Duncan Fisher

5/02/17

project 3

**CryptoMessenger**

The goal of CryptoMessenger is to provide a secure, easy to use, and visually appealing messenger application for Android devices. The target audience for this application is anyone interested in cryptography, color theory, texting, or secure communication over a network via encryption. The inspiration for CryptoMessenger was originally my interest in cryptography, but I decided to include dynamic color schemes as an extra feature.

In researching how I might implement dynamic color schemes I found multiple sources on color theory and algorithms to generate visually appealing color palettes. I originally intended to use an open source API to generate random colors but it seemed feasible and even fun to try and implement my own color algorithms. The primary sources I found for color theory were mostly blogs ranging from light discussions of color design to rigorous studies of color theory with actual implementations of algorithms. Some of the more technical articles were:

* <http://devmag.org.za/2012/07/29/how-to-choose-colours-procedurally-algorithms/>
* <http://martin.ankerl.com/2009/12/09/how-to-create-random-colors-programmatically/>

The second source was the algorithm that I implemented for this phase of project development. It is a simple algorithm for generating nice looking, high contrast colors. I used this to generate colors dynamically for each key in the key list activity.

My research for the cryptographic parts of the application included studying documentation for the Java language security libraries and examining other open source cryptography apps to see how they used the libraries to encode and decode messages. I already have a solid background of knowledge of cryptography theory but I reviewed encryption methods and different forms of key exchanges including RSA and Diffie-Hellman. Also I researched common attacks on cryptographic methods. These sources were:

