Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name:
Student ID:
Supervisor: Michael Bradford

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:_____

Date:_____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

National College of Ireland
BSc in Computing
2016/2017

John Noone
X13360866
X13360866@student.ncirl.ie

GPS Encipher

National College of Ireland

Table of Contents

Executive Summary

The problem being addressed is to find a way to create a prototype for GPS based encryption of mobile data. My initial approach was to create an application that could encrypt a secret message using the combination of a personal password and current GPS coordinates.

The original problem presented several challenges. One of which was to figure out what way should I incorporate the coordinates into the encryption process. My first thought was to store the coordinates and encryption key password in plaintext database. I subsequently concluded that my original idea was unsecure and needed to be implemented differently. The final solution was to use a symmetric encryption algorithm with a customized encryption key.

My solution to address the problem is the GPS Encipher application. GPS Encipher is a native security android application which allows users to create a customised encryption key using a personal password and current GPS coordinates to create a composite key. The composite key will be used in a cryptographic algorithm to transform normal text into unreadable cipher text. To decrypt the text the user needs to return to the original encryption location

The custom encryption key will re-define the way in which the encryption process occurs. It's common to see the key using a personal password as opposed to the combination of GPS Coordinates being encrypted with a custom password to create a composite key hasn't been done before. This custom process makes the location in which you encrypt and decrypt the data a vital part of the process. That is what makes the project so different and really sets it apart from the other encryption projects available on the market.

GPS Encipher also includes a demo of the RSA public key encryption algorithm along with the ability for users to share ciphertext using their desired social media app.

1 Introduction

The purpose of this document is to set out the technical details and requirements for the development of the GPS Encipher application. GPS Encipher is an android application which allows the user to encrypt data with cryptographic algorithms but also as an added layer of security requires the GPS coordinates of the location of encryption. This basically meaning that you must return to the location you encrypted the data to decrypt the data or have retained the GPS coordinates. Users will be able set the accuracy of the GPS coordinates used in the encryption process. They will also be able to navigate all the files and apps on the phone and choose which ones they would like to decrypt and encrypt.

The intended customers are android phone owners who are security conscious and looking to add an extra layer of protection to their files and apps on their android device.

1.1 Background

The background of the idea is to add and extra layer to already secure encryption methods. The main reason I want to develop this application is because I wanted a way for people to have and feel more secure about the data stored on their phones and make it more difficult if not impossible for hackers to access the user's data. Also in my opinion, it was the most challenging project for me and I am always up for a challenge.

I have zero experience in using any cryptographic algorithms and this project will provide the perfect platform to gain and increase my skills to the test and challenge myself. After speaking to my supervisor about my chosen project "Prototype for GPS based encryption of mobile phone data", on what direction to go with the project we agreed that there are main factors to the project. They are the encryption of text using a password and GPS coordinates.

These factors must be individually solved. This will allow me to join the all the separate parts together for the product.

The initial recommendation from my supervisor was to read Applied Cryptography by Bruce Schneir, to which I purchase the book soon after the conversation. This book will be a vital tool in learning how to apply cryptography and encryption in my project. The unique trait about this app idea is the fact that the possibility of cracking the standard encryption is a possibility, but finding the exact geolocation of where the data was first encrypted makes it extremely hard to crack the code. This combined method of encryption will make it more secure for people to encrypt data on their phone.

1.2 Aims

The scope of the project is to develop a fully functioning android application which will encrypt text and user's files, whilst using GPS coordinates and a password as an added layer of security. The android application shall require the user to return to the location where the data was encrypted and supply their password to decrypt the data. All information shall be stored in an encrypted database such as the key which will be a combination of the GPS Coordinates of the encryption location mixed with the user's personal password.

I also aim to have more than one for of encryption in the application. This idea of some form of public key encryption would bring the application to the next level. If I have time geo fencing an area using the GPS coordinates would be also one of the finishing touches but not necessary to make the application function which is the same view thing with the public key encryption.

1.3 Technologies

I will be using all the technologies listed below:

Android Application

- Java language
- Android Studio using intellij IDE.

Testing

- Android Studio Expresso (Android Testing Framework)
- Android Studio Unit Testing

1.4 Definitions, Acronyms, and Abbreviations

AD: Another Definition

GUI: Graphical User Interface

Encipher: Encryption

AES: Advanced Encryption Standard

GPS: Global positioning system

UI: User Interface

Symmetric Encryption: Single key encryption.

Asymmetric Encryption: Public and private key encryption.

2 System

2.1 Requirements

Below are functional requirements that were discussed and provisionally agreed with my project supervisor.

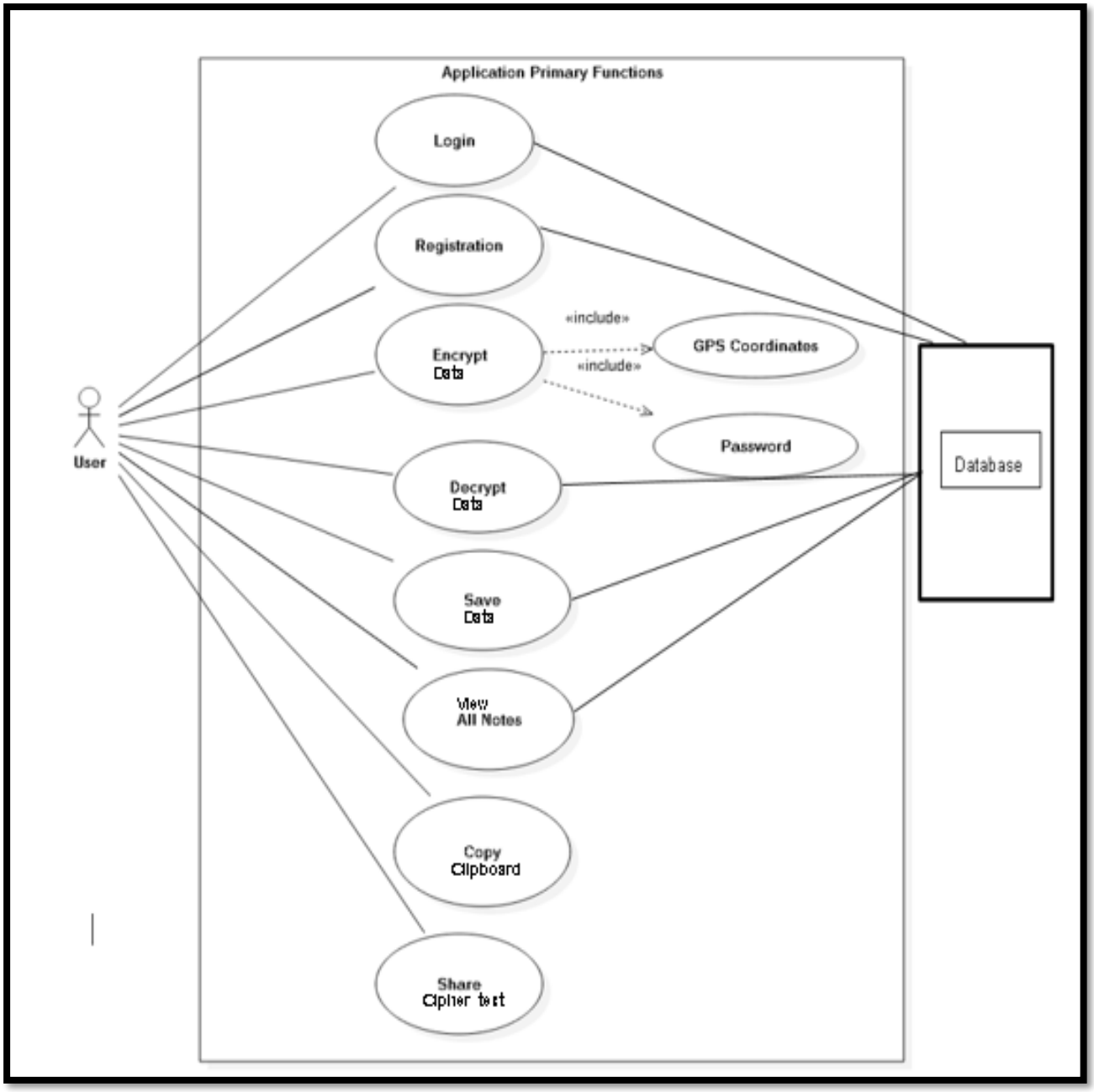
The following are required:

- A user must have GPS enabled
- An existing user must be able to log in by username and password.
- A new user must be able to register.
- Else, if user exists, users must be able to identify themselves.
- A user must be able to view the encryption page.
- A user must be able to view the decryption page.
- A user must be able to view the all notes page.
- A user must be able to enter text to encrypt.
- A user must be able to decrypt cipher text
- A user must be able to set the accuracy of the GPS Coordinates.
- A user must be able to select an encryption algorithm.
- User must be able to encrypt and decrypt notes in the all notes view.
- Else, if user is new and has created an account, they should be able to encrypt files and folders.
- The user should be able to save a notes containing a title and ciphertext.

2.1.1 Functional requirements

2.1.2 Use Case Diagram

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.



2.1.3 Requirement 1 <User Registration>

2.1.3.1 Description & Priority

The user is required to register before being able to use the system.

Priority

The priority of this requirement is high.

2.1.3.2 Use Case Registration

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

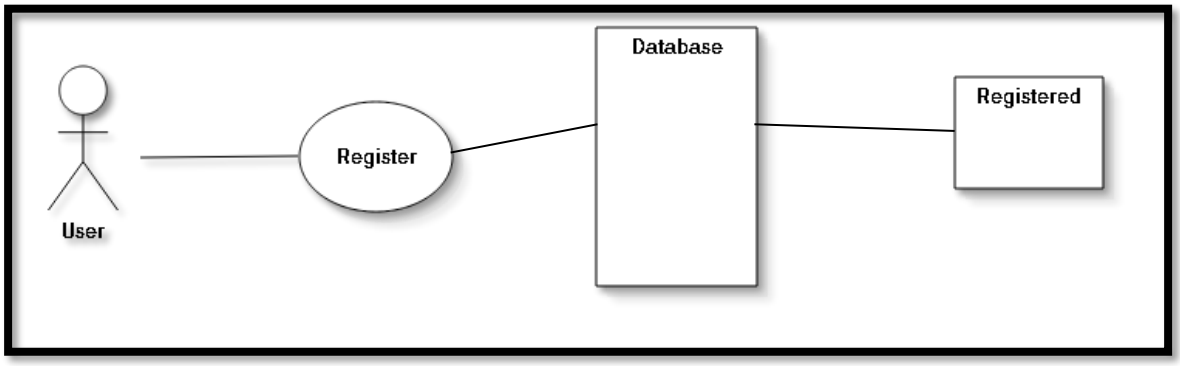
Scope

The scope of this use case is to register a user in the system.

Description

This use case describes the first interaction with the system.

Use Case Diagram



Flow Description

Precondition

The system is in a ready stage.

The user has access to the internet.

Activation

This use case starts when a user (<Actor>) clicks on the Don't have an account? Sign Up

Main flow

1. The system displays a Sign-Up form.
2. The user enters valid data in the form. (See A1+A2)
3. The user submits the information.
4. The system checks the information provided by the user have not been used before.
5. The system successfully signs up and opens on the first page of the app if the provided details have not been already used.

Alternate flow

A1:
<Not valid information>

1. The system does not accept the information provided by the user.
2. The text fields are cleared.
3. The <Actor> is required to re-enter their details.

A2:
<User Already Exists>

1. User enters details.
2. Details already exist on the system.
3. The system outputs an error says details already entered before.
4. User is required to enter new details.

Termination

The user has been registered.

Post condition

The system goes into a wait state.

2.1.4 Requirement 2 <Login>

2.1.4.1 Description & Priority

The user will be required to log in before being able to use the application.

Priority

The priority of this requirement is HIGH.

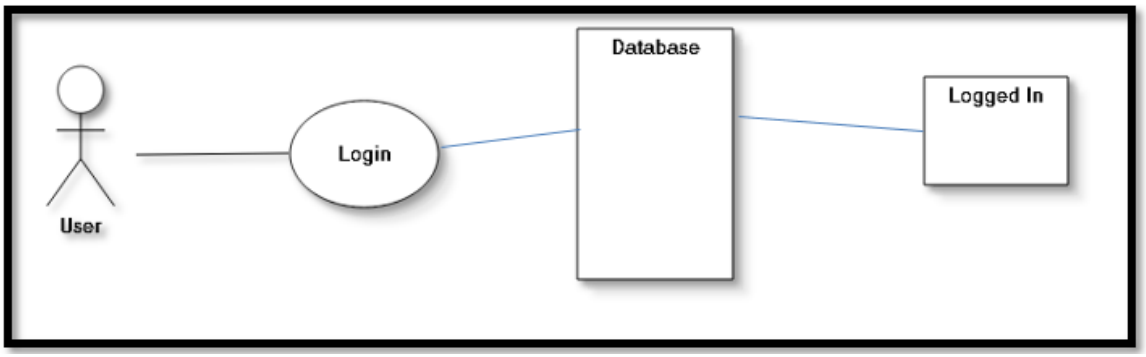
Scope

The scope of this use case is to establish if the user is logged in or not.

Description

This use case describes what the system does when the user is not logged in.

Use Case Diagram



Flow Description

Precondition

The system is in a ready stage.

The user has internet access.

The user is on the Login Page.

Activation

This use case starts when the user (<Actor>) tries to log in.

Main flow

1. The system displays the login screen.
2. The user enters valid information.
3. The user clicks the Login button.
4. The system checks if username and password are valid.
5. The user is logged in the system.

Alternate flow

A1: < Username and password are not valid >

1. The system rejects password and/or username entered by the user.
2. The use case continues at position 2 of the main flow.

Exceptional flow

E1: <The user is already logged in>

1. The system sends the user to his/her home page.

Termination

The user is sent to his/her home page.

Post condition

The system goes into a wait state

2.1.5 Requirement 3 <Encrypt Data>

2.1.5.1 Description & Priority

This requirement describes how the system deals with a user decrypting text.

Priority

The priority of this requirement is HIGH.

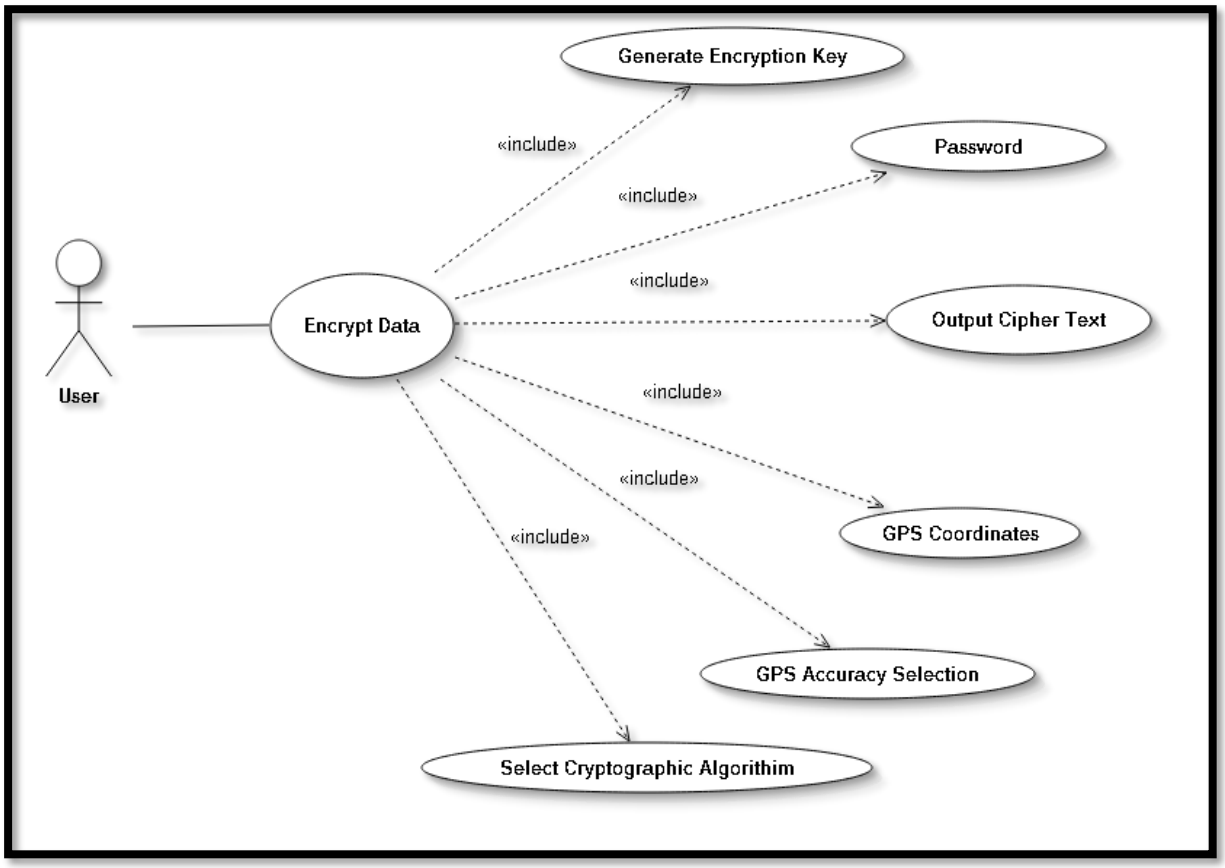
Scope

The scope of this use case is to encrypt input text.

