DEVELOPMENT OF A SECURE COMMUNICATION SYSTEM BASED ON STEGANOGRAPHY FOR MOBILE DEVICES

by
Dasarathan Selvaraj

A thesis submitted in partial fulfillment
of the requirements for the degree
of

MASTER OF SCIENCE (M.Sc.)

in

High Integrity Systems
Frankfurt University of Applied Sciences

Under the guidance of

Prof. Dr. Christian Baun
Advisor
Frankfurt University of Applied Sciences

Prof. Dr. Matthias Wagner
Co Advisor
Frankfurt University of Applied Sciences

Acknowledgement

It has been a pleasure to work with all the concerned members of the thesis. First, I would like to thank my Professor Dr. Christian Baun for his consent to be my examiner. I am grateful for his valuable guidance in planning and development of the thesis work. Without his profound guidance, this thesis would not have been possible. His excitement and willingness to provide feedback made the completion of this research an enjoyable experience. I would like to thank my co-advisor Prof. Mathias Wagner, who has shown the right attitude towards the study. He persistently conveyed the excitement in regards to teaching.

Furthermore, I thank the Frankfurt University of Applied Sciences for making my educational carrier a special one. The facilities provided by the university was exceptionally useful for my whole studies and for completion of this thesis. A special thanks to my colleagues for their ever-consistent and intuitive support in completing the work. Finally, I would like to thank my family, who has supported me in different ways. Even if their contributions are intangible, I express my sincere gratitude to all of them.

Frankfurt am Main, 27 October 2014 Dasarathan Selvaraj.

Abstract

Due to rapid growth in the use of mobile devices, security issues shifts from the personal computing platform to new mobile technology. Mobile devices have become the new source of communication. Divergence of focus towards new audience and their new source of communication is necessary to know about their needs and to provide it with improved quality. In addition to secure communication, hiding the fact that a secret message exists in a digital material is an innovative concept. The distinct approach of combined cryptography and steganography is discussed in this thesis. Steganography is a method to hide data, but it is not the same as cryptography. The goal of modern steganography is to hide the information in an innocent digital carrier, so that it is undetectable by third party.

This unique solution is developed and implemented as an android application for mobile devices. This android application embed the message in a cover image or in an audio file. In addition, it features the ability to extract the concealed message from the embedded cover file. The application support text message and quick voice message. Process of development and implementation of this concept is proved and documented.

Declaration

I hereby declare that this thesis is my own work and effort. It has not been submitted, either in whole or in part, anywhere for any awards. Where other sources of information have been used, they have been acknowledged.

Frankfurt am Main, October 27 2014 Dasarathan Selvaraj

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work introduce the concept of steganography, purpose of this thesis and need for this thesis at modern era. This thesis's objective and optimal way to approach it, is discussed briefly. Additionally a glimpse of other chapters are provided.

1.1 Background

The process of embedding secret information into a carrier is called steganography and its origins can be traced back to ancient times. Moreover, its importance has not decreased since its birth.[3]

Society's daily activities are highly influenced by latest digital technologies and communication. Major economical and social impact are felt in all sphere of its existence. Evidence of the development of smaller, faster and high performance mobile devices, which can support a wide range of features, is supporting the fact of rapid growth in mobile technology. The attributes of personal computers are slowly injected into mobile devices. Mobile hand-held devices which are popularly called smart gadgets includes smart phones, tablets and e-book readers. Now a days, they are becoming essential to everyday social activities. These newly developed devices provides better processing, storing and transmission of information. In this ever changing and evolving environment, establishing secure communication is an important target for researchers.

Undoubtedly, Android becomes the apt destination to keep in touch with the community of digital

device users. Android is an open-source operating system offered to mobile developers to use on their smartphones (Samsung, HTC, Sony, etc.) and free for developers to expand on.[21] Among adults who have mobile phone, 58% of them have smart phone in 2014. [17] Some mobile users does not always stay as a user, instead they turns to mobile addicts. Data suggest 123% of increase in addicts this year when compared to last year.[18] Interesting fact is 65% of global smartphone owners use Android OS.[19] Android has 56% of the mobile market share with 375 million handsets sold already.

More internet users are using photos and videos as a social currency. 54% of internet users have posted original photos or videos to websites and 47% share photos or videos they found elsewhere online. Young adults and women lead the way in each of these activities. Cell phones and smartphones have given rise to photos and video sharing apps. 18% of mobile owners use Instagram and 9% use Snapchat. [20]

In current smart phone era, number of photos are taken with smart phone. All most all newly released phone and tablets have at least one camera. All high end smart phone have both front and back camera. Now a days, it has become so easy to take photos at any time anywhere. Selfies have become so popular that one third of photos taken by phones are selfies. Smart phone is replacing traditional cameras. The study found that the majority of today's photo taking smartphone customers, who've taken at least 10 photos in the last three months use only their smartphone to take photos. [12] Global market have sky rocketed for smart phone while cameras have foundered [13].Thus Android becomes the suitable choice for implementing image steganography.

1.2 Evolution of Steganography

Steganographic techniques have been used for ages and they date back to ancient Greece. The term "Steganography" was first recorded in 1499 by Johannes Trithemius in his book, Steganographia. In world war some messages were written using invisible ink. Liquids such as milk, vinegar and fruit juices were used as invisible inks, because when each of these substances are heated they darken and become visible to human eyes.

Historical methods relied on physical steganography. The employed media were human skin, game, etc. What distinguishes historical steganographic methods from the modern ones is, in fact, only the form of the cover (carrier) for secret data. [10]

In Ancient Greece they used to select messengers and shave their head, then they would write a message on their head. Then hair was grown and the messenger was sent to deliver the message with the secret message inside the hair. The recipient would shave off the messengers hair to see the secret message.

Another method used in Greece were someone would peel wax off a tablet that was covered in wax, write a message underneath the wax then apply the wax again. The recipient of the message would simply remove the wax from the tablet to view the message.

Text steganography is hiding message in a text by abnormally altering its fonts, size, alignment and other techniques.

1.3 Approach

Cryptography and steganography are two techniques used to ensure information confidentiality, integrity and authenticity. Cryptography uses encryption to scramble the secret information in such a way that only the sender and the intended receiver are able to reveal it. Steganography hides the secret information in different carriers in such a way that it becomes difficult to detect. Commonly the carriers are media files or other supports like communication protocols. An example is network steganography.

The advantage of steganography over cryptography is that, there is no attraction of encrypted message. In cryptography, it is well know that the message is encrypted. It is not the case with steganography. Steganography is not only to concealing the contents of the message but also to concealing the fact that a secret message is being sent. One of the limitation of steganography is considered to be advantageous to society. Substantial process of steganography is to store data in image or any other file. Arising question on possibility of steeling data is hammered down immediately as the size of concealed data is very low. Believably, few kilobytes of data stolen is not going to harm a lot. Even though, this case cannot be completely ruled out. Because there

can be small quantity of valuable data.

Both technologies have their limitations and this is why most of the specialists sustain that a good solution for securing the digital information is to combine the two techniques.[9] In this paper I propose an application named protectMSG developed to transmit secret files through Internet and mobile networks using a smart phone that run Android operating system. It involves technical steganography combined with symmetric key cryptography and a pseudorandom selection of the bits.

1.4 Chapters at a Glance

This thesis consists of seven chapters. Their purpose is as follows:

Chapter 1 is introduction to steganography and need for the thesis. Helps to identify the basic methodology of this work and the expected outcome from it.

Chapter 2 provides in-depth knowledge of the technologies that are the building blocks of this thesis. Modern techniques of steganography and cryptography classifications and types of commonly use encryption are elaborated.

Chapter 3 throws light on Android Operating System and its components. Also highlights the advantage of using Android. Elementary installation and configuration required for the development of project is explained.

Chapter 4 describes the design based on requirement of the final product. According to the scope of the project, gathered analysis are put together as functional and non functional requirements. Construction of architecture, algorithm and flow of the applications are explained in detail. Necessary inputs for the development phase such as use case, package model and class diagram are designed.

Chapter 5 gives elaborate details of implementation of this thesis. Details about the helpful tools used during the development and their importance in implementation is provided. All function of the application and the way of execution to obtain the result are written here.

Chapter 6 captures test results, behavior and every approach handled to provide best quality product.

Chapter 7 is the final chapter of this dissertation and details the summary of the entire dissertation and exploits the area of scope for future enhancements.

The next section consists of the appendices, glossary of terms and the bibliography of references.

Chapter 2

Literature Review

This chapter provides in depth knowledge of technologies that are the building blocks of this thesis. Modern techniques of steganography and cryptography classifications are explained. Also the types of commonly used encryption are elaborated.

2.1 Digital Steganography

A famous illustration of steganography is the prisoner's problem where Alice and Bob are in jail, locked up in separate cells, who wish to communicate in order to hatch an escape plan. However, all communication between them is examined by a warden called Wendy, who will put them in solitary confinement at the slightest suspicion of trouble. Specifically, in the general model for steganography, we have Alice wishing to send a secret message M to Bob. In order to do so she "embeds" M into a cover object C, to obtain the stego-object S. The stego-object S is then sent through a public channel. The warden Wendy who is free to examine all messages exchanged between Alice and Bob can be passive or active. A passive warden simply examines the message and tries to determine if it potentially contains a hidden message. If it appears that it does, then she takes appropriate action else she lets the message through without alteration. An active warden on the other hand can alter messages deliberately, even though she does not see any trace of a hidden message,in order to foil any secret communication that can nevertheless be occurring between Alice and Bob.[23]

Given the above framework, the main goal of steganography is to communicate securely in a completely undetectable manner. That is, Wendy should not be able to reliably distinguish in any sense between cover-objects (objects not containing any secret message) and stego-objects (objects containing a secret message). In this context, steganalysis refers to the body of techniques that are designed to distinguish between cover-objects and stego-objects.[23]

There are several steganographic techniques exist and they are still being invented. They are classified into technical steganography and linguistic steganography. Technical steganography is a scientific method of hiding a message, such as the use of invisible ink or microdots and other size-reduction methods. In Linguistic steganography, message is hidden in a carrier in some non obvious ways and is further categorized as semagrams or open codes [4]. Semagrams hide information by the use of symbols or signs. A visual semagram uses innocent looking or everyday physical objects to convey a message, such as doodles or the positioning of items on a desk or Website. A text semagram hides a message by modifying the appearance of the carrier text, such as subtle changes in font size or type, adding extra spaces, or different flourishes in letters or handwritten text. [4] Open codes hide a message in a legitimate carrier message in ways that are not obvious to an unsuspecting observer. The carrier message is sometimes called the overt communication whereas the hidden message is the covert communication. This category is subdivided into jargon codes and covered ciphers.[4] Jargon code use a special language which is understood only by a group of people. It is meaningless to others. Covered or concealment ciphers is a method of hiding message openly in the carrier medium so that it can be recovered by anyone who knows the secret for how it was concealed. A grille cipher employs a template that is used to cover the carrier message. The words that appear in the openings of the template are the hidden message. A null cipher hides the message according to some prearranged set of rules, such as "read every fifth word" or "look at the third character in every word.[4] So, in this thesis, covered cipher is the technique used to hide messages.

Few application of steganography are:

- Copyright
- Watermarks

- Covert military operations
- Keys
- Intelligence agencies
- Medical imagery
- Checksum embedding

2.2 Digital Carrier Methods

In today's world of digital technologies, the carrier in which secret data is embedded, is not necessarily an image or web page source code, but it can also be any other file type or any organizational unit of data. For example, a packet or a frame in a computer network can be used as a carrier for steganography. Data are stored in left out file slack or unallocated space as the remains of previous files. Unused part of file headers are used to hide secret information. Information can also be hidden in a hard disk in a secret partition. In order to retrieve the data, programs can be written to access slack and unallocated space directly. Network protocol is also used as an digital carrier. Many new covert communications are possible through internet. Recently, steganography has widened in the field of networks. Typical network steganography method uses modification of a single network protocol. The protocol modification may be applied to the PDU (Protocol Data Unit), time relations between exchanged PDUs, or both (hybrid methods). Moreover, usage of relation between two or more different network protocols to enable secret communication is possible. It is so called inter-protocol steganography.[10]

Sound has characters such as frequency, phase angle and speech cadence. Slight change in these values are not recognizable by human ears. It is an advantage for hiding message in these values. Most common carrier medium is image and audio file. Operation on these carriers are easier than other carriers. Least significant bit substitution or overwriting is the commonly used steganography method.

Combining the idea of both image and audio forms the video steganography. The strength of the steganographic technique relies mainly on three factors - imperceptibility level in the stego

image, robustness and embedding capacity.[22]

2.3 Steganalysis

Due to increase in number of technique used for steganography, a new study called steganalysis has evolved. This study has evolved for security purpose and its aim is to prevent illegal communication. Purpose of steganalysis is to detect the messages that are hidden using steganography by a third party. In the process of steganlysis, the cover file is tested for its file properties and verified for any pattern in the file content. Like this several other ways are used to identify whether the file has any hidden message. And several tool are available online for detecting whether the file is subjected to steganography.

2.4 Image File Types

Practical use of image and photos in current digital world is engaged with different file formats. All available image formats will fall under any one of these classifications - compressed, uncompressed and vector formats. Part of the reason for the plethora of file types is the need for compression. Image files can be quite large, and larger file types mean more disk usage and slower downloads. Compression is a term used to describe ways of cutting the size of the file. Compression schemes can by lossy or lossless.[8] Quality of the image depends on the accuracy of displaying the color in it. Color depth is the number of bits used to display the color in a pixel. Color depth is directly proportional to the size of the file. Another reason for the evolution of many image types is the different in the color depth they have. Lesser the number of colors, lesser the file size.

A lossless compression algorithm eliminates no information. It looks for more efficient ways to represent an image without making any compromises in accuracy. But in contrast, lossy algorithms accept some degradation in the image to make it smaller in file size. A lossless algorithm will look for a recurring pattern in the file, and replace it with a short abbreviation. By doing this file size is reduced. In contrast, a lossy algorithm might store color information at a lower resolution than the image itself, since the eye is not so sensitive to changes in color of a

small distance.[8] Major image file types available are JPG, GIF, TIFF, PNG, and BMP.

TIFF is, in principle, a very flexible format that can be lossless or lossy. TIFF is used almost exactly as a lossless image storage format that do not compress at all. PNG is also a lossless image format. It actually looks for patterns in the image that it can be used to compress file size. The compression is exactly reversible, so the image is recovered exactly. PNG is the only lossless format that web browsers support and later generation web browsers support PNG. It produces smaller files and allows more colors. PNG supports partial transparency. Partial transparency is used for many useful purposes, such as fadeing and antialiasing of text. GIF is "lossless" only for images with 256 colors or less. For a rich, better, true color image, GIF might "lose" 99.9% of the colors. JPG is designed for photographs and some continuous tone images that contain lots of colors. It can achieve great compression ratios by maintaining really very high image quality.[8] In conclusion, implementation of simple steganography technique like Least significant bit(LSB) technique is possible in uncompressed image formats. For example, PNG format is a suitable choice for LSB technique.

2.5 Audio File Types

Audio formats are majorly classified into uncompressed audio format, lossless compressed audio format lossy compressed audio format. WAV, AU, AIFF,or raw header-less PCM are grouped under the uncompressed audio format. A lossless compressed format stores data in less space without losing any information. Lossy compression enables even greater reductions in file size by removing some of the audio information and simplifying the data. MP3 format is one of the popular lossy compression format. Include about frequency and sampling of audio formats.

2.5.1 Bit depth

The number of binary bits (ones and zeroes) used to record the sample level of the waveform. Thus 8-bit sampling uses an 8 digit binary number to record the level, giving 2^8, or 256 potential values. Bit depth describes the ratio between the quietest and loudest signals the system can record . A 8-bit recording is a low resolution audio, it was used by earliest digital audio systems.

12 bit recording had audible degradation or crunch. A 16 bit has 65,536 possible values (i.e.2^{16}), and hence a far higher dynamic range (96dB) than 12-bit recording. 16 bit audio is a kind of quality found in CD. 24-bit sampling has 16,777,216 discrete levels, giving 144dB dynamic range, which exceeds the tolerances of human hearing. It is the minimum recommended standard for high quality.

2.5.2 Sample rate

Sample rate is the number of sample obtained per second. It is expressed in hertz since it is a frequency. There are three range of sampling rate are in use. Low quality audio is produced by 44.1 kHz sampling rate with limited frequency range. 48 kHz is much professional audio equipment. It is the minimum recommended archival standard. Sampling rate more than 96 kHz offers high resolution and extended frequency response.
Bit depth and sample rate are the two determining factor of audio quality.

2.6 Cryptography

Cryptography is a technique that alter the data so that it is understandable only by sender, receiver or anyone who knows to decrypt it. Cryptography protects data from theft, alteration and also used for user authentication. Cryptography exist in different form from ancient age till now. Cryptography is employed everywhere in today's digital communication for its safety. With proper use of cryptography secure communication can be achieved.Three type of cryptography are secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions. Properties of secure communication includes

- Authentication
- Privacy/confidentiality
- Integrity
- Non-repudiation

2.6.1 Symmetric Encryption

Encryption is a methodology of encoding message or information with the help of a key. Such encrypted messages, cipher do not expose its secret content to third party. Only the intended receiver is able to decrypt the original message from the encrypted message using the legitimate key. Symmetric Encryption is an encryption algorithm where the same key is used for both encryption and decryption. The key must be kept secret, and is shared by the message sender and recipient.

2.6.2 Asymmetric Encryption

Asymmetric Encryption is also called Public-key cryptography. It is a cryptography in which a pair of keys is used to encrypt and decrypt a message so that it is handed to the recipient securely. What one key encrypts, only the other can decrypt. One of which is private key and the other key is public. Predominantly, public key is used for encryption of a message and private key is used decrypt it. Whereas, in the case of digital signature, private key is used to generate digital signature and public key is used to verify digital signature.

2.6.3 Data Encryption Standard (DES)

IBM designed DES and adopted by National Institute for Standards and Technology (NIST) for commercial and unclassified government applications. It is the commonly used symmetric cryptography. Triple-DES (3DES) and DESX are the two variant of DES. But there is a possibility for DES can be susceptible to brute-force attacks. So it has been deprecated, and replaced by the Advanced Encryption Standard (AES).

2.6.4 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is the official successor of DES. AES is the United States Government standard for symmetric encryption, defined in 2001. Rijndael is a block cipher used in AES. It is based on a substitution-permutation network (SPN). AES cipher key length can be 128, 192, or 256 bits. The AES replaced the DES with new and better feature such as block

encryption, data security for 20-30 years and it is easy to implement in overall. It is widely used to encrypt confidential text.

Chapter 3

Android

This chapter throws light on Android Operating System and its components. Also highlights the advantage of using Android. Elementary installation and configuration required for the development of the project is explained.

3.1 Android Operating System

Android is an operating system initially designed for mobile devices like phones and tablets, but now android wear are available for smart watches. Android TV and Android Auto are expected very soon. Android is based on Linux kernel, designed primarily for touch devices. Initially, Android was developed by Android Inc. and later acquired by Google in 2005 to make sure that a mobile operating system (OS) is created and maintained in an open platform. Android is licensed under the Apache License 2.0 Linux kernel patches. Android operating system is an open platform.

Google continues to pump time and resources into the Android project, which has already proved to be beneficial, though devices are available only since October 2008. There are now 900 million Android devices that have been activated. Android devices are activated daily.[15] 48 billion Android apps are already available in app store.[15] In few years, Android have made a great impact. Many analysts believe that the number of Android devices in use is greater than the

number of devices on all other mobile operating systems combined together. Android users trust Google because the app resides in the Google Play Store, which is controlled by Google. So, Android users have an impression that the apps are trust worthy.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world.[16] It's the largest installed base of any mobile platform and growing very fast. Every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content.[16] Android gives a world-class platform for creating apps and games for Android users everywhere, as well as an open marketplace for distributing to them instantly.[16]

3.2 Highlights of Android

Android is highly customizable in nature. From the home screen to the themes or wall papers, everything can be customized. The position of apps and the apps which are to be present on the screen can also be controlled. It is also possible to install a number of third party applications on the device. It has a large community of developers writing apps which extends the functionality of the devices. These apps are written primarily in Java programming language.

3.3 Android Platform Components

Android system is made of several layers on top of Linux 2.6 Kernel. Linux Kernel handles the security and network communication between the top layers and the underlying hardware. Linux Kernel allocates resources to processes as they need them and performs memory management. The goal of Linux is to ensure that the application works. Common functions such as graphic rendering, data storage and web browsing are bundled as libraries. Along with these, core Java libraries for running Android applications and the Dalvik runtime forms a layer. The Figure 3.1 shows the four layers of Android Operating System. The core libraries are following.

Open GL (graphics library): an application program interface (API) is used to produce 2D and 3D computer graphics.

WebKit - an open source web browser engine.

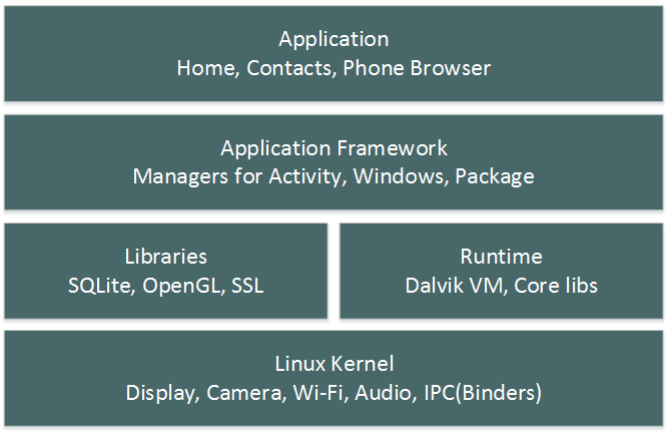

Figure 3.1 – Android Platform Component

SQLite - an open source relational database engine.

Media frameworks - A media library for playback of audio and video media.

Secure Sockets Layer (SSL)- provides Internet and web browser security.

Application Framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.

Application is the main source for users to interact with the device. Android is a multitasking platform, it can simultaneously run more than one application without one affecting the performance of the other. Android Open Source Project has Browser, Camera, Gallery, Music, Phone and many more as default application.

3.4 Android SDK

The Android Software Development Kit (Android SDK) provides API libraries and necessary tools to create, compile, test, debug and package Android applications. These tools are mostly command line based. Dalvik virtual machine is the software that runs the applications on Android devices. Dalvik VM, an open-source software, is designed to address security, performance, and reliability issues in mobile devices having limited power,limited processor speed and limited RAM. Underlying functionality of dalvik is done using Linux kernel. Dalvik Executable (.dex) is the file format executed by DVM. The java source files are converted to Java class files by the Java compiler. As described in the FIgure 3.2, tool called dx(Dex compiler) which converts Java

class files into a .dex (Dalvik Executable). All class files of the application are placed in this .dex file. During this conversion process redundant information in the class files are optimized in the.dex file. So .dex files are smaller in size than the corresponding class files. Android Asset Packaging Tool(aapt) pack .dex files into .apk(Android Package) file. APK file contains necessary files and resources to deploy the application to devices.