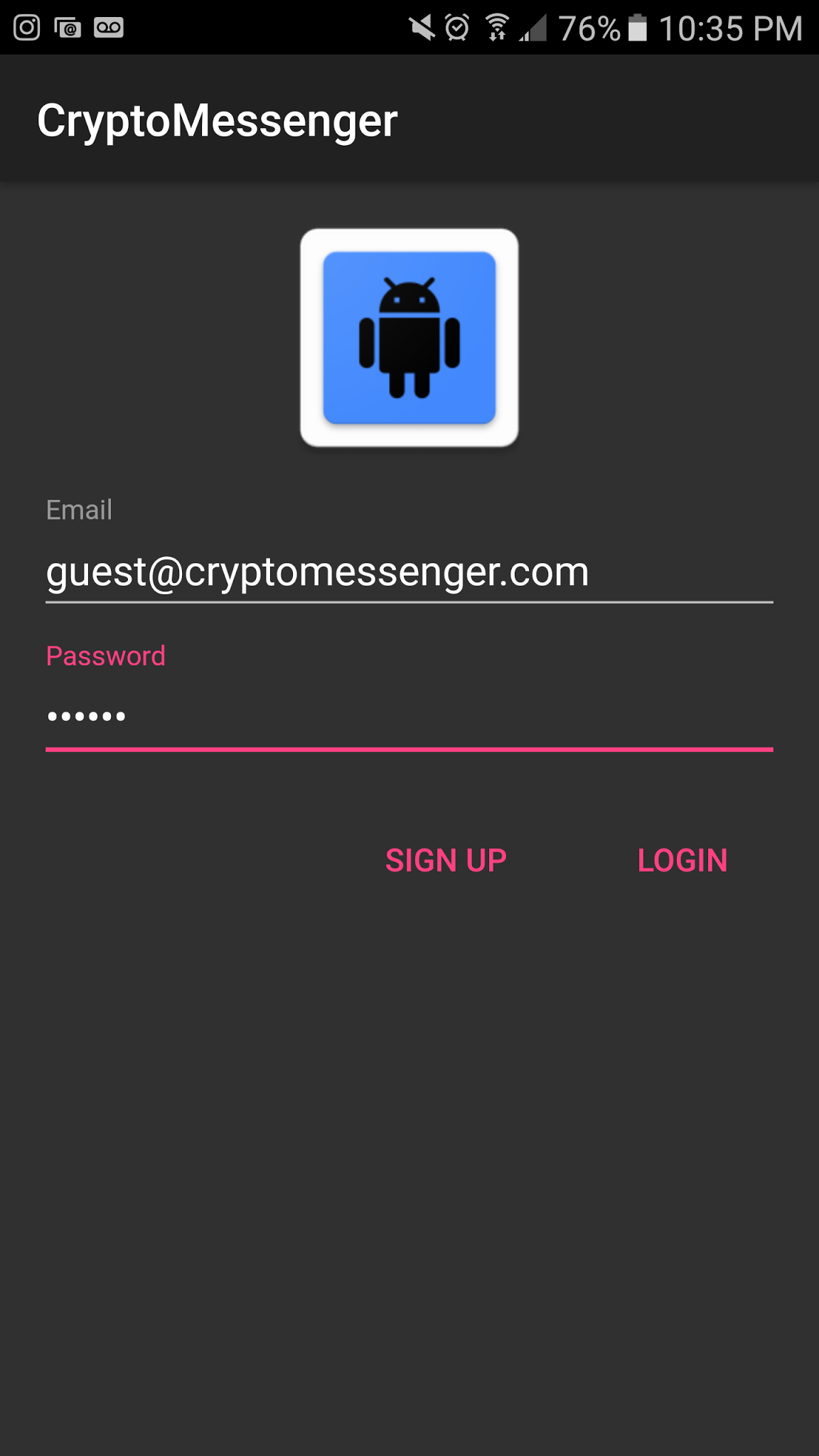
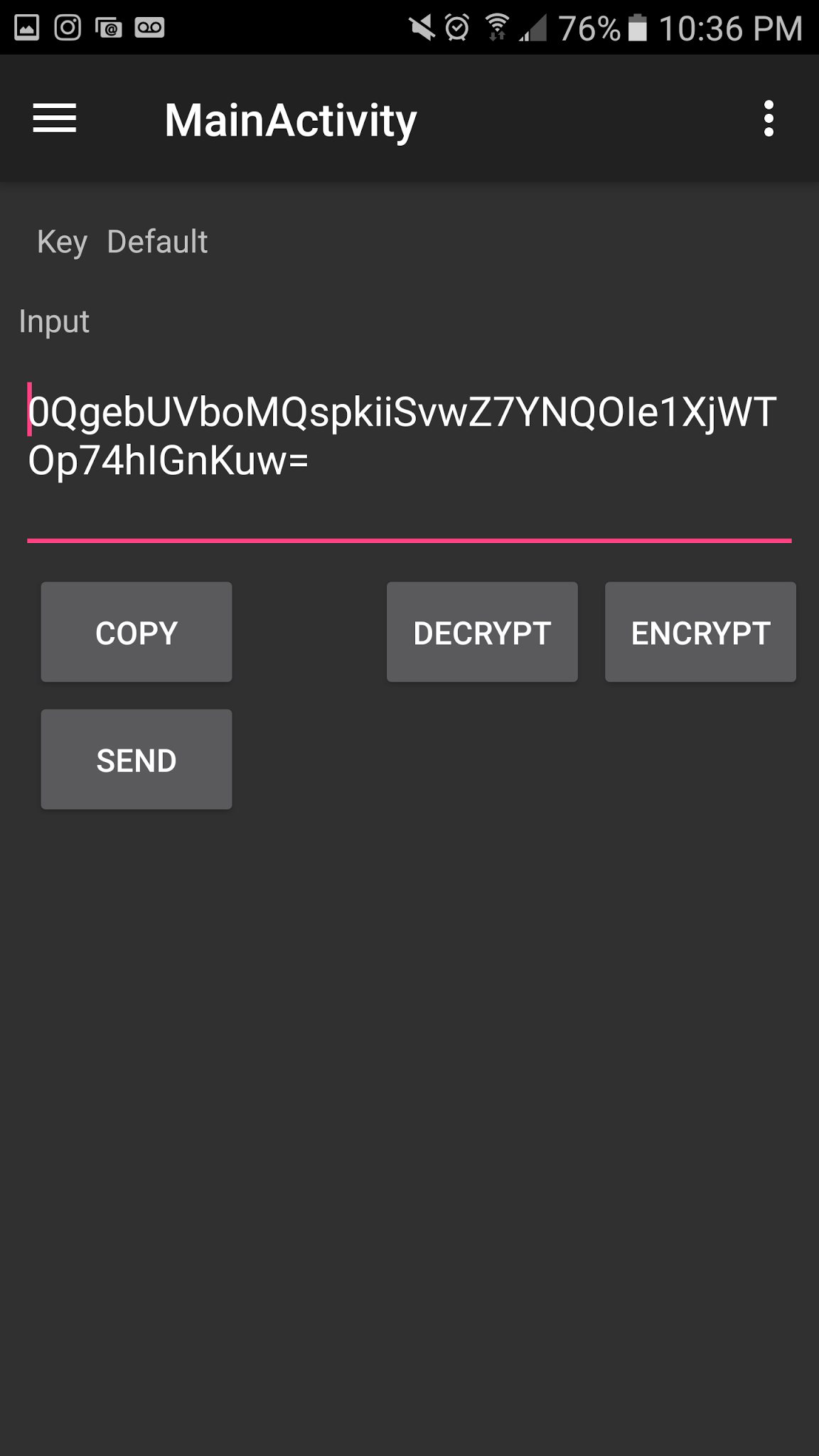
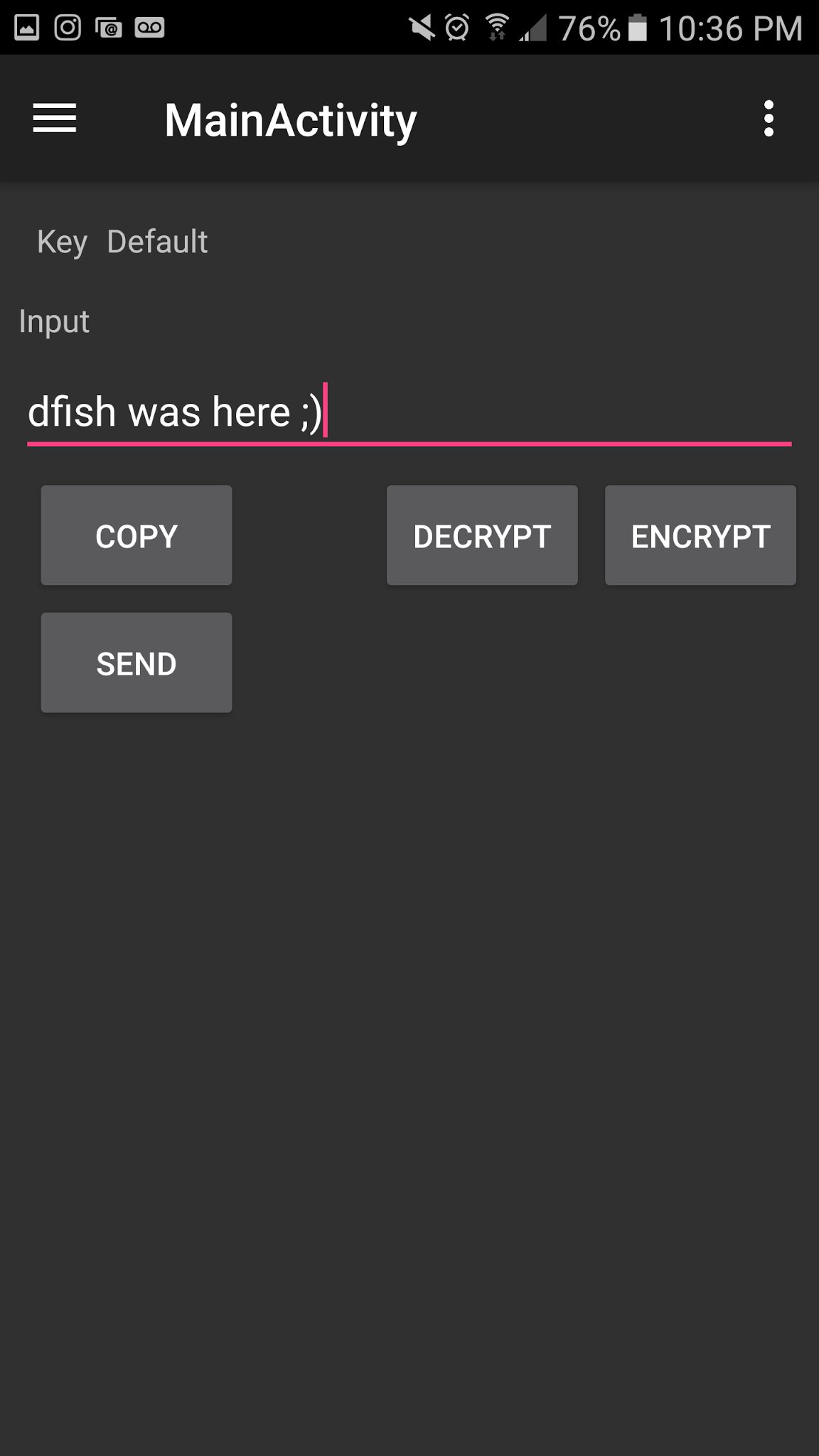