Description

This use case describes what the system does when the user enters text to be encrypted into the application.

Use Case Diagram



Flow Description

Precondition

The system is in a ready stage.

The user has logged into the application and lands on the encryption page.

Activation

This use case starts when the user (<Actor>) clicks the encrypt button.

Main flow

1. The system displays the Encryption screen.
2. The user enters title.

3. The user enters text to be encrypted.
4. The user enters a password.
5. The current GPS Coordinates are retrieved on button click to right.
6. The user selects the accuracy of the coordinates.
7. The user clicks the encrypt button.
8. The application then outputs the cipher text in a text area below.
9. The user can copy the cipher text to clipboard by clicking the copy button.
10. The user clicks the save button to save a note to the database.
11. The user clicks the share button to share the copied cypher text.

Alternate flow

A1: < Password is not valid >

1. The system rejects password entered by the user.
2. The system sends an error message to the user and wipes the password field.
3. The user is required to enter a new password.

A2: <User fails to retrieve GPS Coordinates>

1. The user fails to click the get location button.
2. The system sends an error message to the user.

Termination

The system has successfully encrypted the text and output the cipher text.

Post condition

The system goes into a wait state.

2.1.6 Requirement 4 <Decrypt Data>

2.1.6.1 Description & Priority

This requirement describes how the system deals with a user decrypting text.

Priority

The priority of this requirement is HIGH.

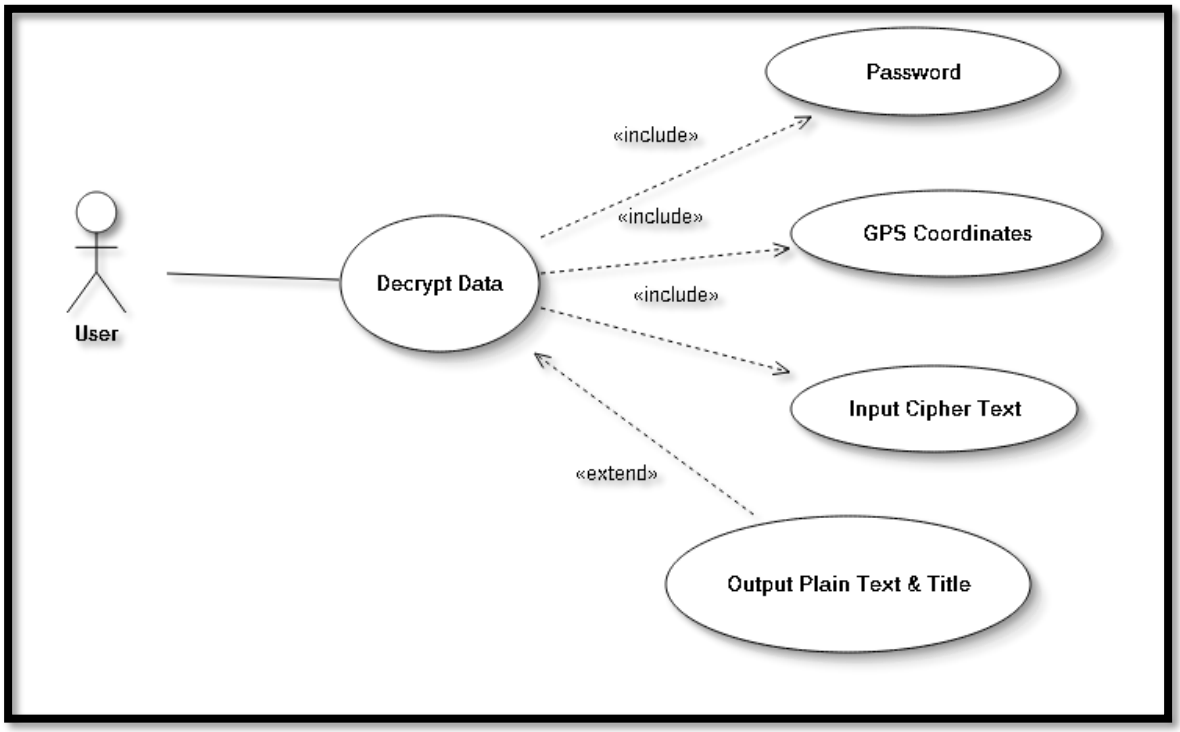
Scope

The scope of this use case is to decrypt cipher text.

Description

This use case describes what the system does when the user enters cipher text to be decrypted in the application.

Use Case Diagram



Flow Description

Precondition

The system is in a ready stage.

The user has logged into the application and on the decryption page.

Activation

This use case starts when the user (<Actor>) clicks the decrypt button.

Main flow

1. The system displays the decryption screen.
2. The user enters cipher text to be decrypted.
3. The user enters a password.
4. The user clicks the decrypt button.

5. The application then outputs the plain text in a text area below.
6. The system successfully decrypts the data.

Alternate flow

A1: < Password is not valid >

1. The system rejects password entered by the user.
2. The system sends an error message to the user and wipes the password field.
3. The password filed is cleared.
4. The user has a set number of attempts to get password correct.
5. The user is required to enter a new password.

A2: <User fails to enter cipher text and clicks decrypt button>

1. The user fails to paste cipher text into the text area.
2. The system sends an error message asking the user to input cipher text.

Termination

The system has successfully decrypted the text and output the plain text.

Post condition

The system goes into a wait state.

2.1.7 Requirement 5 <View All Notes>

2.1.7.1 Description & Priority

The user will be required to log in and be on the all notes page before using this application.

Priority

The priority of this requirement is HIGH.

Scope

The scope of this use case is to view all the saved cipher text.

Description

This use case describes what the system does when the user is not logged in.

Use Case Diagram

Flow Description

Precondition

The system is in a ready stage.

The user has logged into the application and on the view all notes page.

Activation

This use case starts when the user (<Actor>) clicks save button.

Main flow

1. The user logs into the system and navigates the encryption screen.
2. The system displays the encryption screen.
3. The user enters a title for the cipher text.
4. The user clicks the save button.
5. The system save the cipher text and the title of the cipher text.
6. The user navigates to the all notes page.
7. The user locates the desired note using the title as a hint to the cipher text.
8. The user clicks the selected note title.
9. The user is required to enter their login password to access the desired note.
10. The note title with cipher text below opens.
11. The user can copy the cipher text and decrypt it on the decryption page.

Alternate flow

A1: < Password is not valid >

1. The system rejects password entered by the user.
2. The system sends an error message asking the user to enter the correct password.
3. Access to viewing that desired note is denied.

Termination

The user has successfully viewed the note containing the cipher text.

Post condition

The system goes into a wait state

2.1.8 Requirement 6 <Save Data>

2.1.8.1 Description & Priority

The user will be required to log in and be on the Encryption page before using this requirement.

Priority

The priority of this requirement is HIGH.

Scope

The scope of this use case is to save the cipher text along with a plain text title to the local database.

Description

This use case describes what the system does when the save button is clicked after encrypting the data.

Use Case Diagram

Flow Description

Precondition

The system is in a ready stage.

The user has logged into the application and on the Encryption page.

Activation

This use case starts when the user (<Actor>) clicks save button.

Main flow

1. The system has populated the cipher text output field after encrypting text. (See A2)
2. The user then enters a title to save the cipher text under. (See A1+A3)
3. The user click the save button at the bottom of the encryption page.

4. The system saves the title and the cipher text to firebase database. (See A3)
5. The title and cipher text can be viewed on the All Notes page and in the firebase console.

Alternate flow

A1: < Title Field Blank >
1.The system rejects the save by the user.

2.The system sends an error message asking the user to enter the correct password.

A2: <Cipher Text Output Field Blank>

1.The system has not output any cipher.

2.The system will send error message to the user.

3. The error message will prompt user to enter cipher text

A3: <Title already used in the database>

1.The user clicks the save button.

2.The system fails to add that title to the database.

3.The system outputs an error message stating the title is already in use.

4. The user must re-enter a new title.

Termination

The user has successfully saved the note containing the cipher text and title to the all notes page.

Post condition

The system goes into a wait state.

2.1.9 Data requirements

The data requirements will also be an important aspect of the application. One of the main features of the application will be the ability to save the encrypted data as a note. The third tab on the GUI called “All Notes” is where all the saved notes will be stored. The page will show the title of the saved note to serve as a hint to what the cyphertext contains as well as the ciphertext. Another important data requirement is the cloud backup. This will mean if the phone was to be lost or stolen there will be a backup of all the encrypted notes on the firebase cloud.

2.1.10 User requirements

Nowadays android devices are powerful computers that a wide variety of built in sensors such as GPS. The main user requirement objectives with this project is to answer the call of users in recent years by making their data more secure using cryptographic algorithms, passwords, and GPS Coordinates to prevent any attackers hacking your device and stealing your private data. I am simply answering the call of many future customers in helping protect their data.

2.1.11 Usability requirements

The usability requirement is very important in this android application. It is an important requirement as the application is applying complicated cryptographic logic. The inner workings of the application are complicated but the application it’s self will be simple to use. Users will be able to simply navigate the application without prior knowledge. I will also be incorporating a help page in the application which will explain how to use the application. The simple to use UI and help sections available on the top right each page of the application will make the application extremely user friendly.

3 Design and Architecture

The architecture is the art and science of designing structures. The android phone platform by google (Android Studio) is becoming more popular in the software developer's community mainly because of its powerful capabilities and open source architecture. I have chosen this architecture because its: simple to integrate and test, comes with an individual test phase strategy. Also in my opinion I think it is the most common architecture for android devices.

The Encipher GPS application is a native android application which was different from my original idea which was a hybrid application. I was already to build my application in a hybrid format but realized that it was the least secure form as all the information would be live over the internet rather than being stored locally on the android device like a native app. The application makes use of the firebase authentication and realtime database to manage the authentication and database needes of the application.

Firebase

My application is using the firebase authentication and firebase real-time

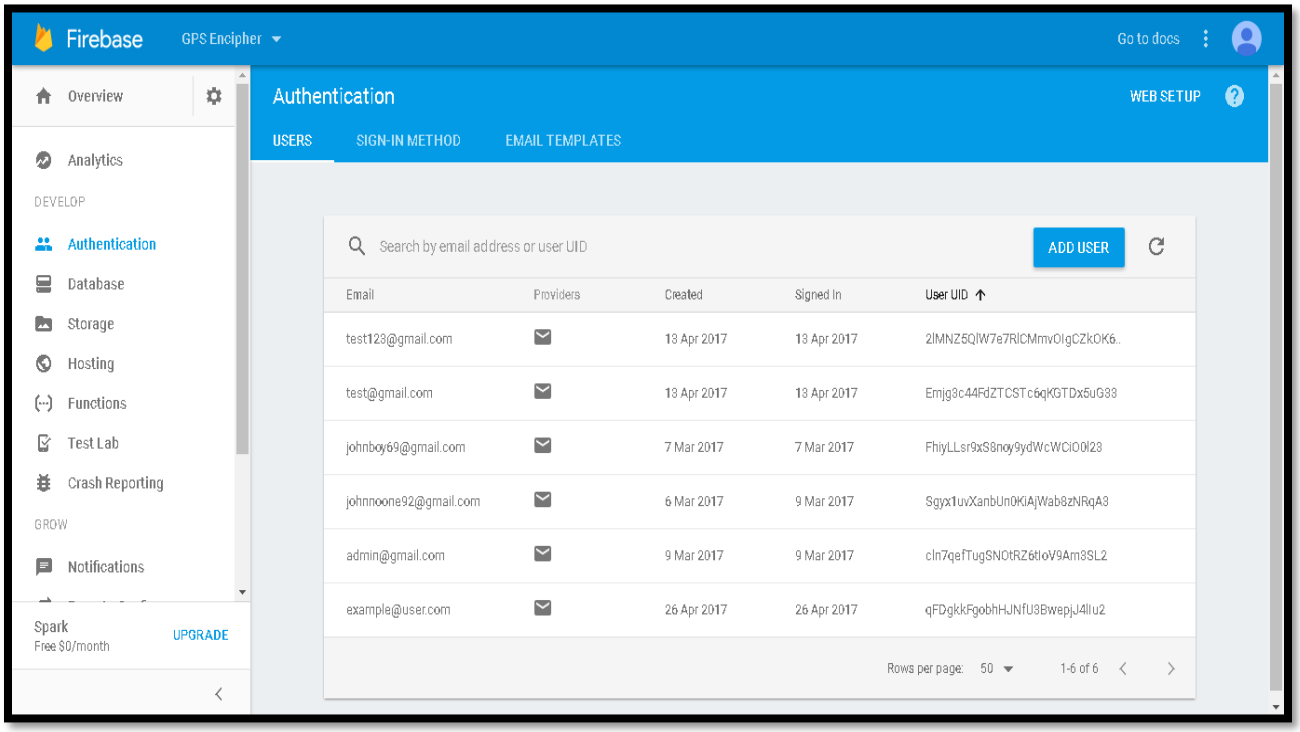


Figure 3.2.1

database. Firebase handles all the backend work and presents all the information in a functional user interface which can be seen below in figure 2.2.1.

The user interface for the authentication gives me all the necessary admin functionality needed to manage all the users on my application. It provides me with the ability to add users, delete account, reset password, and disable accounts. In the below screenshots, you can see the way the data is stored on the database. The data is stored by using an authentication ID. The individual authentication ID can be seen above the ciphertext and title fields in figure 2.2.2.

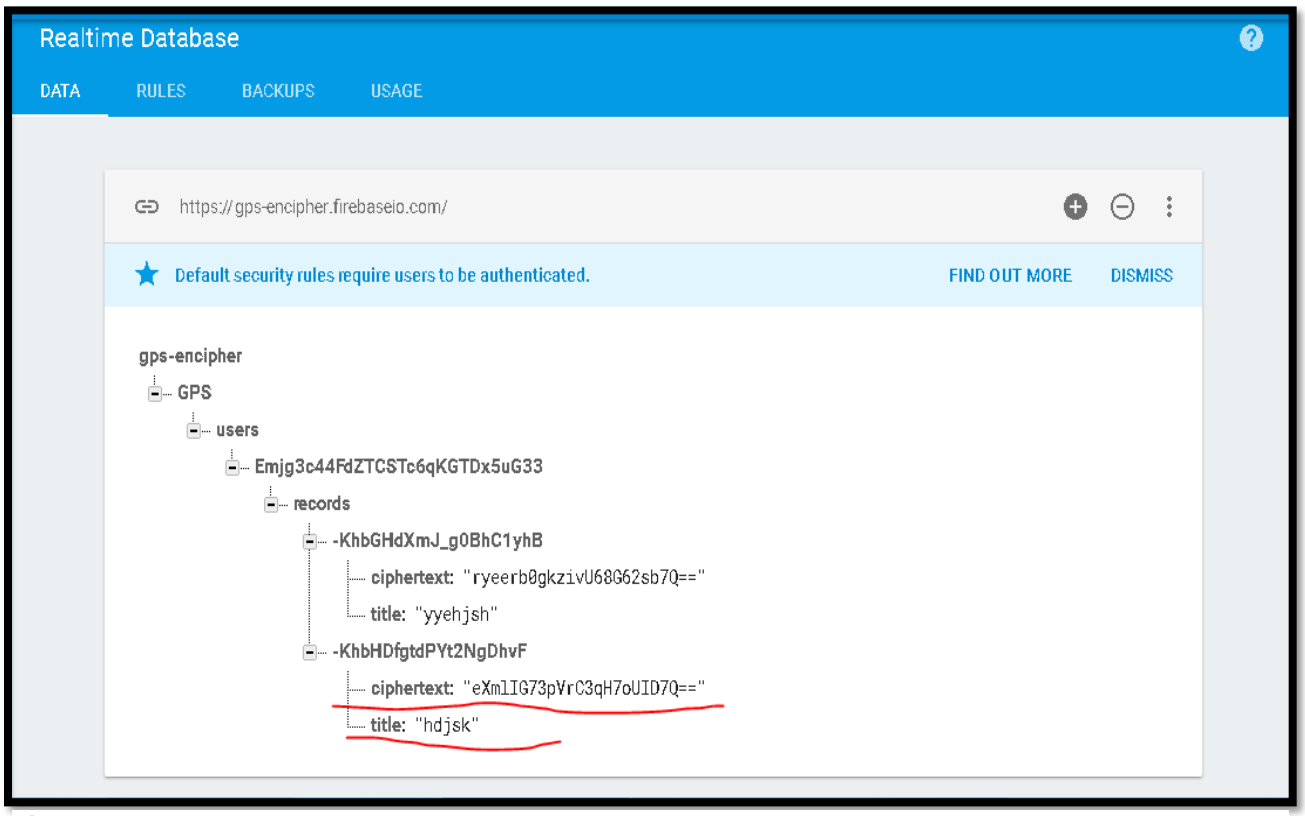


Figure 3.2.2

The authentication ID will store data based on your unique user ID created when you make an account on the app.

