Greg Cernera

Professor Arias

CMPT 220

8 May 2017

Final Project Writeup

For my final project, I have programmed an instant messaging application that connects two computers through a network and allows the users to interact with each other by having a written conversation. This program follows the format of a server/client setup and allows the computers to connect by first setting up a server on one end, and having the opposite user connect to an open port on that server's system. Initially, this program was intended to be an anonymous instant messaging and file sharing application for users who wanted to stay under the radar when sending information through the internet. However, creating this program proved to be much more difficult and time consuming than I had anticipated.

For one, I had never attempted to create a networking project in my computer science career prior to starting this task. So, much of my time was devoted to simply researching and studying the ins-and-outs of networking, how two computers can communicate, and if this idea was even feasible with the Java language. After much research and reading into Java networking, I realized that the best way to go about this project is to create a server/client foundation, and have the two users send information back and forth. Once this was completed, I began researching the necessary components of a proper instant messaging software and how I would be able to implement the ideas that I had found into my own Java application. This took me

several weeks to understand and implement into my code, but I finally figured out how to properly communicate two computers over a network in order for my project to be a success.

I wanted to create this application for multiple reasons. One of which was the fact that I had never created a networking project before, and that all of my previous projects were simply application based. I wanted to transcend the normality of which I have been accustomed to my entire computer science career and push myself to create a project that would force me to learn more about the Java language, and coding in general. So, I decided to create an instant messaging program to suit my desires of having a challenging task. Second, I have always had an interest in cryptography and secrecy, especially when relating to online covertness or cybersecurity. With that in mind, I thought it would be interesting to create an "anonymous" IM application. That is, I would encrypt the message that would have to be sent, send the encrypted phrase, receive it on the other end, decrypt it, and then display it in the text area in decrypted form. So, if someone was trying to intercept the signals across the network, all he would receive is the encrypted message. I wanted to build a program with the intention of making a tool that hackers would use to relay messages to other computers secretly, and not have to run the risk of exposure by using programs such as iMessage, Google, or any other monitored messaging system. Unfortunately, the encryption and decryption became much more difficult than I had previously believed. I tried several different methods to try and implement an encryption/decryption algorithm into my instant messaging code; however, I continually received multiple errors every time I attempted the implementation, but I believe that it is still possible given more time and knowledge of networking.

In order to understand the encryption and decryption part of the program, one must understand how the program is actually constructed. In order for the server/client format to work, I had to create two separate projects: the ***Server*** project and the ***Client*** project. One user (*User1*) on one computer will play the role of the server, and the other user (*User2*) on a different computer will be the client that has to connect to *User1*'s server. The Server project will be initiated first, which will set up a server on *User1*'s computer and begin searching for a signal that points to the server's IP address and port number. Once the server is setup and searching for possible clients, *User2* will open the Client project and provide an IP address and port number that *User1*'s server is transmitting to (similar to the PuTTY setup). Once the server recognizes and accepts the client, the two users are not connected and can send messages. The Server project has classes, MyServer and MyServerTest, and the Client project has classes, MyClient and MyClientTest. MyServer runs and monitors the server and messages being sent by *User1*, while MyClient runs and monitors the client and messages being sent by *User2*. The "test" classes simply execute their respective classes.

The requirements for the initial project idea are as follows:

(1) create a graphic user interface with easy usability

(2) create a working instant messaging application that can send messages to and from both computers

(3) encrypt and decrypt the messages

(4) implement a file sharing button that can send documents between between the two users

I began working on these steps linearly, starting with requirement number 1, all the while asking the question: can I make an easy-to-use instant messaging program that sends encrypted

messages across a network? Steps 1 and 2 took me a good amount of time to research, design, and implement into my program. Once those requirements were working and near completion, I moved on to step 3 and began coding coding methods to encrypt and decrypt strings. I coded the methods rather quickly since Java has multiple libraries that can be recruited to make encrypted strings, and used an Advanced Encryption Standard algorithm for the encrypted messages. However, I began to notice problems when I attempted to call these methods in the code, I received multiple errors that would ultimately break the entire instant messaging algorithm even if I addressed all of the errors and exceptions. Additionally, in order to maintain total anonymity, I attempted to implement the Diffie-Hellman key exchange to have both users fabricate a new key every time the program ran in order to keep the hackers and eavesdroppers out of the loop. But once again, I recieved multiple logic errors that made it nearly impossible to implement these new pieces of code into my instant messaging program. And as the deadline was coming closer to the end, I decided to scrap steps 3 and 4, and rather perfect and tune up the instant messaging GUI and algorithm for my final project submission.

There are several other works that have successfully addressed similar goals and problems to my own. For example, there exists a cryptographic protocol called Off-the-Record Messaging that “provides encryption for instant messaging conversations. OTR uses a combination of AES symmetric-key algorithm with 128 bits key length, the Diffie–Hellman key exchange with 1536 bits group size, and the SHA-1 hash function. In addition to authentication and encryption, OTR provides forward secrecy and malleable encryption” (Off-the-Record Messaging). Similar to my attempted encryption, the OTR protocol has successfully used an AES symmetric-key algorithm along with a successful Diffie-Hellman key exchange that

maintains total secrecy. I discovered this protocol late into the development of this project and tried to discover if I was missing some piece of information by researching OTR more, but unfortunately there is little information about this protocol online that I could carry over to my project. Still, it's nice to see that even though I could not figure out the algorithm myself, my idea and planning was similar to a real cryptographic protocol that people use.

In order for a user to properly use this system, he or she must follow these simple protocols. First one user (*User1*) must have the "Server" package on their computer and the other user (*User2*) must have the "Client" package on their computer. *User1* will execute the "Server" application first, because if *User2* initiated the "Client" application first, the client would have nothing to connect to and be stalled. So once the server is set up, *User2* will start their program and will have to input the IP address and port number that they wish to connect to. Once the client sends this information, the server will pick up this information and connect the two users. Now the two computers can communicate with each other with written messages. When a user wants to leave the conversation, they must simply type "END" in the message box, or just close the window. It is a simple step-by-step process that allows users to start communicating as easily and simply as possible. I wanted to create a graphic user interface that was both easy to use and aesthetically pleasing.

Thus, my goal is that this system will give users the possibility of maintaining complete anonymity, for the most part. I intend to continue with this project and finish the encryption/decryption and Diffie-Hellman key exchange algorithms. This project is not only an exciting and useful application, but also a learning process for me. While I am disappointed that I was not able to build the project which I had initially planned out prior to beginning the project, I

believe that I have learned much about networking. I have never created a networking application before, but many days of research and reading have given me a good amount of knowledge to get started quickly. This IM program has much potential and gives users an insight into how server/client messaging applications actually operate. This system is very promising and will hopefully be a tool that someone will find useful.

References/Bibliography

http://cs.lmu.edu/~ray/notes/javanetexamples/

http://searchexchange.techtarget.com/definition/Internet-Relay-Chat

https://www.tutorialspoint.com/java/java_networking.htm

https://en.wikipedia.org/wiki/Off-the-Record_Messaging