Mobile Computing Deliverable

Written and worked on by

Fayang Pan – Project Leader

Sam Evans-Golden – Project Assistant

Kevin Davison- Project Assistant

Project Features

This project is designed to encrypt messages as well as decrypting them as well. The project was based off of the well-known Caesar Cipher, which shifts the letters the user inputs, which in turn creates an unidentifiable string of text. The key is the operator in which tells the program how many letters up or down the message should shift, inputting the key backward also lets the program decrypt the message as well.

Usage

This project is to demonstrate how encryption and decryption work. This is useful for both tech and non tech-savvy people to really comprehend the mechanics that go behind how powerful a cipher can be. Another use of this project is for users to have fun sharing encrypted texts to one another and create a fun, and secret atmosphere!

This app encrypts and decrypts any and all messages you give it! Ever wanted to send secret codes to your friends? Now you can! How it works is similar to the Caesar Cipher, which shifts the letters you write to other letters in the alphabet. We use a more complicated key implementation where the person you send must have the same key as you do to decrypt your message.

Simple Set up:

Before you can begin sending your hidden messages to your friends you must first create a password to protect your keys.

Now you are able to add your very own unique key! Just give it a name and randomly generate a key, or if a friend sent you one, paste it to the key space and now you can decrypt their messages.

After that you're all set! Happy Crypting!

User Manual:

Upon using this app for the first time, go to "Manage Keys" from the main screen. A window will pop out, asking for password, security question, and answer to that. After you set it up, the app will ask for your password every time you want to do anything to your keys. If you forget your password, enter a wrong password, and you will have the option to answer the security question you set up. If you have the right answer, your password will be shown in a bubble.

After you set up your password, you should be in a screen with two options: "Add a new key" and "Manage your keys". Go to "Add a new key", and put the name of your new key in the slot below "Enter name". Then, if you want to generate a new key, press "New Key" button. If you got a key from your friend, paste the key into the slot below "Enter key". After that, press "Enter". If your input is valid, a "Key Added" bubble will show up. If not, an error will pop out, and please check if the key is correct. NOTE: THE NAMES OF THE KEYS ARE UNIQUE. PLEASE DO NOT ASSIGN THE SAME NAME TO DIFFERENT KEYS.

You can see your key after you click the "Manage your keys" button. If you wish to delete any key, just swipe the entry to the left or right, and it will be deleted.

Assuming the name of your new key is "alpha", and you wish to encrypt some text with that key. You can copy the text (Currently only English is support) and paste it in the textbox after clicking "Encrypt". After pasting, click "Encrypt" in the bottom right corner, you will be asked to input the name of your key, "alpha". A bubble which shows "alpha was found!" will pop out if the key is found. From there, you will find a whole paragraph of random symbols, and two buttons saying "Copy Key" and "Copy Text". If

you want to send your key to your friend, you can click “Copy Key” and paste the key in your email or any other form and send it away. If your friend already has the key, you can just click “Copy Text” and send the text via any form you wish.

If you wish to decrypt a text from your friend, and the friend uses the “alpha” key as well, you can do it in the “Decrypt”, and the procedure is similar to the encryption part.

Features, Design and Implementation

This project uses a plethora of features both in the user-interface and behind the scenes. In the user-interface the home screen greets the user with three buttons, which take the user to different parts of the app. The “Encrypt” and “Decrypt” buttons take to their respective parts, which ask the user to input a key the user created. The “Manage keys” button takes the user to another activity, which then has two more buttons. The “add” key leads to another activity in which the user gives a personalized name to the key and can auto-generate a key or paste a key another user has given them. The “Manage your keys” button takes you to a List View of all the user’s keys they have created, in which they can use a swipe left gesture to delete the key. Behind the scenes there is a myriad number of activities and fragmented dialogs to input key names for the encryption and decryption process. All of the keys are stored in a 2-column SQL database.

Development Log

Github setup:

We had problems with system configurations and how to solve merging conflicts. We searched for the tutorial, went to online forums and asked friends for help.

The Database:

Problems arose when we began to implement the ListView aspect of the database. We fixed this by looking up tutorials and other people who had similar problems on stackoverflow.com

Creating a key seemed harder than previously thought. It took many trial and errors to find a method that worked. Deleting key was also troublesome because you have to check for empty database. We try different methods before finding one that works.

We wanted to set up an easy way to delete key, but we do not know how to do it in the ListView. So we found a Github library developed by an Android developer, copied and pasted the code to our project, and there we have the swipe to delete function.
As we do not want illegal input in the database to cause trouble encrypting/decrypting the text, we spend a long time writing the checks for valid input.

Putting it all together:

As we used different platforms for coding, merging of different codes was hard to us. We had to change the Manifest file. Lots of things need to be added to everyone's part to make easy transitions between each activity, we did so by adding several buttons.

Paste function:

Android let you copy many different data types, so paste has to handle them all. However, the app should only accept text. Eventually, we referred to Android Developer website for help.

Layout:

Relative layout doesn't scale as what we though. This is a current problem; the app looks good on some devices, and weird on others.

Encryption/Decryption:

Because of non-printable ASCII characters, it took some time to write all test cases before we are certain that all printable characters in ASCII are accounted for. Currently, it only decrypts/encrypts characters in ASCII, and return the same characters for others. UTF-8 seems to use hexadecimal representation, so we aim to decipher that in the future.

Password and other user login functionalities:

As AlertDialog only allows one TextView/EditText, we had to create two more class for different DialogFragments.

We wanted to set up a contact email in case one forgets his/her password, but that requires SMTP server and all kinds of authentications. We had to change it to security question and answer.