**Project P5 – Implementation Document**

Nick Forquer, William Greer, Gerard Puhalla, Trevor Rambacher

**API / Libraries Used:**

The project was implemented using a variety of add-ons from the android library. The security portion of this was done using portions of the math, security and crypto libraries. These were used to implement the Diffie Hellman key exchange between the two devices once they were connected with either Wi-Fi or Bluetooth. The crypto libraries had specific DH packages to expedite the process. Security gave the ability to generate keys and key pairs necessary to verify a secure connection. Big integer was used in the calculations for the exchange.

The Bluetooth connection was created using several of the Bluetooth packages, including adapter and device (to use the phone’s own capabilities), as well as socket and socketServer to create a connection between the two phones. In a similar manner, Wi-Fi direct used several packages from net.wifi libraries. These were used to find networks and then connect the two devices to one another (through several p2p packages).

Several of the base packages such as os, widget, and view were used for the base part of the app to have a staring page that connects to others.

**Components:**

This application is divided into three sections, each with their different components. We have the code that is shared by the two different file transfer methods, the code used specifically for the Bluetooth transfer, and the code used specifically for the Wifi-Direct transfer.

The Main Activity is shared by both programs and this activity grants the users the necessary permissions to perform the file transfers as well as allowing a user to select which file they want to use in the file transfer. The app pulls these files from the Downloads folder of the sender and will send the file to the Downloads folder of the receiver under the same name. File selection is optional, as the user could be the receiver. The File selection component is constructed using a spinner. The main activity also contains two buttons which allow the users to select either Wifi-Direct or Bluetooth as their method of file transfer.

Also among the shared code is the Diffie Hellman class which contains two different methods. As mentioned earlier, the components used to build this class came from specific DH packages within java’s crypto library as well as using key components from the java’s security library. The class has to different functions. The first function allows users to generate their specific public key part which they will share with the other transmitting the part over Bluetooth or Wifi-Direct. The second function takes the received public key from the other user and uses that with the information that they already know to generate a shared key. This shared key can then be used to encrypt and decrypt messages via a symmetric key algorithm. For our application, our chosen algorithm is AES, but in theory any symmetric key algorithm can be used.

The Bluetooth file sharing was performed in the SetUpBluetooth Activity utilizing the Bluetooth File Transfer class. The user is able to start this activity by selecting Bluetooth in the main activity. From here the user is presented with three buttons, file sender, file receiver, and find devices. First, the receiver starts the server and makes their device discoverable by hitting the file receiver button. Next, the sender finds devices by hitting the file sender button. A list of devices appears. Then the sender selects the receiver’s device and presses the file sender button. The devices will then proceed to pair if they have not already. The file receiver begins the three way Bluetooth connection. It sets up a server thread called the AcceptThread to begin listening for connections. The file sender then attempts to connect to the receiver by obtaining their MAC Address through the Bluetooth discovery process and sets up a client thread called the ConnectThread which connects to the server thread. Once connected, both threads are closed, and two new ConnectedThreads are created. This thread allows the reading and writing of messages through Input and output streams. The reason for the separate thread is to prevent other users from attempting to connect to the server. From this stage we move to message transfer.

The message transfer for Bluetooth is set up in the current manner. First key parts of Diffie Hellman are exchanged over the connection. Then the shared key is composed by both users. Our key is 192 bits for increased security. Next, the sender will send the file name, file size, and file name size to the receiver. Next, the sender encrypts the file using AES with the shared key, then sends the file to the receiver. At this point the sender’s job is complete. The receiver then recomposes the file and decrypts the file using the shared key. The file is then saved on the receiver’s end. The connection is closed by the receiver once they have completed composing the file and encrypting it.

Initially, when the user chooses to use Wifi-Direct to transfer a file at the main page, their phone’s wifi will be enabled to allow for the transfer. The Wifi-Direct file sharing starts with the user being sent to a screen indicating whether they wish to send or receive a file. Based on the user’s choice, they will be granted with the file sending page or the file receiving page. The base of the Wifi-Direct model is that a client service sends a file to a server service. If the send button is selected, the Wifi-Direct client service and client activity are started. The client activity manages the actions of the client, while the client service manages the sending of the file. These establish a Wifi-Direct opening and search for available server activities running Wifi-Direct to connect to. This will show up in the list view on the client activity page. The user can then select that client to connect to, and the activity will start the process of sending the file to the other phone running the server. The phone that selected receive file will start the server, and then simply waiting at the receive file screen for an incoming connection and file transfer. In the background the phone will be running the server activity and service to receive the file. The server activity handles them management of receiving the file and the network interactions, while the server activity receives the file from the client. Both the client and server activities rely on Wifi-Direct broadcast receiver classes to do the middle-work of actually establishing a connection between devices, and what happens when a device interacts with that connection.

**Bugs / Security Holes / Omissions:**

For Bluetooth a man-in-the-middle attack is only possible if the adversary can trick the user into accepting their address rather than the address of the user, or they beat the client to the server first. In either case, the server has to willingly pair with the adversary for this to work. The file name, file size, and file name size are not encrypted. If one intercepts the file size or file name size and changes it, they could interrupt the file transfer and crash the program. However, this is mitigated by the fact that the data stream is encrypted. Another issue that appears with Bluetooth is the fact that if the connection is lost, the whole process aborts. This makes it hardtop transfer large data files as the connection sometimes drops. Another bug that occurs is that the sender will send the incorrect filename size to receiver causing the program to crash.

For Wifi-Direct, a file transfer cannot be established due to the failure of establishing a wifi group connection. The Wifi-Direct documentation states that the two communicating devise need to be running the same operating system, which our team was not able to provide. A connection from the client to the server could be established, but the server could not recognize the connection due to the failure of the initialization of the wifi group. To send a file, the client must know the IP address of the server, which requires that the server be the owner of the wifi peer2peer group. Since the server could not establish the peer2peer group, the client was always put in the position as group owner, and prevented from sending the file. Based on our findings, we believe this to be a device issue, rather than a software one, as the cals to establish a group relate directly to the android Wifi-Direct API.