The application currently it stores data from the title and ciphertext fields on the encryption page of the app once you click the save button at the bottom of the page. The data can be seen saved in real-time on the all notes page and in the firebase admin console in figure 2.2.2. In figure 2.2.1 you can see the different

options all down the left-hand side of the page such as analytics, testing, crash reports. The analytics in figure2.2.3 would allow me to view the statistics of all the users of my application like where is my app popular in the world or how much revenue the application has generated to date/per quarter.

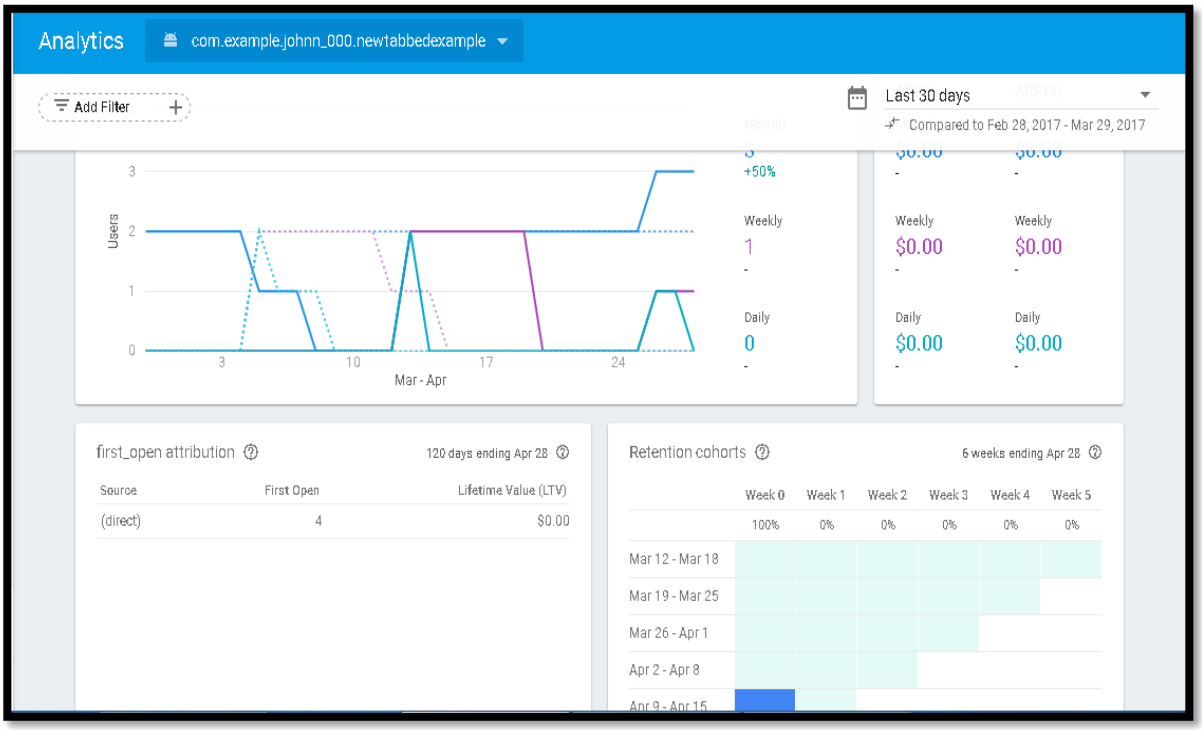


Figure 3.2.3

Below you can see the diagram figure 3.2.4 of the firebase authentication process where you can see the required inputs and depending on the input could get certain outputs. The different inputs and outputs depend on the current state of device (Offline/Online). After login in successfully your auth token is sent to the server using https. The server then verifies the authenticity of the token and provides you with a user ID. The user ID will be used to determine the currently logged in user on the server.

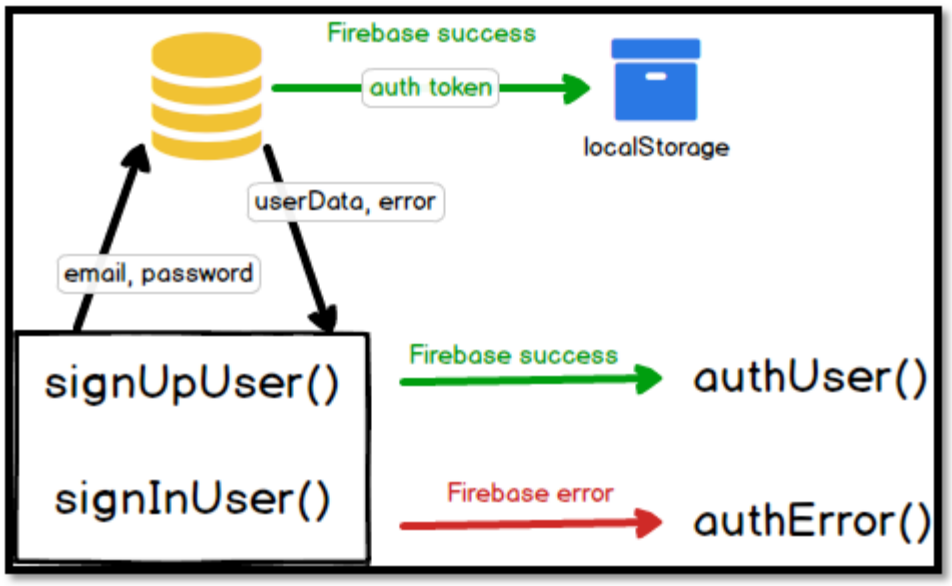


Figure 3.2.4

In the above figure 3.2.4 we can see the broken-down version of the authentication. It shows the creation of a new user or the signs them in using their email or password. The outcome of this being firebase returning error object or a user data object. This also shows how firebase locally stores the auth token in case the page is refreshed.

Figure 3.2.5

The application also utilises the location manager, copy to cipboard, share function along with the AES and RSA algorithims.

The location manager is used in the app to retrive the GPS Coordinates used in the encryption process of the application. The copy to clipboard function is used

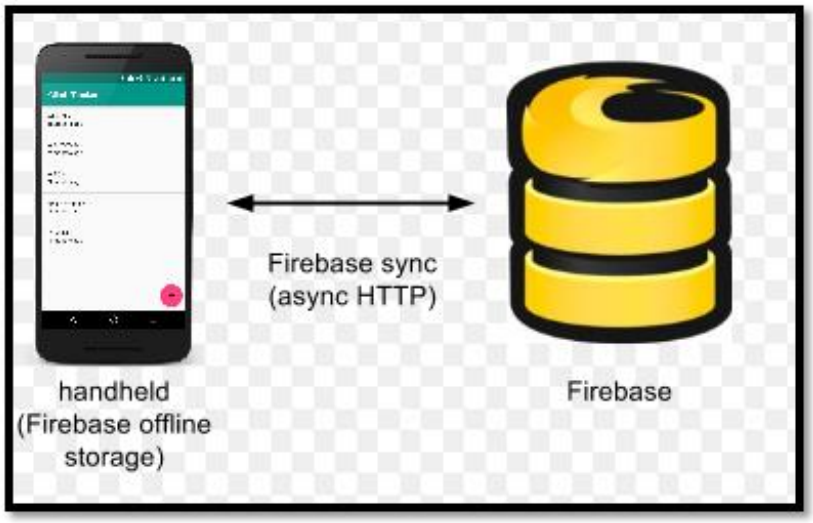


Figure 3.2.6

In figure 3.2.4, 3.2.5 and 3.2.6 are diagrams of how firebase is used in my application to provide real time updates. In figure 3.2.5 and 3.2.6 we can see the architecture of my app. In terms of figure 3.2.5 I choose to go with the real-time database over the storage option. This is because the real-time database offered more flexibility.

3.1 Implementation

The two main algorithms used in my project are the AES and RSA algorithms.

RSA Encryption Algorithm

In the app, I implemented the RSA algorithm in a demo format. I was unable to create a key from string meaning that I can't in turn save the keys. This leaves me with the option of means I'm left with only a demo version where you can only encrypt and decrypt within the given session because new keys are generated on each start up.

The result of the demo is to automatically paste the private key into the decryption tab upon selecting the asymmetric radio button. This is just a simple demo of the RSA Algorithm working as I was not able to accomplish the custom key logic in this instance. Although original goal of incorporating the GPS Coordinates in the encryption was still achieved using the AES algorithm.

AES Encryption Algorithm

AES also known as the Advance Encryption Standard is as symmetric block cipher algorithm. This essentially means that the algorithm uses the same key for encryption and decryption. That would mean that the sender and recipient must know the same secret key. The different key sizes are 128-bit,192-bit and my chosen one the 256-bit algorithm. The downside of using the 256 bit over the 128bit is speed with the 256bit being around 40% slower. A common misconception is to think because the 256bit is the largest key it's the most secure. It's not so much the larger key size but the way it is implemented. Well known weaknesses are in the AES256bit key expansion function. Technically such weaknesses would make the 256bit less complex than the 128bit. My endpoint would be that it depends on the implementation not the key length.

Encryption of AES Algorithm (Online Source)

In the below figures 3.1.1 and 3.1.2 is the encryption part of AES algorithm.

AES encryption algorithm is not my own work and was sourced on stack overflow at : http://stackoverflow.com/questions/992019/java-256-bit-aes-password-based-encryption/992413#992413. The RSA algorithm is also source from an online website cited in the code. I will focus on the AES because that is the algorithm which is fully working with the custom key. In the code snippets below you can see the code is broken down into the encryption and decryption sections.

```
/**
 * Encrypt and encode message using 256-bit AES with key generated from password.
 *
 * @param password used to generated key
 * @param message  the thing you want to encrypt assumed String UTF-8
 *
 * @return Base64 encoded CipherText
 *
 * @throws GeneralSecurityException if problems occur during encryption
 */
public static String encrypt(final String password, String message)
        throws GeneralSecurityException {

    try {
        final SecretKeySpec key = generateKey(password);

        log("message", message);

        byte[] cipherText = encrypt(key, ivBytes, message.getBytes(CHARSET));

        //NO_WRAP is important as was getting \n at the end
        String encoded = Base64.encodeToString(cipherText, Base64.NO_WRAP);
        log("Base64.NO_WRAP", encoded);
        return encoded;
    } catch (UnsupportedEncodingException e) {
        if (DEBUG_LOG_ENABLED)
            Log.e(TAG, "UnsupportedEncodingException ", e);
        throw new GeneralSecurityException(e);
    }
}
```

Figure 3.1.1

In the below figure 3.1.2 we can see the initialization happening. This step is necessary to provide the AES key and the initial vector.

```
/**
 * More flexible AES encrypt that doesn't encode
 *
 * @param key     AES key typically 128, 192 or 256 bit
 * @param iv      Initiation Vector
 * @param message in bytes (assumed it's already been decoded)
 *
 * @return Encrypted cipher text (not encoded)
 *
 * @throws GeneralSecurityException if something goes wrong during encryption
 */
public static byte[] encrypt(final SecretKeySpec key, final byte[] iv, final byte[] message)
        throws GeneralSecurityException {
    final Cipher cipher = Cipher.getInstance(AES_MODE);
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    byte[] cipherText = cipher.doFinal(message);

    log("cipherText", cipherText);

    return cipherText;
}
```

Figure 3.1.2

Decryption of AES Algorithm (Online source)

In the decryption is essentially the same as the encryption only its backwards. The main difference is that the decryption needs an initialization vector as a parameter whereas the encryption uses random IV.

```
/**
 * More flexible AES decrypt that doesn't encode
 *
 * @param key               AES key typically 128, 192 or 256 bit
 * @param iv                Initiation Vector
 * @param decodedCipherText in bytes (assumed it's already been decoded)
 *
 * @return Decrypted message cipher text (not encoded)
 *
 * @throws GeneralSecurityException if something goes wrong during encryption
 */
public static byte[] decrypt(final SecretKeySpec key, final byte[] iv,
                             final byte[] decodedCipherText) throws GeneralSecurityException {

    final Cipher cipher = Cipher.getInstance(AES_MODE);
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
    byte[] decryptedBytes = cipher.doFinal(decodedCipherText);

    log("decryptedBytes", decryptedBytes);

    return decryptedBytes;
}
```

Figure 3.1.3

```
/**
 * Decrypt and decode ciphertext using 256-bit AES with key generated from password
 *
 * @param password                used to generated key
 * @param base64EncodedCipherText the encrpyted message encoded with base64
 *
 * @return message in Plain text (String UTF-8)
 *
 * @throws GeneralSecurityException if there's an issue decrypting
 */
public static String decrypt(final String password, String base64EncodedCipherText)
        throws GeneralSecurityException {

    try {
        final SecretKeySpec key = generateKey(password);

        log("base64EncodedCipherText", base64EncodedCipherText);
        byte[] decodedCipherText = Base64.decode(base64EncodedCipherText, Base64.NO_WRAP);
        log("decodedCipherText", decodedCipherText);

        byte[] decryptedBytes = decrypt(key, ivBytes, decodedCipherText);

        log("decryptedBytes", decryptedBytes);
        String message = new String(decryptedBytes, CHARSET);
        log("message", message);
```

Figure 3.1.4

In the above figure 3.1.4 we can see once the cipher has been provided the key and the IV has been initialized for decryption, and everything is ready for decryption.

In figure 3.1.1 in the encryptionfragment.java class of the implementation we can see where the AES key is created. In the first two lines of the code we can see the password is being used to encrypt the GPS Coordinates. The output of this is a composite key which is used to encrypt the original plaintext. In line three of the code we can see depending on which encryption method is chosen the plaintext is encrypted. The AES algorithm will use the composite key while the RSA algorithm will use the public key to encrypt the text. After this the newly created ciphertext is set to the ciphertext output field. This is a composite encryption of your coordinates and password so it's key will be dynamic and will be the same on same location and password. This is what I used to have my own custom encryption logic in the application.

```
    try {
        // Encrypt GPS coordinates using the password
        String encryptedGPSCoordinates = AESUtil.encrypt(password, gpsCoordinates);
        // Create composite key or the main key for encryption
        String compositeKey = encryptedGPSCoordinates + password;
        // Encrypt original text according to the chosen encryption method
        final String cipher = mEncryptionMethod == MainActivity.SYMMETRIC
                ? AESUtil.encrypt(compositeKey, text)
                : RSAUtil.encrypt(text, mPublicKey);
        mCipherTextView.setText(cipher);
    } catch (GeneralSecurityException e) {
        e.printStackTrace();
        String text2 = "Error In Encryption!";
        displayMessage(text2);
    }
}
```

Figure 3.1.1

In the code snipped below is my copy to clipboard unity which handles my all my copying of text through the app. This is used quite a bit in my app and is a vital part of the implementation. I have it implemented on click of the copy button where it copies text from certain fields into the clipboard for pasting into decryption fields.

```
public static boolean copyToClipboard(Context context, String text) {
    try {
        int sdk = Build.VERSION.SDK_INT;
        if (sdk < Build.VERSION_CODES.HONEYCOMB) {
            android.text.ClipboardManager clipboard = (android.text.ClipboardManager) context
                    .getSystemService(Context.CLIPBOARD_SERVICE);
            clipboard.setText(text);
        } else {
            android.content.ClipboardManager clipboard = (ClipboardManager) context
                    .getSystemService(Context.CLIPBOARD_SERVICE);
            android.content.ClipData clip = android.content.ClipData
                    .newPlainText("Machine ID", text);
            clipboard.setPrimaryClip(clip);
        }
        return true;
    } catch (Exception e) {
        Log.e(TAG, "Error while copying to clipboard", e);
        return false;
    }
}
```

Figure 3.1.5

In figure 3.1.6 we can see the share utility. This is a utility that opens all the social media sharing options from within the application without me exiting the application. Another vital bit of my application and the way it operates.