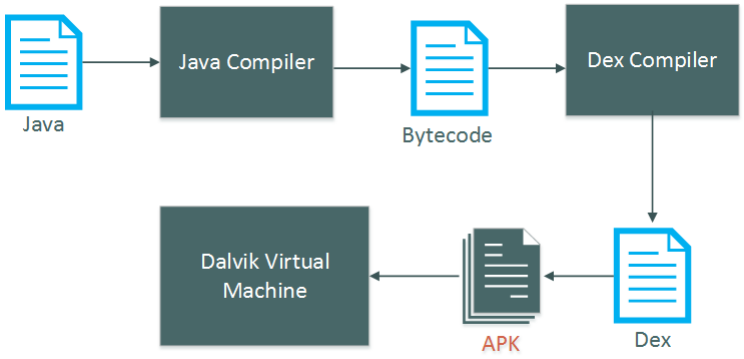

Figure 3.2 – Dalvik Virtual Machine

3.5 System Requirement

In order to develop Android applications one needs a computer with following configuration as specified by android.[26]

3.5.1 Operating Systems

- Windows XP (32 bit), Vista (32 or 64 bit), or Windows 7 or 8 (32 or 64 bit)
- Mac OS X (Intel) 10.5.8 or later (x86 only)
- Linux (tested on Ubuntu Linux, Lucid Lynx)

3.5.2 Development Tools

- JDK 6
- Apache Ant 1.8 or later
- SDK
- Eclipse for better development experience.

3.6 Installation and Configuration

3.6.1 Eclipse

Eclipse is an Integrated Development Environment (IDE) to work with several programming languages. All versions are available for download at free of cost as it is a open source. Android applications are predominantly written in the Java programming language. So Eclipse IDE for Java Developers (Eclipse IDE for JAVA EE Developers works as well) is suitable for android development. Downloaded zip file from eclipse web page contains the eclipse application, which is used to run the Eclipse.

3.6.2 Configuring Eclipse for ADT

ADT is designed to provide a powerful, integrated environment in which a developer can build Android applications. Comfortable to create, develop, debug, compile, and run android applications in Eclipse along with ADT. Most of the files in android development are XML files. ADT provides easy operation with android's XML file by improving the developer experience though structured user interface. It is necessary to make installed Eclipse to configure to work with Android ADT. Android SDK location should be set for the Eclipse. This can be set in the preference window, against the android tab.

3.6.3 Android Virtual Device Manager

AVD Manager is a tools that comes with Android SDK. The AVD Manager provides a graphical user interface used to create and manage Android Virtual Devices (AVDs), which are required by the Android Emulator. Created ADV simulates an actual device. So it is possible to simulate various screen size and test the app before testing on actual devices.

3.7 Android Activity Lifecycle

Activity is an application component which is provided with a window to user. User interact with this window and activity responsible for this window will take care of the user actions here.

Activities in android goes throw several state in its lifecyle. Since android is made for mobile devices, main objective of this life cycle is for memory and resource management. Apart from this android also recreates activities on configuration changes. These states of the application have to be maintained by the developer. When Android components are started a new process with a unique ID mentioned in the AndroidManifest.xml file. The Android system creates a default object if it is not specified in the AndroidManifest.xml. Ultimately, Android app work based on the process status and activity status. The sequence of the life cycle methods are depicted in the Figure 3.3.

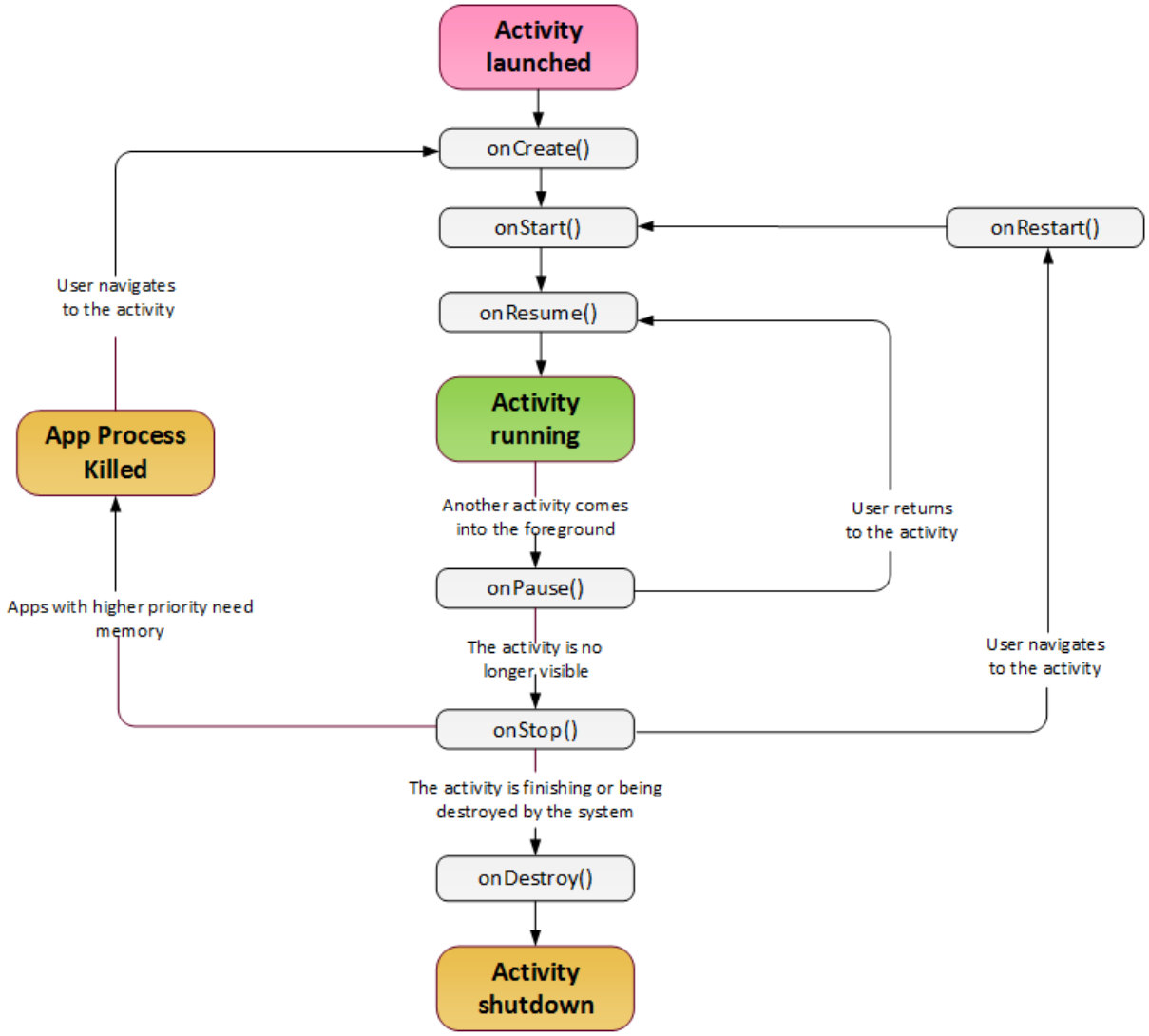

Figure 3.3 – Android Activity Lifecycle

3.7.1 Android Process Status

Foreground

Depicts user is interacting with the activity in an application.

Visible

Depicts user is not interacting with the activity, even though the activity is still visible.

Service

An application which is running but not in foreground or visible.

Background

Depicts the activities that are stopped but it is kept in the Android's least recent used (LRU) list. If required android will terminate the least used activity one by one.

Empty

No active components found in the application.

3.7.2 Activity State

Running

Activity is said to be running if it is visible and interacts with the user.

Paused

Instance of the activity is running but might be killed by the system. Activity is paused until it is visible and not killed but partially obscured.

Stopped

Activity is not visible and the instance is running but it might be killed any time.

Killed

Activity is terminated by the system using finish() method.

3.7.3 Activity Lifecycle Methods

Activity lifecyle methods are provided to implement tasks at different stages of the activity. Most of the initiation task are implemented in onCreate method, because it is called at the start of the activity.

onCreate

This method is called when the activity is created. Used for initialization of the activity and user interface creation.

onResume

This method is called when the activity gets visible again and the user starts interacting with the activity again.It is used to initialize fields, register listeners, bind to services and many more.

onPause

This method is called when another activity gets into the foreground. It is used to release resources, save application data, unregister listeners, intent receivers, unbind from services or remove system service listeners.

onStop

This method is called once the activity is no longer visible. It used to write write information to a database and many more.

onDestroy

This method is called before the activity is destroyed. Since t is the final call to the activity, all resource and process are released.

onRestart

This method is called after the activity has been stopped, just prior to it being started again.

onLowMemory

When the application need to the cleans up memory the Android system will request for it by calling onLowMemory method.

onTerminate

This method is used to terminate a process but it is used only for testing purpose.

onConfigurationChanged

This method is called whenever the configuration changes occurs.

3.8 Android App Components

1. **Activities**

   Every screen is represented by an Android Activity. It take response from user interface and its interaction with the device screen. If application has more than one activity, one which is mention in the manifest XML is initiated during launch of the app.

2. **Services**

   Background operations are ran as a service.For example, audio can be played as background activity while different application is active. Data can be downloaded at background through network.

3. **Broadcast receivers**

   Broadcasts are initiated from an application to notify about an action to other applications. So broadcast receivers can perform any action according to the message. For example, when there is some data is to be downloaded to the device, broadcast receivers get to know about this and start their activity.

4. **Content providers**

   On request by an application, data are supplied by content provider component. ContentResolver class handle those request.

5. **Fragments**

   A fragment is a kind of sub-activity. Part of user interface in an activity is represented by a fragment.

6. **Intent**

   Acts as a glue between activities. Mostly used while launching activities. Intent is passive data structure containing an abstract description of action to be performed

7. **Layouts**

   Structure of the user interface is defined by layout. Layouts are either declared in XML or instantiated at runtime.

8. **Views**

   View is the basic building block of user interface component. Button, text field etc. are the UI components used as view.

9. **Resources**

   Apart from source code android app is composed of images, audio files, and anything relating to the visual presentation of the app. And XML files defining animations, menus, styles, colors, and layout of the activity user interfaces. Resources make it is easy to update various characteristics of the app without modifying the code.

10. **Manifest**

    Before an android app start it must know the components that are existing. The manifest file provide the list of components used. User permission, minimum API level, hardware and software features used or required by the app, such as a camera, bluetooth services, or a multitouch screen are declared in this file.

This thesis is basically an android app development. So all of the above mentioned components of android and its working was very useful to implement the required functionality.

Chapter 4

Design and Requirement Analysis

This chapter describes the design based on the requirements of final product. According to the scope of the project, gathered analysis are put together as functional and non functional requirements. Construction of architecture, algorithm and flow of the applications are explained in detail. Necessary inputs for the development phase such as use case, package model and class diagram are designed.

4.1 Scope

Ideally, implementing few specific features of steganography in android is the objective. The developed android application should be able to hide secrete text message in a cover image or audio file. And it should be capable of hiding quick voice message in a cover image file. Encoding audio data in a cover audio file is out of scope of the project. In same way hiding an image in a cover image is not required. While hiding quick voice message in a cover image the maximum voice message duration is 10 sec. Encoding and decoding of message must be done in real time, i.e. in few moments. According to user action, app must notify appropriate errors and warnings.

4.2 Functional Requirement

Application is a made of group of modules. Each module works on certain task. Requirements based on each task are called functional Requirement. By analyzing the scope of the project and having the mandatory requirements of the target audience in mind, the functional requirements are scripted. The idea is to create an application which is user friendly and perform task in considerably short amount of time. There are mainly six functions. These functions are hiding the text in image, hiding the text in audio, hiding the audio in image and retrieving back the hidden message from those image and audio files. Simultaneously it is significant to provide secure feel for users on their message. So encryption turned out to be a good option along with steganography. Thus six screens are designed to accommodate each module in each screen. Additionally, share, home and help screens are included for supporting features. These collective information forms the main idea and basic requirements. Detailed requirements on each screens and input validations are discussed further.

4.3 Non Functional Requirement

Behavior of a system is also a part of the requirement. They are termed as non functional requirements. Even some constrains and restriction on system are also considered as non functional requirement. These requirement have to be selected such that it is bounded and relevant to the system. Some behaviors considered as non functional requirements are capacity, efficiency, effectiveness, extensibility, maintainability, privacy, portability, quality, reliability, response time, robustness, security and stability.

4.3.1 Use Case

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.[25]. All system activities are recorded in use cases. Use case organize functional requirements into system and

actor interaction. Designing use case helps is managing complexity of the application. The whole requirement is broke down in to several small scenarios. As shown in Figure 4.1, the user has direct access to home screen. From home screen user can travel to any other screens. Share feature is available only in encoding screens. The use case contains the goal of the event flow,

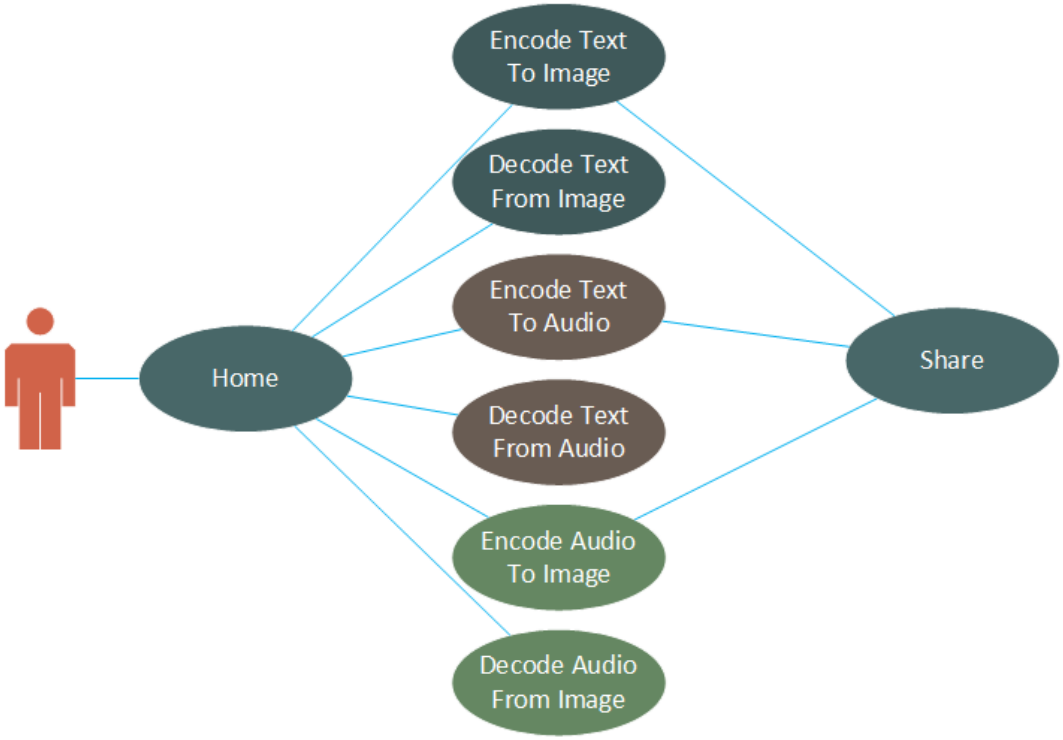

Figure 4.1 – Use case diagram

precondition and post condition of the event. Post condition can be successful or failed post condition. Scenario action describes the path of the event from where it was triggered till the goal. One of the use case of protectMSG is given below. Other use cases are found in appendix.

Use Case 1	Home
Goal in Context	Home screen displays overview of the functionality of protectMSG application. And user has the facility to navigate other pages.
Preconditions	
Successful Post Condition	Overview of the application is present along with a way to access different functionality of the application.
Failed Post Conditions	If user selects medium and message as audio, then the application will inform user that encode or decode audio in audio is not possible. Because it is not included in the scope of the application.
Scenario Action	Step 1: User selects message type and medium in any order. Step 2: On click of "Go" button for encode or decode will leads to the appropriate screen based on the input.

4.4 Architecture

4.4.1 Application Flow

The flow diagram in the Figure 4.2 explains the systematic operation of the application. Since the application have more than one task, no task is depended on other task. Any task can be perform on request of the user. So the user input determine the task to be executed. Fundamental input parameters for any module or task in the application are the medium and the message type. An Image file or an audio file can act as a medium to cover the message. Message can be either text or audio. After the valid selection of medium and message type, operation that needs to be performed is decided. The operation can be either encode or decode. Before encoding or decoding, appropriate error message is thrown if the inputs are invalid. Now either it is encoded or decoded on the selected input. Finally, based on the input, output is drawn. The process used to encode and decode are discussed later in the chapter.

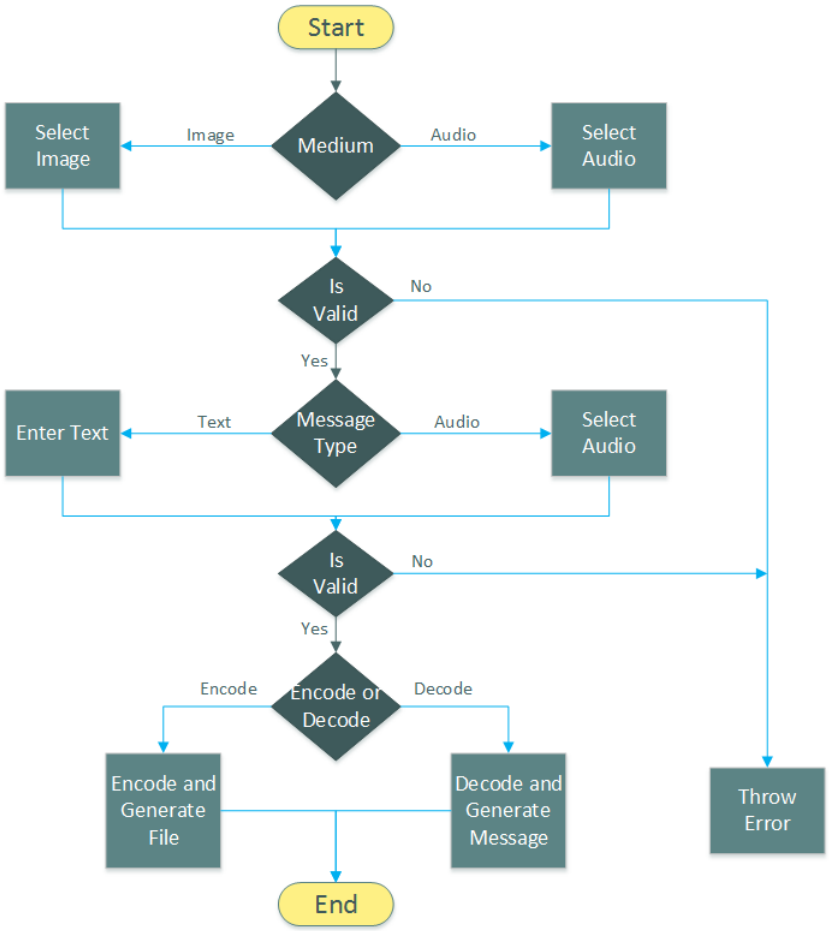

Figure 4.2 – Flow Chart

4.4.2 Encryption Used

In this application at certain stage message is encrypted upon user request. AES encryption of 128 bits cipher key size is used to safe guard the user message. The message is encrypted using the secrete key generated from the password provided by the user. The key length should be 128 bits. so the password undergoes a padding if its size is not exactly 128 bits. If the message is text then the encrypted message is encoded under Base64 encoding scheme. Base64 is a binary to text encoding scheme which represent binary data in ASCII string.

4.4.3 LSB Technique

Least Significant Bit (LSB) is a simple technique to implement steganography. It is mostly used for image and audio steganography. LSB method generally achieve both high capacity and high imperceptibility. [1] High capacity refers to the large size of data can be embedded when compared to other techniques. Imperceptibility refers to natural human ability to detect the difference between the original object and embedded object. The technique works by replacing some of the information in a given pixel with the information from the message data. When the cover object is transformed to bytes, each byte is represented by bits. Replacing the least significant bits seems to be a better option as the value they carry is insignificant when compared to other bits. While the message and cover object is represented in the bit plane, every least significant bit of the cover object's byte is replaced by the message bits. This concept is represented in the Figure 4.3.

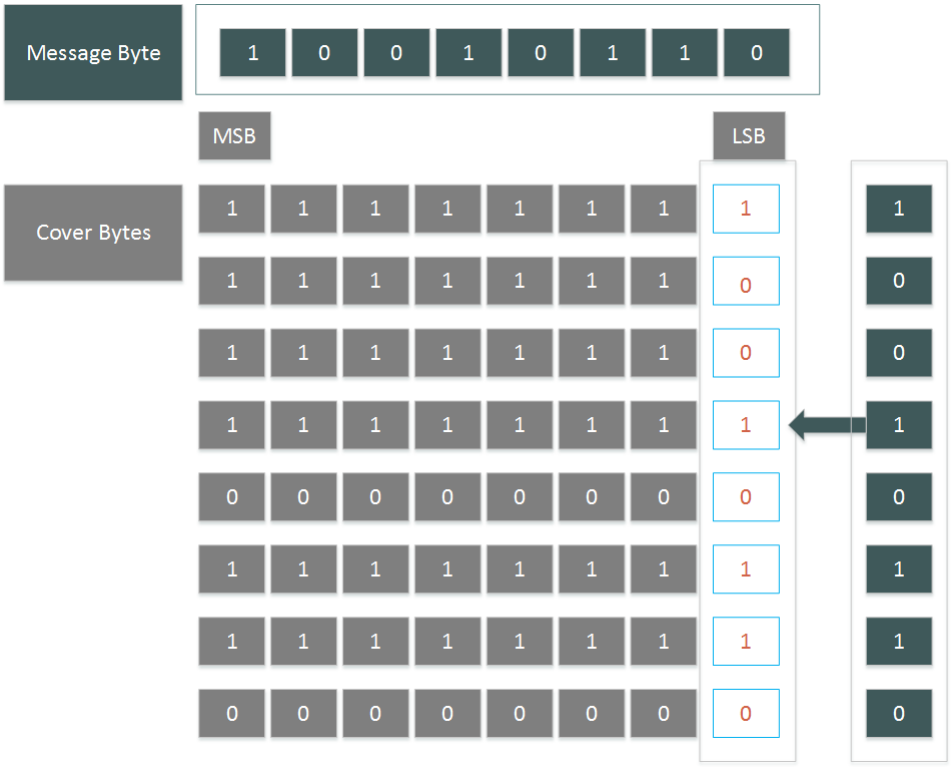

Figure 4.3 – LSB Technique

By doing this, there is some loss of information in the cover object. In case of of image and audio cover file, due to natural weakness of Human Audio/Visual System the loss of data in cover file

is not noticeable.[7]

4.4.4 Encode Data Packet

Just embedding the message into a cover file does not provide a completeness to the application. So the new concept is to create a Encode Data Packet and embed it in the cover file instead of only message. It is useful for the application to work better than normal. Encode Data Packet is a packet that encapsulate the message with some header information which is useful while decoding the message. This packet is designed to achieve the attributes of secure communication. The packet contains the message and a header. Header assure the encoded file is recognized only by the desired reader. Hidden message is retrieved by precise interpretation of this header, which is only known to protectMSG application. As shown in the Figure 4.4, Message Length

Figure 4.4 – Encode Data Packet

and Flag combines to from the header. And header precedes the message. Header is 8 bytes in length. It is divided into two section, namely, Message Length and Flag. Message Length section is used to convey the length of the message that needs to be hidden. It is impossible to extract the precise message without knowing the length of the message hidden in the file. Size of the Message Length frame is 4 bytes. So actual length of the message can vary from 0 to 2^{32} in bytes. This is a huge number. 32 bits are allocated because it will be useful if the message capacity has to be increased in future enhancements.

