Tribhuvan University

Institute of Science and Technology

Himalaya College of Engineering

[Code: CSC-404]

A

Final Year Project Report

On

"Image Steganography"

by

Aakash Shrestha [7212/072]

Sandhya Khadka [7251/072]

Simran Dhoju [7255/072]

Supriya Amatya [7259/072]

Submitted to Department of Computer Science and Information Technology in partial fulfillment of the requirement for the Bachelor's degree in Computer Science and Information Technology

Department of Computer Science and Information Technology

August 5, 2019

DECLARATION BY CANDIDATES

We, the final year (seventh semester) B.Sc. CSIT students of Himalaya College of Engineering, hereby declare that the project report entitled **"Image Steganography"** is carried by us under the guidance and supervision of **Mr. Bhupendra Ram Luhar** (Lecturer, Tribhuwan University)

We assure that this project work embedded the result of the original work and the contents of the project have not submitted to anybody else for the award of degree. This project is purely of academic interest.

We have followed the guide lines provided by the university in writing the report.

Signature of the candidates

Mr. Aakash Shrestha

Date: August 5, 2019 Miss Sandhya Khadka

Miss Simran Dhoju

Miss Supriya Amatya

SUPERVISOR'S CERTIFICATE

This is to certify that the project entitled **"Image Steganography"** is done by Aakash Shrestha (7212), Sandhya Khadka (7251), Simran Dhoju (7255) and Supriya Amatya (7259) of fourth year B.Sc. CSIT under the guidance and supervision of **Mr. Bhupendra Ram Luhar**, Lecturer, Himalaya College of Engineering, towards the partial fulfillment of bachelor's degree of Computer Science and Information Technology.

_____

Signature of Supervisor

Mr. Bhupendra Ram Luhar

Lecturer

Himalaya College of Engineering

Chyasal, Lalitpur

Date:

Seal:

EVALUATION COMMITTEE

This is to certify that the project entitled **"Image Steganography"** is done by Aakash Shrestha (7212), Sandhya Khadka (7251), Simran Dhoju (7255) and Supriya Amatya (7259) of fourth year B.Sc. CSIT under the guidance and supervision of **Mr. Bhupendra Ram Luhar**, Lecturer, Himalaya College of Engineering, towards the partial fulfillment of bachelor's degree of Computer Science and Information Technology.

Signature of HOD	**Signature of Supervisor**
Er. Himal Chand Thapa	Mr. Bhupendra Luhar
Head of Department (CSIT)	Lecturer
Himalaya College of Engineering	Himalaya College of Engineering
Chyasal, Lalitpur	Chyasal, Lalitpur

External Examiner

ACKNOWLEDGEMENT

Firstly, we would like to thank Himalaya College of Engineering for providing us this platform to gain knowledge about different aspects of Computer Science and Information Technology.

We would also like to thank our Head of Department **Er. Himal Chand Thapa** who played an important role in providing us advice and suggestion towards the selection of our project topic as well as for providing us with necessary content and classes regarding the project. We are also grateful for the care and support from our supervisor **Mr. Bhupendra Ram Luhar** and for providing us all necessary suggestions and corrections he provided, for our project.

In addition, we also want to thank our friends and classmates for helping us with the guidelines we needed during the completion of our project.

ABSTRACT

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Cryptography is a technique associated with the process of converting ordinary plain text into unintelligible text and vice-versa. In contrast to cryptography, steganography is not to keep others from knowing the hidden information but it is to keep others from thinking that the information even exists. Together cryptography and steganography can provide a powerful basis for data security.

The main purpose of this project is to produce a security tool based on steganography and cryptography techniques for sending and receiving sensitive information over the internet. The program first encrypts the message data using AES algorithm and then embeds the result of encrypted data in the provided image file using steganography technique. The system also provides the feature for extracting the hidden data from the corresponding image file and decrypting the extracted data for eventually finding the original message. The embedding process follows image's LSB replacement algorithm. To obtain the hidden message, the process is reversed. This project also provides a client/server module for stego image transmission. The client-server connection is established using TCP socket programming on a user given port.

Using the system produced, messages were successfully encrypted and hidden inside an image file. Also, using the generated stego image, the hidden message was extracted and decrypted successfully.

Keyword: Steganography, Cryptography, LSB, AES.

Table of Contents

List of Figures

Abbreviations

AES: Advanced Encryption Standard

API: Application Program Interface

DFD: Data Flow Diagram

ICT: Information and Communication Technology

IDE: Integrated Development Environmen

IP: Internet Protocol

JPG: Joint Photographic Experts Group

LSB: Least Significant Bit

PNG: Portable Network Graphics

SMTP: Simple Mail Transfer Protocol

TCP: Transmission Control Protocol

UI: User Interface

Chapter 1. Introduction

1.1 Introduction to Image Steganography

Due to advances in ICT, most information is kept electronically. Consequently, the security of information has become a fundamental issue. Maintaining the integrity and confidentiality of sensitive information, blocking the access of sophisticated hackers is very important. Steganography is a technique of hiding information in digital media [1]. Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular ones because of their frequency on the internet. In contrast to cryptography, it is not to keep others from knowing the hidden information but it is to keep others from thinking that the information even exists.

Steganography is the art of concealing information in ways that prevent the detection of hidden messages. Similarly, Cryptography is a technique associated with the process of converting ordinary plain text into unintelligible text and vice-versa. Together cryptography and steganography can provide a powerful basis for data security. These techniques become more important as more people join the cyberspace revolution.

This project intends to produce a security tool based on steganography and cryptography techniques for sending and receiving sensitive information over the internet. The program first encrypts the message data using AES encryption and then embeds the result of encrypted data in the provided image file using LSB replacement technique for steganography. The system also provides the feature for extracting the hidden data from the corresponding image file's LSBs and decrypting the extracted data using AES decryption for eventually finding the original message. The generated stego image can either be saved and sent over a user-chosen platform or can also be sent directly using email sending feature of the system. This project also provides a client/server module for stego image transmission. The client-server connection is established using TCP socket programming on a user given port. The sender (server or client) can encrypt-embed message onto a cover image and send the generated stego image to the receiver (client or server) through this connection. On receiver's side, the received image can be used to obtain the original message.

1.2 Problem Statement

With growing digitization in every field, digital security has become a fundamental aspect. Also, because of development in the internet technology, digital media can be transmitted conveniently over the network. This calls for security over the internet. Throughout history steganography and cryptography have been used to secretly communicate information between people. In the past, means of cryptography and steganography were carried out using traditional methods of pen and paper, using invisible ink, etc.

Therefore, with the increasing use of communication of information over digital medium, security for digital methods are to be developed. Steganography hides data onto a stego medium. Steganography along with cryptographic tools can ensure even more data security.

1.3 Objective

The main objectives of this project are:

- To produce security tool based on steganography and cryptography techniques combined.
- To avoid drawing suspicion to the existence of a hidden message.

1.4 Scope and Limitations

The scope of this project is to limit unauthorized access and provide better security during message transmission. To meet the requirements, simple and basic LSB approach of steganography had been used. The program first encrypts the message data using AES algorithm and embeds the result of encrypted data in the provided image file using steganography technique for sending over the network. The system also provides the feature for extracting the hidden data from the corresponding image file and decrypting the extracted data for eventually finding the original message. Further, the system provides a client-server section for image transmission. The client server connection is established using TCP socket programming.

