Door Security Mobile Application

Tyler Beveridge | Nicholas Collier | Christopher Jaress

Design

Our original design for the project was to have a mobile application that would act as an ID card when it came to unlocking doors that had a card reader. We planned to have the mobile application transmit three pieces of data: the unique ID of the user, the door number they wanted to open, and a sequence of red and green lights that the door output. We wanted the phone to receive a random number from the server when it first connected which would be encrypted. The phone would then decrypt using the key provided and would increment the number and send it back to the server. The purpose of this was to prevent against replay attacks. Then if the server received the correct number back, correct ID, door number, and the green and red light sequence, the door would open.

We split up the project by having one person work mainly on the server, one person on the mobile application and then one person working on both. We chose this way because it would be easiest to implement them two separately and then have one person know both sides and put them together.

To make the program efficient and easy to program we divided the tasks in a way that would make putting everything together in the end the easiest. We programmed the server and mobile application separately. To make the program easy to use, we decided to have 3 basic buttons on the main screen of the mobile application. The user would then just enter their ID, the door number they want to open, and when prompted the sequence of green and red lights. Each field is sent over to the server with a click of a button. The results would then be displayed via the card reader, or in our case the server would output the results to the console.

Stages of Implementation

The program began to evolve when we realized we did not know enough about android application development. We had to scrap the idea of having the server send a random number to the phone and having the phone decrypt and send it back over. Instead we authenticate the ID by using the current time. Our program would then send the ID over with the current time appended to it. Then when the server got this message it would check to make sure the time matched what was being sent over. The whole message would be encrypted so it would be impossible for Malice to know what time was being sent over. This would then stop replay attacks on our server since Malice would not know the time being sent over.

We had intermediate versions of the code, but we really meshed the final version together during the end. At first people were working on things independently but then we began to work together on all aspects of the project. We then tested to make sure the application and server were both following the correct protocol when sending and receiving the data. The main reason we decided to make this change was because whenever a message was received on the startup of the mobile application, it would behave unexpectedly. We did not want to have the user prompt to receive the initial message with a button so we opted to go with the hidden time factor authentication. We wanted to incorporate some sort of proximity detection within our application that way the green and red light sequence would be unnecessary. We wanted to either have GPS or Bluetooth let the card reader know that the phone was in fact nearby the door. However, we did not have time to implement this idea.

We finished the server first and got the basic protocol structure laid out. Then we turned our attention to the actual mobile application and got it to send messages to the server. From here we worked on our authentication and process for the controller to access the doors and multiple phones. After we got everything working we polished up our code and outputs so that it looked clean.

Testing

When we finally finished our first working versions of our project, we tested it in various ways. With our C++ code we had originally a C++ program to test to ensure that the C++ code was working properly. We also used the Android application on a very simplistic C++ program at first. This allowed us to work out small problems, such as missing null characters and ensuring that sending and receiving over a TCP connection was working. Eventually we had them both working on the small test programs so we decided to test them together. For a very long time we did not have it working for various reasons. We decided to take apart the C++ program to make sure that it was not something specific causing a problem over the Android application in java. We used tons of print statements in C++. We also used System prints on the android application. We were a lot more experienced with C++, so we naturally wanted to fix it on our stronger side. After a long series of tests we realized that something was wrong without receiving function on the Android application. We then devised a new method of security without the requirement of receiving a message from the C++ program. We after some issues with time and RSA finally got our project working. We used many test programs, many print statements, and testing a variety of cases. This slowly changed our project idea and our code to the point where things were different, but we aimed for the same ideas of security and our main project goal of using a phone application to open a door. We tested a plethora of cases as we developed and finished our project.

We made sure to test many cases and ensure we had no major bug problems.

Division of Labor

We originally divided the labor up into two distinct groups. We had Tyler Beveridge work on the C++ programs. This was considered the easier side due to our prior knowledge of C++. Nicholas Collier and Christopher Jaress worked on the Android application. Nicholas Collier worked on developing the application and learning the basics of android, while Christopher Jaress worked on the TCP connection over java. Slowly as we went on through the quarter, we combined the Android application work and had a phone application that looked decent. However through testing we had to simplify and change it to work with the java TCP connection. This led us to a very simple phone application. Once we had this done, Tyler also had most of the C++ program working correctly. We attempted to test them, but we ran into tons of issues. This led to us basically working on this differently. Nicholas Collier basically worked mostly on the android side of things, while Christopher Jaress worked on fixing or changing things with the C++ program. Tyler would then work on what we still had to complete (RSA encryption, validation). As Nicholas Collier and Christopher Jaress eventually got the connection to work properly (with the help of Tyler Beveridge's ideas) Tyler would then have our next set of things ready to add in. This made things run pretty quickly and smoothly. We could not really have more than two people working on testing both android and C++ so Tyler's use of time on things we had not completed yet was pretty efficient.

Tyler Beveridge – Ideas, C++ programs, C++ program testing, RSA encryption

He chose to work on the C++ side, because he wanted to make the main server.

Nicholas Collier – Ideas, Android basics/layout, Android testing

He chose to work on the Android part, because he wanted to learn android.

Christopher Jaress – Ideas, Android connection, C++ program testing, C++ programs

He worked on the android part, because we needed more help on that end. He also worked on the C++ section to attempt to fix problems and test things out.

Collaboration

At the beginning of the project we divided up the project into tasks, and each individually worked to accomplish our assigned tasks. We would then meet up every Friday in lab and discuss our progress and address any issues we came up with in design. We would contact each other outside of lab via text message and email to pass along what we currently had and ask questions. Once the project became more advanced we started to meet outside of lab. We would email our progress to each other to back everything up, and upload the programs to our virtual machine for universal access among the group.

Results

The final results is of our project is three programs. We have a phone app, a controller and a door. The phone app is a just a simple app with three text fields and three buttons. We have one button for the ID, one for the door, and one for the door sequence. Our program expects messages in a certain order, id, door number then sequence. Our controller is a master server that connects to the doors and phone. The controller will initially connect to the doors

and then wait for a phone connection. Once it receives a phone connection it will authenticate the connection to make sure that it is actually our program. Once the phone and id have been authenticated the controller will wait for a door number. Once the door number has been received, it will output a light sequence to the corresponding door. The controller will then expect that light sequence back from the phone, and the door will open.