

Minor Research Project
on
IMAGE STEGANOGRAPHY

Final Report
Submitted to

P P Savani University
Surat

Submitted by
Keyur Finaviya (17SE02CE015)
Dhruv Gandhi (17SE02CE017)
Mansi Hirpara (17SE02CE020)
Computer Engineering



School of Engineering
P P Savani University
May, 2019-20



P P SAVANI
UNIVERSITY

School of
Engineering

CERTIFICATE

This is to certify that Mr. Keyur Finaviya, Enrollment No. 17SE02CE015 from the Department of Computer Engineering has successfully completed the Minor Project on the Image Steganography during Dec, 2019-May, 2020.

Date:

Name and Sign of Supervisor

Dean, SOE



P P SAVANI
UNIVERSITY

School of
Engineering

CERTIFICATE

This is to certify that Mr. Dhruv Gandhi, Enrollment No. 17SE02CE017 from the Department of Computer Engineering has successfully completed the Minor Project on the Image Steganography during Dec, 2019-May, 2020.

Date:

Name and Sign of Supervisor

Dean, SOE



P P SAVANI
UNIVERSITY

School of
Engineering

CERTIFICATE

This is to certify that Ms. Mansi Hirpara, Enrollment No. 17SE02CE020 from the Department of Computer Engineering has successfully completed the Minor Project on the Image Steganography during Dec, 2019-May, 2020.

Date:

Name and Sign of Supervisor

Dean, SOE

ACKNOWLEDGEMENT

It is indeed with a great pleasure and immense sense of gratitude that we acknowledge the help of these individuals. We are highly indebted to our Dean **Dr. Niraj Shah**, Dean, School of Engineering, P P Savani University, for the facilities provided to accomplish this minor project.

We feel elated in manifesting our sense of gratitude to our project guides Ms. Bannishikha Banerjee, Mr. Ravirajsinh Chauhan, and Mr. Mithilesh Yadav. They have been a constant source of inspiration for us and we are very deeply thankful to them for their support and valuable advice.

We are extremely grateful to our Departmental staff members, Lab technicians and Non-teaching staff members for their extreme help throughout our project.

Finally we express our thanks to all of our friends who helped us in successful completion of this project.

Student Names and Enrollment No.

Keyur Finaviya (17SE02CE015)

Dhruv Gandhi (17SE02CE017)

Mansi Hirpara (17SE02CE020)

ABSTRACT

Steganography is the science that involves communicating secret data in an appropriate multimedia carrier, e.g., image, audio, and video files. It comes under the assumption that if the feature is visible, the point of attack is evident, thus the goal here is always to conceal the very existence of the embedded data. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the Internet. For hiding secret information in images, there exists a large variety of steganography techniques some are more complex than others and all of them have respective strong and weak points. Different applications have different requirements of the steganography technique used. For example, some applications may require absolute invisibility of the secret information, while others require a larger secret message to be hidden.

This project helps in finding the source of widespread of fake news via images. What it does is stores the identity of the user in the image he/she sends. Now that image becomes stego image and if it is found that the news in that image is fake, the person who started it can be easily traced back with the help of the information stored in the image.

INDEX

1. INTRODUCTION	1
1.1 Introduction	1
1.2 Problem introduction	1
1.3 Motivation and Objective	2
2. LITERATURE SURVEY	3
2.1 Introduction	3
2.2 LSB Technique	3
2.3 Steganography using DCT and DWT	3
2.4 Pixel-Value Differencing	4
2.5 7 th Bit Pixel	5
2.6 Color Image Clustering Technique	5
3. REQUIREMENTS SPECIFICATION	6
3.1 Introduction	6
3.2 Hardware requirements	6
3.3 Software requirements	6
4. ANALYSIS	7
4.1 Existing System	7
4.2 Proposed System	7
5. DESIGN	8
5.1 UML Diagram	8
5.2 Data Flow Diagram	9
6. SYSTEM IMPLEMENTATION	10
6.1 Sample code	12
7. SAMPLE SCREENSHOTS	19
7.1 Encryption App	19
7.2 Decryption App	20
7.3 Comparison of Cover Image and Stego Image	22
7.4 Chat Application	23

8. CONCLUSION AND FUTURE SCOPE	27
REFERNCES	28

List of Figures/Tables

Sr. No.	Figure Name	Page No.
5.1.1	UML Diagram	8
5.2.1	Data Flow Diagram	9
7.1.1	Home Page	19
7.1.2	Encryption Page	19
7.1.3	Selecting Image	19
7.1.4	Encryption Page with Image	19
7.1.5	Encryption Completed	20
7.2.1	Home Page	20
7.2.2	Decryption Page	20
7.2.3	Select the Image	21
7.2.4	Selected Image	21
7.2.5	Decryption Page with Secret key	21
7.3.1	Cover Image	22
7.3.2	Stego Image	22
7.4.1	Login Page	23
7.4.2	Sign Up	23
7.4.3	OTP Verification	23
7.4.4	Logging in	23
7.4.5	About the App	24
7.4.6	Account Settings	24
7.4.7	Recent Chat	24
7.4.8	Sending Image	24
7.4.9	Selecting Image	25
7.4.10	Find Friends	25
7.4.11	Friend Requests	25
7.4.12	Contacts	25
7.4.13	Creating Group	26
7.4.14	Group Chat	26

CHAPTER 1

INTRODUCTION

1.1 Introduction

Internet has emerged as the most convenient and efficient medium for communication. Through internet, messages can be transferred in a fast and cheap way in various fields like government offices, private sector, military, and medical areas [1]. Many times, confidentiality of the transferred message needs to be maintained. To ensure that the message is transferred securely and safely over the network, a suitable method is needed. Steganography proves as a trustable method for achieving this aim. In steganography, the data are hidden in the cover media. The cover medium can be in the form of image file, text file, video file, or audio file. Steganography is defined as a science or art of hiding the message inside some cover medium [2, 3]. The word steganography is built up of two words of ancient Greek origin “steganos” meaning “covered, concealed, or protected” and “graphie” meaning “writing.” The concept of steganography is not new; its usage can be seen in the past also. Historical records depict that around 440 BC, Herodotus sent secret messages using the concept of steganography. In ancient times, Greeks also wrote messages on wood and covered them with wax. The concept of invisible ink was also used during the period of World War II. According to Greek history, secret messages were written on the bald scalp of the slaves, and after the growth of hair on their heads, they were sent as messengers. [1]

The most popular medium used is image files because of their high capacity and easy availability over the internet [4]. At the sender’s side, the image used for embedding the secret message is called cover image, and the secret information that needs to be protected is called a message. As soon as data are embedded using some appropriate embedding algorithm, then it is called stego image. This stego image is transferred to the receiver, and he extracts out the secret message using extraction algorithm [5]. Another data hiding technique, cryptography is also used for secure transmission of messages over the internet, but steganography is becoming more popular because of its advantages over cryptography. Cryptography hides the exact meaning of message from the third party whereas steganography hides the very existence of the message itself. [1]

1.2 Problem Introduction

The contemporary world has become entirely based on internet. Communication has become much easier with the growth of internet and as it has become the most prominent medium for transferring information, it has become much quicker and efficient to send a message to anyone across the globe.

With the help of emerging chat applications like Whatsapp, Instagram, Snapchat, etc. passing of information couldn’t have been any easier. But this technology is being misused for spreading fake information and people believe it easily without any

authentication. This could be a huge problem in case of a crisis so it is necessary to stop the widespread of fake news and it becomes really difficult to find out who started it.

1.3 Motivation and Objective

The main objective of this project is to help find the source of fake news and hence, help in reducing its widespread. Since the texting applications are main mode of communication in the modern world, any kind of information can go viral instantly. So it becomes necessary that only credible and authenticated information is passed on.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

In this chapter, a study on various steganography techniques are described. Many existing algorithms for image steganography from 2011 to 2019 are studied and presented in this literature survey. Number of websites, research papers, project reports as well as books are referred for understanding of the topic in detail and also to know about the various existing algorithms that have been modified and the new algorithms that have been designed.