Steganography means hiding data into another data. It can be used to hide data such as text, image, audio, video etc. within a cover image, video etc. While the system program provides a way to hide text data into a cover image file, it is limited only to data of the mentioned types i.e., text data onto image data.

1.5 Report Organization

Chapter 1 includes subtopics as introduction to image steganography, problem statement, objectives, scope and limitations.

Chapter 2 contains literature review.

Chapter 3 comprises of requirement analysis, feasibility study and process models of the system. The requirements analysis further consists of functional and non-functional requirements. Economic, technical and operational feasibilities are some listed feasibilities. Process model includes DFD diagrams.

Chapter 4 consists of system architecture, class diagram activity diagram and algorithm used.

Chapter 5 includes tools used and testing. Testing comprises of unit testing, integrated testing and system testing.

Chapter 6 includes conclusion and future enhancements.

Chapter 2. Literature Review

The word steganography is derived from the Greek words stegos meaning cover and grafia meaning writing [2]. In Image steganography the information is hidden exclusively in images. Steganography is the art and science of secret communication. It is the practice of encoding/embedding secret information in a manner such that the existence of the information is invisible. The actual files can be referred to as cover text and the cover image. After inserting the secret message, it is referred to as stego-medium. A stego-key is used for hiding encoding process to restrict detection or extraction of the embedded data [3].

For more secure data transfer, cryptography is used along with steganography. In cryptography the message is encrypted using encryption algorithm along with secret key and transferred it to the other end, then receiver can decrypt it and get original message by using decryption algorithm. The grouping of these two approaches enhances the security of data. The combination of these two methods will satisfy the requirement such as capacity and security for data transmission over an open channel. If the attacker were able to detect the steganography technique, they would still have to require the cryptographic decoding way to de-cipher the encrypted message and vice versa [4].

The Least Significant Bit Replacement Algorithm is a commonly used straightforward steganographic algorithm used to embed secret information inside a cover medium. In this method, the least significant bits of the original data in the cover medium are altered based on the secret message. In the case of digital images, the alteration is done only at the least significant bits of the original image so as to reduce the effect of degradation of the original image. By inserting the secret message only at the least significant bits, the perceptibility of the original image is not much affected [5].

Currently, AES is regarded as the most popular symmetric cryptographic algorithm. In 1997, NIST advocated making the AES to replace DES to meet the need of message encryption in 21st century [6].

This system implements the image steganography technique of LSB replacement algorithm along with AES encryption for cryptography.

Chapter 3. System Analysis and Feasibility Study

3.1 Requirement Analysis

3.1.1 Functional Requirements

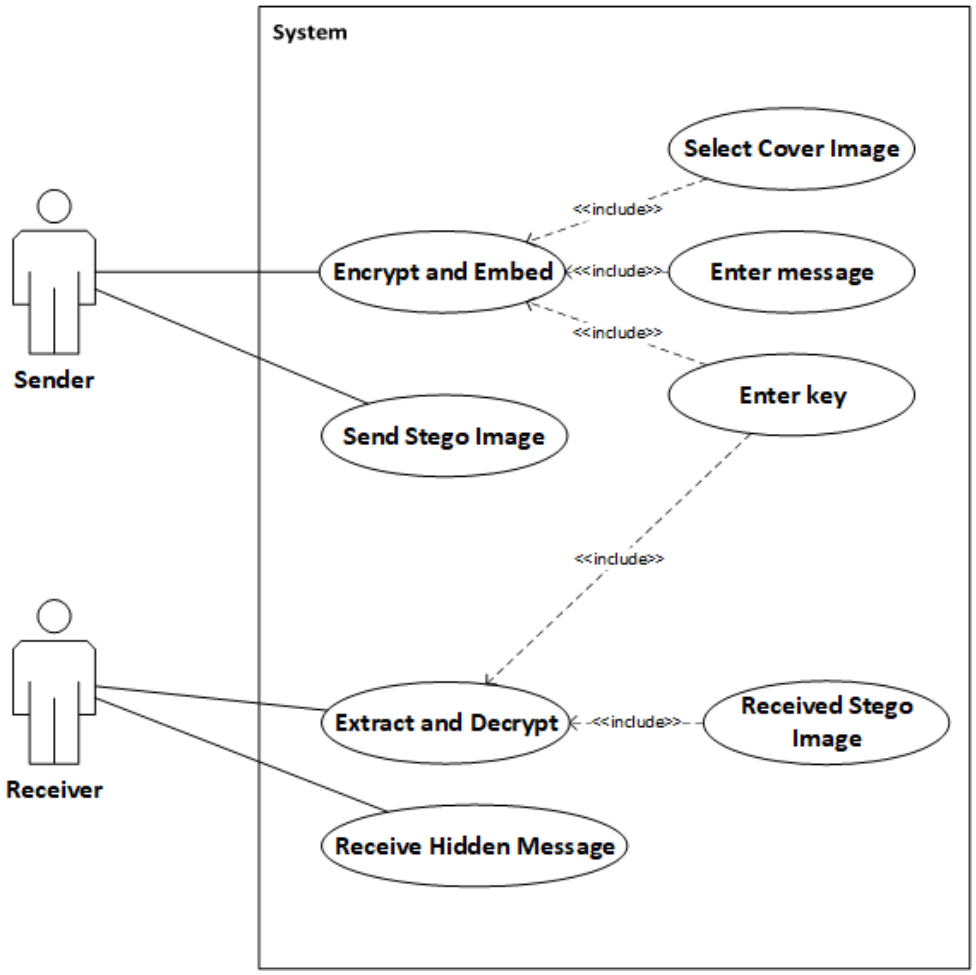


Figure 3-1 Use-Case Diagram for Image Steganography

Figure 3-1 shows the use-case diagram of the system. Sender and receiver are the primary actors. They provide required information and data to the system and receive the required output. Sender's inputs include cover image file, text message (directly or via a txt file) to be hidden and a secret key for message encryption. Upon receiving all valid inputs, the system generates a stego-image which consist of the text message embedded inside it. The sender can then either email the stego-image to the receiver or save the stego image to transfer it over a different media. On receiver's side, the inputs include the received stego

image and associated secret key. The system processes the provided inputs and generates the hidden message in readable text. If needed, the receiver can save the generated message as text file.

In client/Server model of the system, both server and client can act as sender or receiver. They can send stego-images to one another and save or extract hidden messages upon receiving them.

3.1.2 Non-Functional Requirements

Reliability: The system program is reliable as whenever provided with correct information/input, it always gives the corresponding image (on sender's side) or the hidden message (on receiver's side).

Performance: The system performs well as when provided with the correct inputs, it efficiently provides correct output within seconds.

Usability: The system is quite easy to use as anyone with simple skills regarding using a windows computer can use it to create a stego image or extract the messaged hidden on a received stego image.

Maintenance: The system doesn't really need to be maintained regularly but is open to any future enhancement, whenever required.

3.2 Feasibility Study

Feasibility Study is a test of the system according to its workability, impact of the organization, ability to meet user needs and effective use of the resources. A Feasibility Study is generic in nature and can be applied to any type of project, be it for systems and software development, making an acquisition, or any other project. We can test the system by different type of the feasibilities.