Flag

There are several purpose and use in creating header for each encoding. Flag is a part of header in the encode data packet. While decoding, flag provides the information that where the file was

actually encoded by protectMSG. Apart from this, it also provides whether the encoded file was protected by password or not. Two type of flag used are:

Flag 0 - 00 00 00 00 - Represent the file is encoded without password
Flag 1 - FF FF FF FF – Represents the file is encode with password

These flags are used to take several decisions while decoding. While extracting the message, application throws appropriate error if the expected flag is not found. If user does not provide a password but Flag 1 exist in the file, then the application throws error, asking for a password. If any of these flags are not found in the files, then the file is consider to be not encoded by protectMSG app.

4.4.5 Algorithm

In the process of embedding the message into a cover file involves several steps. They are explained below.

Step 1 - Convert the medium to bytes
Medium can be either image or audio file. This file has to converted to byte array for manipulation.
Step 2 - Encrypt Message
If the user has entered password then message is encrypted with the given password.
Step 3 - Convert the message to bytes
Message can be either text or audio file. So they are converted to bytes array for manipulation.
Step 4 - Create Encode Data Packet
Encode Data Packet is formulated using the flag, message byte array and its length. Flag depends on user input.
Step 5 - Encode using LSB technique.
Then encode data packet is embedded in to the byte array of the medium using LSB technique.
Step 6 - Generate encoded file.
Finally the output byte array from the LSB technique, which contains the hidden message, is

convert to either image or audio file based on the input medium type.

The steps involved in extracting the hidden message from the cover file is given below.

Step 1 - Convert the cover file into byte

The cover file containing the hidden message is first converted into byte array.

Step 2 - Read encode data packet header

The converted byte array under goes the reverse LSB technique to extract header and message of the encode data packet.

Step 3 - Extract message

From the encode data packet header and message is separated and validated.

4.4.6 Package Model

In development of the project, to gain molecularity, all the classes are divided into packages. Each package contains several class that performs task in specific area. Model view controller pattern is followed in this project development. The StegoVO is a value object class. It act as a model. Classes in the operation package are the controller which does all core functionality and control the whole system. The fragments classes continuously attach with the view object of the project. They mainly update the user interface when ever required.These relations are represented in the Figure 4.5.

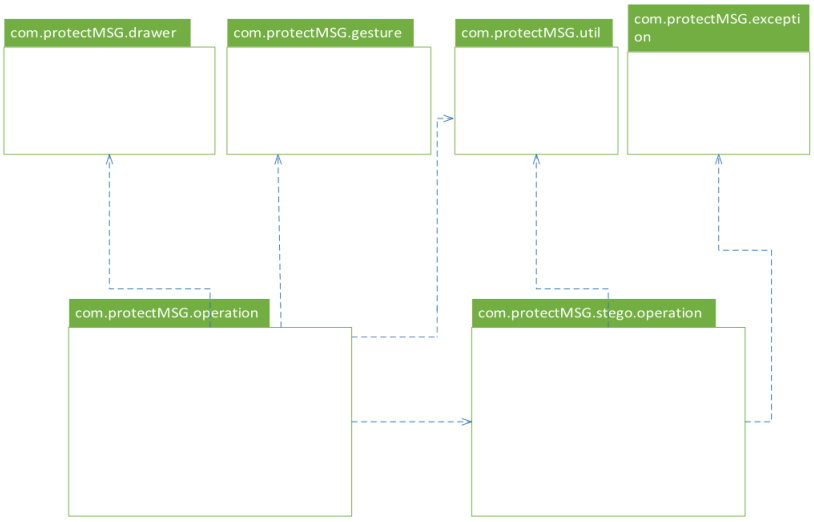

Figure 4.5 – Package Diagram

Drawer

This package is responsible for implementation of the navigation of the application. Android's drawable menu is implemented. It appears at top left corner of the app. Classes in this package are listed below.

Package Name : com.protectMSG.drawer

- NavDrawerItem.java
- NavDrawerListAdapter.java

Exception

Custom exception class is used to catch and throw specific criteria that are treated as error in this app.

Package Name : com.protectMSG.exception

- StegoException.java

Gesture

Gesture package incorporate gesture in the application. Activity class should implement the SwipeInterface to avail this feature. Gesture package is helpful because it simplifies the usage of gesture at any required place very easily.

Package Name : com.protectMSG.gesture

- ActivitySwipeDetector.java
- SwipeInterface.java

Operation

Operation package primarily does the all variety of encoding, decoding and its validations.

Package Name : com.protectMSG.operation

- DecodeAudioFromImage.java
- DecodeTextFromAudio.java
- DecodeTextFromImage.java
- EncodeAudioToImage.java
- EncodeTextToAudio.java
- EncodeTextToImage.java
- Help.java
- HomeFragment.java
- MainActivity.java
- TransparentProgressDialog.java
- Validate.java

Stego Operation

Classes in this package combinedly responsible for the core functionality of the app. LSB technique and encryption are their main duties.
Package Name : com.protectMSG.stego.operation

- StegoEncryption.java
- StegoOperation.java
- StegoVO.java

Util

Util package has classes that majorly does supporting functions for other classes. So they all are helper class.
Package Name : com.protectMSG.util

- AudioRecording.java
- Constant.java
- Customfontloader.java
- Utility.java

4.4.7 User Interface Design

Theme and Icons

When it comes to mobile apps, unlike desktop software, they are used for shot span of time to do quick tasks. User actively use the application for short period of time. An app should be attractive and easy to use it. Thus user interface design plays significant role in the the development of the application. Theme, color and icons used in the application are suitable for all kind of users. Grey background is chosen as it is not flashy or dull. Text and buttons are contrast to the background so that they are clearly visible. And the next is to make the user experience better by fluent navigation in the application.

Navigation and Controls

The protectMSG app has eight screens in total. At the start of the application home screen appears. From home screen, other screens can be visited except Help screen. Apart form home screen, all other functionality of the application can be visited including help screen by accessing the navigation drawer. The navigation drawer is a panel that displays the app's main navigation options on the left edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen or, while at the top level of the app, the user touches the app icon in the action bar. [9] Advantage of this menu is the ability to access any top level content from anywhere in the app.

For example, encode text to image screen can be accessed either from home page or by using the drawer menu at the top left corner of the application. Home page has an option to select message type and the medium to carry the message. The phrase "Encoding Text To Image" denotes the

message is text and its medium is image file. After selecting the message type and medium, “Encode Text To Image” screen can be visited by clicking “Go” button for “Encode” .

Below is the Table 4.1 representing the available screens, its function and access point in the application.

Table 4.1 – Screen Functions

Screen	Functionality	Accessible From
Home	Gives brief description about the application and task that can be performed in this application	Appears at the start of the application. Also accessible using drawer menu.
Encode text to image	Hide text message in an image.	Home and drawer menu.
Decode text from image	Reveals text message from an image.	Home and drawer menu.
Encode text to audio	Hide text message in an audio.	Home and drawer menu.
Decode text from audio	Reveals text message from an audio.	Home and drawer menu.
Encode audio to image	Hide audio message in an image.	Home and drawer menu.
Decode audio from image	Reveals audio message from an image.	Home and drawer menu.
Help	Display detailed specifications of the application that are useful to the user.	Drawer menu.

Feedback and Alerts

Relevant information, warnings and error messages have to be intimated to the user at necessary moments. Android provides a small pop up message called toast messages. It just fills the amount of space required for the message while the current activity remains visible and interactive. It appears only for short period of time and then fades out. This toast message is customized to match with the theme of the application by modifying the background color and including the application icon in the message. Text color in the toast message is orange for warnings, red for errors and green for information as shown in the below Figure 4.6, Figure 4.7 and Figure 4.8 respectively.

Figure 4.6 – Warning Message

Figure 4.7 – Info Message

Figure 4.8 – Error Message

Prototype and Versions

Prototype is the model of the final output. Generally, prototype for GUI of the application is created at first. Then during the process of development several versions of the application are created step by step. So requirement is gathered in terms of user interface to create a basic prototype of the UI design. Many versions of the user interface are created in the process of development. Initial version has the basic required fields and buttons without any special effects. Every other version is reviewed prior designing the next version for but better user interaction, improvised theme and rich UI features. Final version is powered with gestures, rich appearance and colorful buttons. The figure 4.9 shows the initial version of the app.

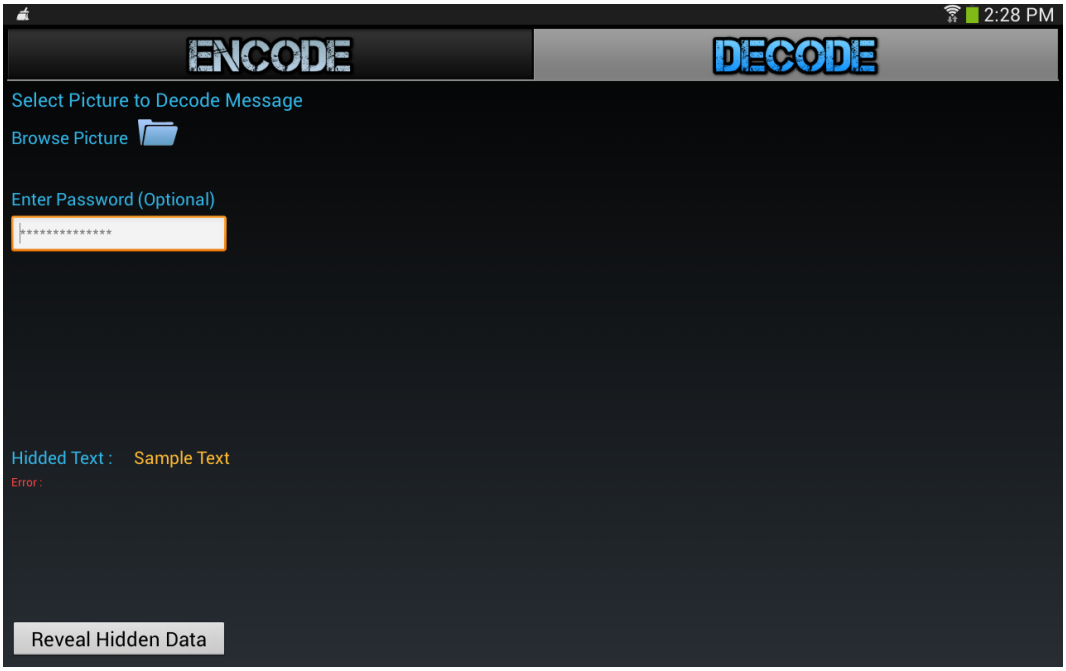

Figure 4.9 – Initial Version Screen

According to the requirement, the functionality of the application is designed for better implementation. As well, the GUI of the application is constructed for high flexibility.

Chapter 5

Implementation

This chapter gives elaborate details of implementation of this thesis. Details about the helpful tools used during the development and their importance in implementation. All function of the application and the way of execution to obtain the result are written here.

5.1 Project Specification

Android manifest XML is where all the components and setting of the android application is mentioned. It also defines the structure and metadata of android application[27]. Permission for the app to access the SD card, camera and audio recording are mentioned in manifest file. Other specifications of the project are given below.

Application name: protectMSG
Minimum Required SDK: API 14
Target SDK: API 19
Target device: Samsung Tab3 GT-P5210
Screen size: 10 inch
Screen mode: Landscape
Online/Offline work: App works perfectly in offline mode. But, internet is needed to share files.

The protectMSG app is compatible with following Android versions.
Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)
Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
Android 4.1 Jelly Bean (API level 16)
Android 4.2 Jelly Bean (API level 17)
Android 4.3 Jelly Bean (API level 18)
Android 4.4 KitKat (API level 19)

5.2 Tools Used

5.2.1 Android Command Line Tools

Android provide several tools that can be used from command line. They are bundled in Android SDK package. tools are classified into SDK tools and platform tools. SDK tools are platform independent while platform tools are customized to support the features of the latest Android platform. SDK tools are frequently used to manage the android project are.

android

Android tool is used to create three types of android projects. A project with all files and resources that are needed to build a project into an .apk file for installation is one among them. Other type is a library project which allows it to be shared with other projects that depend on it. It cannot be installed on devices. Last type is a test projects extend JUnit test functionality to include Android specific functionality.

mksdcard

Quickly creates a FAT32 disk image that can load in the emulator, to simulate the presence of an SD card in the device. The created SD card should be associated to an AVD while creating the AVD in the AVD Manager. Desired memory for the SD card should be specified while creating it. Same SD card image can be shared across multiple emulators.

Dalvik Debug Monitor Server (DDMS)

Dalvik Debug Monitor Server is a debugging tool provided by android. It is a command line tool, but it is easily accessible while using Eclipse with ADT. Eclipse provide a perspective that display all the functions of DDMS. DDMS shows the list of AVDs and devices connected to ADB. It has LogCat, a system to view log messages in real time. A file manager is available to view, push and pull file to the SD card image. Also displays heap usage, slowly running thread and memory allocation of objects.

Android Debug Bridge (ADB)

ADB is used to install Android application file(.apk) file, manage state of an emulator instant, to push and pull files from SD Card image.

5.2.2 BinaCompa

Comparing bytes of data becomes unavoidable necessary in this application development. There exist a tool called BinaCompa. It is used to compare two selected file byte data. For example, it is possible to compare each and every bytes of the two selected files in order.

5.2.3 Checkstyle

Checkstyle is a helping tool for programmers to write java code adhering to coding standards. Eclipse plugin for Checkstyle provide integrated development with eclipse. Naming convention for java is followed.

5.3 Emulator

Android provides mobile device emulator along with Android SDK. Emulator is a virtual mobile device that runs on computer. Before deploying any application directly into the physical device, android provide an option to develop and test app in emulator. Android Virtual Device (AVD) is an emulator configuration tool. Hardware and software parameters to be emulated are defined using AVD. Some important parameters are

AVD Name - Specifies the name of the created virtual device
Device - Devices are available at different screen size. This field determine the screen size of the emulator.
Target - Android API level is mentioned here. It is nothing but the android version.
CPU/ABI - ARM and Intel Atom are the CPU available to choose
Memory Option - RAM size have to specified in order to create a virtual device.
SD Card - SD card image should be created and associated with the virtual device.

5.4 protectMSG

5.4.1 Program Inputs

All core functionality in this application needs some sort of input from the user. Message and the cover file for embedding are accepted as input from the user. User has wide choice of input types when it comes to image or audio. Because there are lot of image and audio formats available now. But only certain image and audio formats are accepted as input to the system. So all inputs undergoes a validity check. Thus these inputs are verified for its characteristics whether they are eligible for application process. The Table 5.1 denotes the valid input criteria. The protectMSG application has more than one screen. Not all screen accept all input. Each screen screen has different type inputs based on its functionality. So the Table 5.1 also depict the association between the input and the screen.

Table 5.1 – Program Inputs

Input	Valid Criteria	Acting Screens
Encode Text	Not null and length between 1 to 50 characters. It can be numeric, alphabets and special characters.	Encode text to image Decode text from image Encode text to audio Decode text from audio
Password	Null or length between 3 to 15 character. It can be numeric, alphabets and special characters.	Encode text to image Encode text to audio Encode audio to image Decode text from image Decode text from audio Decode audio from image
Image	JPG, JPEG, PNG file formats are accepted.	Encode text to image Decode text from image Encode audio to image Decode audio from image
Audio	WAV file formats is accepted.	Encode text to audio Decode text from audio

5.4.2 Encode text to image

Encode text to image is one of the functionality in protectMSG app. Here, text message is hidden in a cover image.

Encode text to image screen has two text field. First text field is for entering the message that needs to be hidden in the image file. Other text field is for the user to enter the password, which is an optional field. It can be seen in the Figure 5.1. If an image is encoded with a password, then it can be decoded only with the same password.

Since Image is the medium here, a valid image has to be selected using the browse image button. On click event of the plus icon initiate the event for image selection from gallery. There are several application available to view and select images. By default, android device manufacturer provide a gallery app installed when the device is sold. Optionally, many applications for viewing image are available in play store. On clicking the browse image button, all the installed gallery applications pops for selection.

Text data is hidden in the image by clicking the encode data button but the input text, password and the cover image undergoes several validity check before encoding. Password is an optional input. If the password is not provided by the user the operation take place as shown in the Figure

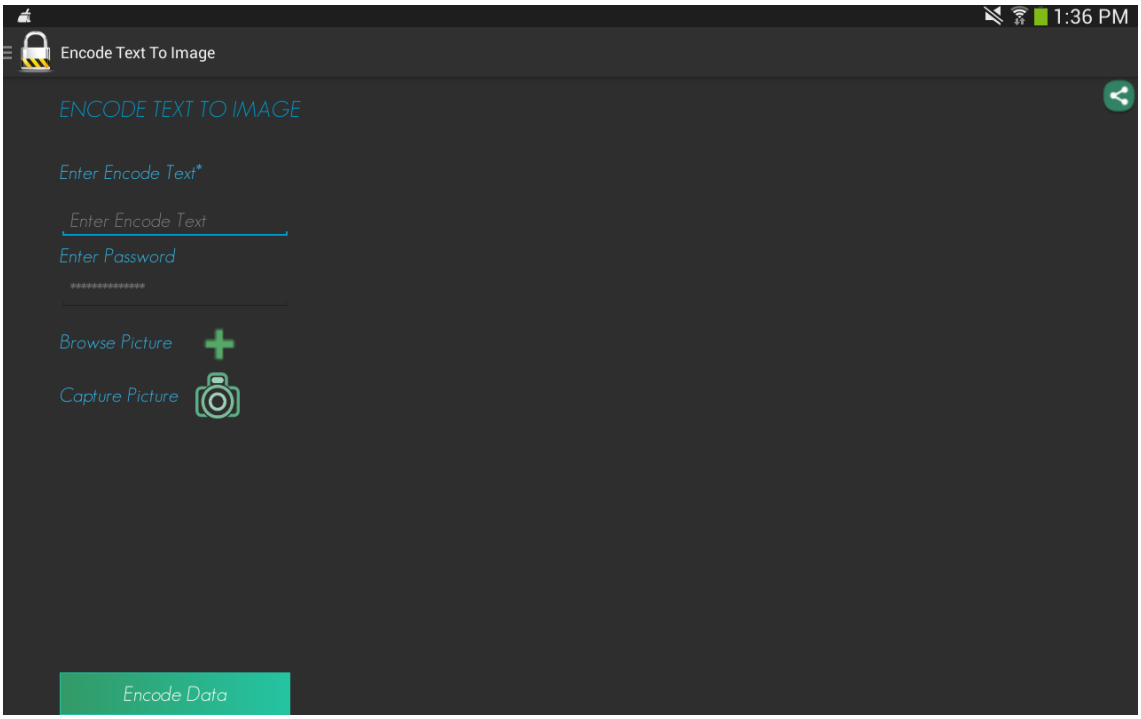

Figure 5.1 – Encode Text To Image

5.2. Directly the message is sent for encoding in to the cover image file.

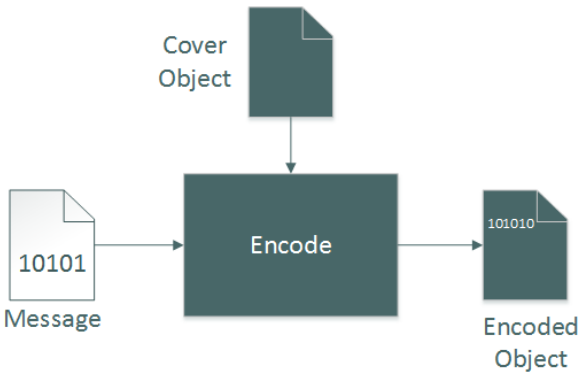

Figure 5.2 – Encode block diagram without encryption

If the user had given a password, an encryption key is generated from the password provided by the user. Using the generated encryption key the text message is encrypted and sent for encoding on cover image as shown in the Figure 5.3.

Encoding

Once the inputs clears verification and arrive the encoding phase, the image is read as a bitmap and it is converted to mutable format. The bitmap is converted to byte array. The text is

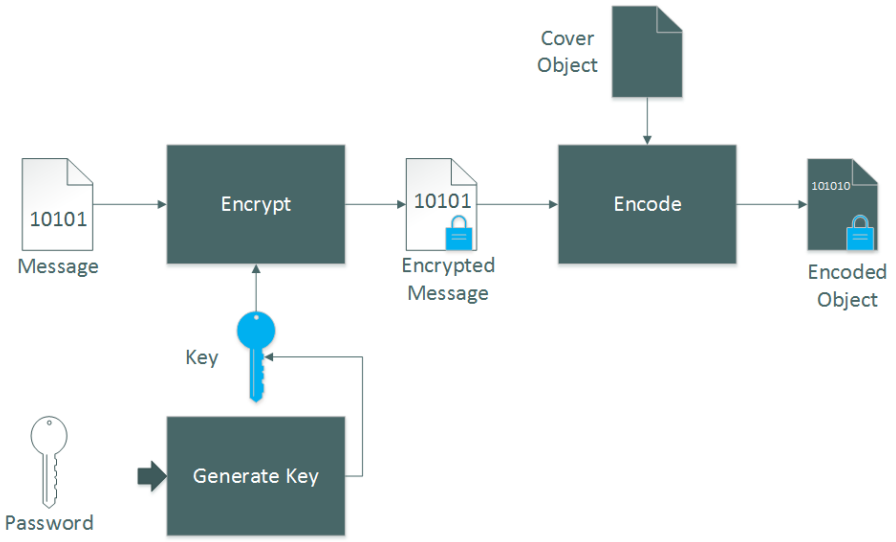

Figure 5.3 – Encode block diagram with encryption

converted to byte array and embedded with image bytes using LSB encoding. Byte array returned from LSB encoding has text hidden in it. This byte array again converted to bitmap and formulated into new image. This generated image is saved in SD card of the device in the location "/sdcard/protectMSG/Image/".

5.4.3 Decode text from image

The encode image is sent to the intended recipient. The recipient will be able to extract the hidden message from the image. But only images that are encoded by protectMSG should be used. This screen has only one text field. This text field is for entering password, which is optional. If an image is encoded with password, then the same password has to be used to unlock it.

The below Figure 5.4 shows the process of decoding if the password is not provided. And the Figure 5.5 shows the process when password is used.

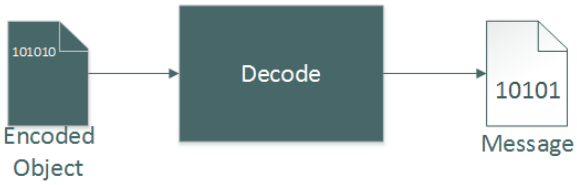

Figure 5.4 – Decode block diagram without decryption

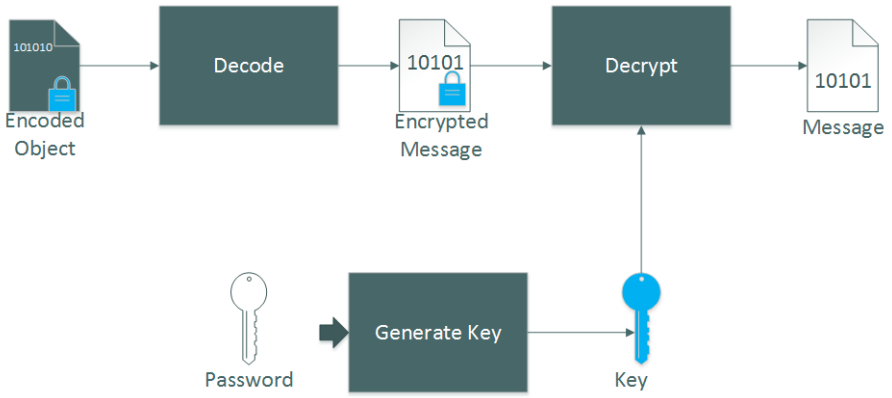

Figure 5.5 – Decode block diagram with decryption

5.4.4 Decode

Click event of "Reveal Hidden Text" button start the decode process with selected image. Since the the image was encoded by LSB technique, the text hidden in it is extracted in same way. If the message was encrypted the it is decrypted using the key generated from the password provided by the user. If the password is wrong, error is thrown asking for correct password.

