# Encrypit

Calculator Password Vault

Milestone

Steven Guarino

## Abstract

The goal of this application is to provide the user with encryption and decryption of a list of text in an encryption app hidden behind a calculator application. The initial plan is to build a functioning calculator using Android's SDK and have it resolve basic mathematical expressions if the user hit's "=" while the correct passcode is in the calculator's expression field. Once through the calculator the user will have access to a text field wherein they can encrypt and decrypt the contents of said text field.

## Introduction

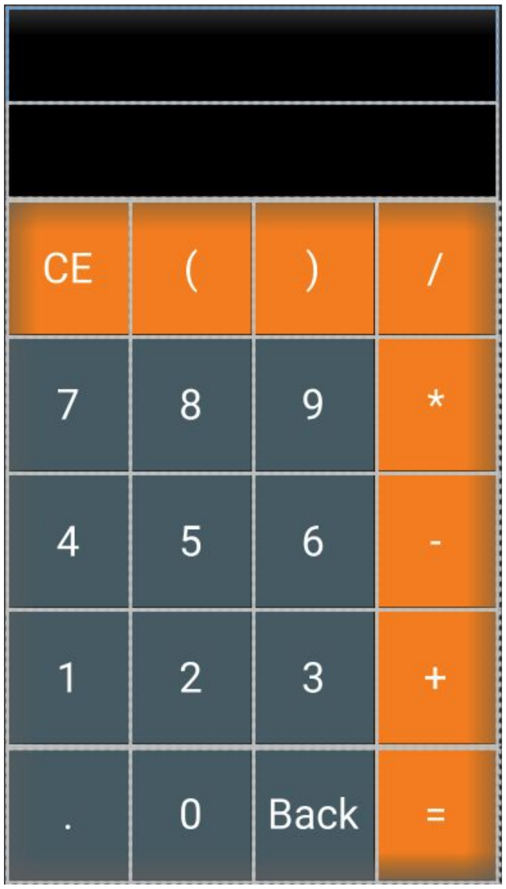
This application was built with a user in mind that maintains a list of passwords on their phone, or someone who simply wants secret, private notes. This application will provide a user with the peace of mind knowing an attacker would not even know where to look for their passwords, instead of them just being in the default notes app, as some less diligent users are known to do.

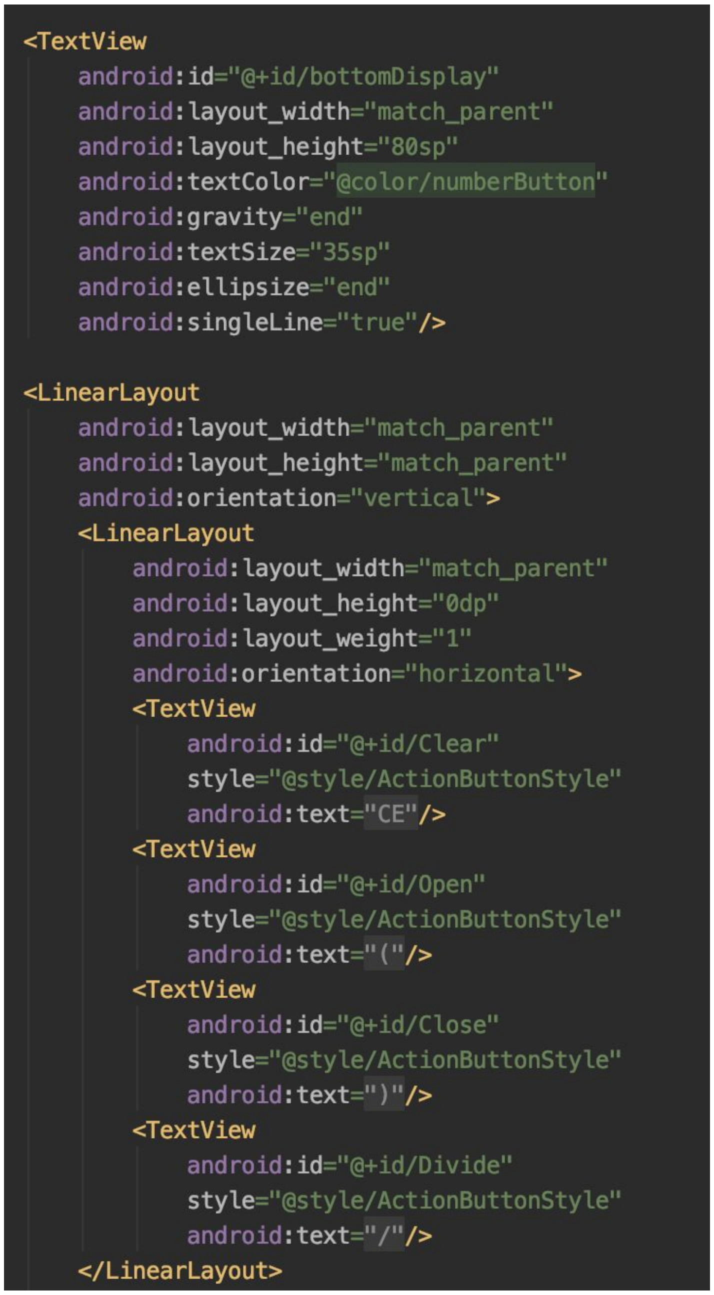
The calculator must check on every "=" if the expression field matches the secret passcode. If it does not it operates as a normal calculator would. Otherwise the user is passed through to the encryption/decryption application.