3.2.1 Economic Feasibility

The system is economically feasible as all the tools and resources required are either cheap or free.

3.2.2 Technical Feasibility

The technical requirements (for e.g. hardware and software specifications of device for operating the system) is easily available and the system can be operated by users with simple knowledge regarding the required technologies.

3.2.3 Operational Feasibility

The system is user friendly as it is easy to use and operating the system doesn't require too complex skills.

3.3 Process Model

3.3.1 DFD Diagram

Level 0 DFD

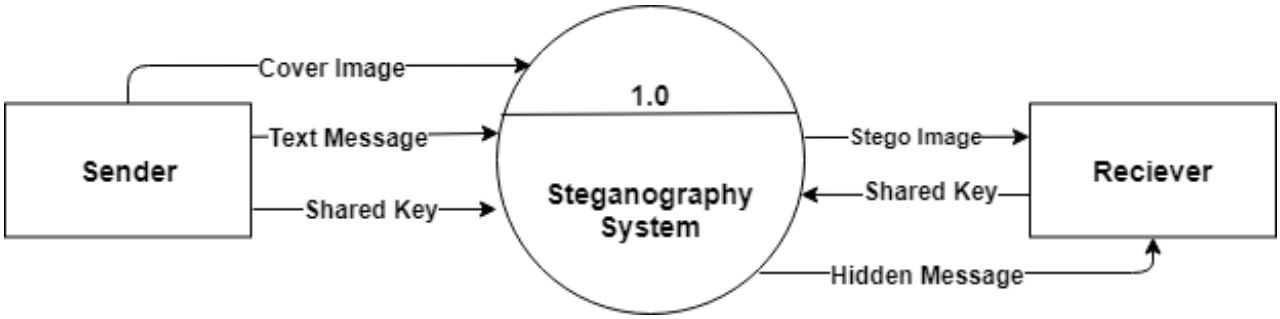


Figure 3-2 Level 0 DFD for Image Steganography

Figure 3-2 explains the Level 0 DFD of the system. The system takes the input as cover image (original image) and a secret text message (directly or via a txt file) from the sender. This application also uses a secret pre-shared key for encryption. With these inputs provided by the sender, the application generates a stego image which is sent to the receiver. The system then takes the associated key from the receiver and displays the hidden message to the receiver. The client server section uses TCP socketing for sending and receiving the stego images generated by using similar techniques as mentioned above.

Level 1 DFD

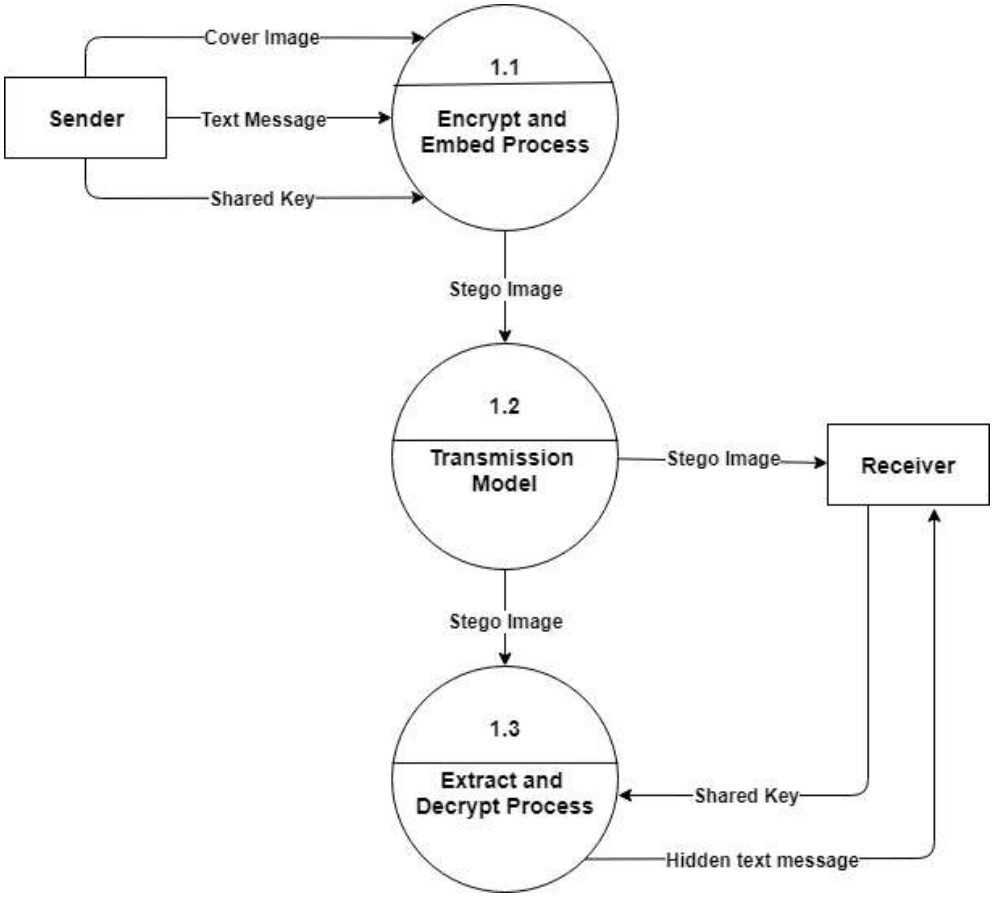


Figure 3-3 Level 1 DFD for Image Steganography

Figure 3-3 represents the Level 1 DFD of the system. In this level, more detailed process of this system is explained. The main task described in this level is that the text message is encrypted using AES encryption and that encrypted message is embedded to the image by replacing least significant bits of the image using LSB technique. Then the generated stego image is sent to receiver through a data transmission channel. The image can be transmitted using the email feature or on client-server section, over the established TCP connection. This system then takes the stego image and shared key from the receiver and first extracts the cipher text message (encrypted hidden message) from the image and with the provided key it decrypts the cipher message and displays the original hidden message to the receiver.

Chapter 4. System Design

4.1 System Architecture

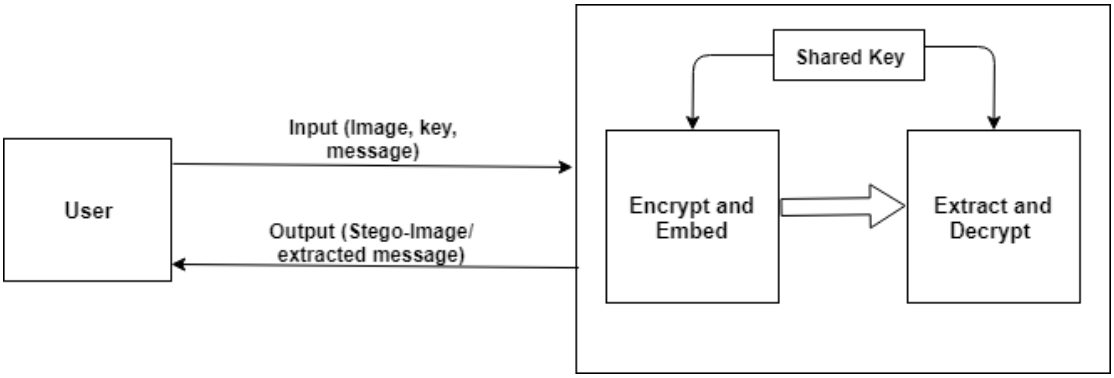


Figure 4-1 System Architecture for Image Steganography

Figure 4-1 shows the basic architecture of the system. User can interact with the system to receive expected outputs.