5.4.5 Encode text to audio

Encode text to audio is the functionality which hides text in an audio file by converting the text data into bytes and encode it in the audio voice signal.This screen has two text field. First text field is for entering text message that needs to be hidden in the audio file. Other field is for the user to enter password, which is optional. If an audio is encoded with password, then the encoded audio can be decoded using the same password only.

Since audio is the medium here a valid audio file have to be selected using the browse audio feature. On click event of the plus icon initiate the event for audio selection from gallery. There are several application available to play and select audio. By default, android device manufacturer provide an audio player app installed when the device is sold. Optionally, many applications for accessing audio files are available in play store. On clicking the browse button, all installed audio player applications pops up for selection.

Recording user voice is also possible. Instead of selecting an already existing audio file, there

exist an option for recording users's voice. Using the record audio button, one can start recording their voice by clicking once. User can continue to record their voice and by clicking the button once again will stop recording. Then the recorded audio is automatically saved in the folder "/sdcard/RecordedAudio/". Name of the recorded audio file is unique every time as it take the current timestamps.

On completion of successful recording new section appears below, containing the recorded audio file with an option to play. When the green play button is clicked it turns to blue pause button and start playing the recently recorded audio. The audio stops playing automatically once it has finished playing. Pause button is used to pause the audio that is currently played. Instant message appears at the bottom when the audio starts playing, paused or finished playing.

On valid input selection both text and audio are converted to byte array and encoded using LSB technique. The resultant byte array is saved as WAV file in the SD card location "/sdcard/protectMSG/Audio/".

5.4.6 Decode text from audio

Hidden text in the audio is decoded and revealed by the exclusive decode function. Audio encoded by protectMSG have to be used for decoding, otherwise appropriate error is thrown. And if the encode audio is secured by password, then then same password have to be used to unlock it. This screen has one text field. This text field is for entering password, which is optional. If an audio is encoded with password, then the encoded audio can only be decoded with the same password.

5.4.7 Encode audio into image

Audio data can be hidden in the image pixels by converting the audio data into bytes and encode it in the image pixels. This functionality is intended to hide quick voice message in an image file. So the voice message is restricted to 10 seconds of recording. This screen has one password field and two section for input selection. First field is for entering password, which is optional. Next section is for recording the audio that needs to be hidden in image file. Next section is for selecting the cover image. If an image is encoded with password, then the encoded image can

only be decoded with the same password.

5.4.8 Decode audio from image

Hidden audio in the image can be decoded and revealed by the exclusive decode function. Image encoded with audio by protectMSG have to be used here. And if the encoded image is secured by password, then then same password has to be used to unlock it.This screen has one text field. First text field is for entering password, which is optional. If an image is encoded with password, then the encoded image can only be decoded with the same password.

5.4.9 Supporting Features

Home Screen

Home screen provides the overview of the application. This screen list the features of the protectMSG application in brief and the ways to achieve it. In home screen user can select the message type and medium type. There exist two option buttons for selecting the message type. One is “Text” button to emphasis that the user message is a text. Other button is “Audio”, which implies audio message is desired to be hidden in a cover file. Since they are option button, only one button stays selected at a time. By selecting one of these button, the other button will go un-selected. Similar functionality applies for the medium type buttons. The medium can be either “Image” or “Audio”. An exception exist here. Audio message cannot be hidden in another audio file as it is not in the scope of this thesis.

Share

Share button at the top right corner of the screen is used to share the recently encoded file. So this feature is available only in encoding screens. It is not available in the screens featuring decoding. On click of the share button, it rotates in the clockwise direction and initiates the intent to share files. Applications installed in the device, which has the ability to share the files are listed in the

pop up screen. User has an option to select his/her own desired application to share the file. For example, if applications like Gmail, or Facebook is installed, they would show up in the listing. So user has an option to share the file either through mail or any messaging apps. This gives flexibility and choice for the user to share file through their own desired application. Screen shot of the home screen is show in the Figure 5.7.

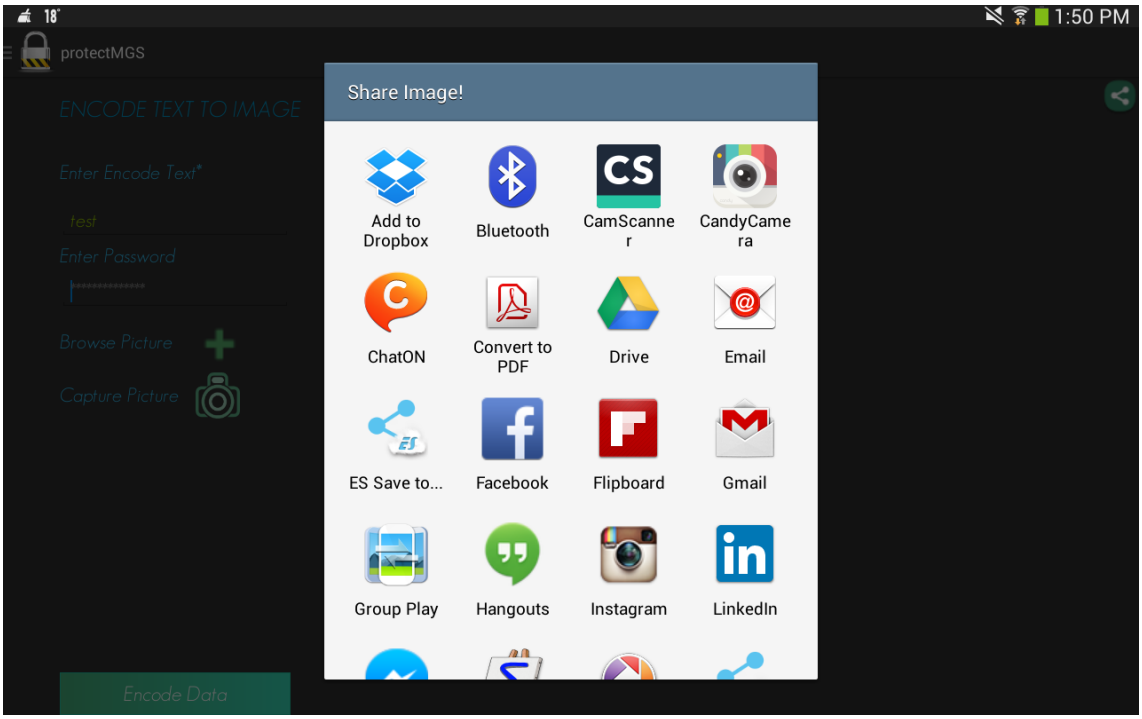

Figure 5.6 – Share Screen

Browse Picture

In most of the screens, user have to input an image to the system. So, the browse picture section in the screen will allows the user to select images from the device internal storage or SD card. The protectMSG app will not do this. But it will transfer this job to any other app that is installed in the device. The protectMSG will communicated with other apps by initiating intent. So specific intent for accessing picture is initiated by clicking the plus icon in this section. Now user will be shown with wide choice of gallery application installed in the device to select the image. The selected image is verified for its type. Only JPEG, JPG and PNG are accepted. Below Figure 5.8 shows the Browse Picture section.

Figure 5.7 – Browse Picture

Capture Picture

User has an option to capture image from the device camera. When the camera button is clicked, camera activity is initiated to take picture. The captured picture is saved in the location "/sdcard/picture/protectMSG/Image/". Below Figure 5.9 shows the capture picture section.

Figure 5.8 – Capture Picture

Browse Audio

In case of an audio file selection, this feature facilitated the action and validates the selected input. The initiated intent will give an option for user to select the audio file through any of the installed app intended to open them. The screen shot for this is shown in the appendix. Below Figure 5.10 shows the browse audio section.

Figure 5.9 – Browse Audio

Record Audio

In few screens of the application instead of selecting an existing audio, user can record his own voice and use in the application. First click of the record button will start the voice recording. And the next click will stop recording and display a new section below the with an option to play the recently record audio. All recorded voice will be stored in the location "/sdcard/protectMSG/RecordedAudio/". Below Figure 5.10 shows the record audio section.

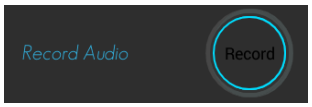

Figure 5.10 – Record Audio

Help

Help screen provides general information, restriction and limitations of protectMSG application. Listed information are significant for the users to access and know about the application. They are categorized into Basic, Audio, Image and Quick Voice Message. Categorization provides clarity in explaining necessary information to the users.

5.5 Class Diagram

Class diagram is a structure that describes the relation between the system's classes. In protectMSG application, several java classes are used but only important class relationships are shown in the diagram below. EncodeTextToImage class is the center of attraction in the diagram. This is shown in the Figure 5.12

MainActivity.java

MainActivity class is specified in the android manifest file to load this class on start of the application. This class is responsible for redirection to other activities on request.

EncodeTextToImage.java

Primary function of EncodeAudioToImage class is to convert the text to byte array, image file to byte array and then send these arrays to StegoOperation class, which performs LSB encoding. The resultant byte array from StegoOperation class is again formulated into an image file. This class is also responsible for activities like sharing the encoded file, capturing image using the camera, browsing picture. Similarly EncodeTextToAudio and EncodeAudioToImage class does convert message and cover file into byte array and responsible for task like sharing and playing selected audio. They also share the similar kind of relationship with other classes as that of

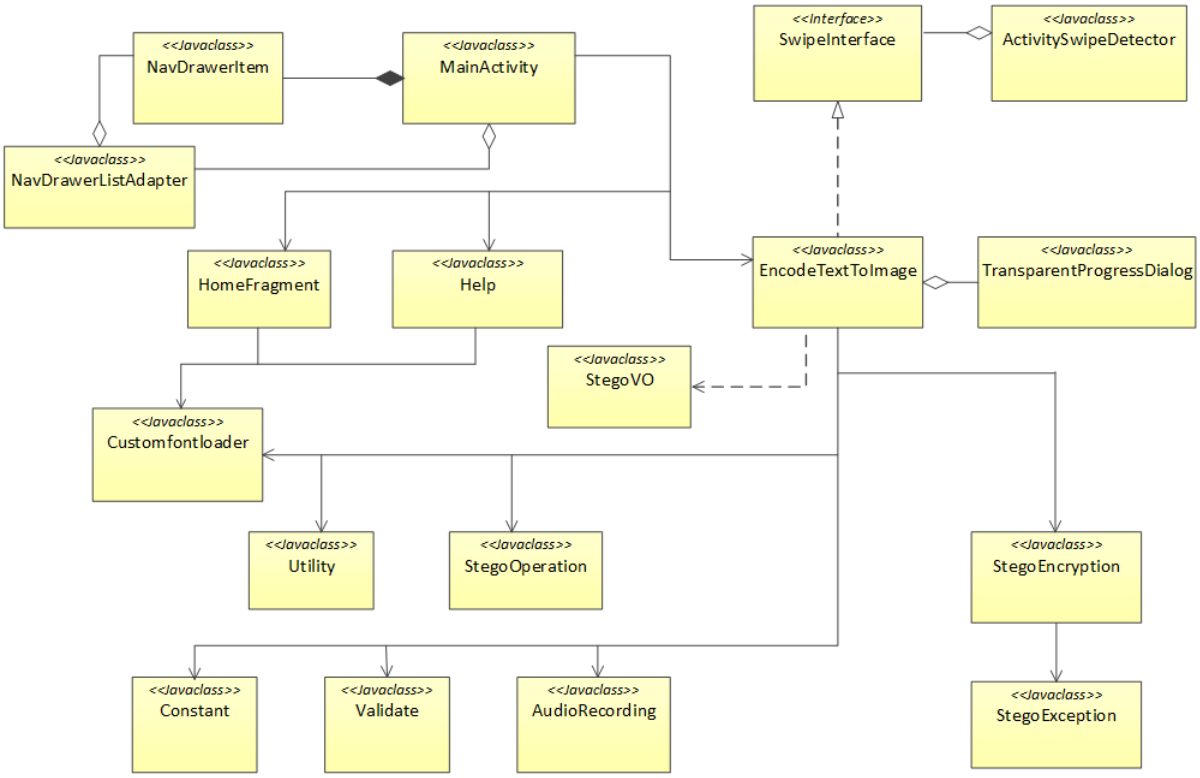

Figure 5.11 – Simplified Class Diagram

EncodeAudioToImage.

DecodeTextFromImage.java

DecodeTextFromImage class is used to extract the text message from the image file. DecodeAudioFromImage and DecodeTextFromAudio class also extract the message from the corresponding cover files.

Help.java

Help class is a fragment used to display the necessary information regarding the application.

HomeFragment.java

This class is related to the functioning of the home page in the app.

Validate.java

Validate class does basics functions to verify the inputs such as password, encode text, input file

size and file extensions.

StegoEncryption.java

AES encryption and decryption is implemented in StegoEncryption class. A dedicated method is available for password padding.

StegoOperation.java

StegoOperation is a significant class implementing the core functionality, LSB technique. Accepts message and cover file as input in bytes and performs encoding operation. Also performs decode operation and extracts the message from the cover file.

StegoVO.java

StegoVO is a value object to store details about message and errors belonging to the encoding or decoding transaction. Object of this class is passed as a parameter to the encode and decode methods to carry messages and errors of the current action.

AudioRecording.java

AudioRecording class handles recording audio and saving it in the appropriate SD card location.

Constant.java

To maintain coding standards, all strings used in this project are grouped in Constant class. The caller can refers to these constants using static references. So the caller will never need to create a Constant object.

Customfontloader.java

In the application there are several fonts used in different places. Static methods from Customfontloader class are used to apply necessary fonts to required UI component easily.

Utility.java

Pulling the locations of the audio and image files from the SD card, generating filenames for newly encoded files and showing custom alert are the functionality of the Utility class.

In the development process, several struggles has been overcame to achieve the desired output. These road blocks were discussed in details in the section 6.5 Challenges Confronted.

Chapter 6

Evaluation

This chapter investigate and captures test results, behavior and every approach handled to provide best quality product.

6.1 Test Strategy

Software testing approach is defined initially to achieve testing objectives. Testing methodology and the testing procedures for each testing are defined prior to the testing. Testing plan is derived from the specified requirement of the application and the test scope is also defined. Test plan contains full details of test approach, test conditions, test cases, expected results, test exclusions. Required environment and tools for testing are identified and prepared.

6.2 Testing Scope

Instead of creating a single master test plan, the strategy is to divide testing into several small test cases. Each test case concentrate on single function and scripts their outcomes. Addressing each functions separately provides modularity and linearity in testing. By simplifying the testing process quality of the test results increases in terms of accuracy. Testing scope is to test all functionality, user interface, maintain the test approach throughout, provide the test deliverable at the end and it also include device specific testing.

6.3 Test method

6.3.1 Functionality Test

Functionality test is similar to black-box testing, where the focus is on functions of the application rather than the way of programming or coding. Several modules in the application are expected to behave in the programmed way. These expected results are grouped to form test cases. Then testing is done with reference to these test cases. One of the test case is shown below in the Table 6.1, rest are attached in the appendix.

Table 6.1 – Test Case no:001

Test Case no:	**001**
Test Case	Encode Text To Image
Test Case Description	On clicking of encode button, text message is encoded into the cover image.
Test Input	Valid text is considered as a message that needs to be hidden. Valid image is also an input from user to conceal the text message.
Expected Result	New image is generated with the given text message hidden in it.

6.3.2 User Interface Test

User Interface Test is a testing technique, which test the application using graphical user interface to find the defects. GUI events like key press and button click are triggered to check for unexpected results.[24] Check list verified during the user interface tests are :

- Check Screen Validations
- Verify All Navigations
- Check usability Conditions
- Verify Data Integrity
- Verify the object states
- Verify the date Field and Numeric Field Formats

6.3.3 Monkey Testing

All bugs are expected to be caught by test cases. It does not happens every time. Sometime random test reveal unexpected error which was not trapped according to the plan. This random test is called monkey test. It is a test where random inputs are giving without any restriction. Advantage of this testing is the possibility of a generation of a new test case, which was originally skipped while preparing the test plan initially.

The Monkey is a command-line tool. It runs both on emulator instance and device. It sends a pseudo-random stream of user events into the system, which acts as a stress test on the application. cite11

Four primary categories of Monkey

- Basic configuration options, such as setting the number of events to attempt.
- Operational constraints, such as restricting the test to a single package.
- Event types and frequencies.
- Debugging options.

When the Monkey runs, it generates events and sends them to the system. It also watches the system under test and looks for three conditions, which it treats specially. If the Monkey is constrained to run in one or more specific packages, it watches for attempts to navigate to any other packages, and blocks them. If the application crashes or receives any sort of unhandled exception, the Monkey will stop and report the error. If the application generates an application not responding error, the Monkey will stop and report the error.[11] Events are generated based on the selected verbosity level.

Monkey is launched using a command line on development machine or from a script. Because the Monkey runs in the emulator/device environment, it must be launched from a shell in that environment. It can be done by prefacing adb shell to each commands or by entering the shell and entering Monkey commands directly

6.3.4 Device Oriented Test

Since the application is designed to run on android version 4.2.2, jelly bean, it was tested on Samsung Galaxy Tab3 GT-P5210. Specification of the device tested is shown below in the Table 6.2.

Table 6.2 – Test Device Specification

Device Specification	**Samsung Galaxy Tab3 GT-P5210**
Form Factor	Tablet, Touchscreen
OS	Android 4.2.2 Jelly Bean
Display Type	TFT capacitive touchscreen, 16M colors
Display Size	10.1 inches (149 ppi pixel density)
Display Resolution	1280 x 800 pixels
Primary Camera	3 MP
Secondary Camera	1.3 MP
Primary Camera	256 ppi
Chipset	Intel Atom Z2560
CPU	Dual-core 1.6 GHz
Internal memory	16 GB
Memory card slot	microSD up to 64 GB
RAM	1GB
WLAN	Wi-Fi 802.11 a/b/g/n, Wi-Fi Direct, Wi-Fi hotspot
Bluetooth	v4.0, A2DP
USB	microUSB v2.0 (MHL 1.2), USB Host

6.4 Non-Functional Requirements Evaluation

General operation of a system in a routine or everyday use is classified as non-functional requirement. Resource requirement such as writable SD Card, sufficient memory is a non functional requirement. Extensibility is also a non functional requirement. Extensibility make sure the developed application is adaptable for additional functionality without much effort and customizable for forthcoming version upgrades. These requirements are verified for its completion along with performance, recovery, security, endurance and usability.

6.5 Challenges Confronted

This thesis was a new learning curve in android application development. On the way of discovering several innovative concepts, several challenges were faced. Listing here few significant difficulties overcame.

6.5.1 Emulator Setup

Emulator tool is provided by android for accessing or testing the application in the PC system before deploying into the actual devices. It is the tool which actually simulates the program for us. It is difficult for a developer to test in all type of hardware devices. Instead emulator can be used to simulate hardware device of different specification. Using emulator it is possible to create emulator for different size of screens. According to the requirement of the application, SD card, camera and microphone has to be associated to the emulator.

6.5.2 Encryption Key Generation

Encryption key generation was a critical challenge in the phase of development. The key used in encryption is 128 bits in length. In the early stage the key generated was random every time. Then after removing random function, an other issue popped up. The key was inconsistent because of the incorrect length of the password used to generate the key. So this problem was solved by padding the password to 128 bits length. This worked perfectly for every attempt of the encryption and also for all kind of passwords.

Chapter 7

Conclusion

This is the final chapter of this dissertation and details the summary of the entire dissertation and exploits the area for scope of future enhancements.

7.1 Summary

Steganography is an effective technique and incredibly versatile for hiding information from plain sight. Although methods for detecting hidden data do exist, they cannot be entirely relied upon, as none of them are fully effective. So, among the loads of available steganographic technique, a suitable one is chosen to implement in android. Attributes of the chosen technique is carefully designed to fit the android environment for better performance. This chosen technique is nothing but the Least Significant Bit (LSB) steganography. It works out well for but image and audio steganography.

The protectMSG Android app was designed to hide text data in image and audio file. And voice message of maximum 10 second duration can also be hidden in an image file. This is achieved by implemented LSB steganographic technique in an innovative way. Cryptography and steganography is mixed elegantly in the design of this app to make it secure and better. AES encryption is used to encrypt the messages which is better than DES encryption. And when it is comes to results, this app is able to successfully extract the hidden message from the cover

file. It is made sure that only intended recipient extract it. The protectMSG app is developed in such a way that it is user friendly. User is well directed at any point time in the app by providing appropriate information, warnings and errors message.

7.2 Future Enhancements

7.2.1 Validity for the message

A new exciting feature is to bind a validity for a message in a cover image or audio file. Since this is an application for mobile devices, it is easy to share a cover image or audio to other person. In order to improve the level of safety, a life time for the message hidden can be set in the cover file. This can be achieved by embedding the expiry date along with message in the cover file. Logic should be implemented to retrieve only unexpired messages.

7.2.2 Other input file type

Currently the input files to the system is restricted based on the file types. In case of image files, only PNG, JPG and JPEG can be used. In case of audio file, only WAV files are accepted as input. There are still many other file formats are in used today. MP3 are the most commonly used audio files. And even many other file formats are available for image files. So to make the protectMSG app support all other file type is also included in the future enhancements.

7.2.3 Hide an image in a cover image

The protectMSG android app has facility to hide a text in image or audio. It can also hide audio in an image. In future, steganography technique of hiding a image in a cover image can be incorporated.

7.2.4 Hide an audio in a cover audio

Other possible combination of message and medium are audio hiding in a audio. This is also possible to implement in future if the audio message size is considerable lesser than the cover audio file.

In whole, scripted requirement is theoretically designed and implemented in practice as an android app named protectMSG. This final product is stable and fulfills all the initial requirements. This application is found to be robust and fit for all kind of audience.