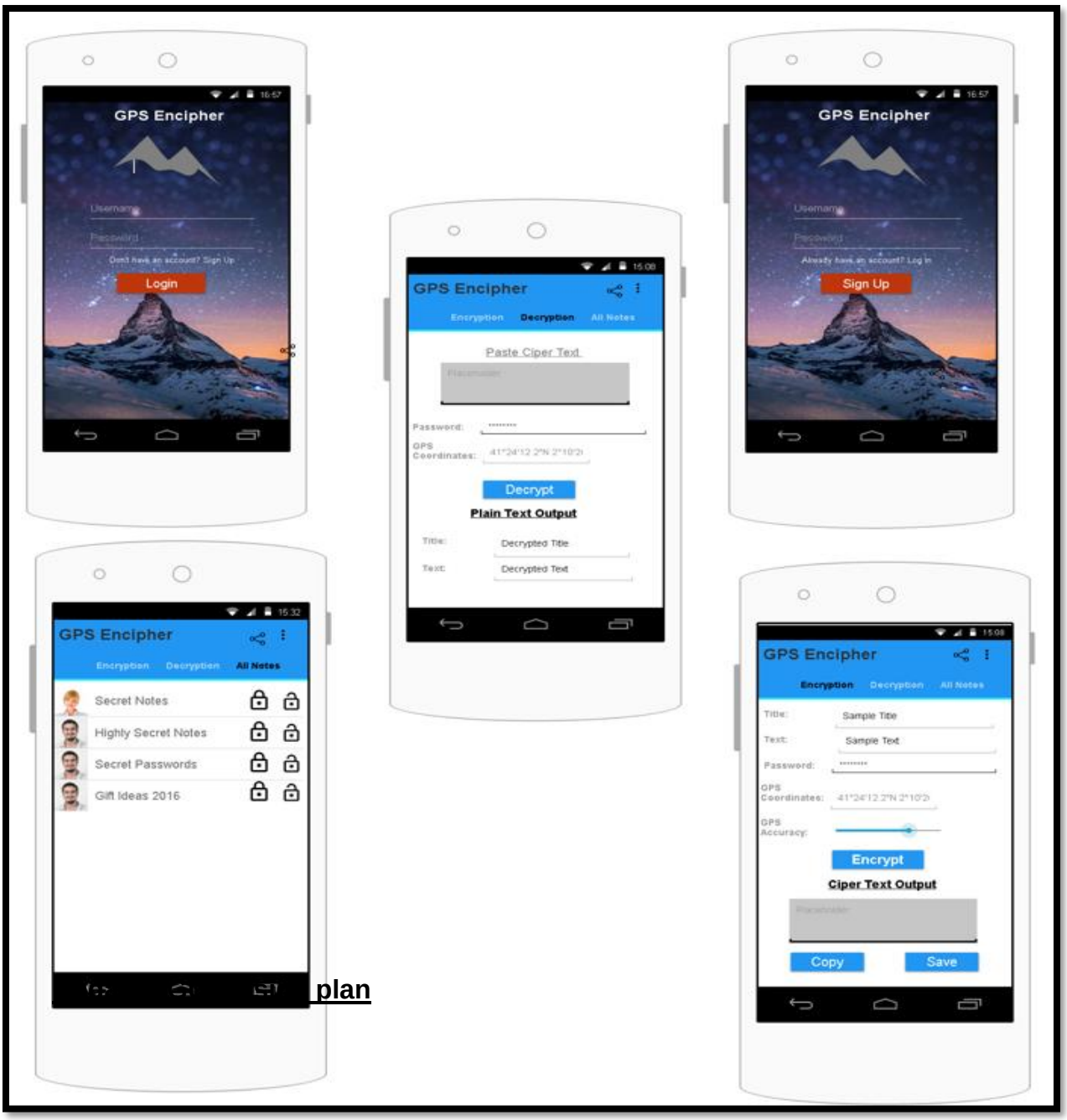
```
public static void shareCipherText(Context context, String text) {
    if (TextUtils.isEmpty(text)) {
        Toast.makeText(context, "No Cipher Text To Share!", Toast.LENGTH_LONG).show();
        return;
    }
    Intent sharingIntent = new Intent(android.content.Intent.ACTION_SEND);
    sharingIntent.setType("text/plain");
    sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT, text);
    context.startActivity(Intent.createChooser(sharingIntent, "Share Cipher Text"));
}
```

Figure 3.1.6

3.2 Graphical User Interface (GUI) Layout

The graphical user interface is designed to be bright, interactive and user friendly. I would also need to remain within the set out android guidelines for developing a user-friendly interface, that supports numerous android phone screen sizes. The GUI consists of several different screens as seen below: Login, Sign up, Encryption, Decryption, and all notes. The designs were made before the implementation to u8

Original GUI design plan



plan

The reason for updating the GUI's in this case was to incorporate the updated database and the public key encryption. When these were designed, I was still considering getting the GPS Accuracy functioning but in the end, I removed that and focused on the public key encryption. The main update is the of the introduction of the asymmetric and symmetric tick boxes. Along with this password field has been changed to accept public keys on the encryption tab and private keys on the decryption tab.

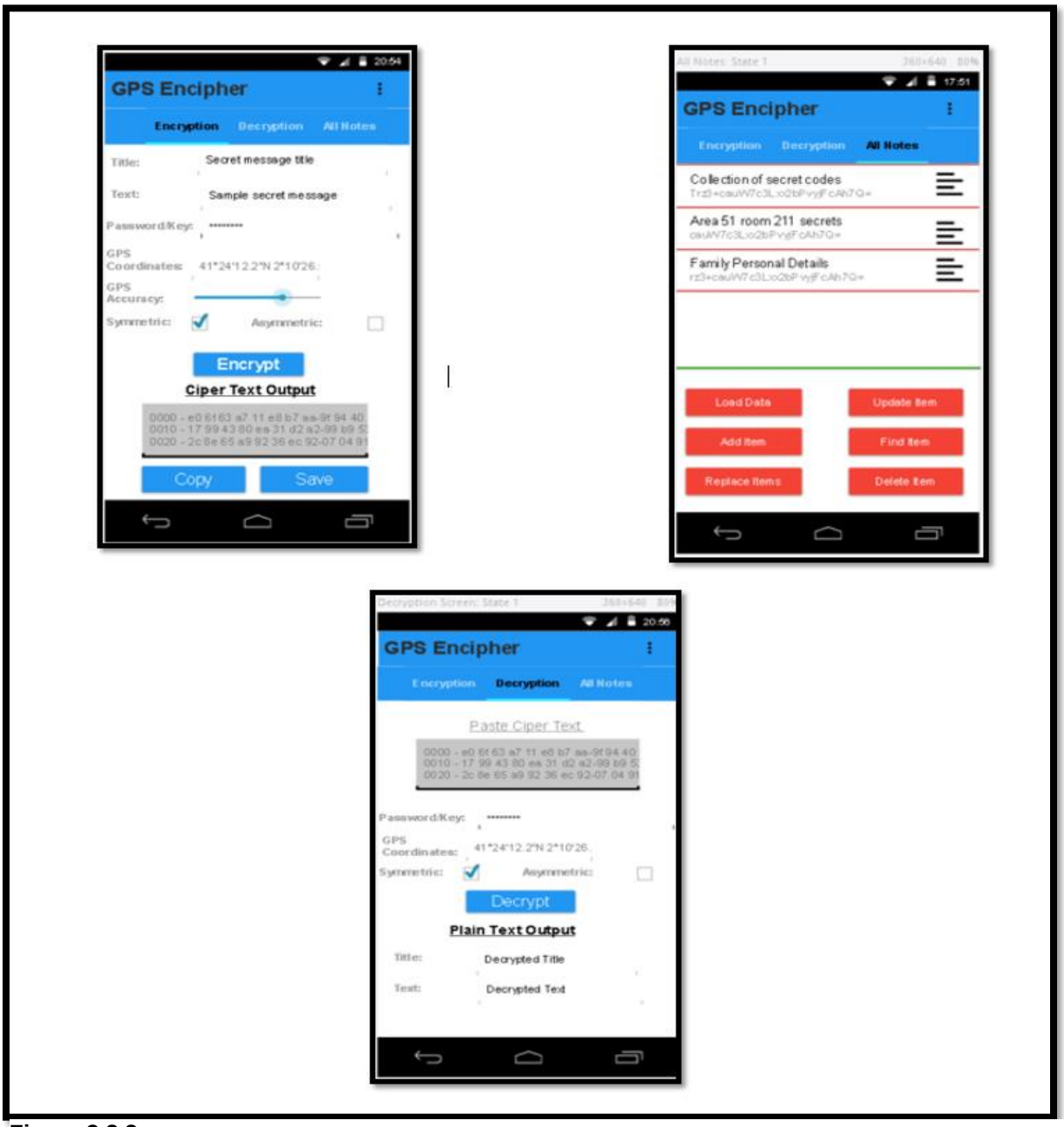


Figure 3.2.2

Final finshed GUI Screenshots

The Login page seen in figure 3.2.3 will be the first page a user will see and will be required to login to access the application. If it is the users first time using the app and they don't have prior login details then they will be required to click the "Not a member? Get registered in firebase now" label/button. The user is required to enter their email address and password to gain access to the application. If the user has forgotten their password they can click the forgot your password button/label. The registration page seen in figure 3.2.4 is where new users create an account. As I said above the authentication in the app is being handled by firebase. When a new user creates an account on the registration page, that user is assigned a uniques user ID. This unique user ID will come in handy in the database part of firebase. The user ID will allow me link a specific user to their uniquly saved content in the database.

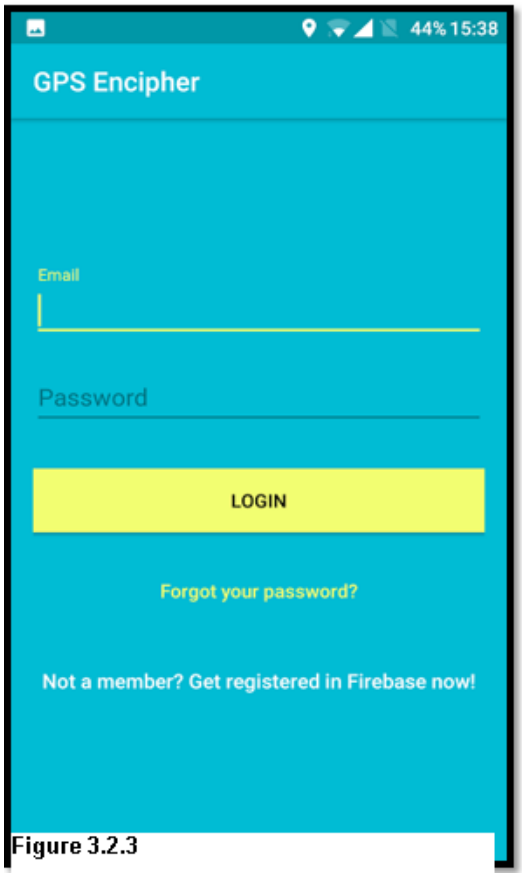


Figure 3.2.3

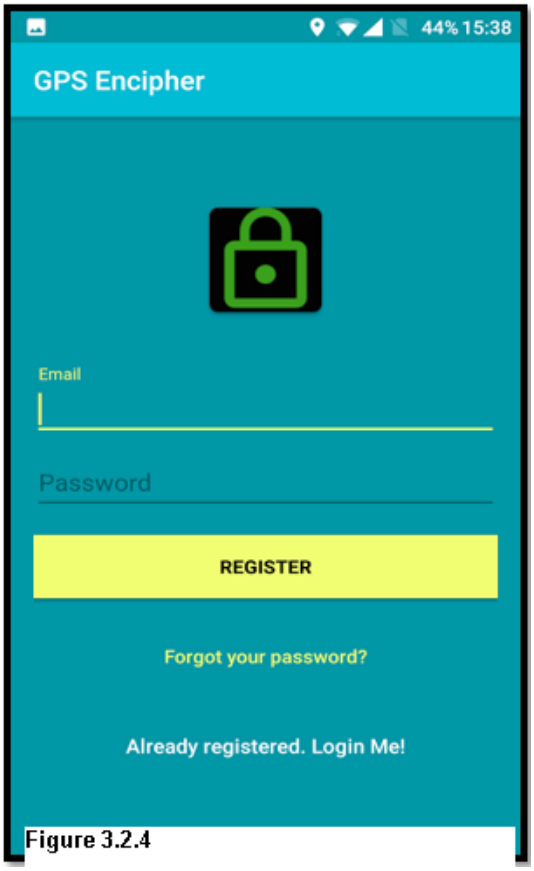


Figure 3.2.4

The encryption tab is where logged in users will first land on. All the fields on the encryption page are required in order to encrypt your desired text.

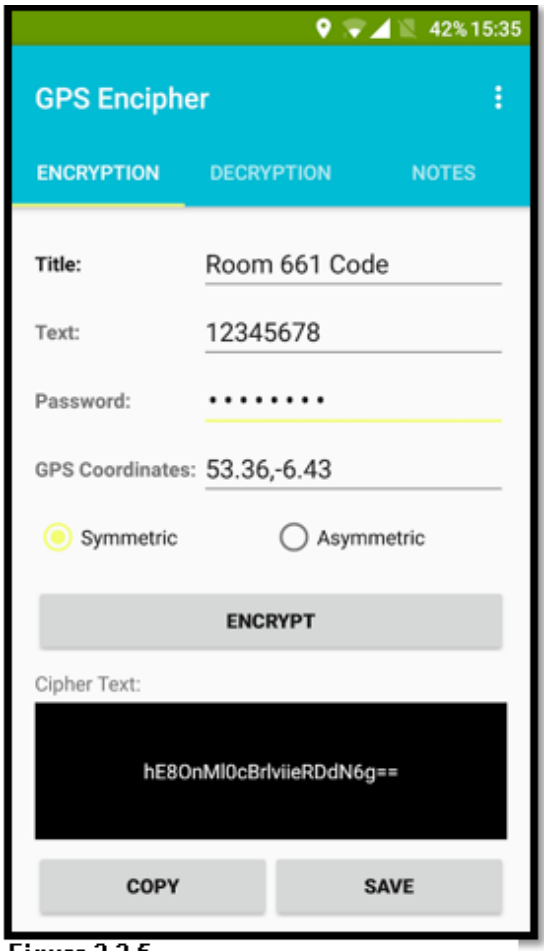


Figure 3.2.5

The first field “title” is designed to be a hint for what the stored cipher text contains. The reason behind this field is so you don’t store random ciphertext in your database. The title allows you to know what the ciphertext contains without decrypting. The “text” field is where you input the text you would like to be encrypted. The “password” field is the personal password used to encrypt the GPS coordinates to create a composite key. The “gps coordinates” field sends live updates of the users GPS coordinates which will be encrypted with the password to create the encryption key. The option of symmetric and asymmetric is to define which type of encryption you would like to use. The asymmetric encryption (public key encryption) is still in beta phase but the symmetric (composite key encryption) is fully working. Once all the fields are populated the user can then click the encrypt button and the subsequent ciphertext will be output into the black box seen above. In figure 3.2.6 you can see the messages you get when clicking the copy button or save button.