## Methodology

The application will be built using Android, and written in kotlin. So far the calculator is implemented in kotlin and functional. The calculator was built using a series of text views nested into rows. This approach makes button generation very easy and clear. Calling these buttons from the kotlin main activity file allows the capture of button presses. The library used for resolving the simple mathematical expressions implements Edsger Dijkstra's "shunting yard" algorithm.

Thus far the calculator has been assembled and is calling the second activity that is planned to house the encryption/decryption application.





```xml
<TextView
    android:id="@+id/bottomDisplay"
    android:layout_width="match_parent"
    android:layout_height="80sp"
    android:textColor="@color/numberButton"
    android:gravity="end"
    android:textSize="35sp"
    android:ellipsize="end"
    android:singleLine="true"/>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/Clear"
            style="@style/ActionButtonStyle"
            android:text="CE"/>
        <TextView
            android:id="@+id/Open"
            style="@style/ActionButtonStyle"
            android:text="("/>
        <TextView
            android:id="@+id/Close"
            style="@style/ActionButtonStyle"
            android:text=")"/>
        <TextView
            android:id="@+id/Divide"
            style="@style/ActionButtonStyle"
            android:text="/"/>
    </LinearLayout>
```

## Discussion

There is some concern for the storage of the calculator password, as it is hardcoded and can be easily found. Storing this in a more secure way will be something I'm looking into as I move forward. As well as the plan for encrypting/decrypting the incoming string values. This will most probably be executed via Android's Cryptography library and Keystore system. Within Android's keystore system there is an option to have the user define the name of the key alias, this will likely be used to ensure the user enters the same key for decryption as encryption, otherwise decryption would fail. This could add another layer of security to the calculator vault. The Keystore system also enables password protection of keys stored within their container. The user could be prompted for a pin or even fingerprint ID before decryption could be successful. These are things in consideration and development

## Conclusion

Thus far the calculator has been successfully operational and now what is left to do is implement Android's Cryptography library and Keystore system to generate keys based on user input, decrypt and encrypt the contents. All while reading and writing these contents in and out of the same file. Once the data is encrypted the user will see the encrypted bytes displayed to them unreadable, only until after the correct key alias is given would the message be decrypted properly and displayed to the user.

## References

https://www.objecthunter.net/exp4j/