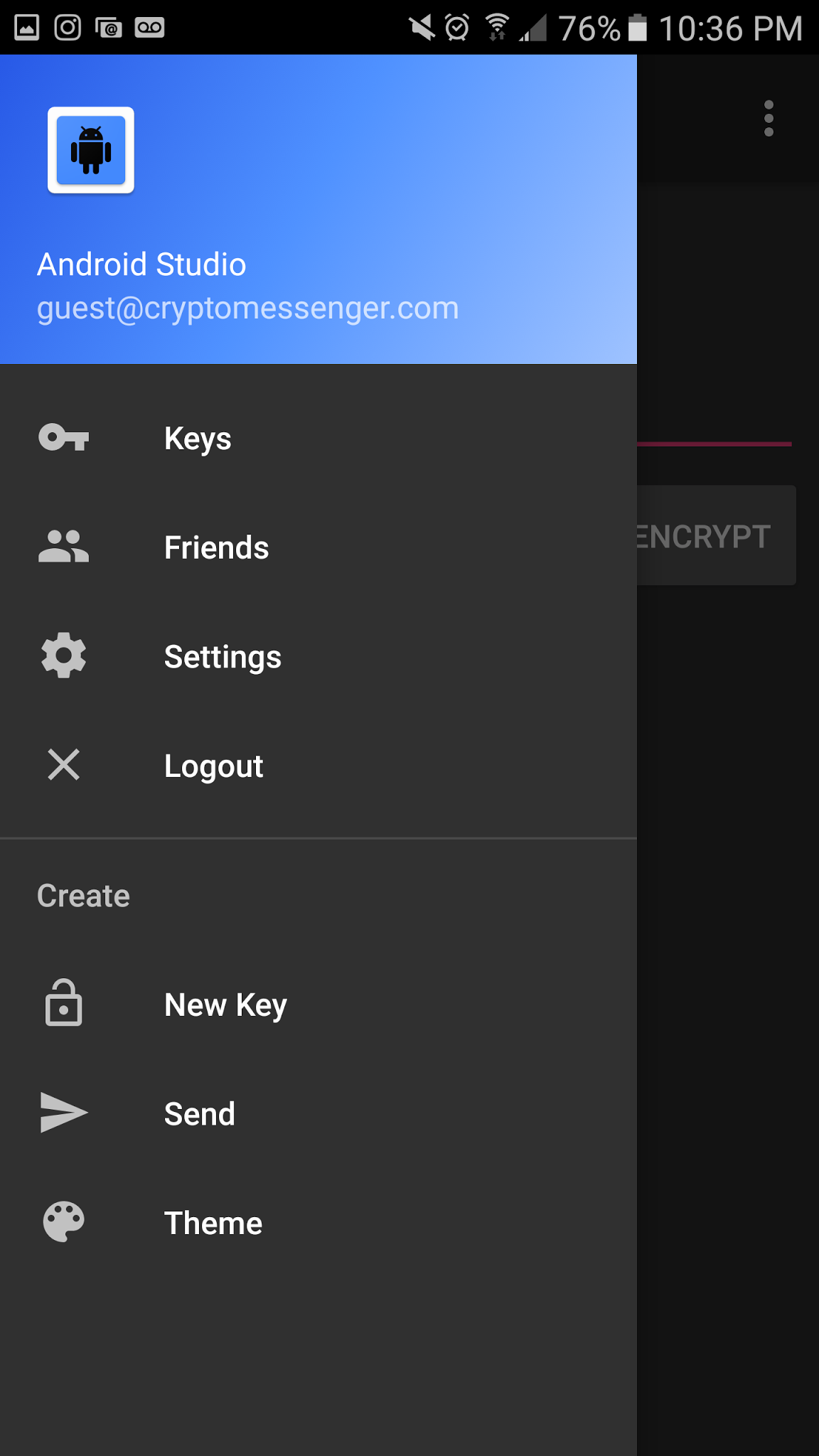
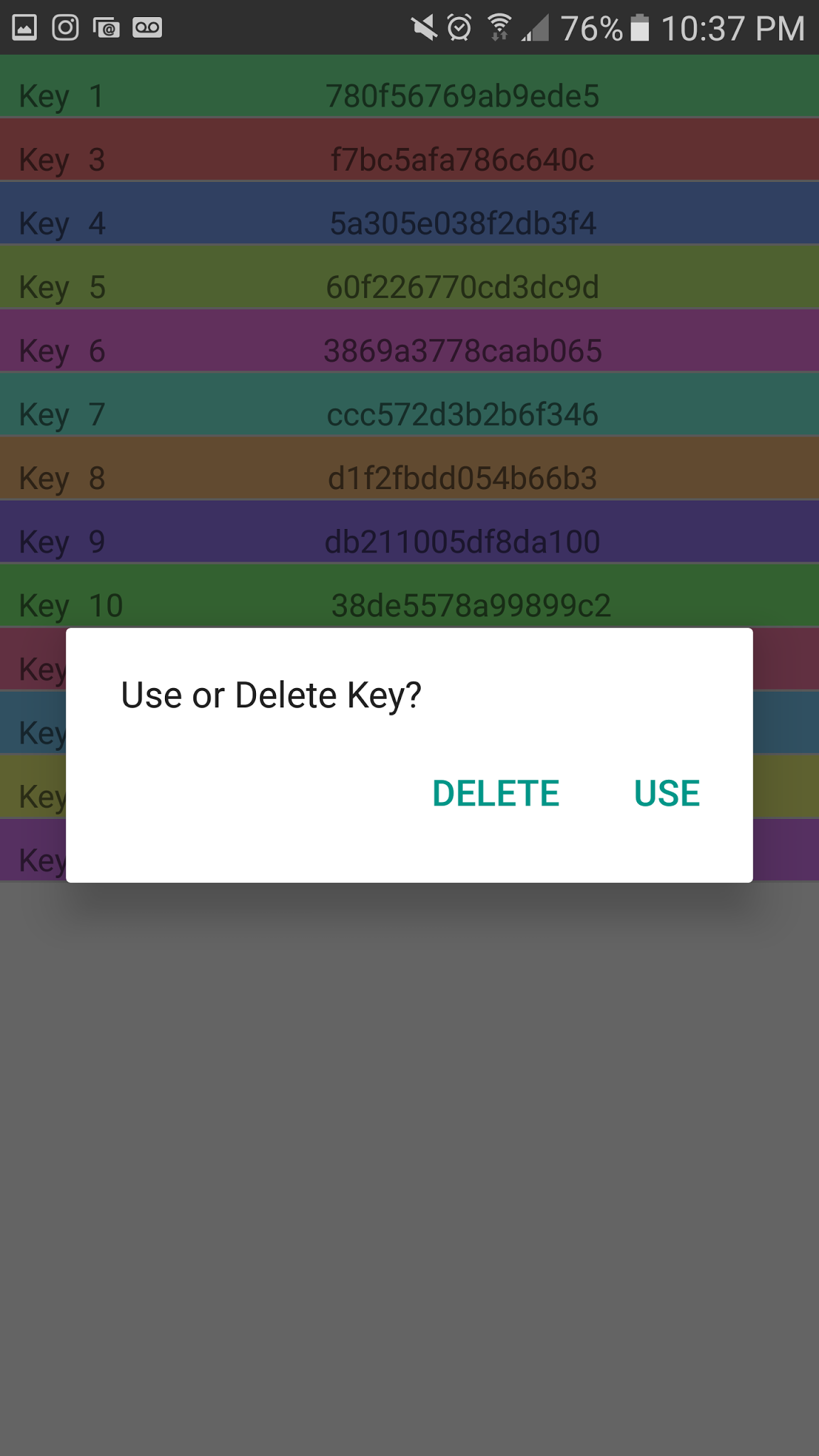