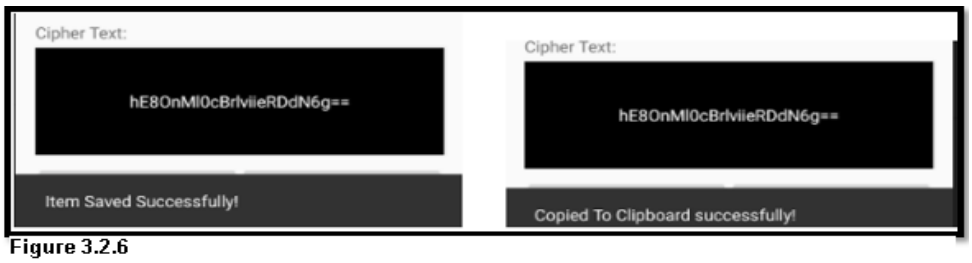


Figure 3.2.6

The decryption tab is the second tab of three tabs. The first field the “ciphertext” field is where users would paste the ciphertext they would like to be decrypted in the black box. The “password” field is where the users enters the password used to originally encrypt the text. The GPS coordinates field is the same as the encryption tab but in this instance, they must match the coordinates that were used to encrypt the data. The radio button field with the “symmetric and asymmetric” options to choose what method of encryption was used to originally encrypt the data. The last text field the “decrypted text” field is another black box which outputs the decrypted ciphertext.

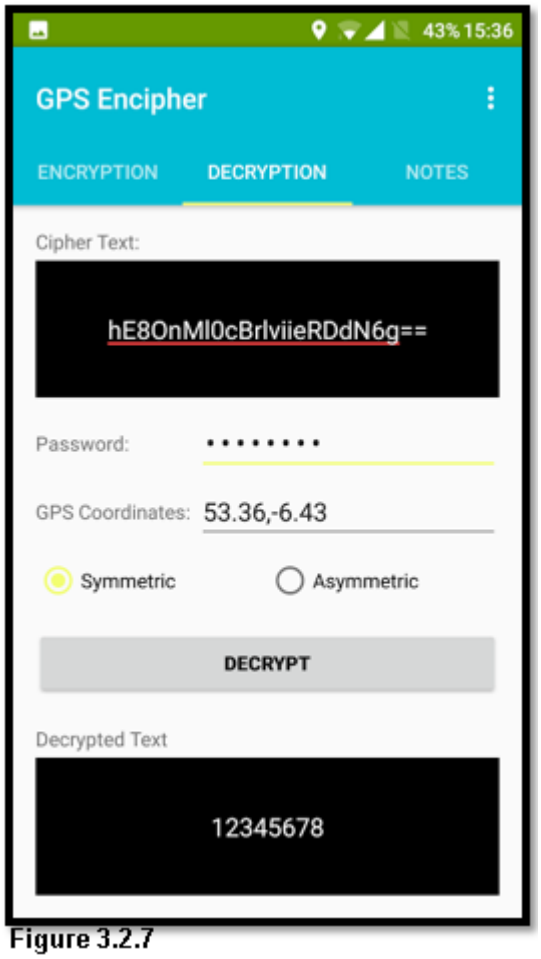


Figure 3.2.7

Another feature on all three tabs of the application is the “share cipher text” and “Logout”. The three dots symbol in figure 3.2.7 can be located at the top right of each of the three tabs. In figure 3.2.8 you can see the options the user is presented with. The logout allows a user to logout.

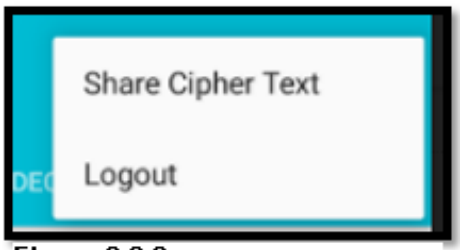


Figure 3.2.8

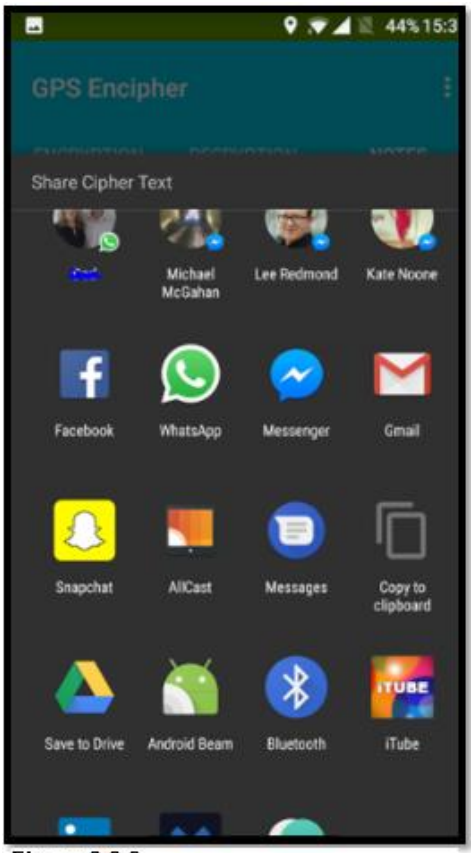


Figure 3.2.9

The share cipher text option gives you the ability to share your encrypted text with a friend over any of your chosen social media platforms. This gives you the ability to share your cipher text with friends but must be in the secret location with the password to decrypt it.

This feature saves time from closing the application and opening your chosen social media platform by linking you to all the available social media sharing applications into the app. The idea for this part of the application was to be able to send your asymmetrically encrypted text to your friend.

The thing is you would have encrypted with their public ally available key and then they would decrypt the cipher text using their personal private key. The sharing option also works with the symmetric encryption. It only works if you have a prearranged location and password with your friend. The issue with sharing this method is that it is not full proof. In the event, you didn't have a prearranged password and location then sending such credentials it over this form of communication would be considered highly insecure. Therefore, the public key encryption is better suited to the sharing part of the application.

The final tab called "All Notes" is where all the saved data is displayed. The ability to save only exists on the "encryption" tab seen at the bottom right of figure 3.2.5. The "all notes" tab can be seen in figure 3.2.10 where some sample data is already stored. I have discussed in detail the firebase backend in the design and architecture section but you can see the way data it is stored in the all notes tab. The title or hint is the heading in bold dark grey text and the cipher text is below it in light grey and non-bold.

If the user holds down on one of the squares of stored data they are presented with what can be seen in figure 3.2.11. In figure 3.2.11 we can see that upon hold down for a few seconds on your chosen block of data the user is presented with a series of options.

The options which can be seen on the centre right of figure 3.2.11 are “delete, copy cipher text and share cipher text”. The “delete” option will delete the record from storage. The “copy cipher text” will copy the ciphertext to the clipboard and not the title. The users can then paste the ciphertext into the decryption field. The “share cipher text” option seen in figure 3.2.11 will open the page seen in figure 3.2.9 above.

Figure 3.2.10

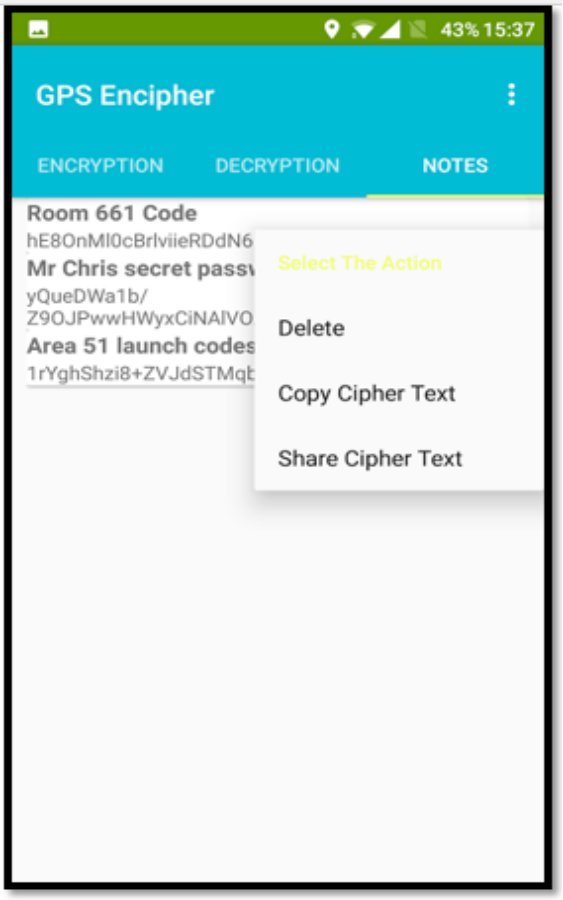


Figure 3.2.11

3.3 Testing

RSA Algorithm Testing

Test Case ID	Test_001	**Test Case Description**	Testing the RSA Algorithim		
Created By	John Noone	**Reviewed By**	John Noone	**Version**	0.19
QA Tester's Log					
Tester's Name	John Noone	**Date Tested**	05-Apr-17	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Logged into the app	1	Testing the encryption and decryption of the algorithim
2	Open the Encryption page of app	2	
3	Select asymetric encryption radio btn	3	
4		4	
Test Scenario	Verifying the algorithim works		

Step :	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Open the Encryption page of app and select asymetric	The app should open on login page	As Expected	Pass
2	Get the encryption and decryption keys	Retrival of both public and private key	As Expected	Pass
3	Encrypt plaintext with your public key	Plain text encrypted into ciphertext using public key	As Expected	Pass
4	Decrypt cipher text with the private key	Ciphertext to be decrypted into plaintext	As Expected	Pass
5	Check that the plaintext and encrypted text are different	The plaintext should be different to the encrypted text	As Expected	Pass
6	Check for spaces in encrypted text	Encrypted text without spaces	As Expected	Pass
7	Check that the plaintext and the decrypted text are the same		As Expected	Pass

In the below figure 3.3.1 we can see the test case 001 coded out. The first thing was to select a sample string which can be seen in the first two lines of code. Once the public and private keys have been generated the next step is the encryption. The next two lines do just that by encrypting the plaintext with the public key generated in the previous step.

The test then moves to the decryption page. In the 5th and 6th lines the decryption begins. The private key is populated in the password field and the ciphertext is decrypted. The next steps after this are to check that the source text and the encrypted text are not the same. After this the encrypted text must be checked for spaces and that the decrypted text and original text match.

```
    @Test
    public void testRSAAlgorithm() throws Exception {
        final String plainText = "This is a sample string";

        // Get keys for encryption and decryption
        PrivateKey privateKey = RSAUtil.getPrivateKey();
        PublicKey publicKey = RSAUtil.getPublicKey();

        // Encrypt plain text with your public key
        String encryptedText = RSAUtil.encrypt(plainText, publicKey);
        System.out.print("Encrypted Text:\n\n" + encryptedText);
        Log.e("RSA Utils", "Encrypted Text:\n\n" + encryptedText);

        // Decrypt cipher text with your private key
        String decryptedText = RSAUtil.decrypt(encryptedText, privateKey);
        System.out.print("Decrypted Text:\n\n" + decryptedText);
        Log.e("RSA Utils", "Decrypted Text:\n\n" + decryptedText);

        // Confirm that the source text and the encrypted text are not the same
        Assert.assertNotSame(plainText, encryptedText);
        // Confirm that the encrypted text does not contains space
        Assert.assertFalse(encryptedText.contains(" "));
        // Confirm that the source text and the encrypted text are same
        Assert.assertEquals(plainText, decryptedText);
    }
}
```

Figure 3.3.1

AES 256bit Algorithm Testing

The following tests in this section test encryption and decryption. There is no need for test cases as you will be able to see input plaintext and the expected output ciphertext. The tests look at the encryption and decryption stages of the algorithm. This series of tests will prove if the encryption and decryption processes are working correctly. The tests will know something is not right if the decryption or encryption output doesn't match the expected outcome.

The following set of tests are based around checking if the encrypted text matches the expected ciphertext. If they match the test is passed and if they don't the test fails.

EncryptionTest | encryptText_shouldReturnCipherText()

```
@Test
public void encryptText_shouldReturnCipherText() throws GeneralSecurityException {
    String text;
    String output;
    String expected;

    text = "Everything we do is connected with heaven: death, milk, definition, dimension.";
    expected = "gFquRtiO5h5Iw2NQlqOZXRklXztQKzUoOwdJdkOuN1FkiKjSH4Kc78Ic7BfJjaz+HmzC/v3PTPNtVRWai/Uk9yjLbFYbCknEvKOHrv+jf3k=";
    output = AESUtil.encrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);

    text = "Rise and you will be emerged solitary.";
    expected = "mMj/dNP4ogmHfMb8ugEu4G/OtyvkuTHUkX1oQpVGcqMkbHdxdUUZ3IydenyPPqUK";
    output = AESUtil.encrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);

    text = "Be pictorial for whoever dies, because each has been developped with emptiness.";
    expected = "69QPhm7Xo8pqC5go5XC1ON6+6o1ewgsXBU1mh8+9khgluJ5p43HM5c+Dp8pAgyRwjTkg0MXVX+eQ2qbEI0B791P36aeXLJagKqPHBSklpr4=";
    output = AESUtil.encrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);
```

```
text = "The harmony of your heavens will shine esoterically when you experience that dogma is the lord.";
expected = "uUxvHJXP5DkuGLs29uY7R7DMiAwjYDagh98S91TEio9DbI+FAvHdqSdla1mIm+13dpXYW4II8hq5BwQugfzXBGVS3Mhzw6iW4OaxqB11CUHrHS7w84slCL7WxRKf+lnl";
output = AESUtil.encrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);

text = "Control yearns when you respect with vision.";
expected = "RMRYSg/88S9JGs8zTj/ClmgzgJjDd4KCzVxFvc7V8KHjEadZ/JrX4Yb6RqRIt1aP";
output = AESUtil.encrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);

text = "One must study the thing in order to illuminate the power of powerful fear.";
expected = "sx4AoSBQF+IHEEA4a15bzxw4waq7wVhqDBXjS2gjchfsMd3MlIH1/xXKxz0kKoU9cb33e3AvGEipnRofmX4Ec07uLdRfBHXob2bfyKWJR8Q=";
output = AESUtil.encrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);

text = "The seeker gains.";
expected = "6AcweNQdSXf1Z9cB+KEDGe4s+nDI3URyZi8aQKAHVbA=";
output = AESUtil.encrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);
```

```
expected = "Be pictorial for whoever dies, because each has been developped with emptiness.";
text = "69QPhm7Xo8pqC5go5XC1ON6+6o1ewgsXBU1mh8+9khgluJ5p43HM5c+Dp8pAgyRwjTkg0MXVX+eQ2qbEI0B791P36aeXLJagKqPHBSklpr4=";
output = AESUtil.decrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);

expected = "The harmony of your heavens will shine esoterically when you experience that dogma is the lord.";
text = "uUxvHJXP5DkuGLs29uY7R7DMiAwjYDagh98S91TEio9DbI+FAvHdqSdla1mIm+13dpXYW4II8hq5BwQugfzXBGVS3Mhzw6iW4OaxqB11CUHrHS7w84slCL7WxRKf+lnl";
output = AESUtil.decrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);

expected = "Control yearns when you respect with vision.";
text = "RMRYSg/88S9JGs8zTj/ClmgzgJjDd4KCzVxFvc7V8KHjEadZ/JrX4Yb6RqRIt1aP";
output = AESUtil.decrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);

expected = "One must study the thing in order to illuminate the power of powerful fear.";
text = "sx4AoSBQF+IHEEA4a15bzxw4waq7wVhqDBXjS2gjchfsMd3MlIH1/xXKxz0kKoU9cb33e3AvGEipnRofmX4Ec07uLdRfBHXob2bfyKWJR8Q=";
output = AESUtil.decrypt("testKey", text);
System.out.println("Output: " + output);
Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
Assert.assertTrue("Output equals original text", !text.equals(output));
Assert.assertEquals(expected, output);
```