Appendix A

Appendix

A.1 Screenshots

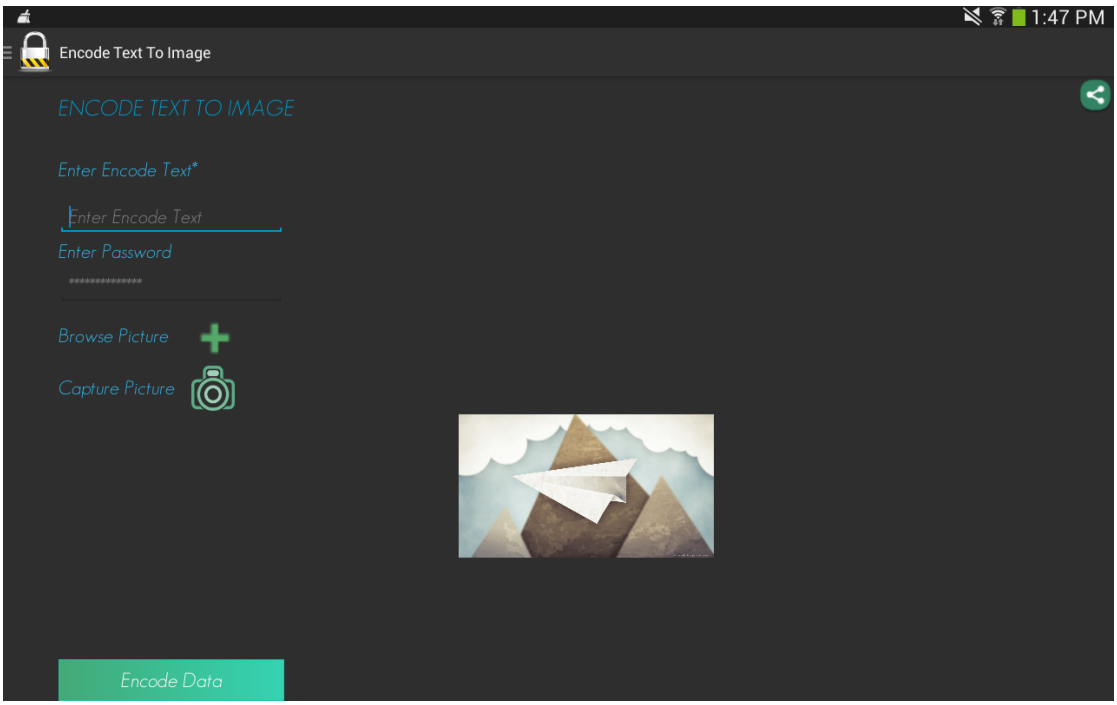

Figure A.1 – Encode Text To Image

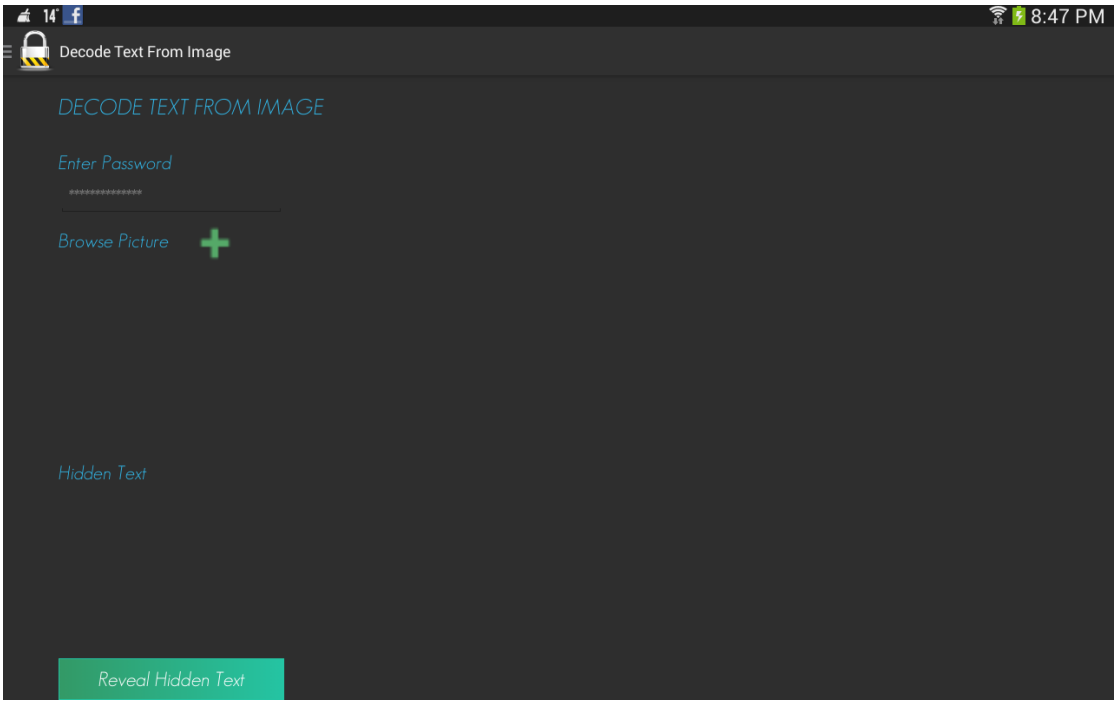

Figure A.2 – Decode Text From Image

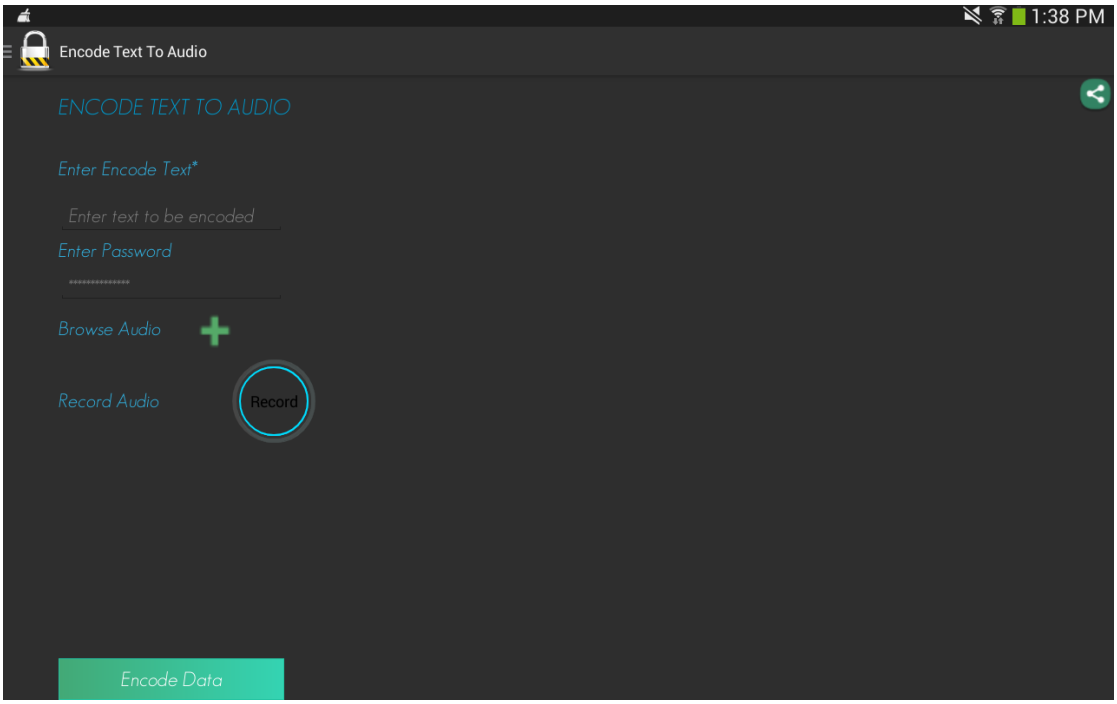

Figure A.3 – Encode Text To Audio

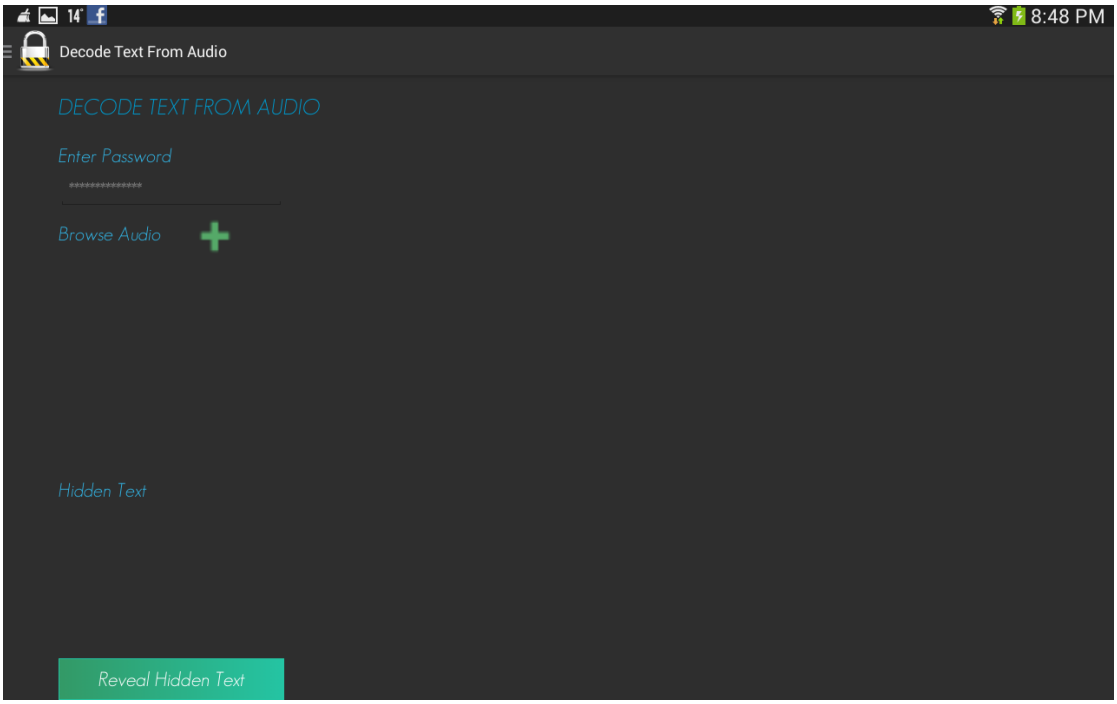

Figure A.4 – Decode Text From Audio

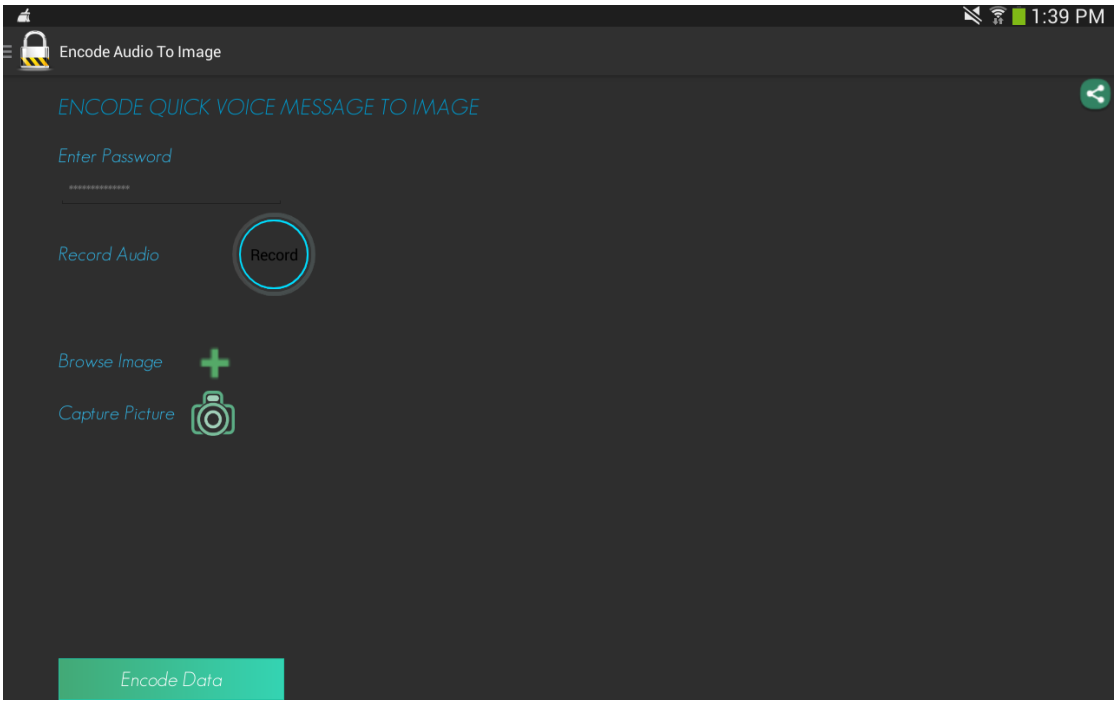

Figure A.5 – Encode Quick Voice Message To Image

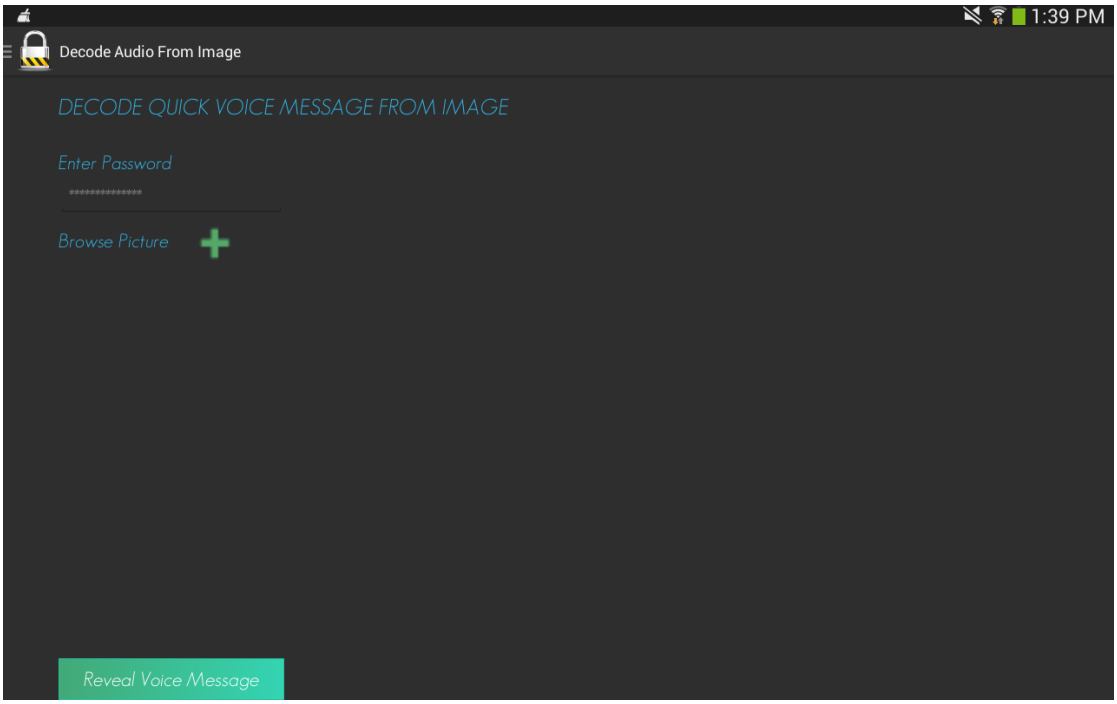

Figure A.6 – Decode Quick Voice Message From Image

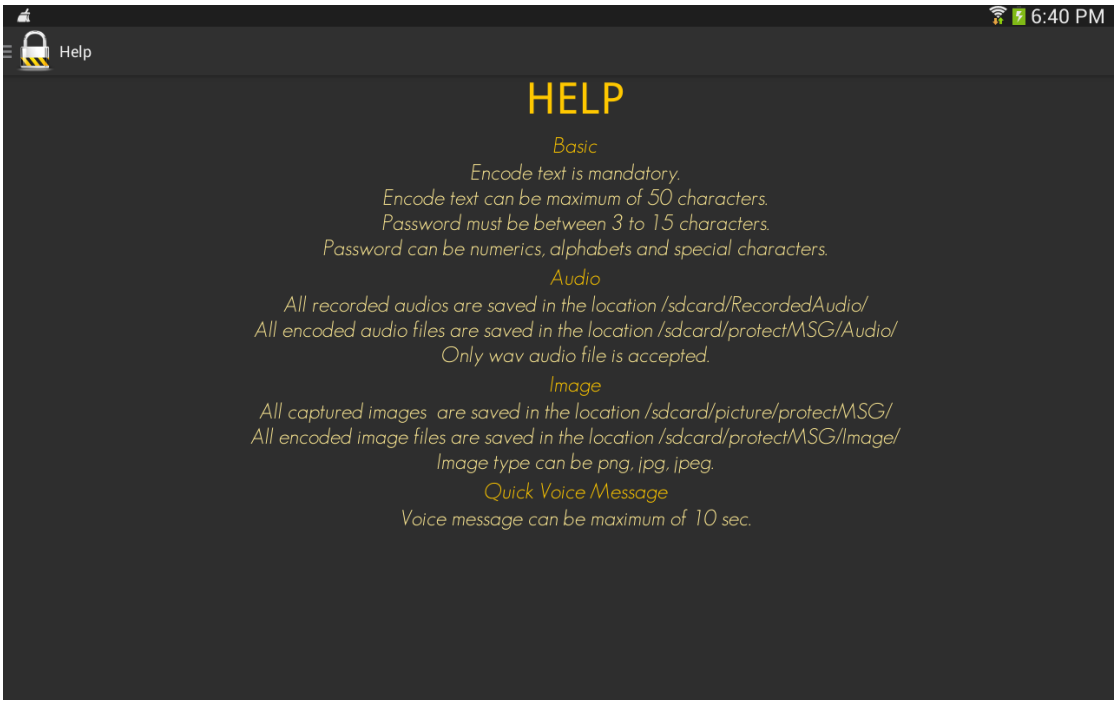

Figure A.7 – Help

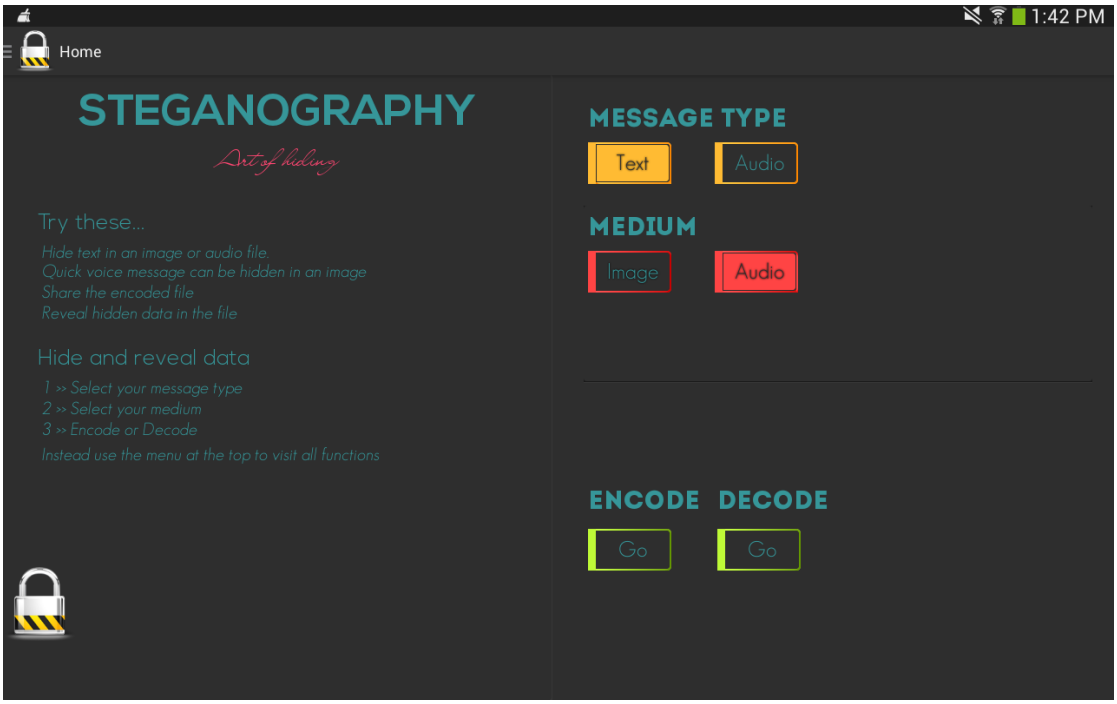

Figure A.8 – Home with button selected

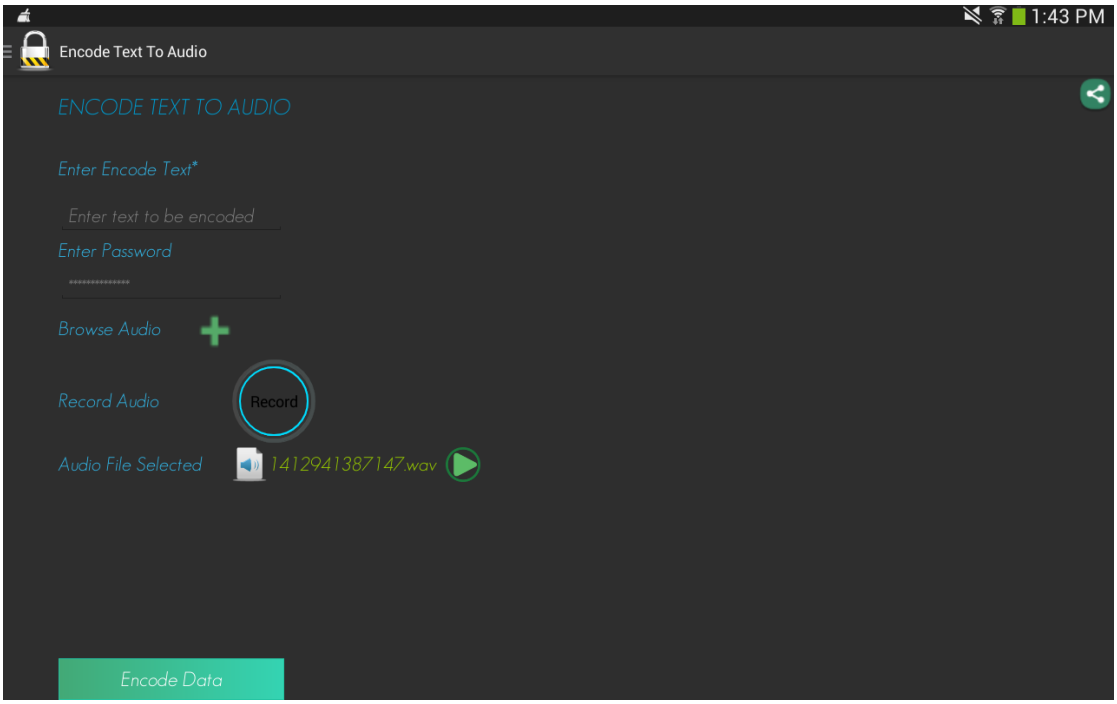

Figure A.9 – Encode Audio To Image with audio recorded

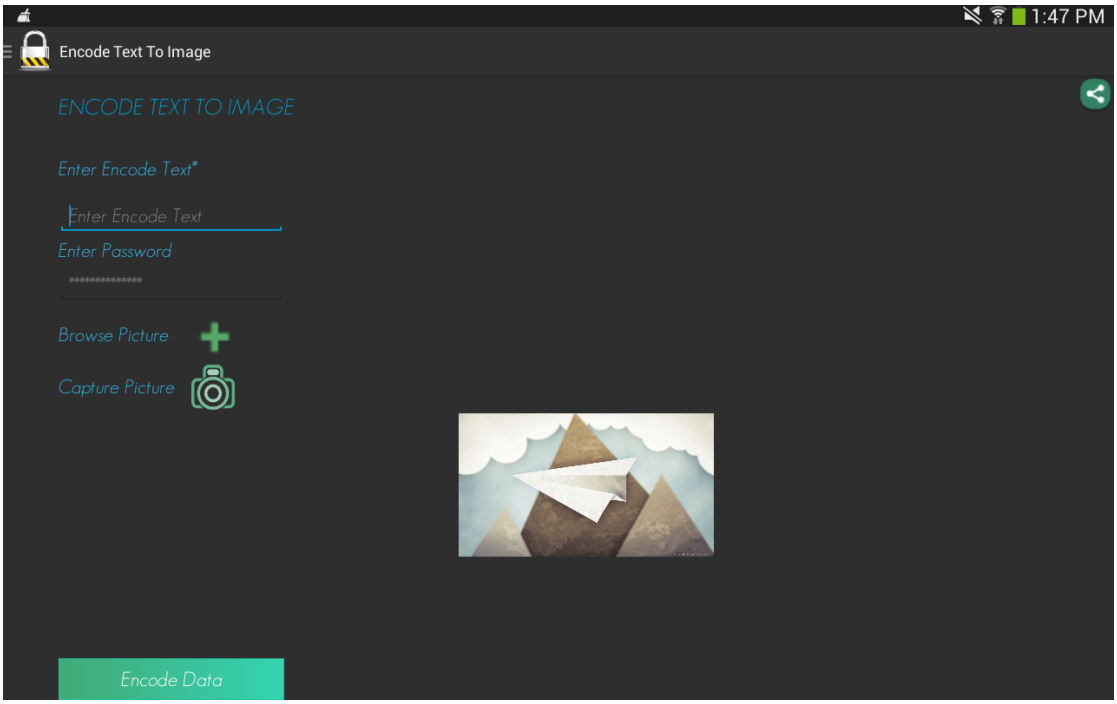

Figure A.10 – Encode Text To Audio With Image Selected

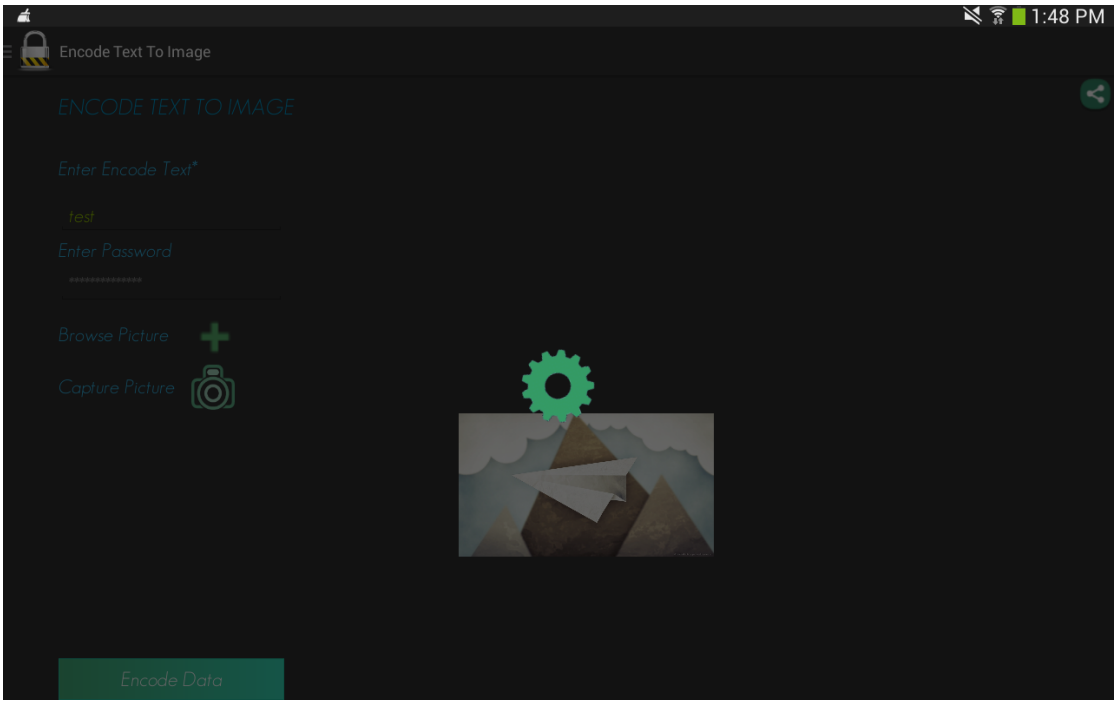

Figure A.11 – Processing

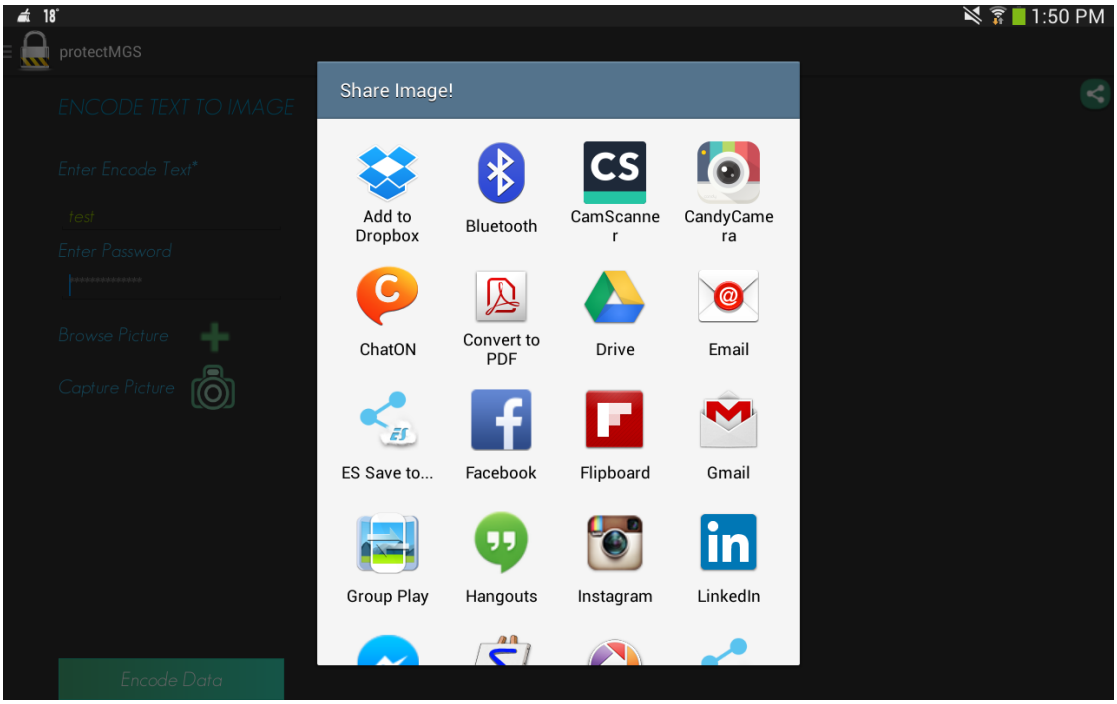

Figure A.12 – Share Image

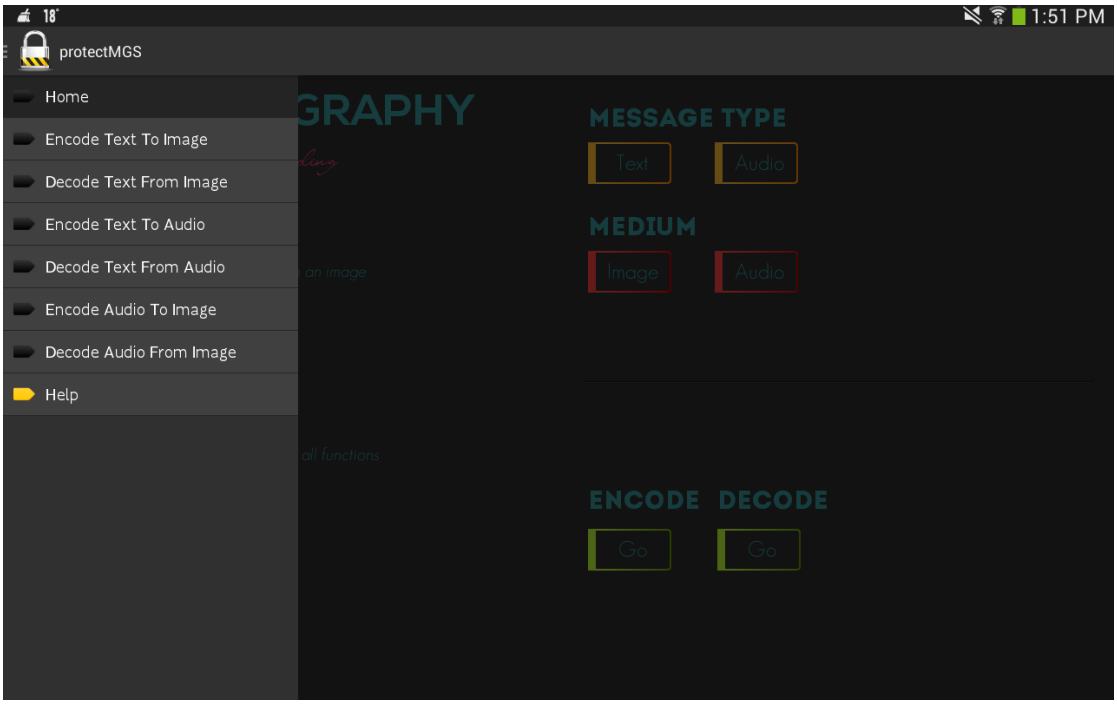

Figure A.13 – Menu

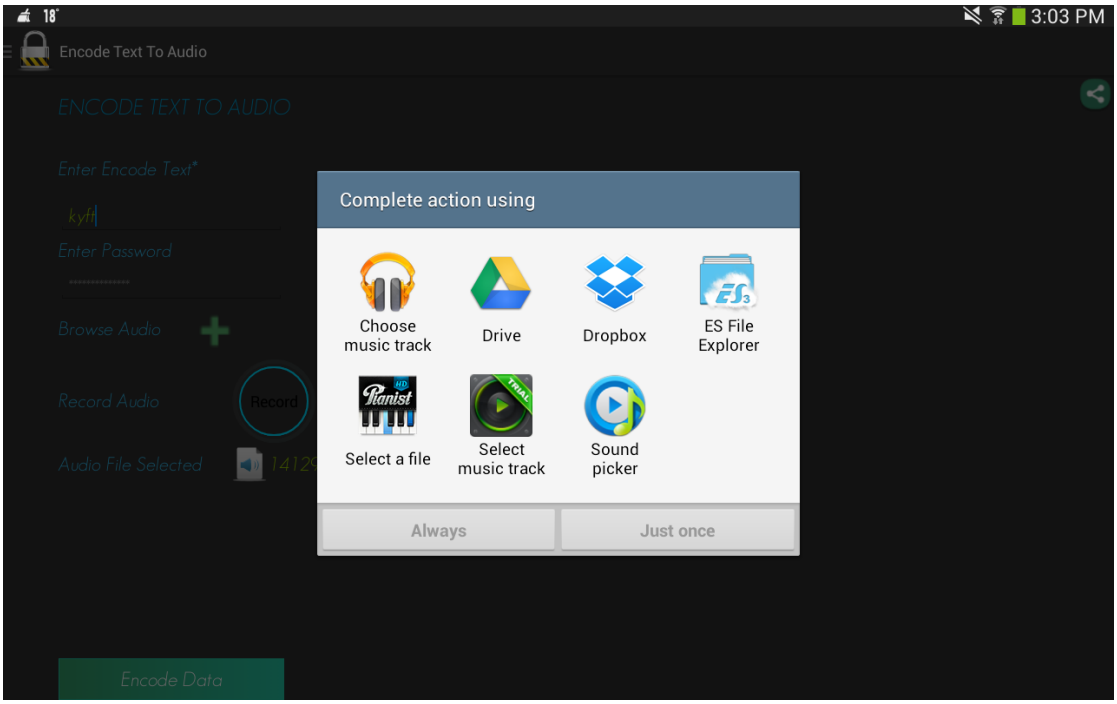

Figure A.14 – Browse Audio

A.2 Use Case

Table A.1 – Use Case no: 1

Use Case 1	**Home**
Goal in Context	Home screen displays overview of the functionality of protectMSG application. And user has the facility to navigate other pages.
Preconditions	
Successful Post Condition	Overview of the application is present along with a way to access different functionalities of the application.
Failed Post Conditions	If user selects medium and message as audio, then the application will inform user that encode or decode audio in audio is not possible. Because it is not included in the scope of the application.
Scenario Action	Step 1: User selects message type and medium in any order. Step 2: On click of "Go" button for encode or decode will leads to the appropriate screen based on the input.

Table A.2 – Use Case no: 2

Use Case 2	**Encode text into Image**
Goal in Context	Hiding text data in an image file.
Preconditions	
Successful Post Condition	Generates a PNG image file with the hidden text data in it.
Failed Post Conditions	Throws error for invalid text, password, and image.
Scenario Action	Step 1: User opens the application. Step 2: Navigate to "Encode text into Image" screen Step 3: Enter encode text. Step 4: Enter password if necessary. Step 5: Select picture Step 6: Click "Encode Data" button.

Table A.3 – Use Case no: 3

Use Case 3	**Decode text From Image**
Goal in Context	Reveal text data from an encoded image file.
Preconditions	Image should be already an encoded image by this application
Successful Post Condition	Reveals the hidden text from the encoded audio file.
Failed Post Conditions	
Scenario Action	Step 1: User opens the application Step 2: Navigate to "Decode text From Image" screen Step 3: Enter password if the encoded file is secured by the password Step 4: Select image Step 5: Click "Reveal Hidden Text" button

Table A.4 – Use Case no: 4

Use Case 4	**Encode text into Audio**
Goal in Context	Hiding text data in an audio file.
Preconditions	
Successful Post Condition	Generates a WAV audio file with the hidden text data in it.
Failed Post Conditions	
Scenario Action	Step 1: User opens the application. Step 2: Navigate to "Encode text into Audio" screen. Step 3: Enter encode text. Step 4: Enter password if necessary. Step 5: Select audio or record audio. Step 6: Click "Encode Data" button.

Table A.5 – Use Case no: 5

Use Case 5	**Decode text From Audio**
Goal in Context	Reveal text data from an encoded audio file.
Preconditions	Audio should be already an encoded audio by this application.
Successful Post Condition	Reveals the hidden text in the encoded audio.
Failed Post Conditions	
Scenario Action	Step 1: User opens the application. Step 2: Navigate to "Decode text From Audio" screen. Step 3: Enter password if the encoded file is secured by the password. Step 4: Select audio. Step 5: Click "Reveal Hidden Text" button.

Table A.6 – Use Case no: 6

Use Case 6	**Encode Audio into Image**
Goal in Context	Hiding audio data in an image file.