The system is divided into two parts: normal mode and client/server mode. When starting the program normally, the user needs to provided necessary input to get necessary output which is either stego-image (when embedding a cover image) or extracted message (when extracting a stego-image for hidden message).

When starting the program in client/server model, firstly a TCP server socket needs to be started. The user provides port number to be associated to the server. When a client connects to the server socket on particular port, the client-server connection starts. Then the client and server can communicate with each other through associated port and IP as the communication medium. For communicating sensitive information, stego-image can also be used. The sender side can generate a stego-image containing a hidden message and a key shared by both ends. Then the receiver can use the shared key to extract the hidden message from the received stego-image.

4.2 Class Diagram

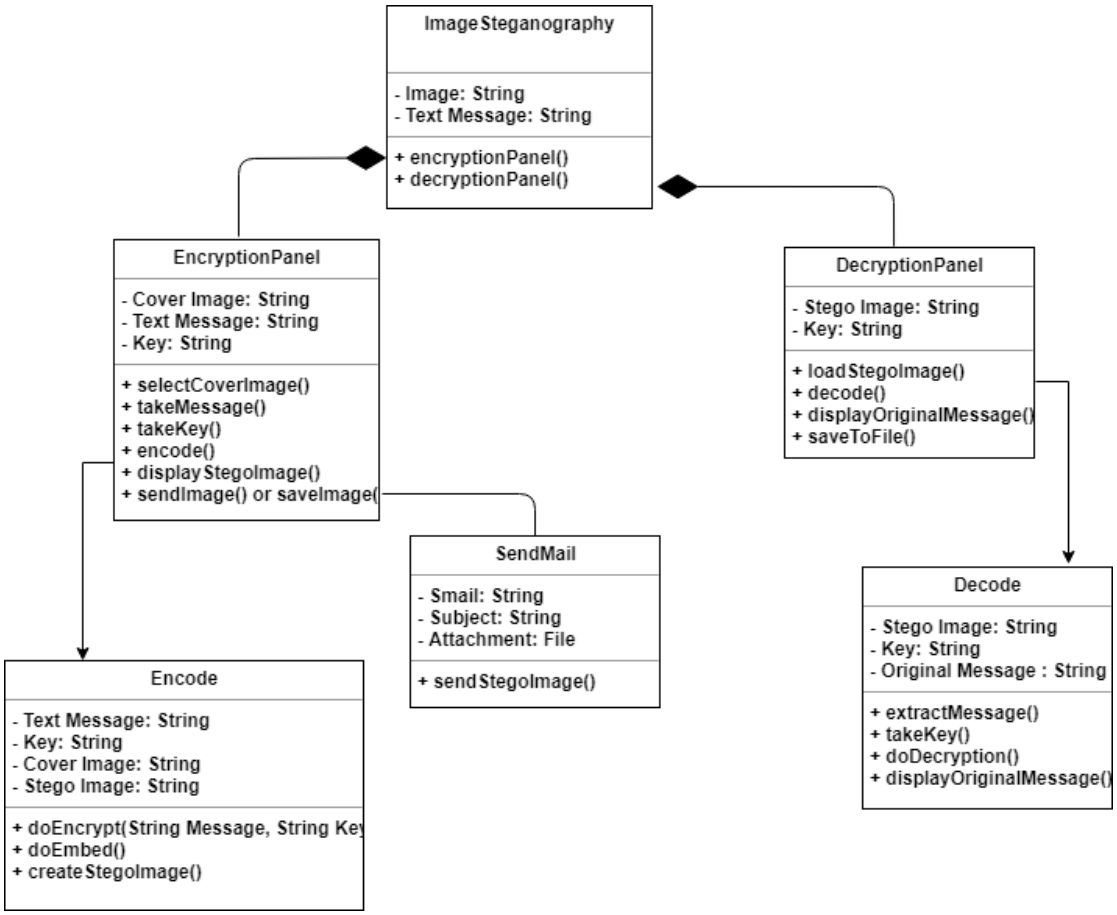


Figure 4-2 Class Diagram for Image Steganography

Figure 4-2 shows the class diagram for the system. Different classes are involved to obtain the preferred result the system. Different classes' instances are created and used whenever necessary. Some major classes involved in the system process are:

ImageSteganography: This is the main class of the system which is accessed first when starting the system. Then, preferred mode of the program can be chosen.

EncryptionPanel: This is the class that contains the panel for encrypting and embedding the cover image file. It takes cover image file, secret message (directly or via a txt file) and shared key as inputs, then calls necessary functions from different classes to reach the goal.

Encode: This class contains the necessary functions for encrypting the given input string and embedding the encrypted string onto provided cover image. It can be accessed from the EncryptionPanel class, which may create and use instance of this class whenever stego-image is to be created.

DecryptionPanel: This is the class that contains the panel for extracting and decrypting the hidden message from the stego image file. It takes the stego image file and shared secret key as inputs, then calls necessary functions from different classes to reach the goal.

Decode: This class contains the necessary functions for extracting the encrypted text from the stego image as well as decrypting the encrypted text to receive the original message. It can be accessed from the DecryptionPanel class, which may create and use instance of this class whenever hidden message is to be extracted.

SendMail: This class is responsible for sending the stego-image given by the Encode class, via email. It contains the frame consisting of the user interface.