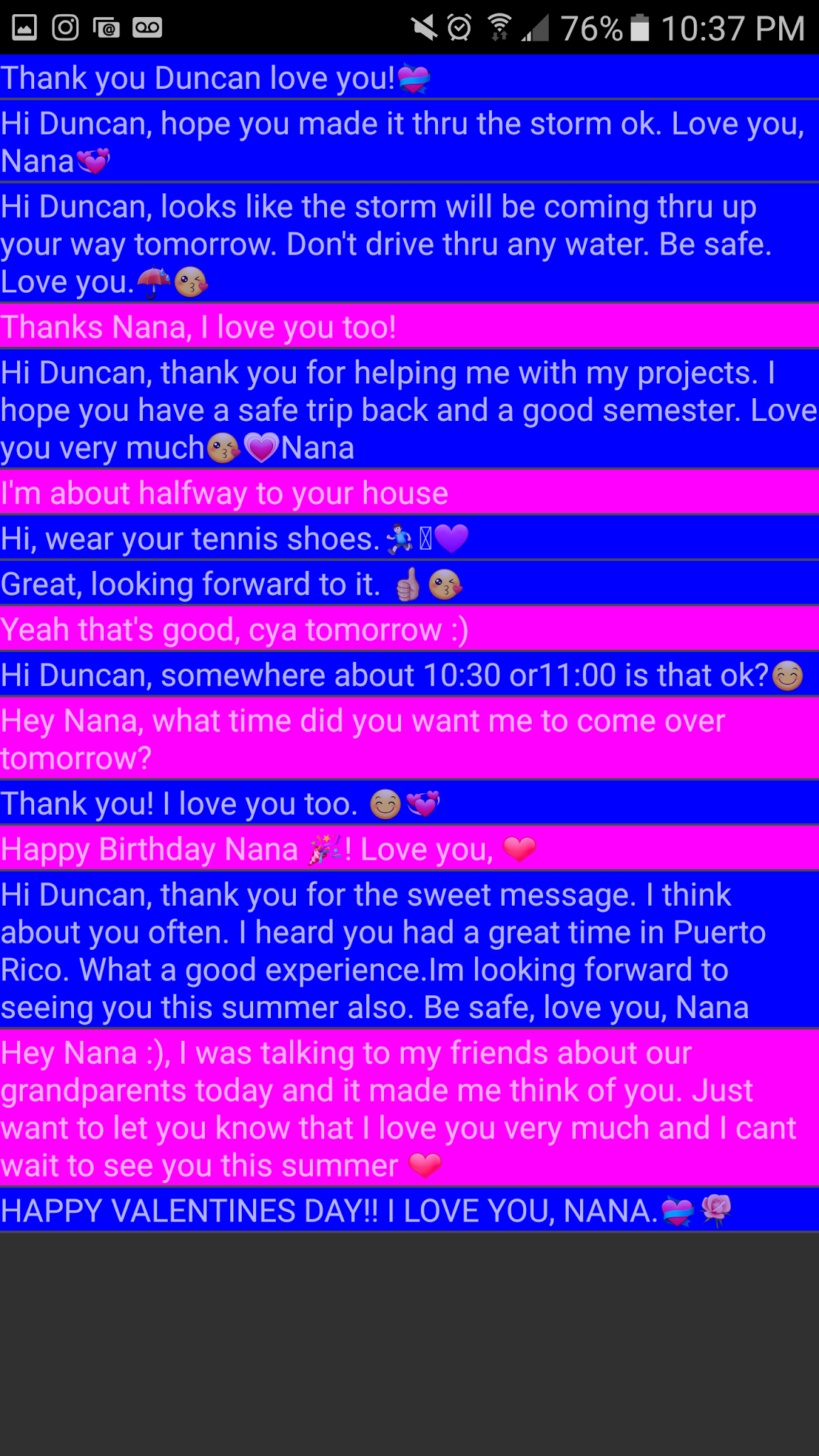
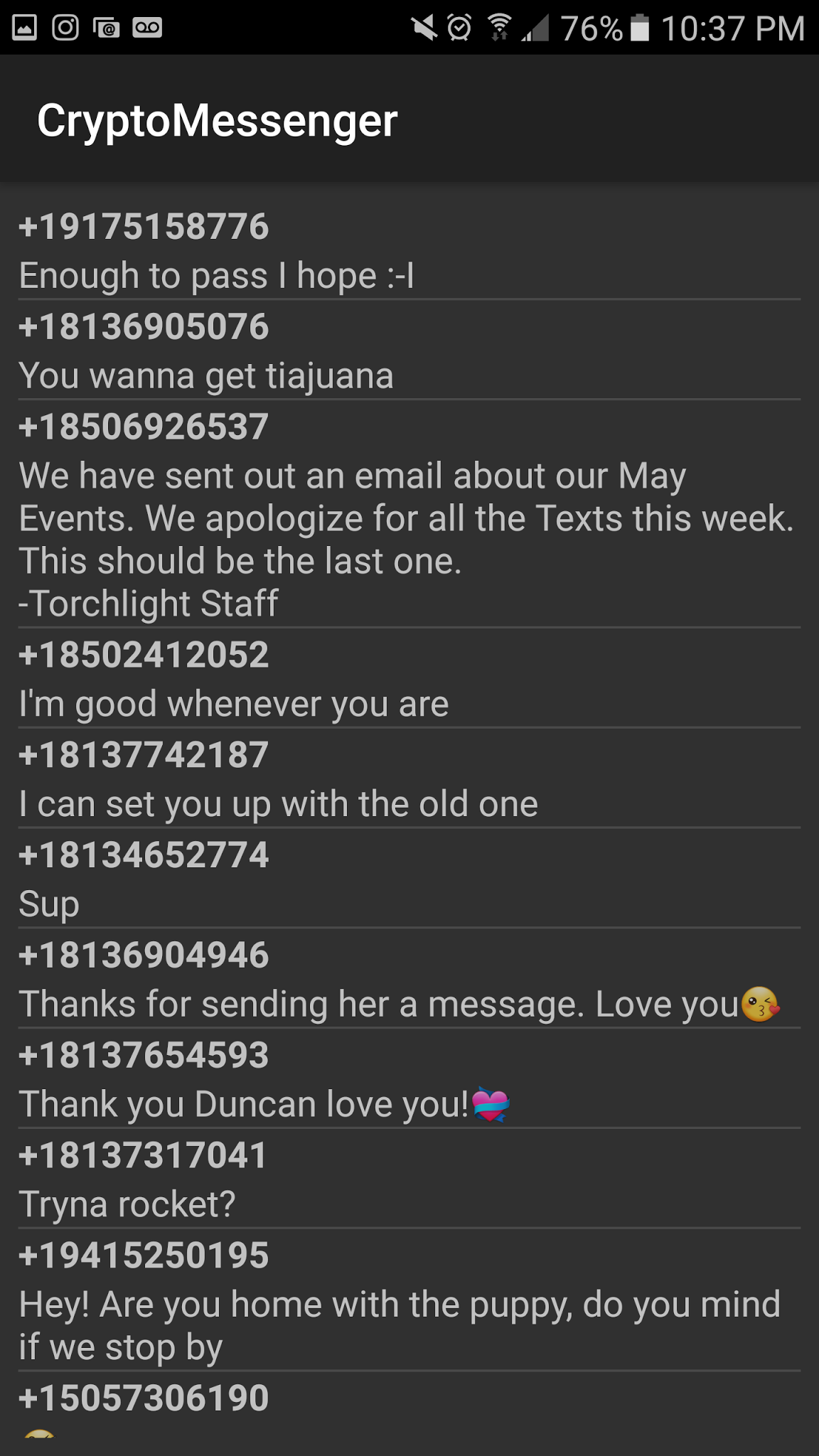
* <http://docs.oracle.com/javase/7/docs/technotes/guides/security/>
* <https://github.com/aidin36/simpletextcrypt>
* <https://www.youtube.com/watch?v=YEBfamv-_do> (Diffie-Hellman)
* <https://en.wikipedia.org/wiki/Birthday_attack> (Great source!)