Preconditions	
Successful Post Condition	Generates a PNG image file with the hidden audio data in it.
Failed Post Conditions	
Scenario Action	Step 1: User opens the application. Step 2: Navigate to "Encode Audio into Image" screen. Step 3: Enter encode text. Step 4: Enter password if necessary. Step 5: Select audio or record audio. Step 6: Select image. Step 7: Click "Encode Data" button.

Table A.7 – Use Case no: 7

Use Case 7	**Decode Audio From Image**
Goal in Context	Reveal audio data from an encoded image file.
Preconditions	Image should be already an encoded image by this application.
Successful Post Condition	Generates the audio file containing the hidden voice data from the image.
Failed Post Conditions	
Scenario Action	Step 1: User opens the application. Step 2: Navigate to "Decode Audio From Image" screen. Step 3: Enter password if the encoded file is secured by the password. Step 4: Select image. Step 5: Click "Reveal Voice Message" button.

Table A.8 – Use Case no: 8

Use Case 8	**Share**
Goal in Context	To share the recently encoded file.
Preconditions	An image or audio should be encoded and generated.
Successful Post Condition	Pop up appears with all installed application's list which has the ability to share file. Example- Gmail, Facebook, Viber and etc.
Failed Post Conditions	Throws error if the encode file is not available or generated previously
Scenario Action	Step 1: After encoding, the encoded file can be shared by clicking the share button at the top right corner.

A.3 Test Case

Table A.9 – Test Case no: 001

Test Case no: 001	
Test Case	Encode Text To Image
Test Case Description	On clicking of encode button, text message is encoded into the cover image, only when the inputs are valid.
Test Input	Valid text is the input message to be hidden. Valid image is also an input from user to conceal the text message.
Expected Result	New image is generated with the given text message hidden in it.

Table A.10 – Test Case no: 002

Test Case no: 002	
Test Case	Decode Text From Image
Test Case Description	On clicking of decode button, text message is extracted from the cover image, only on valid inputs.
Test Input	Image encoded with text by protectMSG is an input to the system. If that image was protected by password then it must be provided as input.
Expected Result	Hidden text is extracted from the cover image and shown to user.

Table A.11 – Test Case no: 003

Test Case no: 003	
Test Case	Encode Text To Audio
Test Case Description	On clicking of encode button, text message is encoded into the cover audio, only on valid inputs.
Test Input	Valid input text is the message to be hidden. Valid audio is also an input from user to conceal the text message.
Expected Result	New image is generated with the given text message hidden in it.

Table A.12 – Test Case no: 004

Test Case no: 004	
Test Case	Decode Text From Audio
Test Case Description	On clicking of decode button, text message is extracted from the cover audio, only on valid inputs.
Test Input	Audio encoded with text by protectMSG is an input to the system. If that image was protected by password then it must be provided as input.
Expected Result	Hidden text is extracted from the cover audio and shown to user.

Table A.13 – Test Case no: 005

Test Case no: 005	
Test Case	Encode Audio To Image
Test Case Description	On clicking of encode button, audio message is encoded into the cover image, only on valid inputs.
Test Input	Valid input audio is the message to be hidden. Valid audio is also an input from user to conceal the text message.
Expected Result	New image is generated with the given text message hidden in it.

Table A.14 – Test Case no: 006

Test Case no: 006	
Test Case	Decode Audio From Image
Test Case Description	On clicking of decode button, audio message is extracted from the cover image, only on valid inputs.
Test Input	Image encoded with audio by protectMSG is an input to the system. If that image was protected by password then it must be provided as input.
Expected Result	Hidden audio is extracted from the cover image and shown to user for playing.

