HW6: First Implementation

by Team Eupheme:
Zack Anderson, Adam Clarke, Eric Happe, Albert Le, Kevin Ngo, Jubal Norman, James Tooze and Jacques Uber

for Customer:
Kevin Tang

on Project:
SMS+

List of User stories:

1) receive message

2) send message

3) mass message

4) read old messages (inbox)

5) delete message

6) ping of success

7) decrypt message

8) encrypt message

9) import / send key

10) sign keys

11) user interface

User Story #1: Receive Message

Requirements:
The SMS+ user receives and is able to read a SMS+ protocol message.

due: week 1

1. Receive file from another device.
Should take minimal time to implement if done after send task 3.

2. Open received file and display relevant information
Should take minimal time to implement.

Programmers: James Tooze and Zack Anderson

Problems encountered: Lack of knowledge programming for the Android platform and using github, both remedied through research (Internet).

Time taken: approximately 2 hours, most of which was learning how to setup the Android programming platform (Eclipse with Android plugins) and other research

Current Status: implemented, minimal hands-on (not automated) testing

Left to complete: N/A

For the user story of receiving text message we (James Tooze and Zack Anderson) were able to find a viable open source application that already did this; namely, the text messaging packages of the Android operating system. We were able to pull the relevant pieces required to receive and read text message from this and have implemented our user story using this code.

This task took us only a few hours spread over two days to implement. We had first considered approaching this problem from scratch but our research showed us that it would prove much easier and quicker to use the existing system. From there we only had to find the relevant sections we needed.

The current status of this user story is complete, barring any errors when attempting to interact with the other user stories, all that's left to do would be a comprehensive test to locate and remove any possible bugs.

User Story #2: Send Message

Requirements:
The user is able to send a SMS+ protocol message to another SMS+ user.

due: week 1

1. Accept user input
Should take a pair of programmers minimal time to implement, time extension may be necessary if programmer does not know how to read input from device.

2. Format input to file
Should take the pair of programmers minimal time to implement.

3. Send file to target device
Day or two to research best means of implementation, 1 day to implement the chosen design. 3 days total.

Programmers: Jacques Uber and Kevin Ngo

Problems encountered: Setting up the Android SDK.

Time taken: approximately 1 hour

Current Status: implemented

Left to complete: N/A

We worked to get the Android SDK to work. We knew that there were “virtual” phones that the SDK offered, but had not tried it out. We were able to get android phones to message each other using the Android SDK. The SDK uses virtual phones to test apps. Right now there is no encryption support. We need to write that code.

User story #4: Read old messages

Requirements:
The user's SMS+ client is able to read messages they have received earlier on their phone.

Due: week 1

1. Store messages on phone
Should take very little time to implement, after file format is decided.

2. Retrieve file from phone.
Again, should take minimal time to implement.

3. Open file and display relevant information.
Very simple to implement after basic send/receive is finished.

Programmers: Albert Le and Jubal Norman

Problems encountered: Lack of knowledge programming for the Android platform and using github

Time taken: approximately 3 hours spent

Current Status: still implementing, no testing done

Left to complete: 90%

User Story #11: GUI

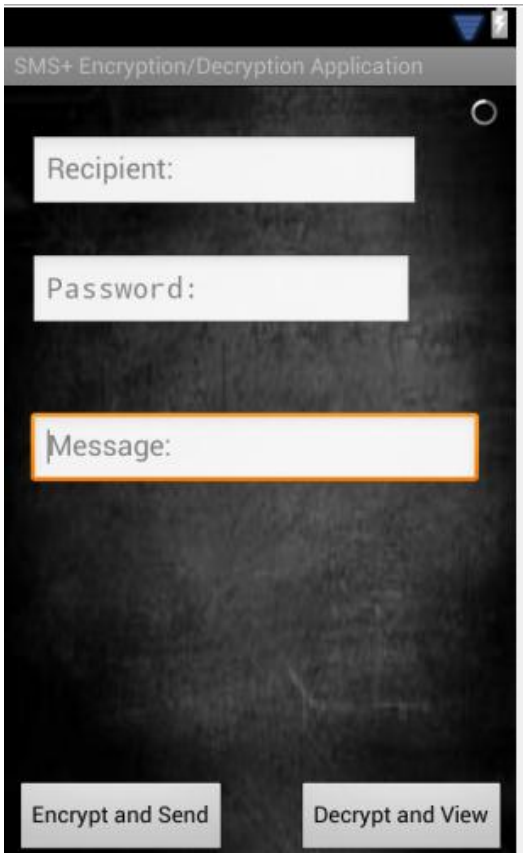
Requirements:
Create user interface in which the user interacts with the SMS+ system.

due: week 1

1. Display relevant data
Should display relevant data on each screen, for example on the send screen the GUI should (minimally) display a textbox for inputting the destination phone number, a box to enter the message text, and a send button.