The open source app above is meant to be used as a means of encrypting private information and storing it for personal use, not for messaging with friends, but I did glean valuable knowledge from reviewing the app. For example, I did not know Java came with security libraries; I initially intended to implement not-so-secure cryptographic ciphers on my own but I figured I might as well try to make CryptoMessenger as secure as possible. One of the pros of this app is that it allows you to use your own passwords to encrypt and decrypt text, however this comes with a decrease in security. CryptoMessenger aims to auto-generate secure keys that can be shared with friends and used to communicate securely over a network.

The development timeline for this app thus far has been, in order:

1. Research topics and come up with a list of features.
2. Design UI framework, including a main screen with a navigation drawer.
3. Implement basic encryption/decryption with a default key (similar to SimpleTextEncrypt)
4. Create a content provider to hold persistent application data i.e. Keys and Contacts.
5. Provide methods for generating keys and storing to content provider.
6. Develop UI elements to allow users to display a list of contacts and keys
7. Implement custom color algorithm for key display and link custom keys to text encrypt/decrypt functionality.
8. Connect to Telephony.Sms provider to display text threads.
9. To be continued...

The features that are currently implemented are:

1. Firebase user authentication
   1. This feature was added early on and it is becoming apparent that it is not strictly necessary to have Firebase user accounts, especially if messaging is going to handled via SMS/MMS.  
      