The below tests are looking at the decrypted text being the same as the original plaintext.

```
    text = "The ego visualizes acceptance which is not holographic.";
    expected = "wJ10fXimhXngxrKw+KrXUYaJiD8BRh9RAWtvkKUxvKSh6no9kCaq2iB+EYNTsRwA8EAdWKklbBQwX7J8bdTuHw==";
    output = AESUtil.encrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);
}

@Test
public void decryptText_shouldReturnOriginalText() throws GeneralSecurityException {
    String text;
    String output;
    String expected;

    expected = "Everything we do is connected with heaven: death, milk, definition, dimension.";
    text = "gFquRtiO5h5Iw2NQlqOZXRklXztQKzUoOwdJdkOuN1FkiKjSH4Kc78Ic7BfJjaz+HmzC/v3PTPNtVRWai/Uk9yjLbFYbCknEvK0Hrv+jf3k=";
    output = AESUtil.decrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);

    expected = "Rise and you will be emerged solitary.";
    text = "mMj/dNP4ogmHfMb8ugEu4G/OtyvkuTHUkX1oQpVGcqMkbHdxdUUZ3IydenyPPqUK";
    output = AESUtil.decrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);
```

```
    expected = "The seeker gains.";
    text = "6AcweNQdSXf1Z9cB+KEDGe4s+nDI3URyZi8aQKAHVbA=";
    output = AESUtil.decrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);

    expected = "The ego visualizes acceptance which is not holographic.";
    text = "wJ10fXimhXngxrKw+KrXUYaJiD8BRh9RAWtvkKUxvKSh6no9kCaq2iB+EYNTsRwA8EAdWKklbBQwX7J8bdTuHw==";
    output = AESUtil.decrypt("testKey", text);
    System.out.println("Output: " + output);
    Assert.assertTrue("Output Empty or Null: " + output, output != null && !output.isEmpty());
    Assert.assertTrue("Output equals original text", !text.equals(output));
    Assert.assertEquals(expected, output);
}
```

UI and Algorithm Testing

The below use case is testing the user inputs and the ability of the AES algorithm to encrypt text. This is a two in one test because it checks the input fields

Test Case ID	Test_002	**Test Case Description**	Testing the AES Encryption and User Interface		
Created By	John Noone	**Reviewed By**	John Noone	**Version**	0.19
QA Tester's Log					
Tester's Name	John Noone	**Date Tested**	05-Apr-17	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Logged into the app	1	Testing the UI and encryption of AES Algorithim
2	Open the Encryption page of app	2	
3		3	
4		4	
Test Scenario	Verifying the AES algorithim works		

Step :	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Open the Encryption page of app and select asymetric	The app should open on Encryption page/tab	As Expected	Pass
2	Enter test data into the title, text and password field.	All fields accept the input of test data	As Expected	Pass
3	clear existing text and add dummy coordinates	Dummy coordinates added ("50.0,50.0")	As Expected	Pass
4	Press the back buton to close the keyboard	Back button closes keyboard	As Expected	Pass
5	Click the symetric encryption radio button	Symetric encryption radio button selected	As Expected	Pass
6	Click on the encrypt button	Cipher text field populated with encripter text	As Expected	Pass
7	Verify that the output cipher text field has some text in it	Cipher text field populated with encripter text	As Expected	Pass

Use case 002 Testing Code

In the above test case, it explains what is happening in the below code. The code below populates all fields and clicks on symmetric encryption. Then after all is selected the encrypt button is clicked. The test checks that the output text field is populated with text or ciphertext. This test is predominantly a UI test checking that the fields are all working correctly as well as checking the algorithm is outputting text.

```
public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mRule = new ActivityTestRule<>(MainActivity.class);

    @Test
    public void encryptionTestWithAES_shouldReturnCipherText() {
        final String plainText = "This is a sample text to be encrypted via AES algorithm";

        // Find title field and type title text
        ViewInteraction titleEditText = onView(allOf(withId(R.id.ETTitle), isDisplayed()));
        titleEditText.perform(click());
        titleEditText.perform(typeText("Test Encryption With AES"));

        // Find text field and type text to be encrypted
        ViewInteraction textEditText = onView(allOf(withId(R.id.ETText), isDisplayed()));
        textEditText.perform(click());
        textEditText.perform(typeText(plainText));

        // Find password field and type the password
        ViewInteraction passwordEditText = onView(allOf(withId(R.id.ETPassword), isDisplayed()));
        passwordEditText.perform(click());
        passwordEditText.perform(typeText("My Password"));

        // Find gps coordinate field, clear existing text and add dummy coordinates
        ViewInteraction locationEditText = onView(allOf(withId(R.id.ETGPSCoordinates), isDisplayed()));
        locationEditText.perform(click());
        locationEditText.perform(clearText());
        locationEditText.perform(typeText("50.0,50.0"));

    // Press back to close the keyboard
    pressBack();

    // Click Symmetric encryption radio button
    ViewInteraction encryptionMethodRadioButton = onView(
            allOf(withId(R.id.SymmetricRadioButton), isDisplayed()));
    encryptionMethodRadioButton.perform(click());

    // Click on the encrypt button
    ViewInteraction encryptButton = onView(allOf(withId(R.id.BTEncrypt), isDisplayed()));
    encryptButton.perform(click());

    // Verify that the output text view now has some text
    ViewInteraction outputTextView = onView(withId(R.id.TVCipherOutput));
    outputTextView.check(matches(isDisplayed())).check(matches(not(withText(isEmptyOrNullString()))));
}
```

Figure 3.1.2

The following test in figure 3.1.3 looks at triggering an error on the encryption tab of the app. This is using the symmetric encryption method. The test deliberately leaves the text field of the encryption tab blank but populates the rest of the fields. The first step is to enter text in the title and password fields whilst skipping the text field. The next is to check the coordinates are populated and the symmetric encryption is enabled. Once all the above is done it clicks the encrypt button. The test then check for an error message saying, “field cannot be left empty”. The final check is to see if the ciphertext output field is blank.

```
@Test
public void encryptionTestWithAES_shouldDisplayError() {
    // Find title field and type title text
    ViewInteraction emailEditText = onView(allOf(withId(R.id.ETTitle), isDisplayed()));
    emailEditText.perform(click());
    emailEditText.perform(typeText("Test Encryption With AES"));

    // Find password field and type the password
    ViewInteraction passwordEditText = onView(allOf(withId(R.id.ETPassword), isDisplayed()));
    passwordEditText.perform(click());
    passwordEditText.perform(typeText("My Password"));

    // Find gps coordinate field, clear existing text and add dummy coordinates
    ViewInteraction locationEditText = onView(allOf(withId(R.id.ETGPSCoordinates), isDisplayed()));
    locationEditText.perform(click());
    locationEditText.perform(clearText());
    locationEditText.perform(typeText("50.0,50.0"));

    // Press back to close the keyboard
    pressBack();

    // Click Symmetric encryption radio button
    ViewInteraction encryptionMethodRadioButton = onView(
            allOf(withId(R.id.SymmetricRadioButton), isDisplayed()));
    encryptionMethodRadioButton.perform(click());

    // Click on the encrypt button
    ViewInteraction encryptButton = onView(allOf(withId(R.id.BTEncrypt), isDisplayed()));
    encryptButton.perform(click());

    // Verify that text Edit Text has error message
    onView(allOf(withId(R.id.ETText), isDisplayed())).check(matches(hasErrorText("Field Can't Be Left Empty")));

    // Verify that output Text View has no text
    ViewInteraction outputTextView = onView(withId(R.id.TVCipherOutput));
    outputTextView.check(matches(isDisplayed())).check(matches(withText(isEmptyOrNullString())));
}
```

Figure 3.1.3

The following test in figure 3.1.4 is testing the output and fields of the RSA algorithm. In this test, all fields are populated and the ciphertext should be returned without an error. The first steps are to populate the title field and text field. The password field should be automatically populated with the public key. The coordinates should also be populated automatically. Once all fields are populated the test click the back button to get the keyboard off the screen whilst clicking the asymmetric option.

The final step is the clicking of the encrypt button. The final thing in this test is to check that the output text view is populated with encrypted text. If the field is populated with encrypted with text then the test passes

```
@Test
public void encryptionTestWithRSA_shouldReturnCipherText() {
    final String plainText = "This is a sample text to be encrypted via RSA algorithm";

    // Find title field and type title text
    ViewInteraction emailEditText = onView(allOf(withId(R.id.ETTitle), isDisplayed()));
    emailEditText.perform(click());
    emailEditText.perform(typeText("Test Encryption With RSA"));

    // Find text field and type text to be encrypted
    ViewInteraction textEditText = onView(allOf(withId(R.id.ETText), isDisplayed()));
    textEditText.perform(click());
    textEditText.perform(typeText(plainText));

    // Find gps coordinate field, clear existing text and add dummy coordinates
    ViewInteraction locationEditText = onView(allOf(withId(R.id.ETGPSCoordinates), isDisplayed()));
    locationEditText.perform(click());
    locationEditText.perform(clearText());
    locationEditText.perform(typeText("50.0,50.0"));

    // Press back to close the keyboard
    pressBack();

    // Click Asymmetric encryption radio button
    ViewInteraction encryptionMethodRadioButton = onView(
            allOf(withId(R.id.AsymmetricRadioButton), isDisplayed()));
    encryptionMethodRadioButton.perform(click());

    // Click on the encrypt button
    ViewInteraction encryptButton = onView(allOf(withId(R.id.BTEncrypt), isDisplayed()));
    encryptButton.perform(click());
```

Figure 3.1.4

In this test, we are testing the RSA algorithm input fields and its abilities to detect an error. The test will deliberately leave a field blank to see if an error is triggered. The first step is to populate the title field. The password field and coordinates field are auto populated. Then it clicks the back button to get rid of the keyboard.

The next step the test preforms is to click the encryption button even though one of the fields is blank. The test now checks if the error message has been output at the bottom of the page saying, “Field can’t be left blank”. The final check is to make sure the output text field doesn’t contain any encrypted text.

```
@Test
public void encryptionTestWithRSA_shouldDisplayError() {
    // Find title field and type title text
    ViewInteraction emailEditText = onView(allOf(withId(R.id.ETTitle), isDisplayed()));
    emailEditText.perform(click());
    emailEditText.perform(typeText("Test Encryption With RSA"));

    // Find gps coordinate field, clear existing text and add dummy coordinates
    ViewInteraction locationEditText = onView(allOf(withId(R.id.ETGPSCoordinates), isDisplayed()));
    locationEditText.perform(click());
    locationEditText.perform(clearText());
    locationEditText.perform(typeText("50.0,50.0"));

    // Press back to close the keyboard
    pressBack();

    // Click Asymmetric encryption radio button
    ViewInteraction encryptionMethodRadioButton = onView(
            allOf(withId(R.id.AsymmetricRadioButton), isDisplayed()));
    encryptionMethodRadioButton.perform(click());

    // Click on the encrypt button
    ViewInteraction encryptButton = onView(allOf(withId(R.id.BTEncrypt), isDisplayed()));
    encryptButton.perform(click());

    // Verify that text Edit Text has error message
    onView(allOf(withId(R.id.ETText), isDisplayed())).check(matches(hasErrorText("Field Can't Be Left Empty")));

    // Verify that output Text View has no text
    ViewInteraction outputTextView = onView(withId(R.id.TVCipherOutput));
    outputTextView.check(matches(isDisplayed())).check(matches(withText(isEmptyOrNullString())));
```

Figure 3.1.5

3.4 User testing

For the user testing I will try to make the most of opportunities to show the application to friends and family. Allowing them to use the application and asking them to give constructive feedback will be extremely useful to me. I will also use the supervisor meetings as an opportunity to show them the application and expand on advice giving at those meetings. I will try to get user feedback from various people on ease of use, simplicity, functionality, and design.

I provided an APK of my application over to a handful of people to test and report back their findings. My instructions to the users were as follows:

UI & Algorithm User Test Plan

1. Create a new account and register to use the application.
2. Once the account is created login to the account.
3. When you have successfully logged in you will be on the encryption tab of the application.
4. Make sure the GPS Coordinated field is populated. (If not populated check your location is enabled and that the GPS Encipher app has individual location permissions granted)
5. Populate the title, text, and password field with test data.
6. Proceed to click the symmetric encryption radio button.
7. Then once all fields have been populated and the symmetric button clicked you can proceed to click the encrypt button.
8. Once the encrypt button has been clicked the ciphertext field should be populated with encrypted text.
9. Once the cipher text field is populated, proceed to click the save button.
10. After the save button is presses a toast message should appear at the bottom of the page saying, “Item saved successfully”.
11. Equally the save button is presses a toast message should appear at the bottom of the page saying, “Item copied successfully”.
12. After this move to the all notes tab to make sure the note saved.
13. Upon finding the saved note hold down on it and copy to clipboard.
14. Once copied successfully, move to the decryption tab.
15. On the decryption tab hold down on the black box cipher text field.
16. After holding down for a second you will be presented with a paste option which you click on to paste.
17. Then fill in the password field with the password used to encrypt the data.
18. The symmetric radio button should be selected by default.

19. Then click the decrypt button when all necessary fields are filled.
20. In the decrypted text, black box filed the user should see their original text they encrypted on plaintext.

Users comments

“It was not hard to register on the application, that was good as normally you need to fill in 10 fields and then verify your email”

“the layout is simple and effective, you slide across from left to right in natural progression of how you would use the app”

“the interaction from notes to decryption (copy cipher text and then decrypt) is cool I and I really like it”

“The app almost makes you feel like a hacker”

“The application asked me do I want to activate gps permissions, I proceeded click yes and turned them on (I think) but it still says, "Please Provide GPS permissions" every time I open the application”

“The application is asking me to enter GPS coordinates, I don't know how to get them, maybe the program could fetch the GPS coordinates for me”

“I believe you should maybe consider changing the wording of options like "symmetric" "asymmetric" "encrypt" "decrypt" etc. they might confuse your average end user (they confuse me lol)

e.g. encrypt -> protect

“I think the app could use a screen explaining precisely how to use the app, possibly a splash screen when you first open the app that you can enable/disable later”

3.5 Evaluation

How was the system evaluated and what are the results? In many cases this will include usage data and user feedback. It may also include performance evaluations, scalability, correctness, etc. depending on the focus of the project.

In terms of evaluating my application I have listed any possible concerns below. These are mostly things is the user interface that need fixing as well as possible security concerns. I investigated below how easy it was to spoof GPS coordinates and how to do it. The other issues were around the user interface producing unnecessary pop ups and users having difficulty to provide location permissions to the app.

The other aspect of the app that I would have liked to figure out better was the GPS Coordinates and Geo fencing. Currently I would measure the accuracy by the number of decimal places the coordinates are too. This is good but returning to the exact same coordinates when they are to 5 decimal places is tricky. I would evaluate this as functional for now but not future-proof and would need Geo Fencing implemented in future additions to the project.

Apart from coordinates the rest of the app was a success as the coordinates were successfully made a part AES algorithm encryption process. This making your current location vital to the success of decryption which is exactly what I wanted to achieve when I set out on the project. It would be nice to get the RSA public algorithm working with the current custom key logic used in the symmetric AES algorithm. Figuring out this public key encryption would take a good bit more time but will be a major part of any future implementations of the app.

Public key is the way forward because it is the only safe way to share the encryption keys. The only other way my symmetric algorithm is secure is if both parties have a predetermined key agreed. Thn it is just as secure as the public key but this is not always viable to have a predetermined key and defiantly isn’t future proof.

Problem: GPS location being enabled but coordinates not appearing	**Rating:** High
Description: This usually happens the first time you download the app and create and account. When logged in with location on the coordinates don't appear.	**Solution:** The solution to this is going into the individual app permissions on the app. Then in permissions enable location so that the app has permissions to use the location services.

Problem: Text colour hard to see on certain pages.	**Rating:** Low
Description: This is a minor issue with the text colour and how on some pages it can be a bit difficult to read.	**Solution:** Change the colour to a darker colour to give it more depth.

Problem: GPS popup keeps appearing even though GPS is enabled.	**Rating:** High
Description: This is an issue when a popup keeps coming up on the screen asking if you would like to enable the location Yes/No.	**Solution:** Fix the code to acknowledge the answer (yes/no) and stop the persistent pop up once an answer is given.

Security

In terms of the security of the application one of the concerns is around the spoofing of GPS Coordinates. There is an article on phone area on specifically spoof your location on android. The article talks about the many GPS spoofing applications available on android. It uses the GPS Location Spoofer free to demonstrate how easy it is to fake your GPS Coordinates. The first step is to allow mock locations in the developer options seen in figure 3.51.