2. allow user to interact as necessary
Should allow the user to use the hardware or software keyboard to input into text boxes, and use the touchscreen or directional-pad as well as other buttons to navigate, make changes and selections.

The GUI for this application is written in XML. It will allow the user to choose recipients, enter their password, compose a message, encrypt and decrypt messages.



Programmers: Eric Happe and Adam Clarke

Problems encountered: Lack of knowledge programming for the Android platform and using github

Time taken: 4 hours spent, mostly setting up and understanding Android programming

Current Status: implemented, but needs to be tied in with other use cases

Left to complete: N/A

GUI XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/AbsoluteLayout1"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/droidwallpaper" >

   <ProgressBar
      android:id="@+id/progressBar1"
      style="?android:attr/progressBarStyleSmall"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="292dp"
      android:layout_y="8dp" />


   <EditText
      android:id="@+id/editText3"
      android:layout_width="280dp"
      android:layout_height="wrap_content"
      android:layout_x="17dp"
      android:layout_y="203dp"
      android:hint="@string/message"
      android:inputType="textMultiLine" >

      <requestFocus />
   </EditText>

   <Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="183dp"
      android:layout_y="433dp"
      android:text="@string/decrypt" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="7dp"
      android:layout_y="433dp"
      android:text="@string/encrypt" />

   <EditText
      android:id="@+id/editText2"
      android:layout_width="239dp"
      android:layout_height="wrap_content"
      android:layout_x="16dp"
      android:layout_y="104dp"
      android:hint="@string/password"
      android:inputType="textPassword" />

   <EditText
      android:id="@+id/editText1"
      android:layout_width="243dp"
      android:layout_height="wrap_content"
      android:layout_x="16dp"
      android:layout_y="30dp"
      android:hint="@string/recipient" />

</AbsoluteLayout>
```

Diagrams:

Was the spike or diagram useful? Why or why not?

- The spike (installing Eclipse with Android plugins) for the receive user story was useful since it allowed us more time to understand how to program for the Android platform
- The UML sequence diagram was not so useful because understanding how a basic SMS text message would be sent or received isn't really worthy of a diagram (IE it's easy to understand)

Were there any diagrams that you *wish* that you had? Why or why not?

- N/A, the diagrams weren't much of a problem for these user stories. We spent most of our time figuring out how to program for Android as we are all new to this.

Refactoring:

We spent some time refactoring the basic Android texting code to make it work by itself (IE not within the context of the entire phone) but since the code is already functional from the resources we found online (mainly the Android SDK and its documentation websites), it did not need much refactoring. We expect to need to do more code refactoring at the next stage of the project when we begin implementation of the encryption.

Integration Testing:

We have not had need to conduct any unit tests yet as this section proved to be quite rudimentary. We did however have some difficulty locating the exact sections of code we needed within the larger scope of the operating system. We did per-user story tests of our code using the built-in Android Manager Virtual Phones, however we have yet to do any integration testing between the different user stories due to time constraints and lack of knowledge on how to put the parts together effectively and efficiently.

Customer Meeting

Our customer spoke with our team in class rather than via email; the customer was reasonable about the work that should be done, and the timeframe in which it is due. We renegotiated with our customer on the priorities of the project: we felt as though we should get the basic SMS (send, receive, inbox, delete) working before we began implementing the encryption and ping of success parts of the project.

Priorities and Work Breakout (for coming week):

Implementation of project will be done using the Android API and SDK (therefore the Java programming language).

Implementation of mass message and delete message

Priority 2

Assigned members: Zack Anderson and James Tooze

Implementation of ping of success

Priority 1

Assigned members: Jubal Norman and Albert Le

Implementation of decrypt and encrypt message

Priority 2

Assigned members: Adam Clarke and Eric Happe

Implementation of import, send and sign keys

Priority 1

Assigned members: Jacques Uber and Kevin Ngo

Creation of powerpoint and demonstration for class

Priority 3

Assigned members: Zack Anderson and James Tooze

Presentation to class:

Priority 3

Assigned members: (whole team)

Team Responsibilities:

- Implement User Story 1: James Tooze and Zack Anderson
- Implement User Story 2: Jacques Uber and Kevin Ngo
- Implement User Story 4: Jubal Norman and Albert Le
- Implement User Story 11: Adam Clarke and Eric Happe
- Diagrams and Refactoring: (whole team)
- Integration Testing: (whole team)
- Customer Meeting and Revised Estimates: (whole team)
- Priorities and Work Breakout: (whole team)
- Compilation and Formatting: James Tooze