A.4 Source Code

A.4.1 EncodeTextToImage

```
package com.protectMSG.operation;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.IntBuffer;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.Fragment;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.Bitmap.CompressFormat;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.provider.MediaStore;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnFocusChangeListener;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

import com.protectMSG.stego.operation.StegoEncryption;
```

```
import com.protectMSG.stego.operation.StegoOperation;
import com.protectMSG.util.Customfontloader;
import com.protectMSG.util.Utility;

public class EncodeTextToImage extends Fragment {

  private static final String TAG = "FindPeopleFragment";
  private TransparentProgressDialog progressBar;
  private static int RESULT_LOAD_IMAGE = 1;
  String picturePath = null;
  String outputPicturePath = null;
  View rootView = null;
   // Activity request codes
   private static final int CAMERA_CAPTURE_IMAGE_REQUEST_CODE = 100;
  public static final int MEDIA_TYPE_IMAGE = 1;
  public static final int MEDIA_TYPE_VIDEO = 2;
  // directory name to store captured images and videos
   private static final String IMAGE_DIRECTORY_NAME = "protectMSG";
   Uri fileUri = null;
   ImageView imageView = null;
   View layout = null;
   EditText encodeEditText = null;
   EditText passwordEditView = null;
  TextView errorEncodeTextView = null;
  TextView errorPasswordTextView = null;
   boolean error = false;
   Handler handler = null;
  public EncodeTextToImage() {
  }

  @SuppressLint("CutPasteId")
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
       Bundle savedInstanceState) {
    Log.i(TAG, "Enter : onCreateView");
    rootView = inflater.inflate(R.layout.encode_text_to_image, container,false);
    TextView tv1 = (TextView) rootView.findViewById(R.id.headerTextView);
    TextView tv2 = (TextView) rootView.findViewById(R.id.enterEncodeTextView);
      TextView tv3= (TextView) rootView.findViewById(R.id.enterPasswordTextView);
      TextView tv4= (TextView) rootView.findViewById(R.id.browsePictureTextView);
      TextView tv5= (TextView) rootView.findViewById(R.id.capturePictureTextView);
      encodeEditText= (EditText) rootView.findViewById(R.id.encodeEditText);
      passwordEditView= (EditText) rootView.findViewById(R.id.passwordEditView);
```

```
 errorEncodeTextView = (TextView)
     rootView.findViewById(R.id.errorEncodeTextView);
 errorPasswordTextView = (TextView)
     rootView.findViewById(R.id.errorPasswordTextView);

 Button encodeButton= (Button) rootView.findViewById(R.id.encodeButton);
 tv1.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 tv2.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 tv3.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 tv4.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 tv5.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 encodeEditText.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 passwordEditView.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 encodeButton.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 errorEncodeTextView.setTypeface(Customfontloader.getTypeface(getActivity(),1));
 errorPasswordTextView.setTypeface(Customfontloader.getTypeface(getActivity(),1));

//validate encode text on out of focus
 encodeEditText.setOnFocusChangeListener(new OnFocusChangeListener() {
    public void onFocusChange(View v, boolean hasFocus) {
      Log.i(TAG, "setOnFocusChangeListener : encodeText");
       if(!hasFocus){
         error =
             Validate.validateEncodeText(encodeEditText.getText().toString());

          if(error)
            errorEncodeTextView.setVisibility(View.VISIBLE);
          else
            errorEncodeTextView.setVisibility(View.INVISIBLE);
       }
    }
 });


final Animation animRotate = AnimationUtils.loadAnimation(getActivity(),
    R.anim.anim_rotate);


final Button shareButton = (Button) rootView.findViewById(R.id.shareButton);
 shareButton.setOnClickListener(new View.OnClickListener() {
  @SuppressLint("NewApi")
  public void onClick(View arg0) {
    Log.i(TAG, "onClick : shareButton");
    arg0.startAnimation(animRotate);
```

```
        if(outputPicturePath!=null){
         Intent share = new Intent(Intent.ACTION_SEND);


            share.setType("image/*");

             File imageFileToShare = new File(outputPicturePath);
              Uri uri = Uri.fromFile(imageFileToShare);
              share.putExtra(Intent.EXTRA_STREAM, uri);

              startActivity(Intent.createChooser(share, "Share Image!"));

        }else{
          Utility.showCustomAlert("No encoded image available to
               share",getActivity(),"Error");
        }
      }
      });

    // Browse Button - onClick - opens the image library
    Button buttonLoadImage = (Button) rootView.findViewById(R.id.browseButton);
    buttonLoadImage.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View arg0) {
        Log.i(TAG, "onClick : browseButton");
        Intent i = new
            Intent(Intent.ACTION_PICK,android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(i, RESULT_LOAD_IMAGE);
      }
    });

    // Capture Button - onClick - opens the Camera app
    Button buttonCaptureImage = (Button) rootView.findViewById(R.id.captureButton);
    buttonCaptureImage.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View arg0) {
        Log.i(TAG, "onClick : captureButton");
        captureImage();
      }
    });


    //Encode Button - onClick - encodes the image file
    final Button button = (Button) rootView.findViewById(R.id.encodeButton);
```

```
button.setOnClickListener(new View.OnClickListener() {
  @SuppressLint("NewApi")
  public void onClick(View v) {
    Log.i(TAG, "onClick : encodeButton");

    //error checking for valid passport
    boolean passwordError =
        Validate.validatePassword(passwordEditView.getText().toString());

       if(passwordError)
        errorPasswordTextView.setVisibility(View.VISIBLE);
       else
        errorPasswordTextView.setVisibility(View.INVISIBLE);

      //error checking for valid encode text
       boolean encodeTextError =
           Validate.validateEncodeText(encodeEditText.getText().toString());

       if(encodeTextError)
        errorEncodeTextView.setVisibility(View.VISIBLE);
       else
        errorEncodeTextView.setVisibility(View.INVISIBLE);

       boolean error = false;
       if(picturePath==null){
        Utility.showCustomAlert("Select Image To
            Encode",getActivity(),"warning");
        error=true;
       }

    if(!passwordError&&!encodeTextError&&!error){

    progressBar = new TransparentProgressDialog(v.getContext(),
        R.drawable.spinner);
    progressBar.setCancelable(true);
    progressBar.show();

    handler = new Handler();
     Runnable runnable = new Runnable() {
        public void run() {
         encodeData();
              handler.post(new Runnable(){
                 public void run() {
```

```
                    Utility.showCustomAlert("Encode Finished", getActivity(),
                        "Info");
                    // close the progress bar dialog
                    progressBar.dismiss();
                  }
                });
                }
        };
        new Thread(runnable).start();

       }
       }

  });


  return rootView;
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  Log.i(TAG, "Enter : onActivityResult()");
  if (requestCode == RESULT_LOAD_IMAGE && resultCode == Activity.RESULT_OK &&
      null != data) {
    Uri selectedImage = data.getData();
    String[] filePathColumn = { MediaStore.Images.Media.DATA };

    Cursor cursor = getActivity().getContentResolver().query(selectedImage,
        filePathColumn, null, null, null);
    cursor.moveToFirst();

    int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
    picturePath = cursor.getString(columnIndex);
    cursor.close();

    if(Validate.validateFileExtension(picturePath,"Image"))
    {
      Utility.showCustomAlert("Select Valid File",getActivity(),"error");

    }else if(Validate.validateFileSize(picturePath,"Image",getActivity())){
      Utility.showCustomAlert("File size is too large",getActivity(),"error");
    }else{
      imageView = (ImageView) rootView.findViewById(R.id.imageView1);
```

```
        imageView.setImageBitmap(BitmapFactory.decodeFile(picturePath));
      }


   }
    // if the result is capturing Image
    if (requestCode == CAMERA_CAPTURE_IMAGE_REQUEST_CODE) {
       if (resultCode == Activity.RESULT_OK) {

        if(data != null) {
        Bitmap thumbnail = (Bitmap) data.getExtras().get("data");
        imageView.setImageBitmap(thumbnail);
        }
       } else if (resultCode == Activity.RESULT_CANCELED) {
          // user cancelled Image capture
          Utility.showCustomAlert("Image Capture Cancelled",getActivity(),"Info");

       } else {
          // failed to capture image
          Utility.showCustomAlert("Sorry, Failed to capture
              image",getActivity(),"Error");
       }
    }
}

//Method to encode data in image file
private void encodeData() {
  Log.i(TAG, "Enter : encodeData()");
  EditText edit = (EditText) rootView.findViewById(R.id.encodeEditText);
  EditText passwordEdit = (EditText) rootView.findViewById(R.id.passwordEditView);
  // encode
  // Read file to bytes
  Bitmap bmp = BitmapFactory.decodeFile(picturePath);
  ByteArrayOutputStream bos = new ByteArrayOutputStream();
  bmp.compress(CompressFormat.PNG, 0, bos);
  String encode_text = edit.getText().toString();
  String password = null;
  boolean isPasswordLocked = false;

  try {
    //Encrypt with password if it not null
    if(passwordEdit.getText().length()!=0)
    {
      isPasswordLocked=true;
```

```
    password = passwordEdit.getText().toString();
    encode_text=StegoEncryption.encrypt1(encode_text,password);
}
StegoOperation operation = new StegoOperation();

// Make the bitmap mutable
bmp = StegoOperation.convertToMutable(bmp);

// Get integer pixel array from the input bitmap
int[] pixels1 = new int[bmp.getHeight() * bmp.getWidth()];
bmp.getPixels(pixels1, 0, bmp.getWidth(), 0, 0,bmp.getWidth(),
    bmp.getHeight());

// Convert int array to byte array
ByteBuffer byteBuffer = ByteBuffer.allocate(pixels1.length * 4);
IntBuffer intBuffer = byteBuffer.asIntBuffer();
intBuffer.put(pixels1);
byte[] byteArray1 = byteBuffer.array();

byteArray1 = operation.encode(byteArray1, encode_text,isPasswordLocked);

Log.i(TAG,"Convert byte array to int array");
Log.i(TAG,"byteArray1 length : "+byteArray1.length);
IntBuffer intBuf =
    ByteBuffer.wrap(byteArray1).order(ByteOrder.BIG_ENDIAN).asIntBuffer();
Log.i(TAG,"Step 1 done");
Log.i(TAG,"intBuf.remaining()"+intBuf.remaining());
Log.i(TAG,"Step 1.5 done");
int[] array = new int[intBuf.remaining()];
Log.i(TAG,"Step 2 done");
intBuf.get(array);
Log.i(TAG,"Step 3 done");
pixels1 = array;

Log.i(TAG,"Set encoded array to bitmap");
// Set bitmap with newly encoded array
bmp.setPixels(pixels1, 0, bmp.getWidth(), 0, 0,bmp.getWidth(),
    bmp.getHeight());

Log.i(TAG,"Create output file from bitmap");
outputPicturePath=Utility.generateImageFileName();
OutputStream fOut = new FileOutputStream(outputPicturePath);
bmp.compress(Bitmap.CompressFormat.PNG, 0, fOut);
fOut.flush();
```

```
            fOut.close();
        } catch (Exception e) {
            e.printStackTrace();
            Utility.showCustomAlert("Error Occurred", getActivity(), "Error");
        }

    }

    /*
     * Capturing Camera Image will lauch camera app requrest image capture
     */

    private void captureImage() {
        Log.i(TAG, "Enter : captureImage()");
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);


        Uri fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

        // start the image capture Intent
        startActivityForResult(intent, CAMERA_CAPTURE_IMAGE_REQUEST_CODE);
    }


    /**
     * ------------ Helper Methods ----------------------
     * */

    /**
     * Creating file uri to store image/video
     */
    public Uri getOutputMediaFileUri(int type) {
        Log.i(TAG, "Enter : getOutputMediaFileUri()");
        return Uri.fromFile(getOutputMediaFile(type));
    }

    /**
     * returning image / video
     */
    private static File getOutputMediaFile(int type) {
        Log.i(TAG, "Enter : getOutputMediaFile()");
        // External sdcard location
```

```
        File mediaStorageDir = new
             File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
               IMAGE_DIRECTORY_NAME);

        // Create the storage directory if it does not exist
        if (!mediaStorageDir.exists()) {
           if (!mediaStorageDir.mkdirs()) {
              Log.d(IMAGE_DIRECTORY_NAME, "Oops! Failed create "
                    + IMAGE_DIRECTORY_NAME + " directory");
              return null;
           }
        }

        // Create a media file name
        String timeStamp = new
             SimpleDateFormat("yyyyMMddHHmmss",Locale.getDefault()).format(new Date());
        File mediaFile;
        if (type == MEDIA_TYPE_IMAGE) {
           mediaFile = new File(mediaStorageDir.getPath() + File.separator + "IMG_" +
                timeStamp + ".jpg");
        } else if (type == MEDIA_TYPE_VIDEO) {
           mediaFile = new File(mediaStorageDir.getPath() + File.separator+ "VID_" +
                timeStamp + ".mp4");
        } else {
           return null;
        }

        return mediaFile;
    }

    /**
     * Display image from a path to ImageView
     */


}
```

A.4.2 MainActivity

```
package com.protectMSG.operation;
```

```
import java.util.ArrayList;

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.content.res.Configuration;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.support.v4.app.ActionBarDrawerToggle;
import android.support.v4.widget.DrawerLayout;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import com.protectMSG.drawer.NavDrawerItem;
import com.protectMSG.drawer.NavDrawerListAdapter;

public class MainActivity extends Activity {
  private DrawerLayout mDrawerLayout;
  private ListView mDrawerList;
  private ActionBarDrawerToggle mDrawerToggle;

  // nav drawer title
  private CharSequence mDrawerTitle;

  // used to store app title
  private CharSequence mTitle;

  // slide menu items
  private String[] navMenuTitles;
  private TypedArray navMenuIcons;

  private ArrayList<NavDrawerItem> navDrawerItems;
  private NavDrawerListAdapter adapter;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mTitle = mDrawerTitle = getTitle();
```

```
// load slide menu items
navMenuTitles = getResources().getStringArray(R.array.nav_drawer_items);

// nav drawer icons from resources
navMenuIcons = getResources()
     .obtainTypedArray(R.array.nav_drawer_icons);

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
mDrawerList = (ListView) findViewById(R.id.list_slidermenu);

navDrawerItems = new ArrayList<NavDrawerItem>();

// adding nav drawer items to array
// Home
navDrawerItems.add(new NavDrawerItem(navMenuTitles[0],
    navMenuIcons.getResourceId(0, -1)));
// Find People
navDrawerItems.add(new NavDrawerItem(navMenuTitles[1],
    navMenuIcons.getResourceId(1, -1)));
// Photos
navDrawerItems.add(new NavDrawerItem(navMenuTitles[2],
    navMenuIcons.getResourceId(2, -1)));
// Communities, Will add a counter here
navDrawerItems.add(new NavDrawerItem(navMenuTitles[3],
    navMenuIcons.getResourceId(3, -1)));
// Pages
navDrawerItems.add(new NavDrawerItem(navMenuTitles[4],
    navMenuIcons.getResourceId(4, -1)));
// Encode Audio on Image
navDrawerItems.add(new NavDrawerItem(navMenuTitles[5],
    navMenuIcons.getResourceId(5, -1)));
// Decode Audio from Image
navDrawerItems.add(new NavDrawerItem(navMenuTitles[6],
    navMenuIcons.getResourceId(6, -1)));
// Help
navDrawerItems.add(new NavDrawerItem(navMenuTitles[7],
    navMenuIcons.getResourceId(7, -1)));
// Recycle the typed array
navMenuIcons.recycle();

mDrawerList.setOnItemClickListener(new SlideMenuClickListener());

// setting the nav drawer list adapter
```

```
    adapter = new NavDrawerListAdapter(getApplicationContext(),
        navDrawerItems);
    mDrawerList.setAdapter(adapter);

    // enabling action bar app icon and behaving it as toggle button
    getActionBar().setDisplayHomeAsUpEnabled(true);
    getActionBar().setHomeButtonEnabled(true);

    mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
        R.drawable.ic_drawer, //nav menu toggle icon
        R.string.app_name, // nav drawer open - description for accessibility
        R.string.app_name // nav drawer close - description for accessibility
    ) {
      public void onDrawerClosed(View view) {
        getActionBar().setTitle(mTitle);
        // calling onPrepareOptionsMenu() to show action bar icons
        invalidateOptionsMenu();
      }

      public void onDrawerOpened(View drawerView) {
        getActionBar().setTitle(mDrawerTitle);
        // calling onPrepareOptionsMenu() to hide action bar icons
        invalidateOptionsMenu();
      }
    };
    mDrawerLayout.setDrawerListener(mDrawerToggle);

    if (savedInstanceState == null) {
      // on first time display view for first nav item
      displayView(0);
    }
  }

  /**
   * Slide menu item click listener
   * */
  private class SlideMenuClickListener implements
      ListView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position,
        long id) {
      // display view for selected nav drawer item
      displayView(position);
    }
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.main, menu);
  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // toggle nav drawer on selecting action bar app icon/title
  if (mDrawerToggle.onOptionsItemSelected(item)) {
    return true;
  }
  // Handle action bar actions click
  switch (item.getItemId()) {
  case R.id.action_settings:
    return true;
  default:
    return super.onOptionsItemSelected(item);
  }
}

/* *
 * Called when invalidateOptionsMenu() is triggered
 */
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
  // if nav drawer is opened, hide the action items
  boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerList);
  menu.findItem(R.id.action_settings).setVisible(!drawerOpen);
  return super.onPrepareOptionsMenu(menu);
}

/**
 * Diplaying fragment view for selected nav drawer list item
 * */
private void displayView(int position) {
  // update the main content by replacing fragments
  Fragment fragment = null;
  switch (position) {
  case 0:
    fragment = new HomeFragment();
    break;
```

```
    case 1:
      fragment = new EncodeTextToImage();
      break;
    case 2:
      fragment = new DecodeTextFromImage();
      break;
    case 3:
      fragment = new EncodeTextToAudio();
      break;
    case 4:
      fragment = new DecodeTextFromAudio();
      break;
    case 5:
      fragment = new EncodeAudioToImage();
      break;
    case 6:
      fragment = new DecodeAudioFromImage();
      break;
    case 7:
      fragment = new Help();
      break;
    default:
      break;
    }

    if (fragment != null) {
      FragmentManager fragmentManager = getFragmentManager();
      fragmentManager.beginTransaction()
          .replace(R.id.frame_container, fragment).commit();

      // update selected item and title, then close the drawer
      mDrawerList.setItemChecked(position, true);
      mDrawerList.setSelection(position);
      setTitle(navMenuTitles[position]);
      mDrawerLayout.closeDrawer(mDrawerList);
    } else {
      // error in creating fragment
      Log.e("MainActivity", "Error in creating fragment");
    }
  }

  @Override
  public void setTitle(CharSequence title) {
    mTitle = title;
```

```
        getActionBar().setTitle(mTitle);
    }

    /**
     * When using the ActionBarDrawerToggle, you must call it during
     * onPostCreate() and onConfigurationChanged()...
     */

    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        // Pass any configuration change to the drawer toggls
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

}
```

A.4.3 StegoEncryption

```
package com.protectMSG.stego.operation;

import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.protectMSG.exception.StegoException;

import android.annotation.SuppressLint;
```

```
import android.util.Base64;

public class StegoEncryption {

  private static final String TAG = "StegoEncryption";

    @SuppressLint("TrulyRandom")
    public static String encrypt1(String message, String password) throws Exception
    {
      String salt = "StegoSalt";
      message = salt+message;
      SecretKeySpec key = new SecretKeySpec(padPassword(password), "AES");
      Cipher c = Cipher.getInstance("AES");
      c.init(Cipher.ENCRYPT_MODE, key);
      byte[] encVal = c.doFinal(message.getBytes());
      String encrypted=Base64.encodeToString(encVal, Base64.DEFAULT);
      return encrypted;
    }

    public static String decrypt1(String message, String password) throws
        StegoException
    {
      String salt;
      Cipher c;
      String decryptedValue = null;
      try {
        c = Cipher.getInstance("AES");

      SecretKeySpec key = new SecretKeySpec(padPassword(password), "AES");
        c.init(Cipher.DECRYPT_MODE, key);
      byte[] decordedValue = Base64.decode(message.getBytes(), Base64.DEFAULT);
      byte[] decValue;
      decValue = c.doFinal(decordedValue);
      decryptedValue = new String(decValue);
      salt = decryptedValue.substring(0, 9);
      if(salt.equals("StegoSalt"))
      {
      decryptedValue = decryptedValue.substring(9);
      }else{
        // Throw error
        throw new StegoException("Wrong Passoword");
      }

      } catch (Exception e) {
```

```
            e.printStackTrace();
            throw new StegoException("Wrong Passoword");
        }
        return decryptedValue;
    }

    public static byte[] encrypt(byte[] message, String password) throws Exception
    {
        String salt = "StegoSalt";
        SecretKeySpec key = new SecretKeySpec(padPassword(password), "AES");
        Cipher c = Cipher.getInstance("AES");
        c.init(Cipher.ENCRYPT_MODE, key);

        //
        byte[] encVal1 = c.doFinal(salt.getBytes());
        String encrypted=Base64.encodeToString(encVal1, Base64.DEFAULT);
        byte[] encVal2 = encrypted.getBytes();
        System.out.println("encVal2.length " +encVal2.length);
        System.out.println("message.length " +message.length);
        byte[] result = new byte[encVal2.length + message.length];
        // copy a to result
        System.arraycopy(encVal2, 0, result, 0, encVal2.length);
        // copy b to result
        System.arraycopy(message, 0, result, encVal2.length, message.length);
//
        return result;
    }

    public static byte[] decrypt(byte[] message, String password) throws Exception
    {
        Cipher c = Cipher.getInstance("AES");
        SecretKeySpec key = new SecretKeySpec(padPassword(password), "AES");
        c.init(Cipher.DECRYPT_MODE, key);
//

        byte[] salt = Arrays.copyOfRange(message,0,25);
        byte[] saltByte = Base64.decode(salt, Base64.DEFAULT);
        byte[] decSalt = c.doFinal(saltByte);
        String saltString = new String(decSalt);
        System.out.println("saltString "+saltString);
        if(!saltString.equals("StegoSalt")){
          throw new StegoException("Wrong Passoword");
        }
```

```
        byte[] messageByte = Arrays.copyOfRange(message,25,message.length);
//
        return messageByte;
      }
      private static byte[] padPassword(String password){
        int keyLength = 128;
        byte[] keyBytes = new byte[keyLength / 8];
        // explicitly fill with zeros
        Arrays.fill(keyBytes, (byte) 0x0);

        // if password is shorter then key length, it will be zero-padded
        // to key length
        byte[] passwordBytes;
        try {
          passwordBytes = password.getBytes("UTF-8");

        int length = passwordBytes.length < keyBytes.length ? passwordBytes.length :
            keyBytes.length;
        System.arraycopy(passwordBytes, 0, keyBytes, 0, length);

        } catch (UnsupportedEncodingException e) {
          e.printStackTrace();
        }
        return keyBytes;
      }
   }
```

A.4.4 StegoOperation

```
package com.protectMSG.stego.operation;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import android.graphics.Bitmap;
import android.os.Environment;
import android.util.Log;
```

```
public class StegoOperation {

  private static final String TAG = "StegoOperation";

  public byte[] encode(byte bytes[], String encode_text, boolean isPasswordLocked)
  {
    Log.i(TAG,"encode()");
    byte img[] = bytes;
    byte msg[] = encode_text.getBytes();
    byte len[] = bitConversion(msg.length);
    System.out.println("len.length"+len.length);
    byte header[];
    byte[] encoded_image = null;
    if(isPasswordLocked){
      byte byte1 =(byte) 0xFF;
      header = new byte[] {byte1,byte1,byte1,byte1};
    }
    else{
      byte byte1 =(byte) 0x00;
      header = new byte[] {byte1,byte1,byte1,byte1};
    }
    try
    {
      encoded_image=encodeText(img, len, 0);
      encoded_image=encodeText(img, header, 32);
      encoded_image=encodeText(img, msg, 64); //4 bytes of space for length:
          4bytes*8bit = 32 bits
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    return encoded_image;
  }

  public byte[] encodeAudio(byte bytes[], String encode_text, boolean
      isPasswordLocked)
  {
    Log.i(TAG,"encode()");
    byte img[] = bytes;
    byte msg[] = encode_text.getBytes();
    byte len[] = bitConversion(msg.length);
    System.out.println("len.length"+len.length);
    byte header[];
```

```
    byte[] encoded_image = null;
    if(isPasswordLocked){
      byte byte1 =(byte) 0xFF;
      header = new byte[] {byte1,byte1,byte1,byte1};
    }
    else{
      byte byte1 =(byte) 0x00;
      header = new byte[] {byte1,byte1,byte1,byte1};
    }
    try
    {
      encoded_image=encodeText(img, len, 44);
      encoded_image=encodeText(img, header, 76);
      encoded_image=encodeText(img, msg, 108); //4 bytes of space for length:
          4bytes*8bit = 32 bits
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    return encoded_image;
}

//added by me
public byte[] encode(byte bytes[], byte encodebyte[], boolean isPasswordLocked)
{
    Log.i(TAG,"encode()");
    byte img[] = bytes;
    byte msg[] = encodebyte;
    byte len[] = bitConversion(msg.length);
    byte[] encoded_image = null;
    byte header[];
    if(isPasswordLocked){
      byte byte1 =(byte) 0xFF;
      header = new byte[] {byte1,byte1,byte1,byte1};
    }
    else{
      byte byte1 =(byte) 0x00;
      header = new byte[] {byte1,byte1,byte1,byte1};
    }
    try
    {
      encoded_image=encodeText(img, len, 0);
      encoded_image=encodeText(img, header, 32);
```

```
        encoded_image=encodeText(img, msg, 64); //4 bytes of space for length:
            4bytes*8bit = 32 bits
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    return encoded_image;
}

public String decode(byte bytes[])
{
    Log.i(TAG,"decode()");
    byte[] decode;
    try
    {
      decode = decodeText(bytes);
      return(new String(decode));
    }
    catch(Exception e)
    {
      e.printStackTrace();
      return "";
    }
}

public String decodeAudio(byte bytes[])
{
    Log.i(TAG,"decode()");
    byte[] decode;
    try
    {
      decode = decodeTextFromAudio(bytes);
      return(new String(decode));
    }
    catch(Exception e)
    {
      e.printStackTrace();
      return "";
    }
}

public byte[] decodeBytes(byte bytes[])
{
```

```
    Log.i(TAG,"decode()");
    byte[] decode = null;
    try
    {
      decode = decodeText(bytes);
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }
    return decode;
  }

  private byte[] bitConversion(int i)
  {
    Log.i(TAG,"bit_conversion()");
    byte byte3 = (byte)((i & 0xFF000000) >>> 24); //0
    byte byte2 = (byte)((i & 0x00FF0000) >>> 16); //0
    byte byte1 = (byte)((i & 0x0000FF00) >>> 8 ); //0
    byte byte0 = (byte)((i & 0x000000FF) );
    return(new byte[]{byte3,byte2,byte1,byte0});
  }

  private byte[] encodeText(byte[] image, byte[] addition, int offset)
  {
    Log.i(TAG,"encode_text()");
    if(addition.length + offset > image.length)
    {
      Log.e(TAG,"File not long enough!");
      throw new IllegalArgumentException("File not long enough!");
    }
    //loop through each addition byte
    for(int i=0; i<addition.length; ++i)
    {
      //loop through the 8 bits of each byte
      int add = addition[i];
      for(int bit=7; bit>=0; --bit, ++offset)
      {
        int b = (add >>> bit) & 1;
        image[offset] = (byte)((image[offset] & 0xFE) | b );
      }
    }
    System.out.println("Final image byte lenght"+image.length);
    return image;
```

```
}

private byte[] decodeText(byte[] image)
{
  Log.i(TAG,"decode_text()");
  int length = 0;
  int offset = 64;
  for(int i=0; i<32; ++i)
  {
    length = (length << 1) | (image[i] & 1);
  }
  byte[] result = new byte[length];
  //loop through each byte of text
  for(int b=0; b<result.length; ++b )
  {
    //loop through each bit within a byte of text
    for(int i=0; i<8; ++i, ++offset)
    {
      result[b] = (byte)((result[b] << 1) | (image[offset] & 1));
    }
  }
  return result;
}

private byte[] decodeTextFromAudio(byte[] image)
{
  Log.i(TAG,"decode_text()");
  int length = 0;
  int offset = 108;
  //loop through 32 bytes of data to determine text length
  for(int i=44; i<76; ++i)
  {
    length = (length << 1) | (image[i] & 1);
  }
  byte[] result = new byte[length];
  //loop through each byte of text
  for(int b=0; b<result.length; ++b )
  {
    //loop through each bit within a byte of text
    for(int i=0; i<8; ++i, ++offset)
    {
      result[b] = (byte)((result[b] << 1) | (image[offset] & 1));
    }
  }
```

```
    return result;
}

public int decodeAudioLen(byte[] image)
{
    Log.i(TAG,"decode_text()");
    int length = 0;
    for(int i=0; i<32; ++i)
    {
      length = (length << 1) | (image[i] & 1);
    }
    return length;
}
public static Bitmap convertToMutable(Bitmap imgIn) {
    Log.i(TAG,"convertToMutable()");
     try {
        File file = new File(Environment.getExternalStorageDirectory() +
            File.separator + "Temp.jpg");

        RandomAccessFile randomAccessFile = new RandomAccessFile(file, "rw");

        // get the width and height of the source bitmap.
        int width = imgIn.getWidth();
        int height = imgIn.getHeight();
        Bitmap.Config type = imgIn.getConfig();

        FileChannel channel = randomAccessFile.getChannel();
        MappedByteBuffer map = channel.map(FileChannel.MapMode.READ_WRITE, 0,
            imgIn.getRowBytes()*height);
        imgIn.copyPixelsToBuffer(map);
        imgIn.recycle();
        System.gc();// try to force the bytes from the imgIn to be released
        imgIn = Bitmap.createBitmap(width, height, type);
        map.position(0);
        //load it back from temporary
        imgIn.copyPixelsFromBuffer(map);
        //close the temporary file and channel , then delete that also
        channel.close();
        randomAccessFile.close();

        // delete the temp file
        file.delete();

     } catch (FileNotFoundException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return imgIn;
    }
}
```