4.3 Activity Diagram

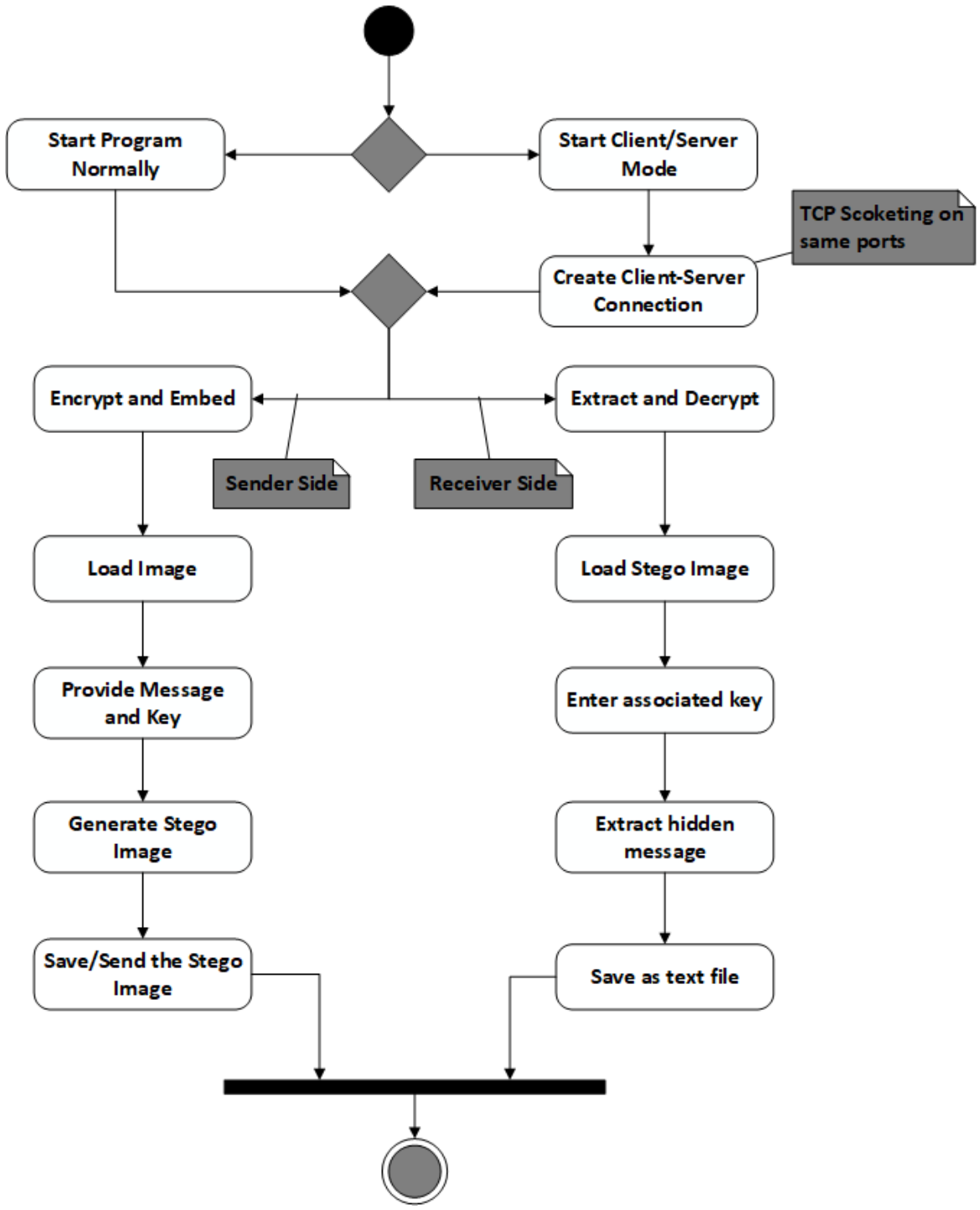


Figure 4-3 Activity Diagram for Image Steganography

Figure 4-3 explains basic activity of the system. The program can either be started in regular mode or client server mode. Either of the modes will support image steganography procedures.

The ‘encrypt and embed’ option takes a cover image, secret key (directly or via a txt file) and the message to be hidden as input. Then provides a stego-image as output. Similarly, the ‘extract and decrypt’ option takes the generated stego image and associated key the extracts the hidden message. Then message can be saved as a file.

In client server mode, firstly a TCP server socket needs to be started. The user provides port number to be associated to the server. When a client connects to the server socket on particular port, the client-server connection starts. Then the client and server can communicate with each other through associated port and IP as the communication medium. Sender loads the image, provides message (directly or via a txt file) and key. Then stego image is generated. The stego image can be saved or send to the receiver. Receiver loads the stego image sent by the sender then enters associate key to extract the hidden message. Finally, the hidden message is extracted from stego image and can be saved as a file.

4.4 Algorithm

Least Significant Bit (LSB) Replacement Method

LSB based technique is simple approach in which message bits are embedded in the least significant bits of cover image [7]. In this technique, the least significant bit of cover image is used to hide the secret message The least significant bit of each pixel of an image is altered to a bit of a message that is to be hidden [8].

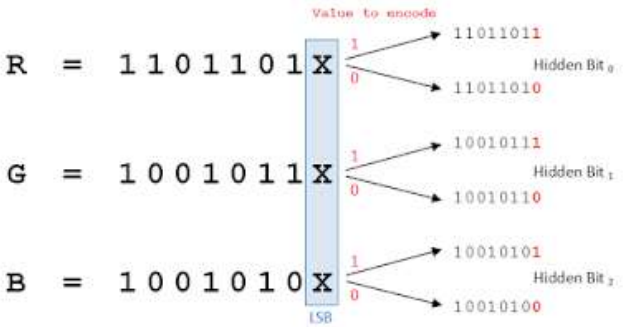


Figure 4-4 LSB Replacement

Message embedding procedure:

Input: Cover image, secret key and message.

Output: Stego image.

1. Read the cover image and secret text information which is to be embedded into the image.
2. Convert the secret information into encrypted text by using AES encryption (for java, Base64Encoder can also be used) and secret key shared by receiver and sender.
3. Add the value of encrypted text's length at the beginning of the text along with a '/' character (to mark the actual cipher length during decryption).
4. Convert encrypted text message into binary form – which will give the text message's characters' bits.
5. Find LSBs of each RGB pixels of the cover image.
6. Embed the bits obtained on step 4 into LSBs of RGB pixels of step 5.
7. Continue the procedure until the secret information is fully hidden in cover image file.

Message extraction procedure:

Input: Stego Image and associated secret key.

Output: Hidden message.

1. Read the stego image and secret key.
2. Retrieve LSBs of each RGB pixels of the stego image.
3. Convert binary strings formed by every 8 RGB pixels of step 2 to character and append the characters to a string builder.
4. Upon finding the first '/' character from string builder of step 3, save its previous characters as text length and discard all characters till that index.
5. Continue the process for more (text length obtained on step 4 * 8) times to fully extract the hidden encrypted text.
6. Using the secret key and AES decryption (Base64Decoder for java), decrypt secret information obtained on step 4 to get original information.

Chapter 5. Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

While many additional software tools were used when developing the project, there was no need for any special hardware requirement. The system was built on a 64-bit computer running with Windows 10. Various Software tools used include:

- IDE: NetBeans 8.0.2.
- Programming Language: Java programming language.
- Swing- GUI widget toolkit for java: Used for creating GUI.
- Java AWT packages: For event handling and working with files and images.
- Java Mail bean: Used to send stego image via Email.
- Java Base64 Encoder: Used to generate cipher text of given message.
- Java Base64 Decoder: Used to decrypt extracted cipher text.
- Java Socket Programming: Used for client-server connection.
- Online Image Histogram Generator (sisik.eu/histo): For image histogram.
- Additionally, various java packages and libraries were also used.

5.1.2 Implementation of Encryption and Embedding Process

The implementation of encoding a message onto a cover image is completed in two steps. The process takes in an image file (JPG, PNG or JPEG), a message string (directly or by uploading a .txt file consisting of the message string) and a key string (up to 16 characters) as inputs. Firstly, the message is encrypted using AES 128-bit encryption. Java Base64 encoder API has been used. The encryption process takes in a 128-bit key string and the required message as input. If the user-entered key is not 128-bit i.e., 16 characters, the provided key string is right-padded with the letter “a” in order to reach the required length of 16 characters. However, if the entered key string exceeds the required length of 16, characters, the program displays a dialog box indicating error. The key string is then converted to the required key datatype. The AES cipher instance is taken which is initialized with ENCRYPT_MODE and given key. The message is then encrypted and cipher string is generated. After the encryption process of given

message, the length of the generated cipher text multiplied by 2 along with a "/" character, is added to the beginning of that cipher text and LSB replacement process of the image with the whole string is done. The LSBs of each RGB pixel is replaced with the character bits of cipher text until all the cipher text bits are embedded onto the image. The rest of the image bits are left unchanged. However, if the total number of bits in the cipher text is greater than the total number of image's RGB bits, a dialog box, indicating that a larger image is required is displayed. After the embedding process completes, the resulting stego image is displayed on the screen. The stego image can then be sent via email or saved to the operating computer. The emailing process takes in required inputs and sends the email using Gmail SMTP host. If any input like sender's credentials are invalid, an error message will be displayed. By default, the image is saved as a JPG image file. If the user explicitly provides a different extension like PNG, JPEG or BMP, the image is saved accordingly. However, if some other unsupported file extension is entered, the image will still be saved as JPG file.