2.2 LSB Technique

This technique is the base of image steganography. It is the most basic algorithm. In a gray scale image each pixel is represented in 8 bits. Similarly for a colour (RGB-red, green, blue) image, each pixel requires 24 bits (8bits for each layer). [2] The last bit in a pixel is called as Least Significant bit (LSB) as its value will affect the pixel value only by “1”. So, this property is used to hide the data in the image and helps in storing extra data. This approach is very simple. In this method the least significant bits of some or all of the bytes inside an image is replaced with a bits of the secret message. The LSB embedding approach has become the basis of many techniques that hide messages within multimedia carrier data. LSB embedding may even be applied in particular data domains – for example, embedding a hidden message into the color values of RGB bitmap data, or into the frequency coefficients of a JPEG image. LSB embedding can also be applied to a variety of data formats and types. Therefore, LSB embedding is one of the most important steganography techniques in use today. [3]

One researcher combined LSB with cryptography and used Advanced Encryption Standard (AES) to encrypt the text before using LSB to hide it in the image. [4] The system first encrypts the given message using AES encryption and then it is written on to the cover image. This provides very high security. The system supports 24-bit and 32-bit BMP images. The system uses a unique LSB technique which stores more data than original algorithm and also maintains good quality while doing so.

2.3 Steganography using DCT and DWT

Many researches have been done on Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). Steganographic approach for securing image using DCT is a widely used method. DCT allows an image to be broken up into three frequency bands namely the Low-frequency band (FL), High-frequency band (FH) and Mid-frequency band (FM). In this approach, the secret data is embedded into the DCT blocks containing mid frequency (FM) sub band components whereas the high frequency sub band components remain unused. Using frequency domain steganography is safe, sound, and flexible approach, and these are its added advantages. It has different techniques for

management. Steganography using DWT has more advantages over DCT because it provides high compression ratios and also it avoids interferences due to artefacts. So comparatively DWT is a better method for hiding confidential data.

Della Baby et al. [5] In this paper, a new algorithm based on DWT has been proposed. The cover image is split up into R, G and B planes. Secret images are embedded into these planes. An N-level decomposition of the cover image and the secret images are done and some frequency components of the same are combined. Secret images are then extracted from the stego image. Here, the stego image obtained has a less perceptible changes compared to the original image with high overall security.

M Ramesh et al. [6] a new approach towards steganography is proposed in this paper. The proposed algorithm is a hybrid Steganography scheme based on Quick Response (QR-code) and Discrete Wavelet Transform (DWT). This technique includes encoding and decoding operation in frequency domain. The text message is hidden in the QR-code image. The QR code image is hidden into the Discrete Wavelet Transform. This technique performed well and additional security was given to the information. The performance evaluation was done by using statistical parameters. This work was compared to other techniques. The proposed method achieved high security and more imperceptibility.

Another work related to DWT was done by Samreen Sekhon Brar et al. [7] They proposed an algorithm in which they used inverse discrete wavelet transform (iDWT) to compress the image, they the image was encrypted using Blowfish encryption. They made use of HAAR wavelet decomposition for the purpose of steganography. This system ensures more image security during transmission by facilitating the quick image transfers.

2.4 Pixel-Value Differencing

Based on the fact that our human vision is sensitive to slight changes in the smooth regions, while can tolerate more severe changes in the edge regions, the PVD-based methods have been proposed to enhance the embedding capacity without introducing obvious visual artefacts into stego images. In PVD-based schemes, the number of embedded bits is determined by the difference between the pixel and its neighbour. The larger the difference amount is, the more secret bits can be embedded. Usually, PVD based approaches can achieve more imperceptible results compared with those typical LSB-based approaches with the same embedding capacity. [8]

Shivam Negi et al. [9] In this method a cover image is first mapped into a 1D pixels sequence by Hilbert filling curve and then it has been divided into non-overlapping embedding units. The division is made such that it gives two consecutive pixel values. As human eye has limited tolerance when it comes to texture and edge areas than in smooth areas, and as the difference between the pixel pairs in those areas are larger, therefore the method exploits pixel value difference to solve out overflow underflow problem.

J. K. Mandal et al. [10] In this paper, PVD) method for secret data embedding in each of the component of a pixel in a colour image is used. But when PVD method as image steganographic scheme is used, the pixel values in the stego image may exceed the range 0-255. The overflow problem of each component pixel has been eliminated. Furthermore,

for providing more security, different number of bits in different pixel components have been used. It would be very difficult to trace how many bits are embedded in a pixel of the stego image. From the experiments it is seen that the results obtained in proposed method provides better visual quality of stego-image compared to the PVD method.

2.5 7th Bit Pixel

Kamaldeep Joshi et al. [11] the proposed method works on gray images. A mathematical function is applied to the 7th bit of the pixels. The 7th bits of the selected pixel and pixel + 1 value are extracted, and on the basis of a combination of these two values, 2 bits of the message can be extracted from each pixel. There can be four possible combinations 00, 01, 10, and 11. This method provides various advantages such as two bits of message storage in each pixel and non-dependency of the technique on the 8th bit.

2.6 Color Image Clustering Technique

Mohamed Tahar Ben Othman et al. [12] presented a more secured and nonspecifically dependent on image formats steganography technique for any bmp or JPEG digital images, based on clustering technique. It is a new color image clustering technique used for robust steganography. It can be defined as the persistence of the embedded steganography under image unrests resulting from attacks. Firstly, the image under consideration is divided into 8x8 blocks and then the clusters of blocks, using the Content Addressable Method (CAM). The cluster of blocks is divided into sub-clusters and Payload data are embedded into next level of cluster i.e. sub clusters. Duplication of the same payload data has been achieved through these sub-clusters which prove the robustness of our proposed steganography technique. Moreover, the distribution rate of image sub-clusters over a cluster it is another parameter that can be tuned to improve this robustness.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Introduction

Information gathering is usually the first phase of any project. Its purpose is to identify the exact requirements for the system and for the users. The proposed system is only for android devices.

3.1 Hardware Requirements

For running this application, user needs an android smartphone with minimum requirement of:

- 1 GB of RAM
- 100 MB of ROM

3.1 Software Requirements

Minimum software requirements is

- Android Marshmallow

CHAPTER 4

ANALYSIS

4.1 Existing System

There are many existing system for detecting the source of fake news but none of them is even remotely similar to the method that we have proposed.