A.4.5 AudioRecording

```
package com.protectMSG.util;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import android.os.Environment;
import android.util.Log;

public class AudioRecording {

  private static final String TAG = "AudioRecording";

  public static String getFilename(){
      String filepath = Environment.getExternalStorageDirectory().getPath();
      File file = new File(filepath,Constant.AUDIO_RECORDER_FOLDER);
      if(!file.exists()){
            file.mkdirs();
      }
      return (file.getAbsolutePath() + "/" + System.currentTimeMillis() +
          Constant.AUDIO_RECORDER_FILE_EXT_WAV);
  }

  public static String getTempFilename(){
      String filepath = Environment.getExternalStorageDirectory().getPath();
      File file = new File(filepath,Constant.AUDIO_RECORDER_FOLDER);
      if(!file.exists()){
            file.mkdirs();
      }
```

```
    File tempFile = new File(filepath,Constant.AUDIO_RECORDER_TEMP_FILE);
    if(tempFile.exists())
          tempFile.delete();

    return (file.getAbsolutePath() + "/" + Constant.AUDIO_RECORDER_TEMP_FILE);
}

public static void deleteTempFile() {
  Log.i(TAG, "deleteTempFile()");

  System.out.println("deleteTempFile >> ");
  File file = new File(getTempFilename());

  file.delete();
}

public static void copyWaveFile(String inFilename, String outFilename) {
  Log.i(TAG, "copyWaveFile()");
  System.out.println("copyWaveFile >> ");
  FileInputStream in = null;
  FileOutputStream out = null;
  long totalAudioLen = 0;
  long totalDataLen = totalAudioLen + 36;
  long longSampleRate = Constant.RECORDER_SAMPLERATE;
  int channels = 2;
  long byteRate = Constant.RECORDER_BPP * Constant.RECORDER_SAMPLERATE * channels
      / 8;
  byte[] data = new byte[Constant.bufferSize];
  try {
    in = new FileInputStream(inFilename);
    out = new FileOutputStream(outFilename);
    totalAudioLen = in.getChannel().size();
    totalDataLen = totalAudioLen + 36;
    WriteWaveFileHeader(out, totalAudioLen, totalDataLen,longSampleRate,
        channels, byteRate);
    while (in.read(data) != -1) {
      out.write(data);
    }
    in.close();
    out.close();
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
```

```
    }
  }

  public static void WriteWaveFileHeader(FileOutputStream out, long totalAudioLen,
      long totalDataLen, long longSampleRate, int channels, long byteRate)
      throws IOException {
    Log.i(TAG, "WriteWaveFileHeader()");
    byte[] header = new byte[44];
    header[0] = 'R'; // RIFF/WAVE header
    header[1] = 'I';
    header[2] = 'F';
    header[3] = 'F';
    header[4] = (byte) (totalDataLen & 0xff);// Problem here - diff
    header[5] = (byte) ((totalDataLen >> 8) & 0xff);
    header[6] = (byte) ((totalDataLen >> 16) & 0xff);
    header[7] = (byte) ((totalDataLen >> 24) & 0xff);
    header[8] = 'W';
    header[9] = 'A';
    header[10] = 'V';
    header[11] = 'E';
    header[12] = 'f'; // 'fmt ' chunk
    header[13] = 'm';
    header[14] = 't';
    header[15] = ' ';
    header[16] = 16; // 4 bytes: size of 'fmt ' chunk
    header[17] = 0;
    header[18] = 0;
    header[19] = 0;
    header[20] = 1; // format = 1
    header[21] = 0;
    header[22] = (byte) channels;
    header[23] = 0;
    header[24] = (byte) (longSampleRate & 0xff);
    header[25] = (byte) ((longSampleRate >> 8) & 0xff);
    header[26] = (byte) ((longSampleRate >> 16) & 0xff);
    header[27] = (byte) ((longSampleRate >> 24) & 0xff);
    header[28] = (byte) (byteRate & 0xff);
    header[29] = (byte) ((byteRate >> 8) & 0xff);
    header[30] = (byte) ((byteRate >> 16) & 0xff);
    header[31] = (byte) ((byteRate >> 24) & 0xff);
    header[32] = (byte) (2 * 16 / 8); // block align
    header[33] = 0;
    header[34] = Constant.RECORDER_BPP; // bits per sample
    header[35] = 0;
```

```
    header[36] = 'd';
    header[37] = 'a';
    header[38] = 't';
    header[39] = 'a';
    header[40] = (byte) (totalAudioLen & 0xff);// Problem here- diff
    header[41] = (byte) ((totalAudioLen >> 8) & 0xff);
    header[42] = (byte) ((totalAudioLen >> 16) & 0xff);
    header[43] = (byte) ((totalAudioLen >> 24) & 0xff);
    out.write(header, 0, 44);
  }
}
```

A.4.6 Utility

```
package com.protectMSG.util;


import java.io.File;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Environment;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.protectMSG.operation.R;

public class Utility {

  private static final String TAG = "Utility";
  static String AUDIO_RECORDER_FOLDER="";

  public static String getImageSDLocation() {
    Log.i(TAG, "getImageSDLocation");
    String filepath = Environment.getExternalStorageDirectory().getPath();
```

```
    System.out.println("filepath >> "+filepath);
    File file = new File(filepath, Constant.MAIN_FOLDER);
    if (!file.exists()) {
      file.mkdirs();
    }
    File file1 = new File(filepath+"/"+Constant.MAIN_FOLDER,Constant.IMAGE_FOLDER);
    if (!file1.exists()) {
      file1.mkdirs();
    }
    return Environment.getExternalStorageDirectory()+"/protectMSG/Image";
  }

  public static String getAudioSDLocation() {
    Log.i(TAG, "getAudioSDLocation");
    String filepath = Environment.getExternalStorageDirectory().getPath();
    File file = new File(filepath, Constant.MAIN_FOLDER);
    if (!file.exists()) {
      file.mkdirs();
    }
    File file1 = new File(filepath+"/"+Constant.MAIN_FOLDER,Constant.AUDIO_FOLDER);
    if (!file1.exists()) {
      file1.mkdirs();
    }
    return Environment.getExternalStorageDirectory()+"/protectMSG/Audio";
  }

  public static String generateImageFileName() {
    Log.i(TAG, "generateImageFileName()");
    return (getImageSDLocation() + "/" + "Image_" + System.currentTimeMillis() +
        Constant.PNG_FILE_EXT_WAV);
  }

  public static String generateAudioFileName() {
    Log.i(TAG, "generateAudioFileName()");
    return (getAudioSDLocation() + "/" + "Audio_" + System.currentTimeMillis() +
        Constant.AUDIO_RECORDER_FILE_EXT_WAV);
  }
  public static String generateResultAudioFileName() {
    Log.i(TAG, "generateAudioFileName()");
    return (getAudioSDLocation() + "/" + "Audio_Message_" +
        System.currentTimeMillis() + Constant.AUDIO_RECORDER_FILE_EXT_WAV);
  }
  static public boolean resetExternalStorageMedia(Context context) {
     if (Environment.isExternalStorageEmulated())
```

```
            return (false);
        Uri uri = Uri.parse("file://" + Environment.getExternalStorageDirectory());
        Intent intent = new Intent(Intent.ACTION_MEDIA_MOUNTED, uri);
        context.sendBroadcast(intent);
        return (true);
    }

    public static void showCustomAlert(String text,Activity activity,String errorType)
     {

         Context context = activity.getApplicationContext();
         // Create layout inflator object to inflate toast.xml file
         LayoutInflater inflater = activity.getLayoutInflater();
         // Call toast.xml file for toast layout
         View toastRoot = inflater.inflate(R.layout.toast, null);
       TextView text1 =(TextView) toastRoot.findViewById(R.id.text);
       text1.setText(text);
       if(errorType.equalsIgnoreCase("info")){
          text1.setTextColor(Color.parseColor("#7BD4A8"));
       }else if(errorType.equalsIgnoreCase("warning")){
          text1.setTextColor(Color.parseColor("#FFC800"));
       }if(errorType.equalsIgnoreCase("error")){
          text1.setTextColor(Color.parseColor("#DF151A"));
       }
       text1.setTypeface(Customfontloader.getTypeface(activity,1));
         Toast toast = new Toast(context);
         // Set layout to toast
         toast.setView(toastRoot);
         toast.setGravity(Gravity.BOTTOM,0, 0);
         toast.setDuration(Toast.LENGTH_LONG);
         toast.show();

     }
}
```

A.4.7 Validate

```
package com.protectMSG.operation;

import java.io.File;

import com.protectMSG.util.Utility;
```

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.util.Log;

public class Validate {
  private final static String TAG = "Validate";
  public static boolean validatePassword(String password) {
    if(password.length()>0){
      if (password.length() < 3 || password.length() > 15) {
        return true;
      } else {
        return false;
      }
    }else{
      return false;
    }
  }

  public static boolean validateEncodeText(String encodeText) {
    if (encodeText.length() < 1 || encodeText.length() > 50) {
      return true;
    } else {
      return false;
    }
  }

  public static boolean validateFileSize(String fileType, String fileName,Activity
      activity) {
    boolean isValid = false;
    if (fileType.equalsIgnoreCase("Audio")) {
      // TODO
    } else if (fileType.equalsIgnoreCase("Image")) {
      // TODO
      File file = new File(fileName);
      System.out.println("File size"+file.length());
      Utility.showCustomAlert("File size"+file.length(),activity,"error");
      if(file.length()>2621440){
        isValid = true;
      }
    }
    return isValid;
  }
```

```
  @SuppressLint("DefaultLocale")
  public static boolean validateFileExtension(String fileName,String fileType) {

    if (fileType.equalsIgnoreCase("Audio")) {
      // TODO
      if(fileName.toLowerCase().endsWith(".wav")){
          return false;
       }
       else{
         System.out.println("Browsed dest file extension must be .txt");
         return true;
       }
    } else if (fileType.equalsIgnoreCase("Image")) {
      // TODO
      if(fileName.toLowerCase().endsWith(".jpg")
          ||fileName.toLowerCase().endsWith(".jpeg")
          ||fileName.toLowerCase().endsWith(".png")){
          return false;
       }
       else{
         System.out.println("Browsed dest file extension must be .txt");
         return true;
       }
    }
    return false;
  }

  public static String validateFile(byte[] image)
  {
    Log.i(TAG,"validateFile()");
    int length = 0;
//   int offset = 32;
    int offset = 64;
    byte byteZero=(byte) 0x00;
    byte byteOnes=(byte) 0xFF;
    int byteZeroCount = 0;
    int byteOnesCount = 0;
    String result="NotEncoded";

    for(int i=32; i<36; ++i)
    {
      length = (length << 1) | (image[i] & 1);

    }
```

```
    if(length == 0){
      result="EncodedWithoutPassword";
    }else if(length == 15){
      result="EncodedWithPassword";
    }else{
      result="NotEncoded";
    }

    return result;
  }
  public static String validateAudioFile(byte[] image)
  {
    Log.i(TAG,"validateFile()");
    int length = 0;
    String result="NotEncoded";

    for(int i=76; i<80; ++i)
    {
      length = (length << 1) | (image[i] & 1);
    }

    if(length == 0){
      result="EncodedWithoutPassword";
    }else if(length == 15){
      result="EncodedWithPassword";
    }else{
      result="NotEncoded";
    }

    return result;
  }
}
```

A.4.8 Encode Audio To Image XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:background="#2E2E2E" >
```

```
<LinearLayout
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#2E2E2E"
   >

<TextView
   android:id="@+id/headerTextView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
   android:layout_marginLeft="@dimen/activity_horizontal_margin"
   android:fontFamily=""
   android:gravity="center_horizontal"
   android:paddingBottom="22dp"
   android:paddingTop="22dp"
   android:shadowColor="@color/list_background"
   android:text="ENCODE QUICK VOICE MESSAGE TO IMAGE"
   android:textColor="@android:color/holo_blue_dark"
   android:textSize="25sp"
   android:textStyle="italic" />

<Space
   android:layout_width="0dp"
   android:layout_weight="1"
   android:layout_height="20dp" />
<TextView
     android:id="@+id/shareError"
      android:layout_width="153dp"
      android:layout_height="wrap_content"
      android:gravity="center_vertical"
      android:text="Share Audio"
      android:visibility="invisible"
      android:textColor="@android:color/holo_blue_light"
      android:textSize="22sp"
      android:textStyle="italic"
      />
 <Button
      android:id="@+id/shareButton"
      style="?android:attr/buttonStyleSmall"
      android:layout_width="43dp"
```

```
        android:layout_height="43dp"
        android:background="@drawable/sharebutton" />
    </LinearLayout>

    <TextView
       android:id="@+id/enterPasswordTextView"
       android:layout_width="match_parent"
       android:layout_height="40dp"
       android:gravity="center_vertical"
       android:text="Enter Password"
       android:textColor="@android:color/holo_blue_light"
       android:textSize="22sp"
       android:layout_marginLeft="@dimen/activity_horizontal_margin"/>
   <LinearLayout
         android:orientation="horizontal"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:background="#2E2E2E"
         >
      <EditText
         android:id="@+id/passwordEditView"
         android:layout_width="260dp"
         android:layout_height="wrap_content"
         android:ems="10"
         android:hint="**************"
         android:textColor="@android:color/holo_green_light"
         android:inputType="textPassword"
         android:layout_marginLeft="@dimen/activity_horizontal_margin" />
     <TextView
         android:id="@+id/errorPasswordTextView"
         android:layout_width="match_parent"
         android:layout_height="40dp"
         android:gravity="center_vertical"
         android:visibility="invisible"
         android:text="Enter valid password"
         android:textColor="@android:color/holo_red_light"
         android:textSize="22sp"
         android:layout_marginLeft="5dp"/>
   </LinearLayout>
   <LinearLayout
       android:id="@+id/recordAudioLayout"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:visibility="visible"
```

```
    android:paddingTop="5dp"
    android:layout_marginLeft="@dimen/activity_horizontal_margin">
  <TextView
     android:id="@+id/recordAudioTextView"
     android:layout_width="200dp"
     android:layout_height="40dp"
     android:gravity="center_vertical"
     android:text="Record Audio"
     android:textColor="@android:color/holo_blue_light"
     android:textSize="22sp" />
    <Button
       android:id="@+id/buttonRecord"
       android:layout_width="96dp"
       android:layout_height="96dp"
       android:background="@drawable/recordaudiobackground"
       android:text="Record" />
</LinearLayout>

<LinearLayout
    android:id="@+id/audioFileLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="invisible"
    android:paddingTop="5dp"
    android:layout_marginLeft="@dimen/activity_horizontal_margin">

    <TextView
       android:id="@+id/audioFileTextView"
       android:layout_width="200dp"
       android:layout_height="40dp"
       android:layout_gravity="left"
       android:gravity="left|center_vertical"
       android:text="Audio File Selected "
       android:textColor="@android:color/holo_blue_light"
       android:textSize="22sp" />

     <ImageView
       android:id="@+id/audioImageView"
       android:contentDescription="Audio File"
       android:layout_width="40dp"
       android:layout_height="40dp"
       android:layout_gravity="left"
       android:paddingLeft="50dp"
       android:background="@drawable/audiofile"
```

```
        />

    <TextView
        android:id="@+id/audioTextView"
        android:layout_width="wrap_content"
        android:layout_height="40dp"
        android:layout_marginRight="10dp"
        android:layout_gravity="left"
        android:gravity="left|center_vertical"
        android:paddingLeft="2dp"
        android:text=""
        android:textColor="@android:color/holo_green_light"
        android:textSize="22sp" />

    <ImageView
        android:id="@+id/playImageView"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_gravity="left"
        android:paddingLeft="5dp"
        android:background="@drawable/playerplay"
        />

</LinearLayout>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/activity_horizontal_margin" >
 <TextView
      android:id="@+id/browseAudioTextView"
       android:layout_width="153dp"
       android:layout_height="60dp"
       android:gravity="center_vertical"
       android:text="Browse Image"
       android:textColor="@android:color/holo_blue_light"
       android:textSize="22sp"
       android:textStyle="italic"
       />
   <Button
       android:id="@+id/browseButton"
       android:layout_width="50dp"
       android:layout_height="52dp"
       android:background="@drawable/folderbutton" />
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/activity_horizontal_margin" >

    <TextView
       android:id="@+id/capturePictureTextView"
       android:layout_width="153dp"
       android:layout_height="60dp"
       android:gravity="center_vertical"
       android:text="Capture Picture"
       android:textColor="@android:color/holo_blue_light"
       android:textSize="22sp"
       android:textStyle="italic"/>

    <Button
       android:id="@+id/captureButton"
       android:layout_width="50dp"
       android:layout_height="52dp"
       android:background="@drawable/camerabutton" />
 </LinearLayout>
 <ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="0dip"
    android:layout_weight="0.34"
    android:layout_marginLeft="@dimen/activity_horizontal_margin"/>

 <TextView
    android:id="@+id/errorText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:visibility="invisible"
    android:text="error"
    android:textColor="@android:color/holo_red_light" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dip"
    android:layout_weight="1"
    android:gravity="bottom"
    android:orientation="vertical" >
 <Button
    android:id="@+id/encodeAudioButton"
    android:layout_width="260dp"
```

```
        android:layout_height="wrap_content"
        android:background="@drawable/button"
        android:textColor="@android:color/white"
        android:text="Encode Data"
        android:textSize="23sp"
        android:layout_marginLeft="@dimen/activity_horizontal_margin"/>
    </LinearLayout>

</LinearLayout>
```

Appendix B

Glossary

2D	Two Dimension
3DES	Triple Data Encryption Standard
ABI	Application Binary Interface
ADB	Android Debug Bridge
ADT	Android Developer Tools
AES	Advanced Encryption Standard
AIFF	Audio Interchange File Format
API	Application Programming Interface
AVD	Android Virtual Device
BMP	Bitmap
CPU	Central Processing Unit
DDMS	Dalvik Debug Monitor Server
DES	Data Encryption Standard

DESX	A variant of Data Encryption Standard
DVM	Dalvik Virtual Machine
FAT32	File Allocation Table 32
GIF	Graphics Interchange Format
GUI	Graphical User Interface
IDE	Integrated Development Environment
JAVA EE	Java Platform, Enterprise Edition
JDK	Java Development Kit
JEPG	Joint Photographic Experts Group
LRU	Least Recently Used
LSB	Least Significant Bit
MP	Megapixel
NIST	National Institute of Standards and Technology
PCM	Pulse Code Modulation (
PDU	Protocol Data Unit
PNG	Portable Network Graphics
RAM	Random Access Memory
SD Card	Secure Digital Card
SDK	Software Development Kit
SPN	Substitution-Permutation Network
SSL	Secure Sockets Layer

TIFF	Tagged Image File Format
UI	User Interface
WAV	Waveform Audio File Format
XML	Extensible Markup Language

Bibliography

[1] Silvia Torres-Maya, Mariko Nakano-Miyatake and Héctor Perez-Meana, *An Image Steganography Systems Based on BPCS and IWT*, In Proceedings of the 16th IEEE International Conference on Electronics, Communications and Computers (CONIELECOMP 2006)

[2] Moazzam Hossain, Sadia Al Haque, and Farhana Sharmin, *Variable Rate Steganography in Gray Scale Digital Images Using Neighborhood Pixel Information* in The International Arab Journal of Information Technology, Vol.7, No.1, January 2010

[3] Elżbieta Zielińska, Wojciech Mazurczyk, Krzysztof Szczypiorski, *Trends in Steganography* in Communications of the ACM, Vol. 57 No. 3, Pages 86-95

[4] Gary C. KesslerAn, *Overview of Steganography for the Computer Forensics Examiner* July 2004, Forensic Science Communications.

[5] *Smartphones vs. Digital Cameras*, 24 July 2013, LOS ALTOS, California http://www.ireachcontent.com

[6] The Android SDK Installation https://developer.android.com/sdk/installing/index.html

[7] C. Yi-zhen, et al., *An adaptive steganography algorithm based on block sensitivity vectors using HVS features* in Image and Signal Processing (CISP), 3rd International Congress on, 2010, pp. 1151-1155, ISBN: 978-1-4244-6513-2.

[8] Digital Image File Types http://users.wfu.edu/matthews/misc/graphics/formats/formats.html

[9] Android Navigation Drawer http://developer.android.com/training/implementing-navigation/nav-drawer.html

[10] Evolution of Steganography http://stegano.net

[11] Monkey Tool http://developer.android.com/tools/help/monkey.html

[12] Suite 48 Analytics, a report on 24 July 2013 http://suite48a.com

[13] The Wall Street Journal, a statistic, http://liesdamnedliesstatistics.com/2013/09/the-digital-camera-market-continues-to-drop-thanks-to-smartphones.html

[14] Michael Burton and Donn Felker,*Android application development for dummies, 2nd edition* 23 October 2012, ISBN-10: 1118387104 ISBN-13: 978-1118387108

[15] *Google I/O Conference 2014*

[16] Info on Android http://developer.android.com/about/index.html

[17] Mobile Technology Fact Sheet http://www.pewinternet.org

[18] Simon Khalaf, *The Rise of the Mobile Addict*, 22 April 2014 http://www.flurry.com

[19] Christopher Ratcliff, *65% of global smartphone owners use Android OS: stats* 21 February 2014, https://econsultancy.com

[20] Maeve Duggan, *Photo and Video Sharing Grow Online*, 28 October 2013, http://www.pewinternet.org

[21] Mobile Statistics, http://www.mobilestatistics.com

[22] C. Vanmathi, S. Prabu, *A Survey of State of the Art techniques of Steganography* in International Journal of Engineering and Technology, Feb-Mar 2013, Vol 5,ISSN : 0975-4024

[23] Chandramouli, *R Analysis of LSB based image steganography techniques*,pages 1019 - 1022 vol.3 in Image Processing, 2001. Proceedings. 2001 International Conference on (Volume:3) ISBN:0-7803-6725-1

[24] Tutorials Point, http://www.tutorialspoint.com/

[25] Tech Target, http://searchsoftwarequality.techtarget.com/

[26] Android SDK, https://developer.android.com/sdk/index.html

[27] Simple Developer, http://simpledeveloper.com