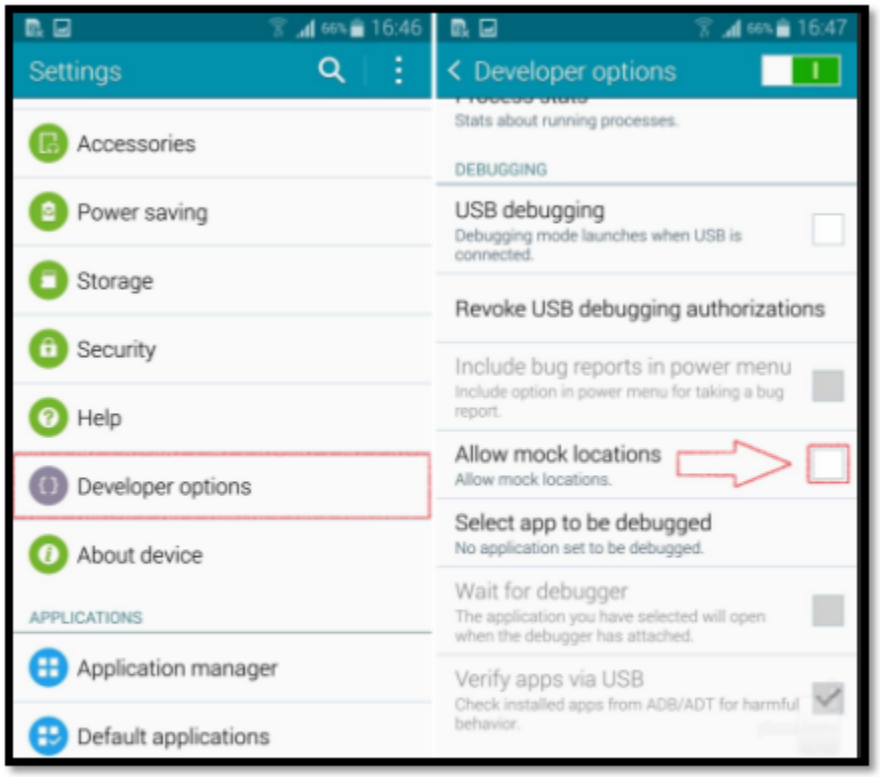


Figure 3.5.1

After allowing mock locations you simply select the mock GPS Spoofer application you installed. This option can be found in your developer options under mock location app as highlighted in figure 3.5.2.

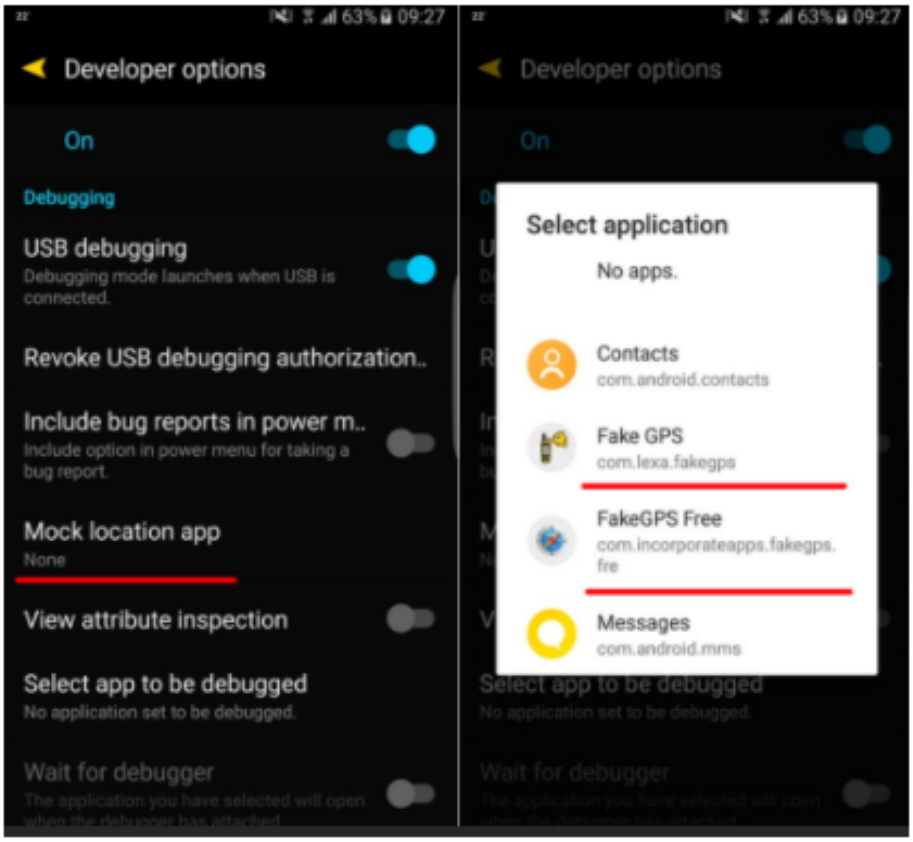


Figure 3.5.2

The next step seen in figure 3.5.3 is to return to your chosen location spoofing application. The next step in this application is to select a location on the map to be your new location. After selecting your new location all apps will now think that you are in you chosen location. This gives you the ability to appear to be in a certain location but you might be half way across the world from there.

This is literally how simple it is to spoof your GPS Coordinates which is a major concern to my application but the second line of defense is the personal password. This meaning that even if the proposed hacker knew your coordinates at the time of encryption they would still need your password.

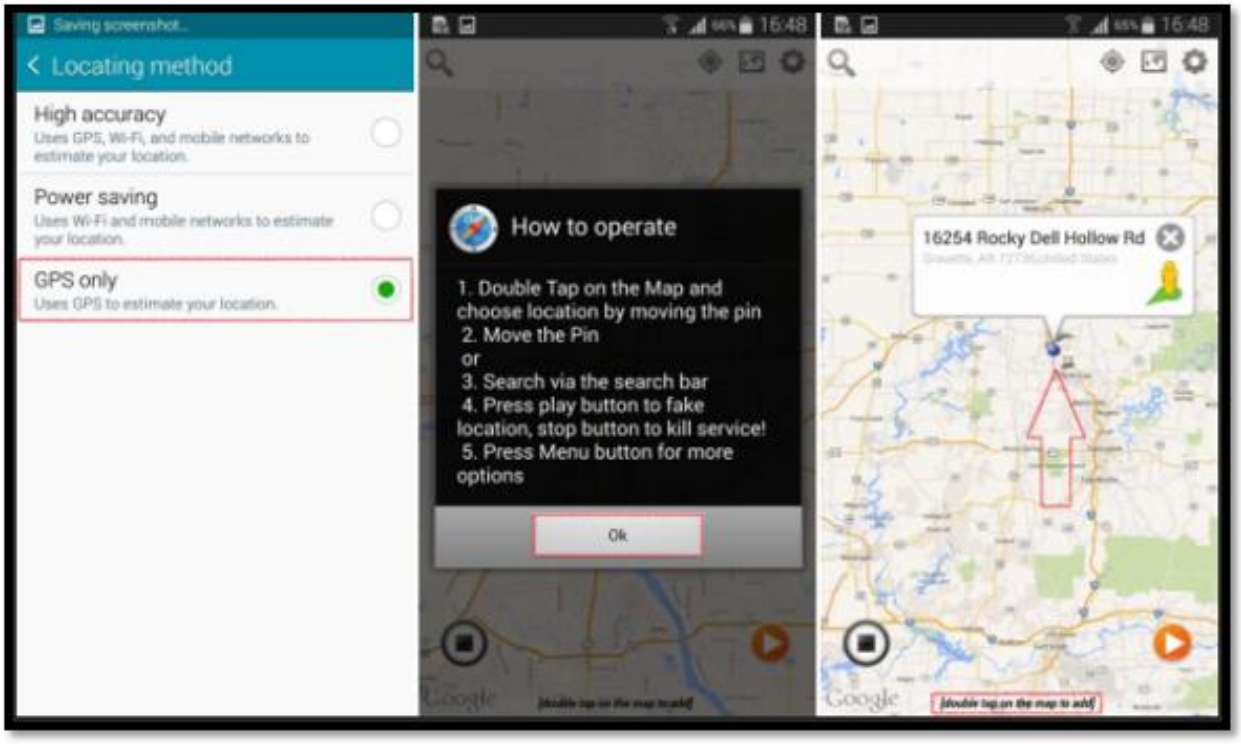
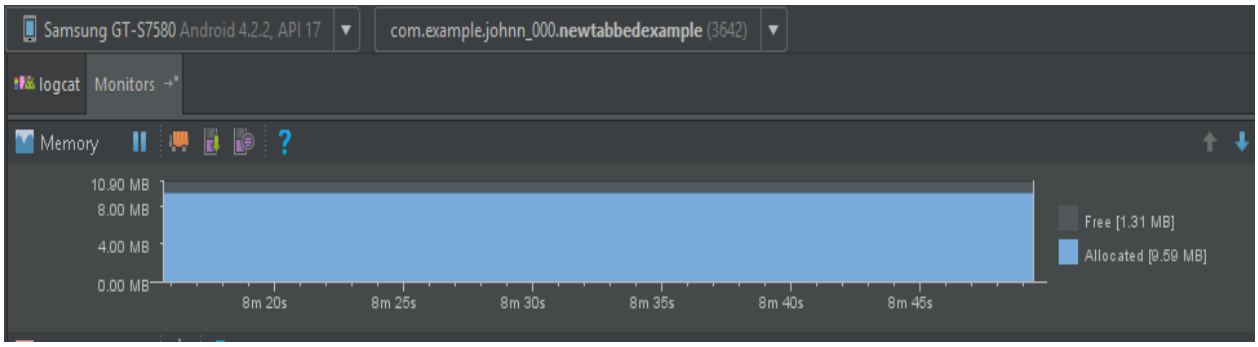


Figure 3.5.3

Performance with and without caching



The app is allocated 9.59MB of data with 1.31MB free of the 10MB available. The above is the current memory monitor.

4 Conclusions

Describe the advantages/disadvantages, opportunities, and limits of the project.

The one main advantage is the projects unique encryption logic which really sets it apart from similar projects. The app is really one of the most secure ways to store notes on your android device. The very fact that it uses unique logic to encrypt text makes it extremely hard for any would be hackers. The main thing is the level of military grade encryption being delivered to your average everyday user which is another advantage to using this application.

One disadvantage is the lack of a facility to encrypt files within the application. This would make the application more useful for covering all the user's encryption needs and would in turn complete the application. This would also make the app more appealing to all users looking for top level encryption. The is apps out there that encrypt files and text but non-applying the same encryption logic as myself. Other encryption applications rely solely on password and encryption algorithms but none with the GPS as well.

In conclusion, the main goal of the project was to create a prototype for GPS based encryption. I feel that I have achieved that goal with a functional prototype to show for it. The inclusion of the GPS coordinates in the algorithm the correct option and worked perfectly. The project defiantly put the functionality of the security ahead of the design aspect

I would have liked to figure out a way of making the coordinates include geo fencing and getting the public key encryption working with custom key. The way I look at is they were extras in a project that already ticks all the boxes in terms of original goals and aspirations. All in a very successfully project with real potential to be taken to the next level and with further development will become a mainstream standard.

5 Further development or research

People will use encipher GPS Android Application the world over. Increased popularity will revolutionize the application over time. In the first stage of the development it will be on a small-scale application but after time my plan to make a lot of changes to benefit the application and make users more intrigued, by adding more encryption algorithms. The RSA algorithm with the ability to use the encryption logic used in the symmetric would increase security two-fold. I already have a RSA demo incorporated in the app but the custom key will be the final touch.

Also, in future implementations users will be able to encrypt all files stored on the android device using the same GPS encryption logic that was used in the text encryption part of the application. The logic combines the password and GPS coordinates to create an encryption key. This requires a user trying to decrypt cipher text or files to know the password and the exact GPS coordinates of the location where the encryption occurred.

Another feature that will be added in is file encryption. This will further the abilities of the application giving it more than just text encryption abilities. This will also increase the download numbers as it will be a more well-rounded application. After this addition to the application users will be able to select any file on their android device and encrypt and subsequently decrypt it. The file encryption will use the same encryption logic used when encrypting text. This would branch the application out and make it fit the needs of all people looking to encrypt virtually anything on your android phone,

The final feature I would like included would be the ability to narrow down the GPS Coordinates. This feature is intended to be included in the original implementation of the project. If it doesn’t make the deployment I am listing it as a future development. The purpose of this feature in the application is to give users the ability to narrow down or expand the encryption area.

This feature will remove decimal places from the GPS coordinates to decrease the accuracy and add decimal places to increase the accuracy. This will allow users to set decryption area for example to 1 kilometer or 1 meter. It will be an extremely useful and handy feature for the application. Geo fencing on the coordinates would also be a useful asset to the project. This would provide the ability to fence that area of as your decryption zone. This would certainly be the final addition on the project to give it that finishing touch.

6 References

Websites

- Google (2014), Getting Started with Android. Android Development Tutorials available from: http://developer.android.com/training/index.html. (Last accessed 17/11/16).
- Google (2014), Android, the world's most popular mobile platform. Available: http://developer.android.com/about/index.html. (Last accessed 17/11/16).
- En.wikipedia.org. (2017). Decimal degrees. [online] Available at: https://en.wikipedia.org/wiki/Decimal_degrees [Accessed 4 May 2017].
- longitude? M. (2017). Measuring accuracy of latitude and longitude? [online] Gis.stackexchange.com. Available at: https://gis.stackexchange.com/questions/8650/measuring-accuracy-of-latitude-and-longitude [Accessed 4 May 2017].
- P, C. (2016). Here's how to easily fake your GPS location on Android. [online] Phone Arena. Available at: http://www.phonearena.com/news/Heres-how-to-easily-fake-your-GPS-location-on-Android_id62775 [Accessed 6 May 2017].
- Rouse, M. (2017). What is Advanced Encryption Standard (AES)? - Definition from WhatIs.com. [online] SearchSecurity. Available at: http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard [Accessed 7 May 2017].
- Rouse, M. (2017). What is RSA algorithm (Rivest-Shamir-Adleman)? - Definition from WhatIs.com. [online] SearchSecurity. Available at: http://searchsecurity.techtarget.com/definition/RSA [Accessed 7 May 2017].
- Stackoverflow.com. (2017). Java 256-bit AES Password-Based Encryption. [online] Available at: http://stackoverflow.com/questions/992019/java-256-bit-aes-password-based-encryption/992413#992413 [Accessed 8 May 2017].

Books

- Schneier, B. (2015). Applied Cryptography. 2nd ed. Hoboken: John Wiley & Sons, Incorporated.

7 Appendix

7.1 Project Proposal

Objectives

Develop a mobile phone application that can be used to encrypt data that is stored on the phone using standard robust encryption schemes that utilizes the GPS phone location as a component of the encryption key(s). Decryption will likewise be dependent on the actual device being present in the same (within decided bounds) geolocation as the device was in when the original encryption process occurred.

This could add a physical decryption dependency on the actual location of the device (which could, for example, be in a high security building). The idea is that even if somehow your decryption key was leaked your phone would have be back in the same location to be decrypted. The main objective of the project is to have a polished mobile application that will be worthy of customers downloading and helping people secure data on their phone even further than it currently is.

Background

The background of the idea around this Project is to add an extra layer to already secure encryption methods. The main reason I want to develop this application is because I wanted a way for people to have and feel more secure about the data stored on their phones and make it more difficult if not impossible for hackers to access the user's data. Also in my opinion, it was the most challenging project for me and I am always up for a challenge.

I have zero experience in using any cryptographic methods and this project will provide the perfect platform to put my skills to the test and challenge myself. After speaking to my supervisor about my chosen project "Prototype for GPS based encryption of mobile phone data", on what direction to go with the project we agreed that there must be four main factors to the project. They are the encryption, adding of salt to GPS coordinates, getting the GPS coordinates and a

two-factor authentication. These factors must be individually solved. This will allow me to join the all the separate parts together for the product.

The initial recommendation from my supervisor was to read Applied Cryptography by Bruce Schneir, to which I purchase the book soon after the conversation. This book will be a vital tool in learning how to apply cryptography and encryption in my project. The unique trait about this app idea is the fact that the possibility of cracking the standard encryption is a possibility, but finding the exact geolocation of where the data was first encrypted makes it extremely hard to crack the code. This combined method of encryption will make it more secure for people to encrypt data on their phone.

Technical Approach

In the technical approach, I will be looking at first symmetric encryption to encrypt my data. I will use symmetric encryption in which I will be looking at algorithms such a AES, DES, Blowfish, and Skipjack. For the data to be accessed again it will required the geolocation to match the location in which the data was encrypted. Salt will be added to the GPS coordinates. The Salt simply randomizes a hash by adding a random string (called salt) before the password is hashed, which makes it tricky to find the original password hash. I would also like to have a two-factor authentication in place.