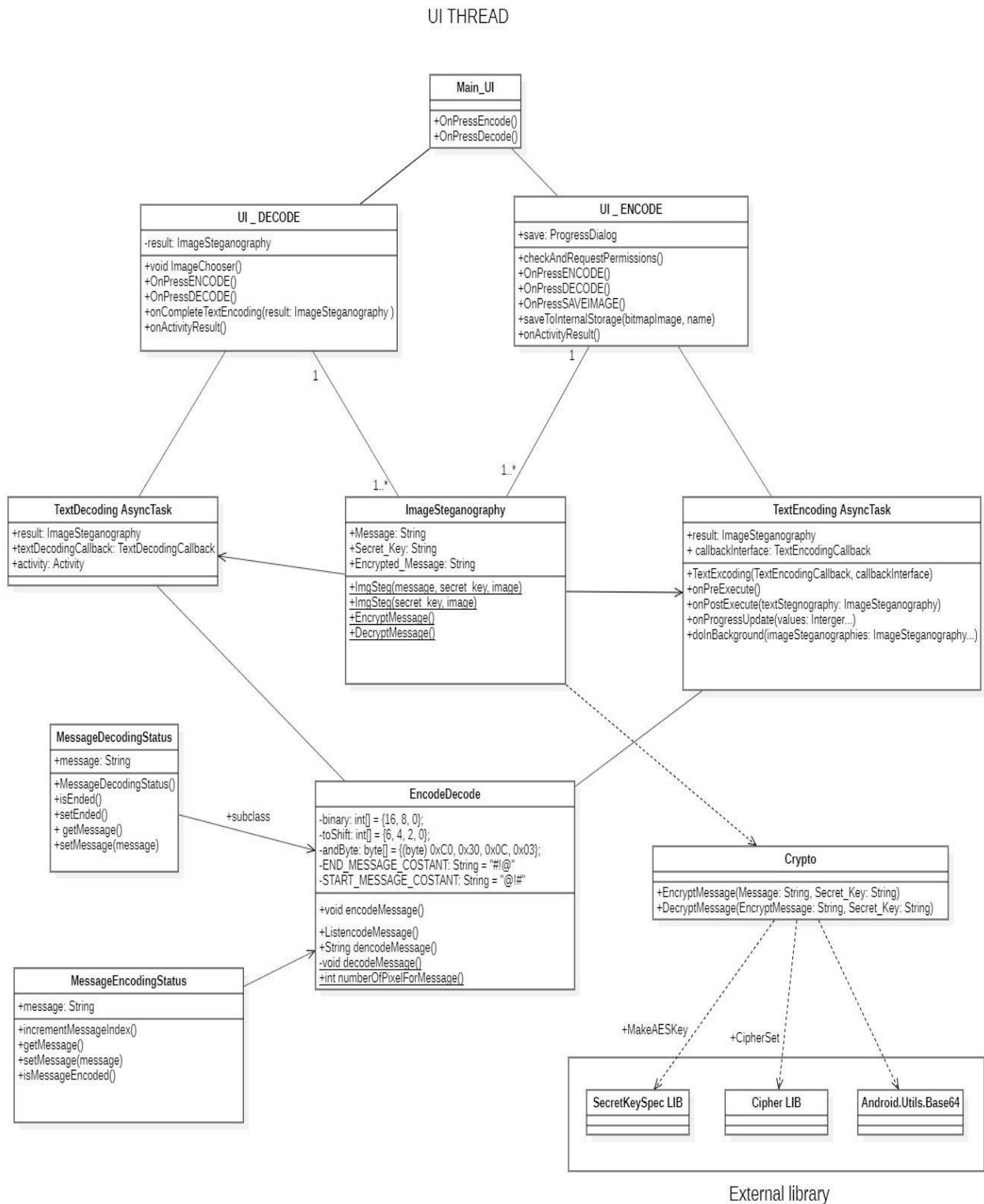
4.2 Proposed System

We are proposing a system to find out the source of widespread of fake news in chat applications like Whatsapp. We will be doing this with the use of steganography techniques. Currently, our main focus is finding the source of the fake news shared via images.

Whenever a new user joins the platform, by default he/she is assigned a permanent and unique id. When an image is shared, the application will check if there is any unique id already present in the image. If it is present, no changes will be made to the image but if it is not present, the application will add the current user's id into the image so that wherever it is passed on, the information of the source will also be passed on. So basically every image which is shared on the platform will have a unique user id stored in it. To hide this information in the image, image steganography is used.

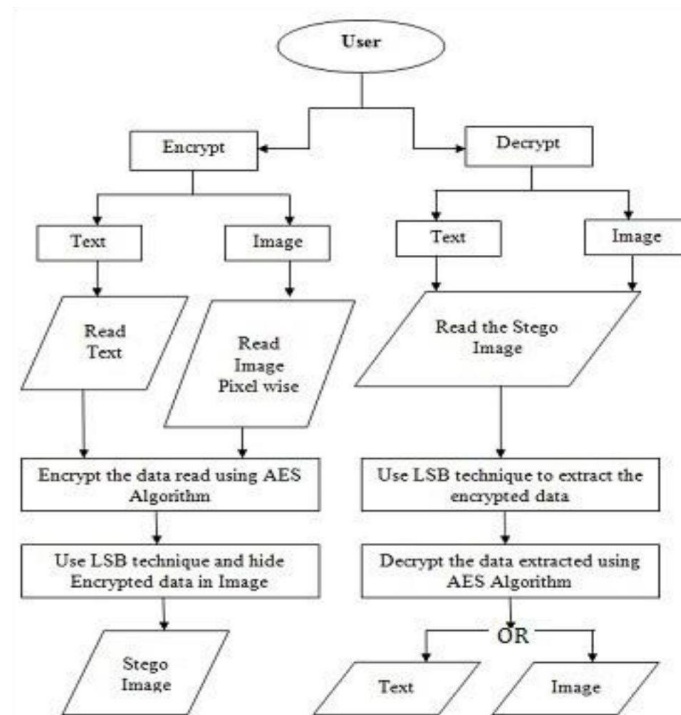
When an image with fake news will be identified, the extracting algorithm will be used to fetch the unique id of the user which has been hidden in the image. The id will then be matched with the data stored in the database and that is how the source will be found.

5.1 UML Diagram



5.1.1 UML Diagram

5.2 Data Flow Diagram



5.2.1 Data Flow Diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

The chat application is created in Android Studio along with Firebase connectivity and programming is done in Java. For storing the unique user id in the image that is sent via the chat application, Least Significant Bit (LSB) embedding technique of image steganography is used. Since LSB technique does not provide much security, the text is encrypted via Advanced Encryption Standard (AES) algorithm.

Images are made up of pixels which usually refer to the color of that particular pixel. The idea behind LSB embedding is that if we change the last bit value of a pixel, there won't be much visible change in the color. For example, 0 is black. Changing the value to 1 won't make much of a difference since it is still black, the only difference is that there is an extremely minor change in the shade that no human eye can detect. The image used to hide the data is called cover image and the image obtained after the message is hidden is called stego image. [13]

AES algorithm:

1. **Byte Substitution:**
The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.
2. **Shift rows:**
Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows –
 - 2.1 First row is not shifted
 - 2.2 Second row is shifted by one (byte) position to the left
 - 2.3 Third row is shifted two positions to the left
 - 2.4 Fourth row is shifted three positions to the left
 - 2.5 The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other
3. **Mix Columns:**
Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.
4. **Add round key:**
The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the cipher text. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round. [14]

Embedding Algorithm:

1. Copy the entire cover image as ARGB_8888 with pre-multiplied feature = false
2. Convert secret text into stream of bits stored as String
3. Check if this stream of bits can be fit inside the cover image, return null otherwise
4. Generate 24 bit random key represented as integer array of length 24 and store in (0,0)th pixel
5. Set flag that indicates the type of secret message (text) in (0,1)th pixel of stego image
6. Perform embedding of secret message stream of bits in the following way:
 - i) Initialize 2 nested for-loops in respective interval $[0 \leq x \leq \text{width}]$, $[2 \leq y \leq \text{height}]$
 - ii) Select each pixel of stego image from (x,y) coordinates
 - iii) Extract Red, Green, Blue colors from each pixel and store in integer array
 - iv) Store 1 bit of secret text in each of the colors. Decision of which color to choose is conducted by xor operation of one bit of random key and 2nd LSB of stego image. If XOR equals to 1 then we store that bit, otherwise we skip the color.
 - v) Update the pixel of stego image at (x,y) with above mutated RGB colors
 - iv) Repeat above given steps until the whole secret text as stream of bits is embedded

Extracting Algorithm

1. Initialize Hash Map which will store 2 key-value pairs (message_type and stream_of_bits)
 2. Extract random 24 bit key from (0,0)th pixel and store in array of length 24
 3. Extract message type (TEXT, IMAGE, or UNDEFINED) from (0,1)th pixel and store in HashMap
 4. Perform extraction of secret message in the following way:
 - i) Initialize 2 nested for-loops in respective interval $[0 \leq x \leq \text{width}]$, $[2 \leq y \leq \text{height}]$
 - ii) Select each pixel of stego image from (x,y) coordinates
 - iii) Extract Red, Green, Blue colors from each pixel and store in integer array
 - iv) Extract 1 bit of secret message from each color. Decision of which color to choose is conducted by xor operation of one bit of secret key and 2nd LSB of Stego image. If xor equals to 1 then we extract LSB of that color, otherwise we skip the color.
 - v) Append each bit to StringBuilder
 - vi) Repeat above given steps until we hit end flag at (x,y) coordinates
 5. We cut unnecessary [0-7] bits from StringBuilder
 6. Store stream of bits of secret message as String in HashMap
- stegoImage is Bitmap image where the secret data is hidden
 - HashMap which contains the message_type and stream_of_bits values