5.1.3 Implementation of Extraction and Decryption Process

The implementation of extraction process of the hidden message from a stego image is done in two steps, as well. The process takes in a stego image file (generated by the program), and the associated key string as inputs. Firstly, the cipher text embedded onto the stego image is extracted. Starting from the beginning of the image i.e., the very first pixel, the LSBs of every 8 bits (one pixel consisting of 3 bits- RGB) are taken and converted to characters. Upon finding the first "/" character, the process is stopped and the string made by characters leading up to the "/" is separated and taken as integer and divided by 2, indicating the total length of embedded cipher. This length is taken as reference for stopping the extraction process. Then the rest of the characters until the cipher length are extracted in the same way as mentioned earlier. After the cipher length extraction process, the cipher text thus obtained is decrypted using 128-bit AES decryption. Java Base64 decoder API has been used. The decryption process takes the associated 128-bit key string as input. If the user-entered key is not 128-bit i.e., 16 characters, the provided key string is right-padded with the letter "a" in order to reach the required length of 16 characters. However, if the entered key string is invalid or the provided stego image is not a stego image generated by the same program, a dialog box

indicating input error is displayed. The key string is then converted to the required key datatype. The AES cipher instance is taken which is initialized with DECRYPT_MODE and given key. The message is then decrypted and the original message is obtained and displayed on the screen. The user can then save the displayed message as text file on the operating computer's memory or discard it. The text file is saved as a .txt file by default.

5.1.4 Implementation of Client-Server Model

The client-server section of the program is implemented using java socket programming. The server thread is started on a user-entered port number using ServerSocket tag. If the server couldn't be started on given port number, a dialog box indicating the failure is displayed. Whenever a client wishes to establish a connection with the server, the client-side thread is started and the Socket tag is used along with server's port number (entered by user) and the server's IP address (localhost by default). However, if a server at given port and IP is not found, a dialog box indicating the error is displayed. The Socket tag is used to accept the client's connection request, on server side. After the connection has been established, both server and client can send and receive stego images from each other. The maximum number of clients that can be handled by server at once is one. The sender side can generate and send stego images to receiver's side by providing the required inputs. On receiver's side, whenever an incoming image is detected, an image viewer window is prompted automatically. The receiver is then provided with options to either extract the hidden message from the image or to save the image file. The extraction process requires the associated key. Further, after providing the correct key, as used by the sender and extracting the hidden message, the receiver can save the extracted message on operating computer's memory. Once the client side closes the connection to the server, the server will be open for new client requests. Also, at this point, if the server tries to send a stego image, a message indicating that the program is unable to send that image is displayed. Similarly, if the client tries to send an image, after server has been closed, error is detected and the message indicating that the server is inactive is displayed.

5.2 Testing

5.2.1 Unit Testing

- Message encryption and embedding module:

  Test Case 1: String Encryption (using 128-bit AES encryption)

  Input message string: “This is a test string.”

  Input key string: “hello12”

  Expected output: Encrypted string.

  Obtained output:

```
run:
vyoeY0tNi5amaMLoPoU5OMrn7xCV3vQjyZMGsCRzWOw=
BUILD SUCCESSFUL (total time: 2 seconds)
```

Output 1: Encrypted text

  Remarks: String Encrypted successfully.

  Test Case 2: Image’s LSB replacement

  Input image: 800 x 500 jpg image

  Input message string: “test”

  Expected output: Image with LSBs changed with message string’s

Output: 800 x 500 PNG image

Output 2: Stego-image

```
run:
Given message: test
Message's bits: [0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0]
Original Image pixel bits:
01101011 10000111 01011111 01001110 01101000 01000011 00111010 01001111
00101110 01001000 01011000 00111011 00101101 00111010 00011110 00011111
00101001 00001110 00010001 00011011 00000000 00010000 00011101 00000011
00010101 00100100 00001111 00110000 01000011 00110000 01110000 10000010
01110110 01010110 01101011 01011100 01001000 01011011 01000111 00111100
New pixel bits:
01101010 10000111 01011111 01001111 01101000 01000011 00111010 01001110
00101110 01001001 01011001 00111010 00101100 00111011 00011110 00011111
00101000 00001111 00010001 00011011 00000000 00010000 00011101 00000011
00010100 00100101 00001111 00110001 01000010 00110001 01110000 10000010
01110110 01010110 01101011 01011100 01001000 01011011 01000111 00111100
New Image created and saved successfully
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output 3: Image and message bits

Remarks: Message embedded in image successfully.

- Message extraction and decryption module:

  Test Case 3-1: String Decryption (using 128-bit AES decryption)

  Input message string: Output of test case 1

  (vyoeY0tNi5amaMLoPoU5OMrn7xCV3vQjyZMGsCRzWOw=)

  Input key string: “hello”

  Expected output: Decrypted string.

Obtained output:

```
run:
Given String: vyoeY0tNi5amaMLoPoU5OMrn7xCV3vQjyZMGsCRzWOw=
Key: hello
Exception in thread "main" javax.crypto.BadPaddingException: Given final block not properly padded
        at com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:966)
        at com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:824)
        at com.sun.crypto.provider.AESCipher.engineDoFinal(AESCipher.java:436)
        at javax.crypto.Cipher.doFinal(Cipher.java:2165)
        at imagesteganography.test.main(test.java:278)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output 4: Decryption error

Remarks: Invalid Key Error.

Test Case 3-2: String Decryption (using 128-bit AES decryption)

Input message string: Output of test case 1

(vyoeY0tNi5amaMLoPoU5OMrn7xCV3vQjyZMGsCRzWOw=)

Input key string: “hello12”

Expected output: Decrypted string.

Obtained Output:

```
run:
Given String: vyoeY0tNi5amaMLoPoU5OMrn7xCV3vQjyZMGsCRzWOw=
Key: hello12
Decrypted String: This is a test string.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output 5: Decrypted text

Remarks: String decrypted successfully.

Test Case 4: String extraction from image (Up to first 32 RGB bits)

Input image: Output from Test Case 2 (800 x 500 PNG image)

Expected output: Extracted hidden string/message.

Obtained Output:

```
run:
First 32 LSBs: [0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0]
Extracted String: test
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output 6: String extracted from image

Remarks: String extracted successfully.

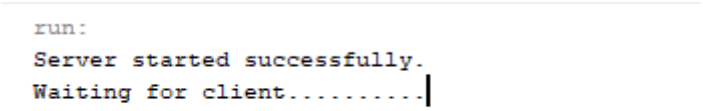
- Client-Server module

Test Case 5: Starting Server Socket

Input port number: 80

Expected output: Server Started and waiting for client message.

