could use the one least significant bit of each channel, which is more expensive than our approach, as more pixels will be used to hide the message. Other implementations also used 3 bits of each channel of each pixel, which is worse than our approach since it will cause a larger distortion to the used pixels since a bigger portion of each pixels is altered, and this could jeopardize the secrecy aspect of the algorithm, because the bigger the distortion of the image, the more suspicious it is to unauthorized users. The other implementations also fix the starting position of the pixels, which is worse than our randomizing algorithm.

4. The possible attack could be done by the man in the middle who can try to acquire the message exchanged between the entitled users. When the man in the middle gets the message, it would be hidden in the image, so he wouldn’t expect that the message is

hidden in the image and even if he knows that it is hidden in it, he won't have the key to be used in decoding and understanding the message. However, if a hacker was somehow able to get access to the key, he would easily be able to decode the message, but since Steganography is based on the secrecy not on the security if the message, this is not a concern.

5. Explanation of each computer security technique you used along with its advantages/disadvantages trade-offs.

   a) Authentication:
   Authentication was one of the important security techniques used. It was needed so that not any user can access the application, as only registered users are allowed to use it. He has to have a username and password.

   Typically authentication would be implemented by allowing new users to register and then their credentials would be saved in a database, and when the user logs in with his username a password, they should be compared to the ones saved in the database, and then he is granted access if they are correct, but for the simplicity of the implementation of our project, we injected 4 users (usernames and passwords) in the source code, and in order to log in the user has to use the credentials of one of those users.

   Advantages: As mentioned before, this supports the privacy of the application by limiting the access to specific authorized users only.

   Trade-offs: in our project, the feature of allowing new users to register was compromised, so only a limited number of users can be authorized if they use the credentials already saved by the application developers, we had to do so because our application is not a large scale application, and we are not using a database

   b) Capacity:
   3 different images with 3 different sizes were made available to be used for hiding the messages exchanged.

   The image used depends on the length of the message needed to be hidden and sent:
   If the message is less than 16 character, an image of 144*144 pixels is used
   If the message is between 16-36 characters, the image used is of 360*360 pixels
   If the message is more than 36, an image of 720*720 pixels is used

Advantages: the ratio of the secret message to the source image is maintained.

Trade-off:

If the message length is very long, sending and receiving a 720*720 image every time there is a new message would be expensive.

c) Fidelity:
To avoid the distortion of the image containing the secret message, we are making use and altering only the least 2 significant bits of each channel of each pixel (not more than 2), this gives us the advantage of preserving the image as much as possible as not so many bits of each channel are changed, so the overall change of the image is not obvious so the image stays as close to the original one as possible.

This is also better as it won't be suspicious to the users who receive the image, as they wouldn't expect that the image received is manipulated and contains a secret message

Tradeoff:
The number of pixels used might be larger to substitute the low number of bits changed, so this might be expensive while decoding the message

d) Access control:
- Accessing the chat room:
A user can enter a chat room only if he has the name of the room, so if the other users don't want a user to enter the chat room they wouldn't tell him about the chat room name

- Accessing the message:
If user A has the room name, or he is already in the chat room, but another user B wants to send a private message to C, B can make use of steganography to send the private message to C so A doesn't have access to it

6. Libraries/frameworks used:
   - Opencv: we used opencv to read the images into arrays.
   - PIL: process the image (resize).
   - Tkinter: for the user interface.

7. References, links and other aids:
For the chat application: https://github.com/kanakkabara/P2P-Chat
Steganography:
https://www.garykessler.net/library/steganography.html
https://ccrma.stanford.edu/~eberdahl/Projects/Paranoia/