6.1 Sample Code

6.1.1 EncryptionDecryptionAES.java

```
package com.Image.Decrypt.algorithms;

import android.os.Build;
import android.support.annotation.RequiresApi;

import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class EncryptionDecryptionAES {
    private static Cipher cipher;
    private static KeyGenerator keyGenerator;

    public EncryptionDecryptionAES() throws Exception {
        init();
    }

    private static void init() throws Exception {
        keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128);
        cipher = Cipher.getInstance("AES");
    }
    @RequiresApi(api = Build.VERSION_CODES.O)

    public String[] encrypt(String plainText)
        throws Exception {
        SecretKey secretKey = keyGenerator.generateKey();
        byte[] plainTextByte = plainText.getBytes();
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedByte = cipher.doFinal(plainTextByte);
        Base64.Encoder encoder = Base64.getEncoder();
        String encryptedText = encoder.encodeToString(encryptedByte);
        String encodedKey = encoder.encodeToString(secretKey.getEncoded());
        String[] encryptionResult = {encryptedText, encodedKey};
        return encryptionResult;
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    public String decrypt(String encryptedText, String secretKey)
        throws Exception {
        Base64.Decoder decoder = Base64.getDecoder();
        byte[] decodedKey = decoder.decode(secretKey);
```

Image Steganography

```
        SecretKey originalKey = new SecretKeySpec(decodedKey, 0, decodedKey.length, "AES");
        byte[] encryptedTextByte = decoder.decode(encryptedText);

        cipher.init(Cipher.DECRYPT_MODE, originalKey);
        byte[] decryptedByte = cipher.doFinal(encryptedTextByte);
        String decryptedText = new String(decryptedByte);
        return decryptedText;
    }
}
```

6.1.2 Embedding.java

```
package com.Image.Decrypt.algorithms;

import android.graphics.Bitmap;
import android.graphics.Color;
import android.support.annotation.Nullable;

import com.Image.Decrypt.utils.Constants;
import com.Image.Decrypt.utils.HelperMethods;
import com.Image.Decrypt.utils.StandardMethods;

import org.jetbrains.annotations.Contract;

import java.util.Random;

public class Embedding {

    public static Bitmap embedSecretText(String secretText, Bitmap coverImage) {

        Bitmap stegoImage = coverImage.copy(Bitmap.Config.ARGB_8888, true);
        stegoImage.setPremultiplied(false);

        String sTextInBin = HelperMethods.stringToBinaryStream(secretText);

        int secretMessageLen = sTextInBin.length();
        int action, embMesPos = 0, keyPos = 0;

        int width = coverImage.getWidth();
        int height = coverImage.getHeight();

        if (secretMessageLen > width * height * 2) {
            return null;
        }

        int key[] = generateKey();
        int temp_number;
```

```
int red_sum = 0;

for (int j = 0; j <= 7; ++j) {
    if (key[j] == 1) {
        temp_number = (int) Math.pow(2, 7 - j);
    } else {
        temp_number = 0;
    }
    red_sum += temp_number;
}

int green_sum = 0;
for (int j = 8; j <= 15; ++j) {
    if (key[j] == 1) {
        temp_number = (int) Math.pow(2, 15 - j);
    } else {
        temp_number = 0;
    }
    green_sum += temp_number;
}

int blue_sum = 0;
for (int j = 16; j <= 23; ++j) {
    if (key[j] == 1) {
        temp_number = (int) Math.pow(2, 23 - j);
    } else {
        temp_number = 0;
    }
    blue_sum += temp_number;
}

stegoImage.setPixel(0, 0, Color.rgb(red_sum, green_sum, blue_sum));
StandardMethods.showLog("EMB", "Key1: " + red_sum + " " + green_sum + " " + blue_sum);

stegoImage.setPixel(0, 1, Constants.COLOR_RGB_TEXT);

int endX = 0, endY = 2;

outerloop:
for (int x = 0; x < width; x++) {
    for (int y = 2; y < height; y++) {
        int pixel = coverImage.getPixel(x, y);

        if (embMesPos < secretMessageLen) {
            int colors[] = {Color.red(pixel), Color.green(pixel), Color.blue(pixel)};

            for (int c = 0; c < 3; c++) {
```

```
        if (embMesPos == secretMessageLen) {
            break;
        }

        if ((key[keyPos] ^ LSB2(colors[c])) == 1) {
            action = action(colors[c], sTextInBin.charAt(embMesPos));
            colors[c] += action;
            embMesPos++;
            keyPos = (keyPos + 1) % key.length;
        }
    }

    int newPixel = Color.rgb(colors[0], colors[1], colors[2]);
    stegoImage.setPixel(x, y, newPixel);
} else {

    if (y < height - 1) {
        endX = x;
        endY = y + 1;
    } else if (endX < width - 1) {
        endX = x + 1;
        endY = y;
    } else {
        endX = width - 1;
        endY = height - 1;
    }

    break outerloop;
}
}

stegoImage.setPixel(endX, endY, Constants.COLOR_RGB_END);

return stegoImage;
}

@Contract(pure = true)
private static int LSB(int number) {
    return number & 1;
}

@Contract(pure = true)
private static int LSB2(int number) {
    return (number >> 1) & 1;
}

private static int action(int color, char bit) {
```


Image Steganography

```
        if (LSB(color) == 1 && bit == '0') {
            return -1;
        } else if (LSB(color) == 0 && bit == '1') {
            return 1;
        } else {
            return 0;
        }
    }
}

private static int[] generateKey() {
    final int[] bits = {0, 1};
    int[] result = new int[24];

    int n, i;
    Random random = new Random();

    for (i = 0; i < result.length; ++i) {
        n = random.nextInt(2);
        result[i] = bits[n];
    }
    return result;
}
```

6.1.3 Extracting.java

```
package com.Image.Decrypt.algorithms;

import android.graphics.Bitmap;
import android.graphics.Color;

import com.Image.Decrypt.utils.Constants;
import com.Image.Decrypt.utils.StandardMethods;

import org.jetbrains.annotations.Contract;

import java.util.HashMap;
import java.util.Map;

public class Extracting {
    public static Map extractSecretMessage(Bitmap stegoImage) {
        Map<String, Object> map = new HashMap<String, Object>();

        int width = stegoImage.getWidth();
        int height = stegoImage.getHeight();
```

Image Steganography

```
int key[] = new int[24];

int keyPixel = stegoImage.getPixel(0, 0);

int red = Color.red(keyPixel);
int green = Color.green(keyPixel);
int blue = Color.blue(keyPixel);

StandardMethods.showLog("EXT", "Key2: " + red + " " + green + " " + blue);

String red_bin = Integer.toBinaryString(red);
red_bin = "00000000" + red_bin;
red_bin = red_bin.substring(red_bin.length() - 8);

for (int i = 0; i <= 7; i++) {
    key[i] = (red_bin.charAt(i) == '1' ? 1 : 0);
}

String green_bin = Integer.toBinaryString(green);
green_bin = "00000000" + green_bin;
green_bin = green_bin.substring(green_bin.length() - 8);

for (int i = 0; i <= 7; i++) {
    key[i + 8] = (green_bin.charAt(i) == '1' ? 1 : 0);
}

String blue_bin = Integer.toBinaryString(blue);
blue_bin = "00000000" + blue_bin;
blue_bin = blue_bin.substring(blue_bin.length() - 8);

for (int i = 0; i <= 7; i++) {
    key[i + 16] = (blue_bin.charAt(i) == '1' ? 1 : 0);
}

int typePixel = stegoImage.getPixel(0, 1);
int tRed = Color.red(typePixel);
int tGreen = Color.green(typePixel);
int tBlue = Color.blue(typePixel);

if (tRed == 135 && tGreen == 197 && tBlue == 245) {
    map.put(Constants.MESSAGE_TYPE, Constants.TYPE_TEXT);
} else if (tRed == 255 && tGreen == 105 && tBlue == 180) {
    map.put(Constants.MESSAGE_TYPE, Constants.TYPE_IMAGE);
} else {

    map.put(Constants.MESSAGE_TYPE, Constants.TYPE_UNDEFINED);
    map.put(Constants.MESSAGE_BITS, "");
    return map;
}
```