Obtained Output:



Output 7: Server started

Remarks: Server socket started successfully.

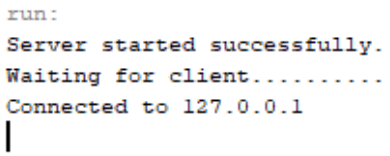
Test Case 6: Client-Server connection

Input port number: 80

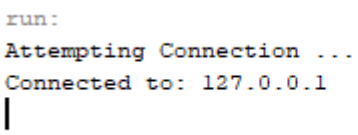
Input Server IP: localhost (127.0.0.1)

Expected output: Client-Server connection established.

Obtained Output:



Output 8: Server connected to client



Output 9: Client connected to server

Remarks: Client-Server connection established successfully.

5.2.2 Integration Testing

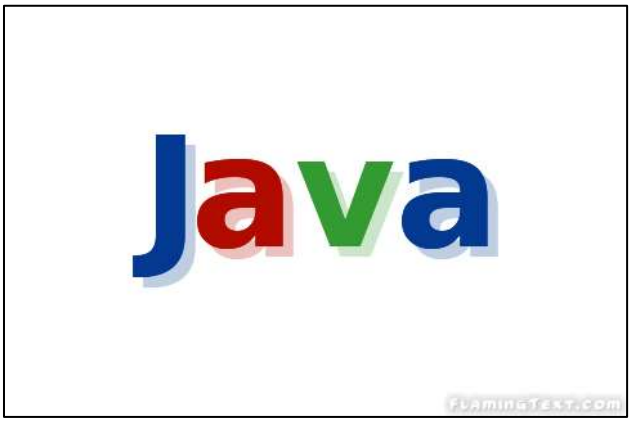
The test cases 1 and 2 were integrated together and tested for output. An image file, a message and a key were provided as inputs. After the encryption process of given message, the length of the generated cipher text along with a "/" character, was added to the beginning of that cipher text and LSB replacement process of the whole string was done. The generated stego image was saved for further testing.

Test Case 7: Encryption and LSB replacement process integration.

Input message: A .txt file containing String- “This is a test.”

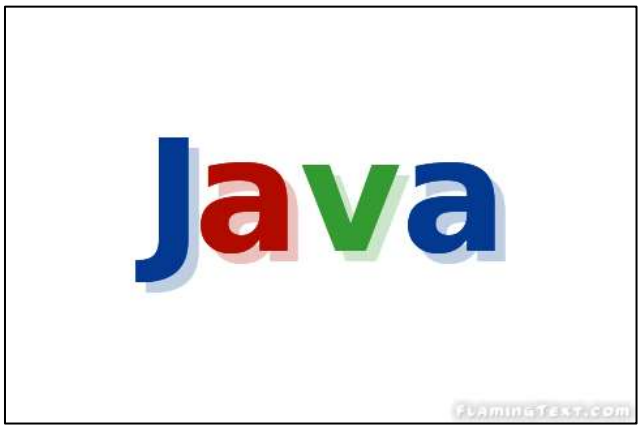
Input Key String: “hello12”

Input Image: 484 x 319 PNG image



Expected output: Stego image

Obtained Output: 484 x 319 PNG image



Output 10: Stego-image

Remarks: String extracted successfully.

Also, the message extraction and decryption process, as referenced on test cases 3 and 4, were tested as a whole. At first, a random key string and stego image produced by test case 7 were provided as inputs, which produced an error. Later the stego image produced by test case 7 and the correct associated key were provided as inputs, then the hidden message was extracted from the image and displayed on the screen.

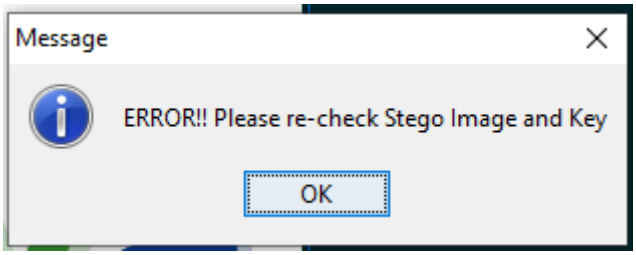
Test Case 8-1: Message Extraction and Decryption process integration.

Input Key String: “hello”

Input Image: Stego image generated on test case 7

Expected output: Hidden message in decrypted form

Obtained Output: Input error.



Output 11: Input error

Remarks: Invalid Key.

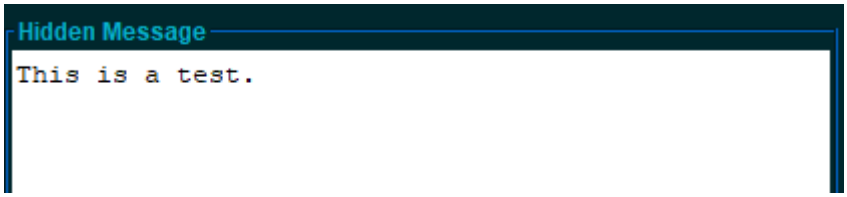
Test Case 8-2: Encryption and LSB replacement process integration.

Input Key String: “hello12”

Input Image: Stego image generated on test case 7

Expected output: Hidden message in decrypted form.

Obtained Output: Hidden message in decrypted form.



Output 12: Extracted and decrypted message

Remarks: Hidden message extracted successfully.

Test Case 9: Creating Client-Server connection and sending stego images

Input port number (on both client and server side): 80

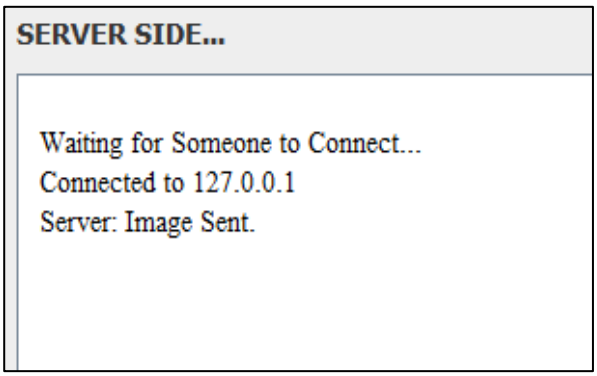
Input message (on server side): “Client-Server test”

Input Key String (on server side): “hello”

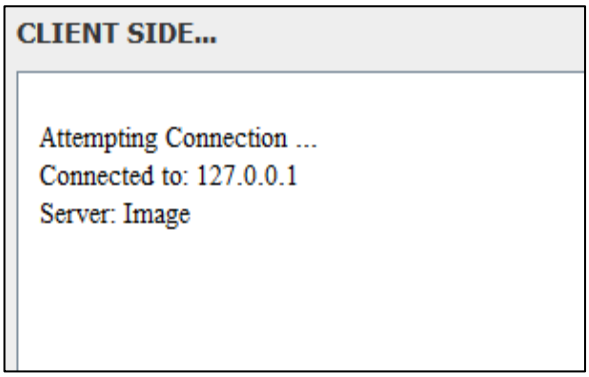
Input image (on server side): 800 x 500 jpg image

Expected output: Image sent from server, received on client side and extracted message

Obtained Output: Image sent from server, received on client side and extracted message

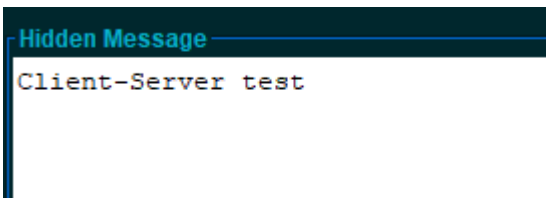


Output 13: Image sent from server



Output 14: Image received on client side

Output 15: Obtained image on client side



Output 16: Extracted and decrypted message on client side

Remarks: Stego image sent, received and message extracted successfully.