2. Simple text encryption/decryption
   1. This feature is in an awkward stage. I am trying to move from password based encryption to actual secure private/public key encryption so for now this part of the app just uses a default key.  
      
3. Navigation drawer
   1. This feature facilitates moving easily between activities.  
      
4. Custom colors generated with color theory based algorithms
   1. Key list activity loads with new colors each time you start the activity.
   2. Uses the golden ratio method described in the blog by Martin Ankerl to generate a list of high contrast colors based on an initial random seed.
   3. Can choose to delete or use a key to encrypt/decrypt text in the MainActivity.  
      
5. Integration with Telephony.Sms provider to display a list of text message threads
   1. Click on a thread and load all the texts from that conversation.
   2. This part of the project was confusing as I am not sure how to connect the thread\_id and contact information found in the Telephony.Sms provider to the Contacts provider. For this reason, I am only able to display the phone number and not the contact’s name.  
      

This project currently makes use of the Google Firebase API to allow user authentication. However, this feature seems unnecessary so future iterations will probably do away with it. I intend to use the [RANDOM.ORG - True Random Number Service](https://www.random.org/) API in the future to ensure random seeding for key generation and also random colors.

The feature that is entirely incomplete is messaging and exchanging keys between users. The reason this has been challenging to implement is the complicated relationship between the Contacts provider and the Telephony.Sms provider. I am unsure how to go about handling communication between users, and I want to do more research before I make a decision. One concern is expandability: is SMS/MMS broad enough to handle all of the communication features I eventually want to implement like steganography which involves sending images?

Another feature that is lacking is the dynamic themes that I promised. For this feature, I also need to do more research to see how a can dynamically change themes for the entire application and how I might store persistent theme data. Also I need to find a good algorithm for producing visually appealing color palettes of 3-5 colors and not just a list of high contrast colors. Or perhaps I can find a good API but I have yet to find one.

To conclude, there are still many more features I need to complete but I feel the progress I have made, given the time constraints and the fact that I am working alone, is satisfactory. I am very interested in the topics that inspired this application and I intend to continue working on CryptoMessenger even after the project deadline. My hope is that I will have it working well enough by the end of this summer to post it on F-Droid or even the Google Play Store.