Image Steganography

```
}
StringBuilder sb = new StringBuilder();

int keyPos = 0;
outerloop:
for (int x = 0; x < width; ++x) {
    for (int y = 2; y < height; ++y) {
        int pixel = stegoImage.getPixel(x, y);

        int colors[] = {Color.red(pixel), Color.green(pixel), Color.blue(pixel)};

        if (colors[0] == 96 && colors[1] == 62 && colors[2] == 148) {
            break outerloop;
        } else {

            for (int c = 0; c < 3; c++) {

                if ((key[keyPos] ^ LSB2(colors[c])) == 1) {
                    int lsb = LSB(colors[c]);
                    sb.append(lsb);
                    keyPos = (keyPos + 1) % key.length;
                }
            }
        }
    }
}

String sm = sb.toString();
int sL = sm.length();

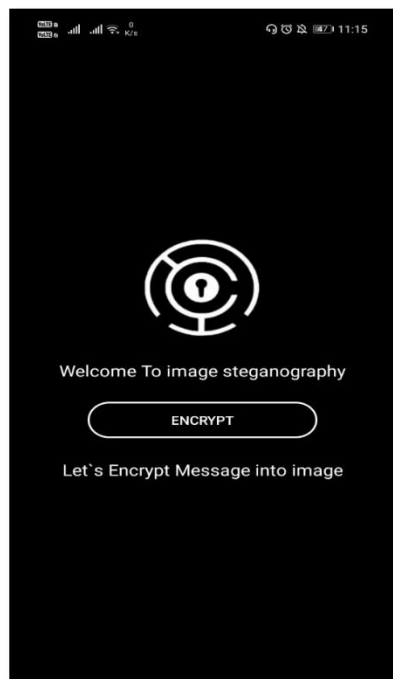
sm = sm.substring(0, sL - sL % 8);

map.put(Constants.MESSAGE_BITS, sm);
return map;
}
@Contract(pure = true)
private static int LSB(int number) {
    return number & 1;
}
@Contract(pure = true)
private static int LSB2(int number) {
    return (number >> 1) & 1;
}
}
```