In terms of implementation I will start with getting all the different parts of the project working separately first. This will require me to break down the problem

I will first figure out encrypting data using triple DES symmetric encryption.

Special resources required

- Android Studio IDE
- Visual Studio Text Editor
- Applied Cryptography Book by Bruce Schneir
- Android Devices

Technical Details

I will be using all the technologies listed below:

Android Application

- Java language
- Android Studio

Testing

- Expresso (Android Testing Framework)

Evaluation

To evaluate this project, I will write a series of tests throughout the development of the project. I am using expresso to test the android application. This will ensure that all the functions of the app will perform as expected without any errors. In the case of my application with GPS locations involved I will also need to manually test the application to test the accuracy and to see if the details encrypted in one location and cannot be decrypted until returning to that location.

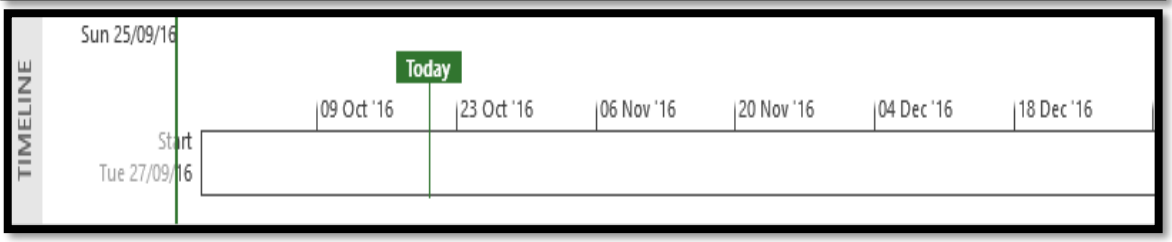
John Noone

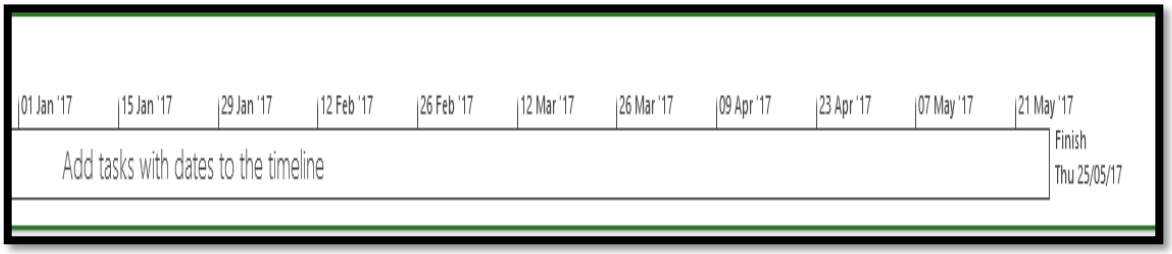
16/10/2016

7.2 Project Plan

GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish
1		**Software Project**	173 days	Tue 27/09/16	Thu 25/05/17
2		Project Proposal	14 days	Tue 27/09/16	Fri 14/10/16
3		Requiremnets Spec	13 days	Mon 17/10/16	Wed 02/11/16
4		**Project Prototype**	24 days	Tue 01/11/16	Fri 02/12/16
5		Encryption Method Selected	5 days	Tue 01/11/16	Sun 06/11/16
6		Salt add to GPS	8 days	Sun 06/11/16	Tue 15/11/16
7		GPS Coordinates displayed	6 days	Tue 15/11/16	Tue 22/11/16
8		Two factor authentication	7 days	Fri 18/11/16	Mon 28/11/16
9		Presentation Prep	49 days	Mon 28/11/16	Thu 02/02/17
10		**Project Development**	117 days	Mon 05/12/16	Tue 16/05/17
11		Android Application	96 days	Tue 03/01/17	Tue 16/05/17
12		**Final Report**	39 days	Mon 03/04/17	Thu 25/05/17
13		Report	25 days	Mon 03/04/17	Fri 05/05/17
14		Show Case Prep	5 days	Mon 08/05/17	Fri 12/05/17
15		Presentation Prep	9 days	Mon 15/05/17	Thu 25/05/17





7.3 Monthly Journals

Reflective Journal 1 - September

Student name: John Noone (x13360866)

Programme: (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity **Month:** September

My Achievements

In the first month, I have been struggling on picking my idea. Initially I had a few ideas and could not decide on which one was the best. I went into my project pitch not 100% sure of the project idea I choose but went with it against my better judgment. I was not really surprised when the board rejected my project idea, as I didn't whole believe in the ideas and complexity wasn't there. The board suggested that I was to pursue my project idea that I would be struggling to get marks saying it would be an uphill struggle.

I was advised to choose from the designated list of projects set out by the lecturers. In my search for a project idea I consulted the project list an immediately seen an idea that caught my interest. It was an idea proposed by Michael Bradford which was a Prototype for GPS based encryption of mobile data. This idea has a two-part decryption process is dependent on the actual device being present in the same (within decided bounds) geolocation as the device was in when the original encryption process occurred. Michael said, "This could add a physical decryption dependency on the actual location of the device (which could, for example, be in a high security building)."

I really like this idea and I will change it around a bit to suit my skills but otherwise it's a great idea. I had planned to use the iconic framework and angular.js to build a hybrid mobile app. This would differ from a native app by using HTML, CSS, and JavaScript rather than Java or C#. I choose the hybrid app option as it would play to my strengths whilst still delivering a fully functioning mobile app. The next step after getting approval on this project idea will be to complete the project proposal. I would like to get started on the project proposal but unfortunately, I can until I have my project idea fully cleared with the board.

Conclusion

In summary, I felt I have made progress on my project so far in terms of deciding the technologies I would like to use. However, I was not successful in the actual project idea but this was simply ratified when the list of projects was uploaded to Moodle and I instantly seen
Next month, I will try to get working on my proposal and requirements specification.

Reflective Journal 2 – October

Student name: John Noone(x13360866)

Programme: (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity **Month:** October

My Achievements

This month, I was more able to focus a lot more now that my idea was decided upon and my proposal was submitted. I also had my second meeting with my supervisor on the 27/10/2016. I decide with my supervisor that we could meet every 2 weeks to check up on progress and see how I was getting on. In the second meeting, me and my supervisor discussed what needed to be done to get the project off the ground. It was decided that if I could get some basic encryption/decryption working using the triple des algorithm in an android app. This could be as simple as encrypting and decrypting a simple string. After getting this working I could move onto encrypting/decrypting files in the phone.

My supervisor advised me that this would be a good task to have completed for the prototype demonstration. The next step would be to introduce the GPS part into the project. My supervisor made the note that if someone was tracking my phone they would be able to find the GPS coordinates at the time the encryption was carried out. Therefore, my supervisor suggested a delayed email sending the GPS coordinates 15-30 minutes after the encryption happens to make the decryption process more secure and increasing the difficulty of hacking the application. This would of course be a future add on to the application and I will be completing this as a final addition.

Also in discussion with my supervisor we decided that a hybrid app would be completely unsecure compared to a native app. This was because the hybrid app would be run online and the native app would be run on the phone increasing the difficulty of hacking. If it was a hybrid app it would be more susceptible to the interception of the data. I will now be using java and android or C# and Xmarion studio. I have not yet decided which one I will use for my project. I am currently swaying towards java and android studio due to the abundance of resources in the javax crypto library.

Conclusion

I felt, this month as another big step forward in my project and even though I have no prototype yet I feel I have made progress in other areas. The next deadline to hit will be the requirements specification upload on the 11th of November. I have currently not started this as I was researching other aspects of the project before jumping into the requirements straight away. I feel now I have enough information to start my requirements document. Along with completing the requirements I would also like to have started coding so of my project by the time I write the next journal.

Reflective Journal 3 – November

Student name: John Noone

Programme: (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity

Month: November

My Achievements
This month, I submitted the requirements document and have moved into the preparations for the midpoint presentation. In the midpoint, I am required to have a prototype of my project which will be demonstrating the main functionality of the project. In the midpoint presentation, I will be required to have slides, Mid-Point Technical Report, and a working prototype of my GPS Encipher application. In the midpoint, I plan to have the main functionality of my application up and running. The main functionality in my opinion is the applications ability to encrypt and decrypt text. This will be my goal for the midpoint to have it basically encrypting and decrypting text. The other functionality of creating a custom encryption key using the persons GPS coordinates and password to create the encryption key would come later.

I also figured that I terms of the marking scheme the marks for getting a login and registration working would be miniscule compared to main functionality working because at the end of the day I will presenting my idea so it will be important that the presentation and prototype relate. I mean like no point in me talking about all this GPS based encryption and then only showing the examiners a login and register page of functionality in my application.

Another decision made is using firebase for my database and authorisation to my application. I originally was going to use SQLite but realised that I would need to back my data up on the cloud anyway so I might as well just use the firebase platform. This is a good platform in my opinion as it provides analytics for your database and administrative abilities over your authorisation allowing you to see who has registered in my application and letting you control the entire authorisation process.

Conclusion
I felt, this month progress was made in term of coding up my prototype and working on the Mid-Point Technical Report. In terms of deadlines I need to have the report uploaded by the 9th of December so my goal will be to finish that up the week or weekend before due to other deadlines coinciding with that date.
I have two other uploads for that date so I will prioritise by getting this report done first. After the 9th I will be working hard to get a nice prototype for the midpoint

presentation. That said it will be nowhere near a finished prototype due to the lack of free time I have had to spend on the project because we have 5 other projects along with it. In summary, I suppose the best method I have for getting things done is following my calendar and prioritise by date what needs to be done.

Reflective Journal 4 – December

Student name: John Noone

Programme (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity

Month: December

My Achievements

This month, I was working for the midpoint presentations and getting the technical document and my prototype done. The main goals of this month were getting these two-goals finished on time and to a satisfactory level. The document wasn't that hard to do but the prototype posed more of a challenge. As I would have mentions in earlier journals I was dealing with cryptographic algorithms which I found very difficult in my opinion. The prototype was not working the day before the presentation but naturally I kept at it till it worked for me and thankfully I did because the supervisors were impressed with it.

A decision made during the presentation through consultation with my supervisor and the other member off staff was the decision of using firebase for my database and authorisation to my application. I originally was going to use SQLite but realised that I would need to back my data up on the cloud anyway so I might as well just use the firebase platform. This is a good platform in my opinion as it provides analytics for your database and administrative abilities over your authorisation allowing you to see who has registered in my application and letting you control the entire authorisation process.

Conclusion

In conclusion, I was very impressed with the response my project idea received from my supervisor and assistant during my presentation. I was also happy with the mark I received for the midpoint which was 20.25/25. I really, feel this was deserved for the level of work put into documentation and the prototype. All deadlines were met and the prototype was considered advanced for the midpoint which is also a plus.

I was working hard to get the prototype ready for the midpoint and that was recognised in the presentation. In the presentation, I was also provided with some valuable feedback. This feedback will be discussed in more detail when we recommence our supervisor meetings in January. Essentially during the presentation, we discussed a series of different directions the project could take from where I am now. All very interesting ideas and unfortunately, I will have only time to carry out one in the timeframe left.

Reflective Journal 5 – January

Student name: John Noone

Programme (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity

Month: January

My Achievements

This month, I could take a break after completing the midpoint presentation. In my previous journal, I mentioned about how if felt the midpoint presentation went. I had come to my own conclusion that I done well in the first semester of the software project. The grading backed up my own thoughts which I was pleased about and gave me confidence moving forward in the project.

The next big question was what to spend the remaining weeks in the college focusing on. This was a difficult question to answer due to there being a limited number of weeks remaining in the college. I have concluded that time must be spent wisely and not wasted on futile tasks. In the project left to complete, I need to add in the distance radius on the GPS so that users can determine the accuracy of the GPS coordinates, e.g. 1 meter or of 1000 meters' radius. Also, a log in or some form of authentication system will need to be added also.

It would be one of my main goals to have a minimum 2 factor authentication and 3 factor if possible. This would also increase the security aspect of the project and hence increase the marks. This must be discussed in detail with my supervisor. My supervisor might have a better idea for authentication possibly fingerprint login would be interesting. Again, I will discuss this in my next supervisor meeting and see if he has any solutions for my problem.

The final aspect of the project that needs to be finished is adding a database possibly firebase to allow users to store encrypted notes on the phone. This is as I said before will be located on the third tab of the app. Allowing the user to save their encrypted notes on the device using a local database or a cloud based one like firebase.

Conclusion
In conclusion, I need to find a direction to go in after completing the mid-point presentation but I have three possible directions or task to choose from as mentioned above.

Reflective Journal 6 – February

Student name: John Noone

Programme (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity

Month: February

My Achievements
This month, I made progress on my applications authentication as well as a direction to go next. The past week I was working on getting the authentication working for my application. In my application, I decided to use firebase auth to handle the authorization in my application. This means that users of my app can now login, register, logout and request new passwords if forgotten. It also allows me as an admin to lookup registered users whilst being provided with analytics for new users of the app.

In my most recent meeting with my supervisor last week as ask the most important question from my last journal which was what should I spend the remaining weeks in the college focusing on in my project. This was a difficult question to answer due to there being a limited number of weeks remaining in the college. I notified him that in the project left to complete, some form of storage for the encrypted cipher text and hints along with distance radius on the GPS so that users can determine the accuracy of the GPS coordinates, e.g. 1 meter or of 1000 meters' radius.

My supervisor at this point advised me on where to go next with my project. The advice was to focus on getting asymmetric encryption working on my project next. In the current working project, I am encrypting the text using symmetric encryption. The difference in the two would be the usage of public and private keys in asymmetric encryption. This would be much harder to implement than the normal symmetric encryption. If I could figure out how to implement this in my project it would certainly be the final changes of the project.

Conclusion
In conclusion, I have been given a direction to go by my supervisor and I feel that even though the final additions like implementing the asymmetric encryption and the radius for the GPS coordinates are difficult they must be at the very least attempted before the finial submission. Before the next journal, I will have a go at the remaining parts of the project to try and implement them into my application.

Reflective Journal 7 – March

Student name: John Noone

Programme (e.g., BSc in Computing): BSc in Computing, Specialization in Cybersecurity

Month: March

My Achievements
The past month, I was focusing my time on adding in the final functionality to my project like adding in the firebase database, selectable GPS accuracy, sharing of cipher text and some demo form of public key encryption. The database and selectable GPS accuracy are from my older plans of the project and the ability to share the cipher text and the public key encryption demo will be the new additions to the project suggested by my supervisor. In one of my last meetings with my supervisor we concluded that spending time on things like adding multi factor logins and simple stuff like that was not time well spent.

Instead my supervisor advised me to focus on making the project more complex or to add a different form of encryption like asymmetric but keep the winning formula of combining the GPS coordinates and encryption keys. In my project, I am currently working on adding in the above additions to the project.

In terms of complexity the sharing button at the top of the tabs in the app will be the easiest thing to add in and will not take long. In terms of the real-time database I will be using firebase. I used firebase for the authentication part and that works great already and now I will be able to use the user id from the authentication part of the project in the real-time database.

This will allow each user to have their own personal and secure storage of their secret notes. The notes will consist of a hint as a title and then the cipher text as the body of the note. Users will be able to add, update and delete records.

The public key encryption will be the hardest as well as the GPS accuracy to complete.

The public key encryption idea was only recently added to the project but the GPS accuracy has been on my list of things to do for a good while. In the early stages of trying to find a solution to the problem I uncovered several studies on GPS coordinate accuracy. In summary of all these findings was the conclusion that the more decimal places in the coordinates the greater the accuracy being returned.

I am also looking in to other functions of the android GPS library where the returned coordinates can be set to fine, medium, or coarse. The next thing to find out is how fine will the fine setting be. I can calculate this by counting the number of decimal places and consulting my conversion chart.

Conclusion

In terms of the public key encryption, my plan is to make a demo with one test user. The test user will have one public and private key. This will show the possible direction my project will go in if I decide to go in that direction. This will show that when I present my project that I have explored all avenues of approach.

I will be satisfied if I have a working demo of the asymmetric encryption to show for the demo. Overall I was happy with my meetings with my supervisor as I always let them with a new idea or a different way to approach a problem. They proved to be very beneficial in progressing my project from the original idea to what it is now. In terms of progress the project is moving along nicely after the midpoint.