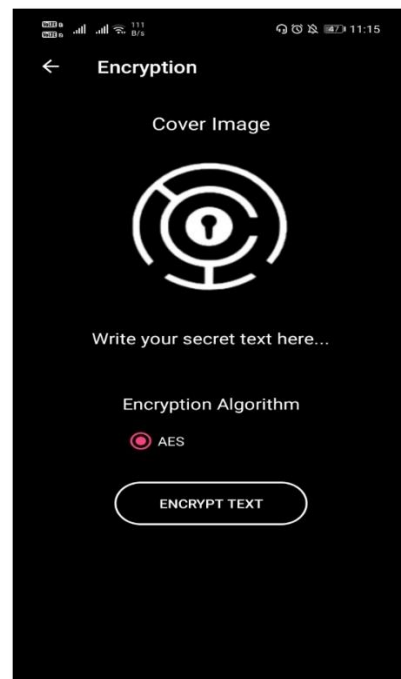
CHAPTER 7

SAMPLE SCREENSHOTS

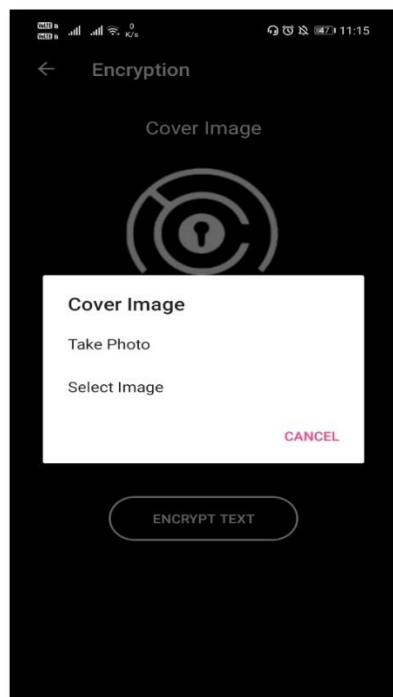
7.1 Encryption App



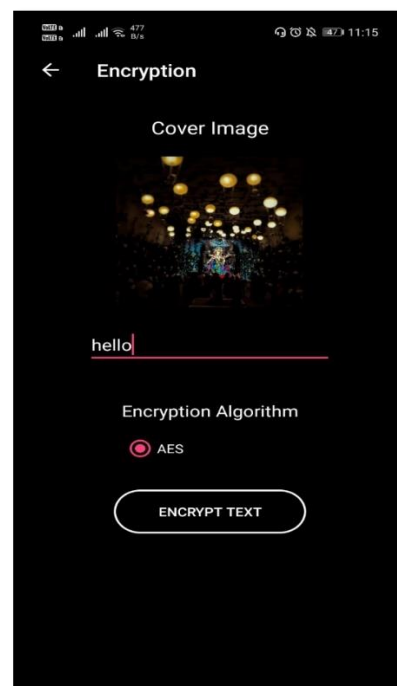
7.1.1 Home Page



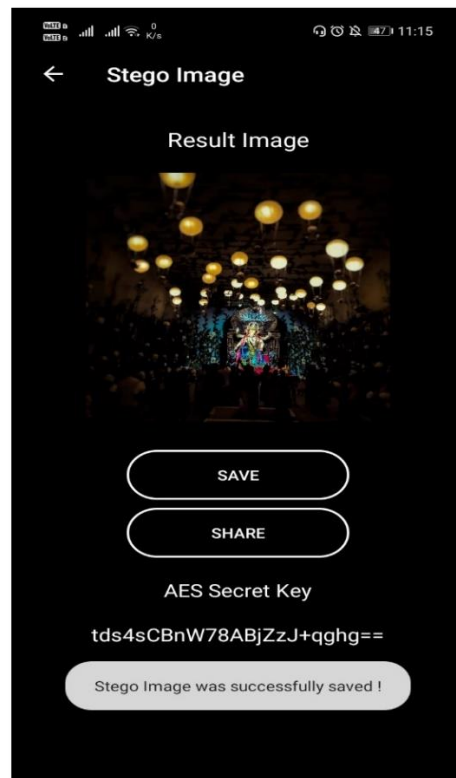
7.1.2 Encryption Page



7.1.3 Selecting Image



7.1.4 Encryption Page with Image

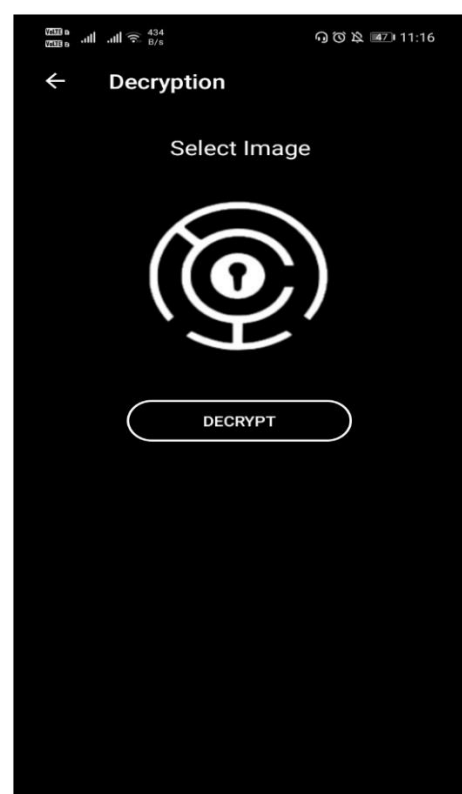


7.1.5 Encryption Completed

7.2 Decryption App

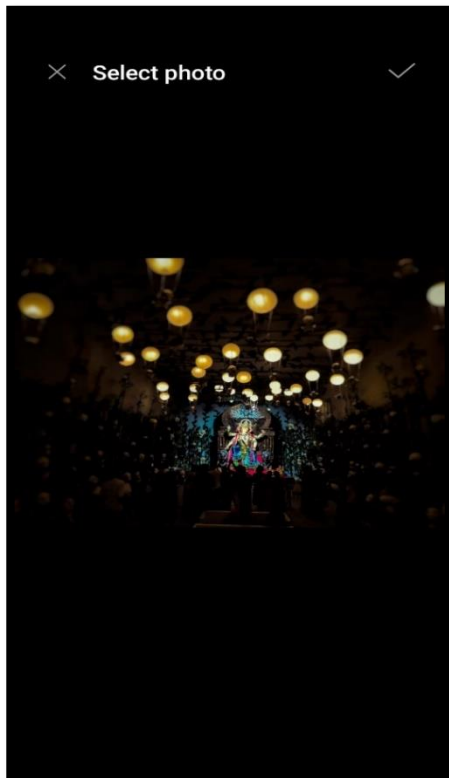


7.2.1 Home page

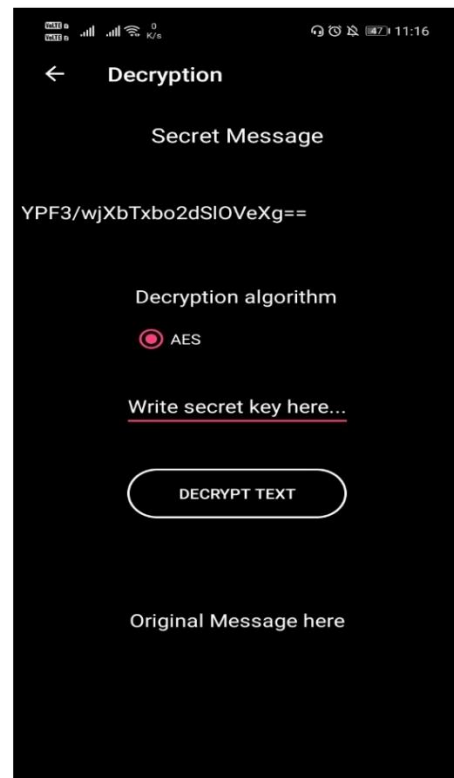


7.2.2 Select the image

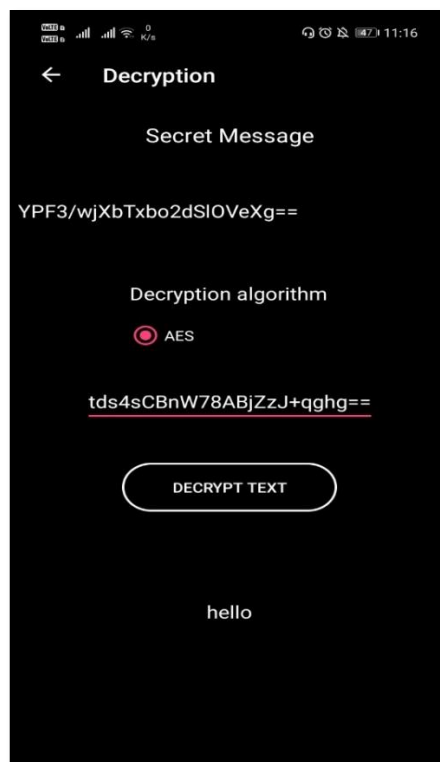
Image Steganography



7.2.3 Selected image

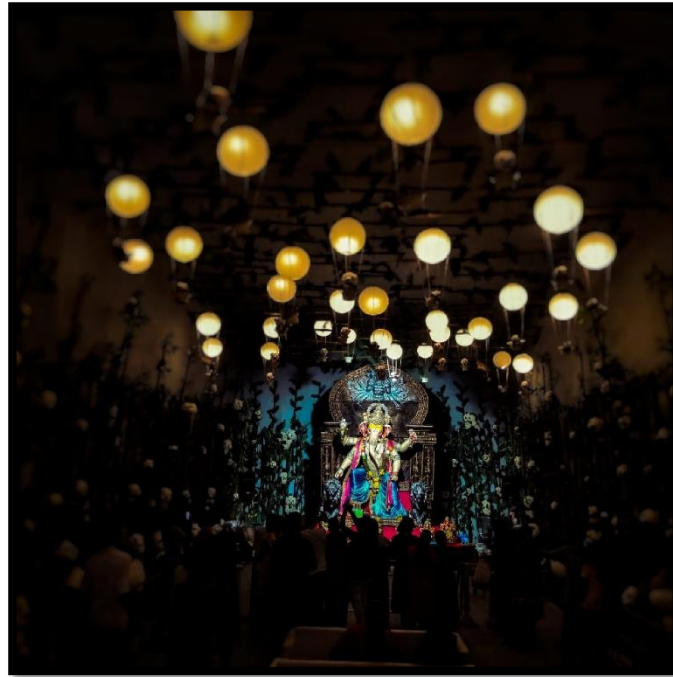


7.2.4 Decryption Page

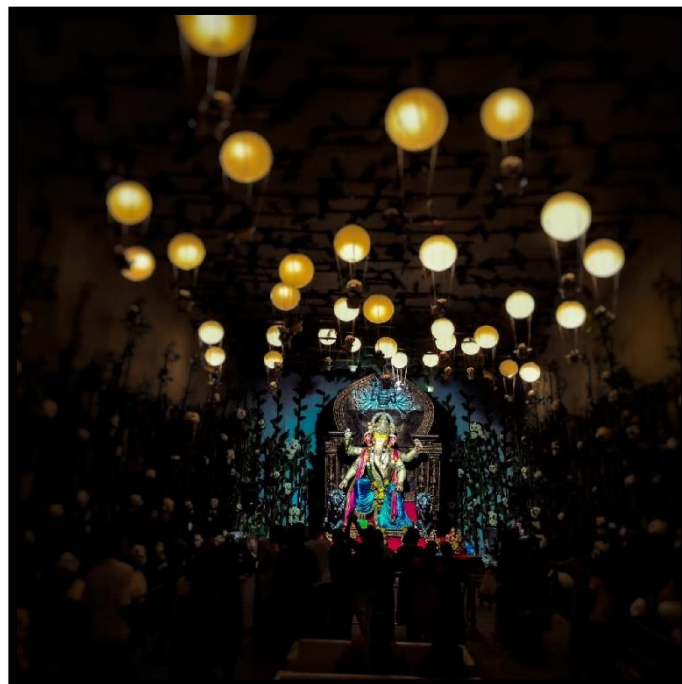


7.2.5 Decryption Page with Secret key

7.3 Comparison of Cover image and Stego image

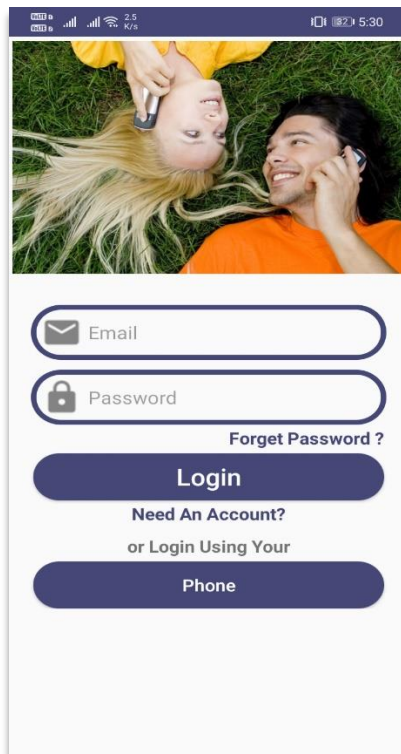


7.3.1 Cover Image



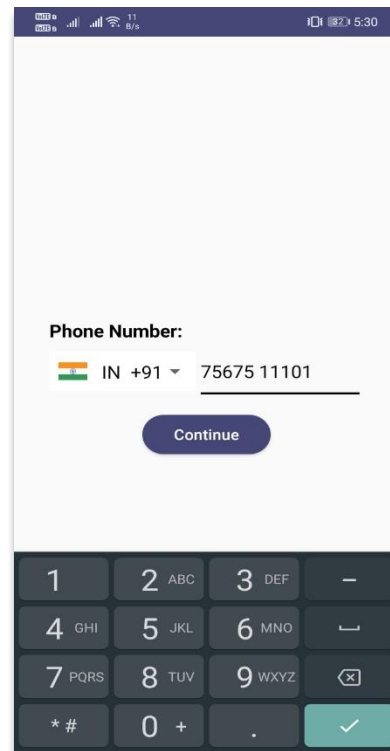
7.3.2 Stego Image

7.4 Chat App



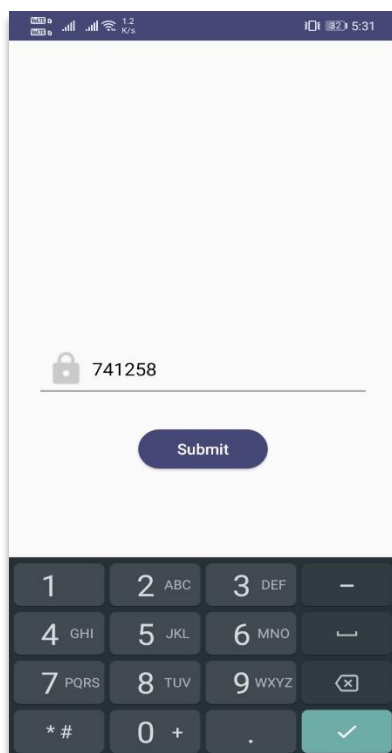
The Login Page UI features a header image of a man and a woman talking on the phone. Below the image are two input fields: 'Email' and 'Password'. A 'Forgot Password ?' link is positioned to the right of the Password field. Below these fields is a large blue 'Login' button. Underneath the button, the text 'Need An Account?' is displayed, followed by 'or Login Using Your' and a blue 'Phone' button.

7.4.1 Login Page



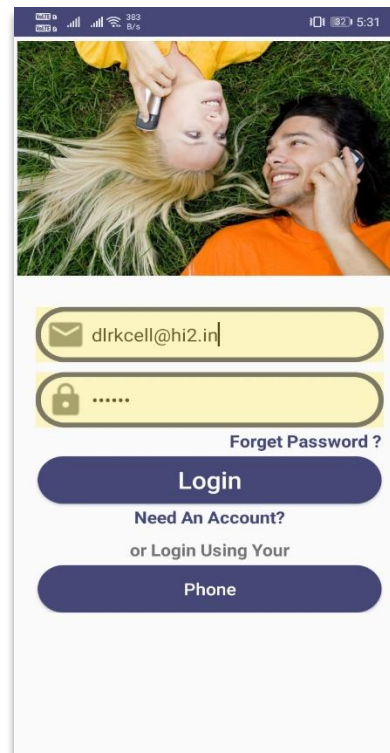
The Sign Up UI features a header image of a man and a woman talking on the phone. Below the image is a 'Phone Number:' label. A dropdown menu shows 'IN +91' and a text input field contains '75675 11101'. A blue 'Continue' button is below the input field. At the bottom is a numeric keypad with digits 1-9, *, #, 0, and a checkmark button.

7.4.2 Sign Up



The OTP Verification UI features a header image of a man and a woman talking on the phone. Below the image is a lock icon and a text input field containing '741258'. A blue 'Submit' button is below the input field. At the bottom is a numeric keypad with digits 1-9, *, #, 0, and a checkmark button.

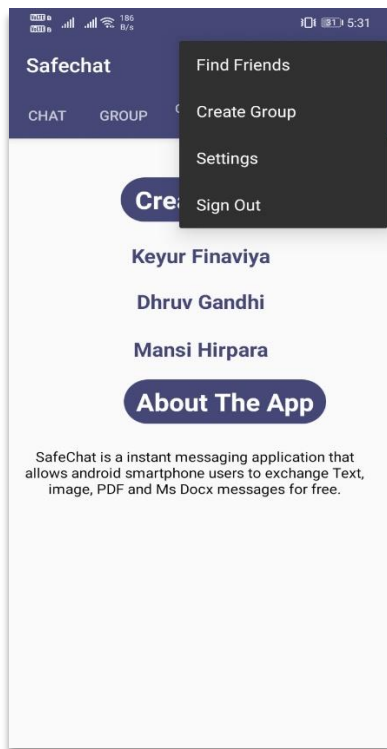
7.4.3 OTP Verification



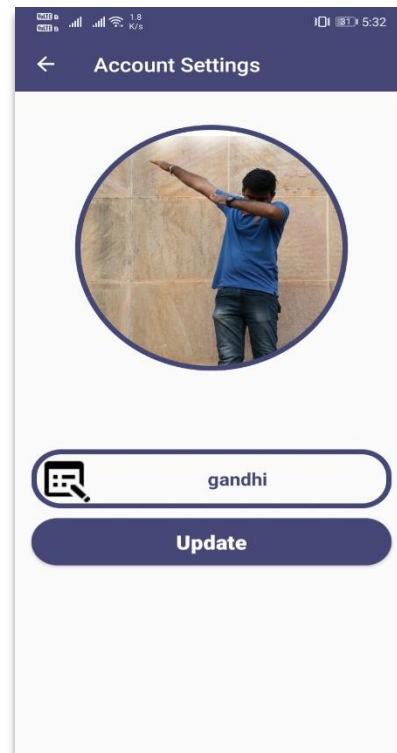
The Logging in UI features a header image of a man and a woman talking on the phone. Below the image are two input fields: 'Email' (containing 'dlrkcell@hi2.in') and 'Password' (containing '.....'). A 'Forgot Password ?' link is to the right of the Password field. Below these fields is a large blue 'Login' button. Underneath the button, the text 'Need An Account?' is displayed, followed by 'or Login Using Your' and a blue 'Phone' button.

7.4.4 Logging in

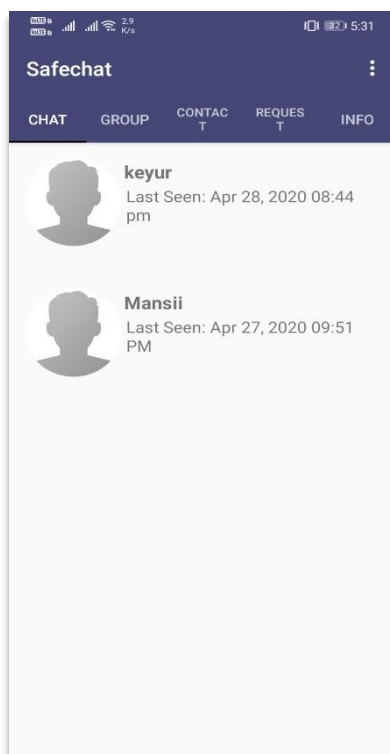
Image Steganography



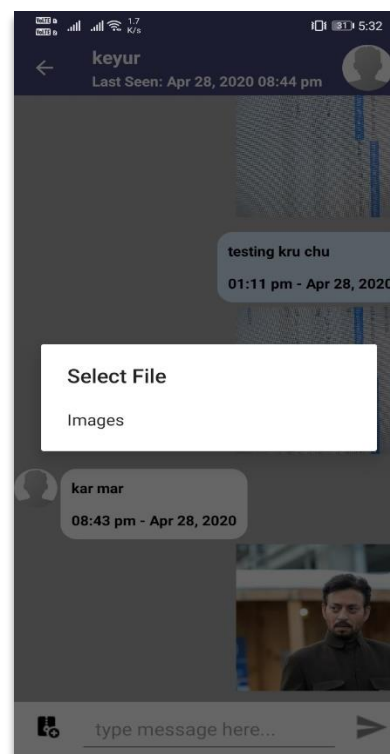
7.4.5 About the App



7.4.6 Account Settings

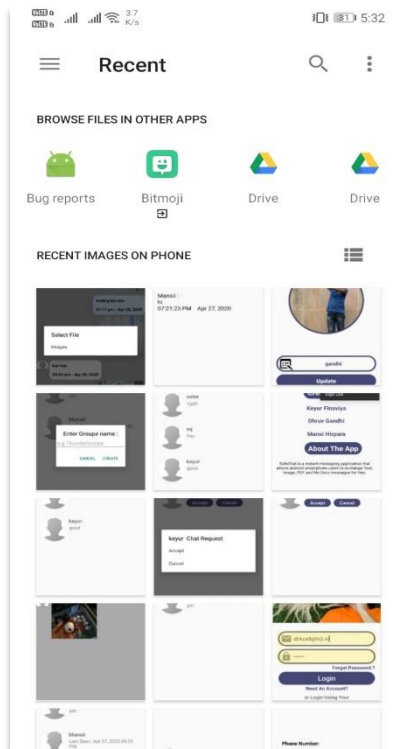


7.4.7 Recent Chat

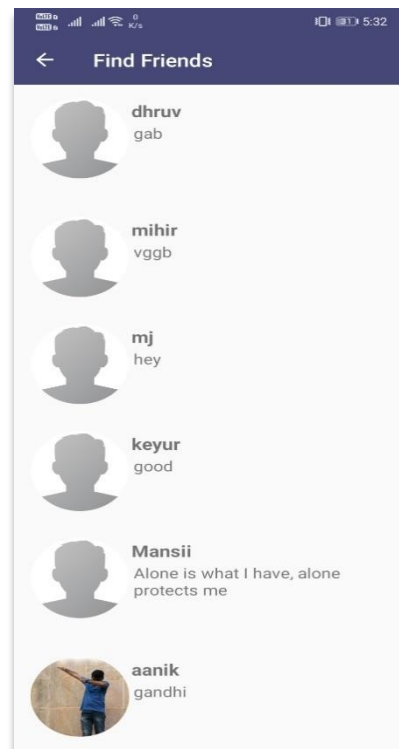


7.4.8 Sending Image

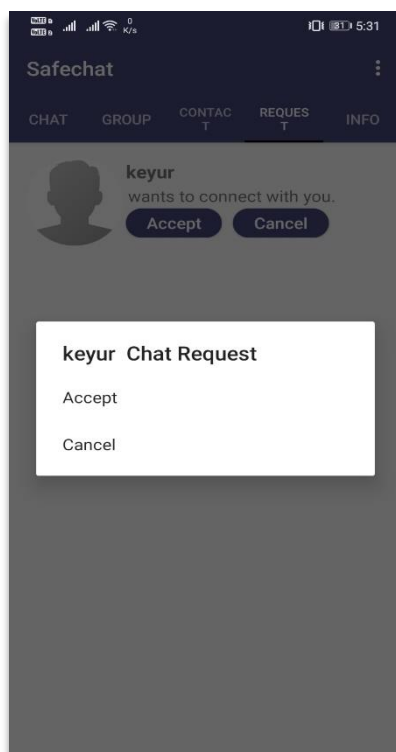
Image Steganography



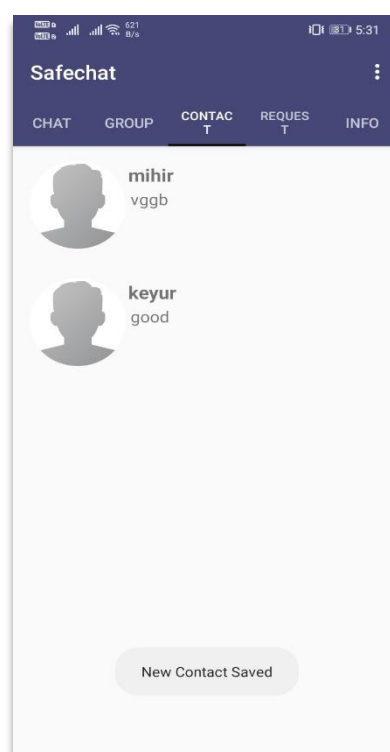
7.4.9 Selecting Image



7.4.10 Find Friends

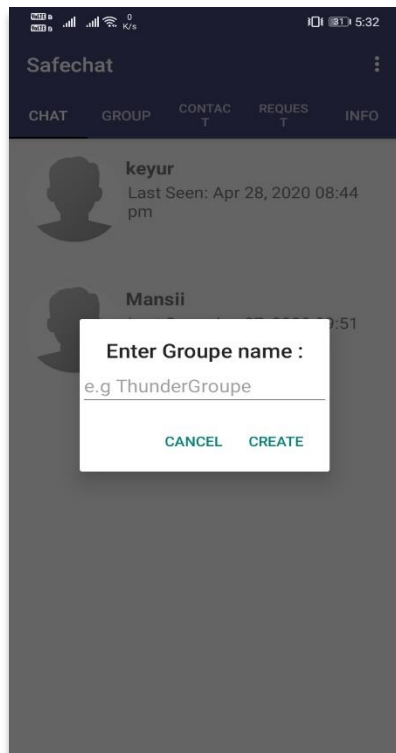


7.4.11 Friend Request



7.4.12 Contacts

Image Steganography



7.4.13 Creating Group



7.4.14 Group Chat

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

In this report, we introduce a basic idea of how we can work with the concepts of steganography and combine it with already existing technology to solve a real life problem.

This project is a very simple implementation of the idea. There is still much more that can be improved. The algorithm used in this project has some downsides, security and message size being one of the few. LSB is the most basic algorithm and the steganography community has many more advanced algorithms which overcome the limitations of LSB embedding. Also, the current scope of the project is only to implement steganography in images sent via texting. This scope can be further broadened by using different forms of steganography like audio and video steganography for the same purpose. And furthermore, the project is only limited to android devices. We can work with other operating systems and extend that limitation.

REFERENCES

- [1] K. Joshi, 2018. [Online]. Available: <https://www.hindawi.com/journals/jcnc/2018/9475142/>.
- [2] A. H. Khire, S. S. Mahadik, M. V. Solanke and S. S. Panavkar, *Steganography*, Raigad, Maharashtra, 2017-18.
- [3] J. Singh, D. H. V. Singh and A. K. Singh, "Steganography in Images Using LSB Technique," *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, vol. 5, no. 1, pp. 426-430, 2015.
- [4] V. M, Y. E, Veeresh, S. K. S and N. .M, "Hybrid Approach to Text & Image Steganography using AES and LSB Technique," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 4, pp. 1500-1502, April 2018.
- [5] D. Baby, "A Novel DWT based Image Securing Method using Steganography," *Procedia Computer Science*, vol. 46, pp. 612-618, 2015.
- [6] M. Ramesh, G. Prabakaran and R. Bhavani, "QR- DWT Code Image Steganography," *International Journal of Computational Intelligence and Informatics*, vol. 3, no. 1, pp. 9-13, April-June 2013.
- [7] S. S. Brar and A. Brar, "Double Layer Image Security System using Encryption and Steganography," *I. J. Computer Network and Information Security*, vol. 3, pp. 27-33, 2016.
- [8] A. Tiwari, S. R. Yadav and N. Mittal, "A Review on Different Image Steganography Techniques," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 3, no. 7, pp. 121-124, January 2014.
- [9] S. Negi and R. Jaat, "Images Steganography using Pixel Value," Vellore, 2015.
- [10] D. D. J. K. Mandal, "Colour Image Steganography Based on Pixel Value," *International Journal of Information Sciences and Techniques (IJIST)*, vol. 2, no. 4, pp. 83-93, July 2012.
- [11] K. Joshi, S. Gill and R. Yadav, "A New Method of Image Steganography Using 7th Bit of a Pixel as Indicator by Introducing the Successive Temporary Pixel in the Gray scale Image," *Journal of Computer Networks and Communications*, vol. 2018, August 2018.
- [12] M. T. B. Othman, A. S. Ansari and M. S. Mohammadi, "Digital color image steganography for nonspecific format and secured based on Clustering," *IJCSNS International Journal of Computer Science and Network Security*, vol. 19, no. 4, pp. 20-27, April 2019.
- [13] A. Sridhar, "GeeksforGeeks," [Online]. Available: <https://www.geeksforgeeks.org/lsb-based-image-steganography-using-matlab/>.
- [14] [Online]. Available: https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm. [Accessed April 2020].

- [15] A. Agarwal. [Online]. Available: <https://github.com/aagarwal1012/Image-Steganography-Library-Android>.