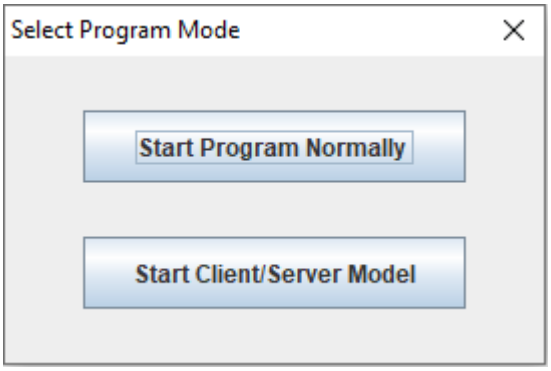
5.2.3 System Testing

On program's normal mode; firstly, to test the message hiding process, the encrypt and embed option was chosen. A cover image file, a key and a message to be hidden were provided as inputs. After being provided with valid inputs, the system was successfully able to generate a stego image. After receiving the stego image file, it was saved to the hard drive. Sending the generated stego image file over email was also successfully tested. Sending the image over the internet required inputs such as sender's email and password as well as the receiver's email and password. The provided email attributes need to be correct and the sender's id should have permission to send emails via windows applications.

After the successful generation of the stego image, it was tested for message extraction. To test the image sent over the email, the image was firstly downloaded using the option provided by Gmail. After choosing the extract and decrypt option in the program, the stego image and associated key were provided as inputs. The program was able to extract the hidden message from the stego image in decrypted form. The hidden message was displayed on the screen. After receiving the hidden message, it was saved to the hard drive as text file.

On the client-server mode of the system; firstly, server was started on a random port. On client side, the same port number was provided with local host as default server IP. After the connection was successfully established, a stego image was generated on server's side by providing necessary inputs. The stego image was then sent to the client where an automatic image viewing frame was prompted. Then by selecting the "extract and decrypt" option provided on the frame, the message hidden on the server side was successfully displayed on the screen.

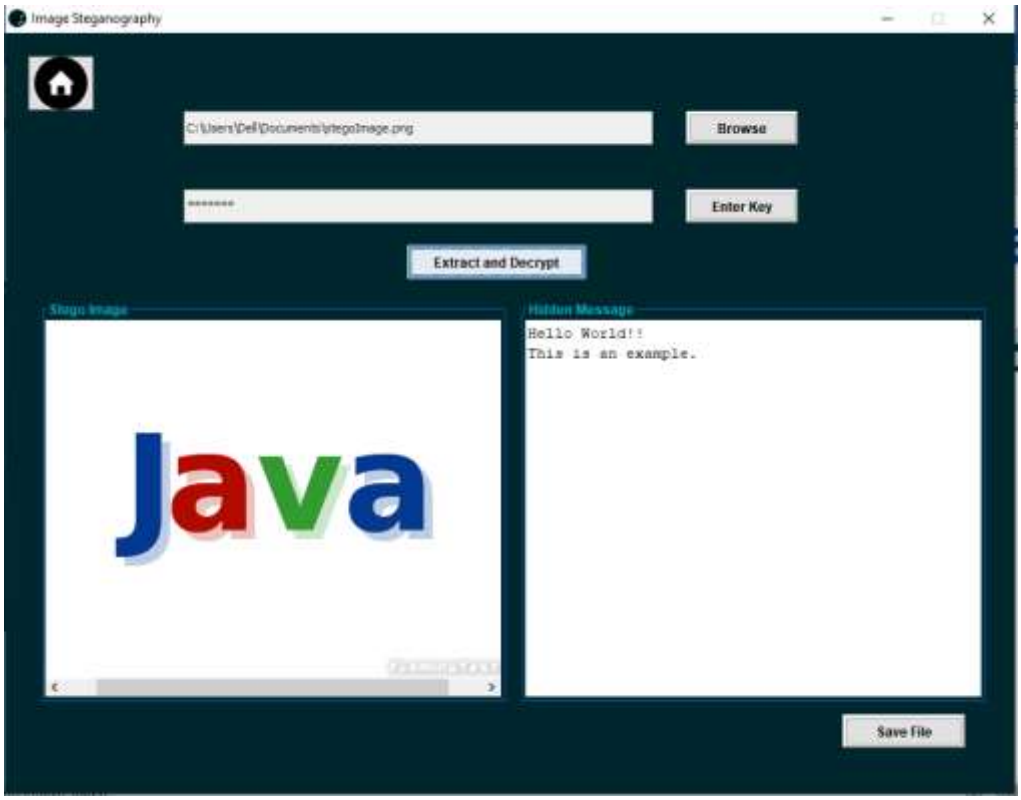
Screenshots



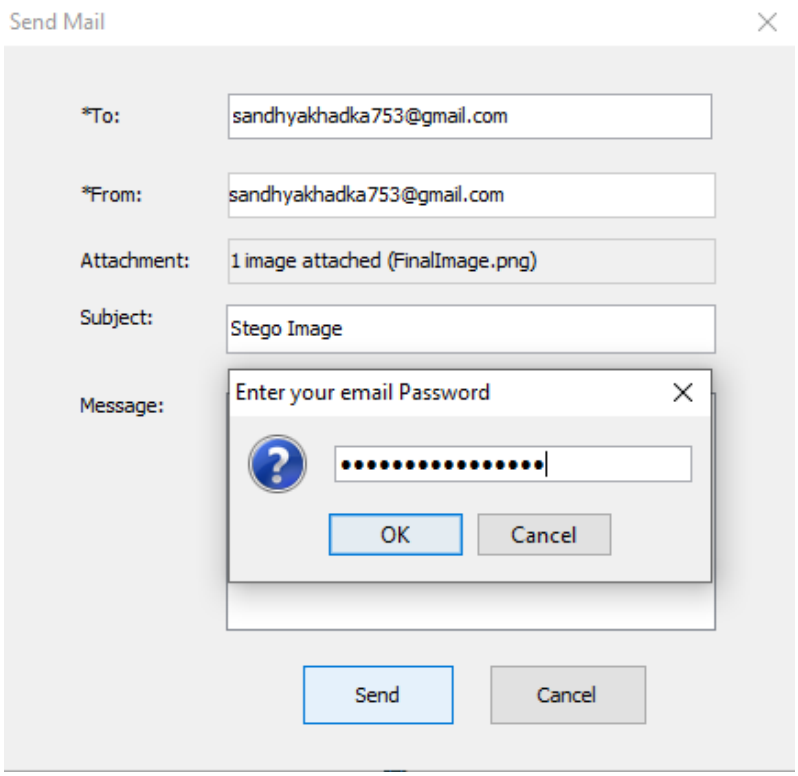
Screenshot 1: Program mode selection

Screenshot 3: Normal mode user Interface

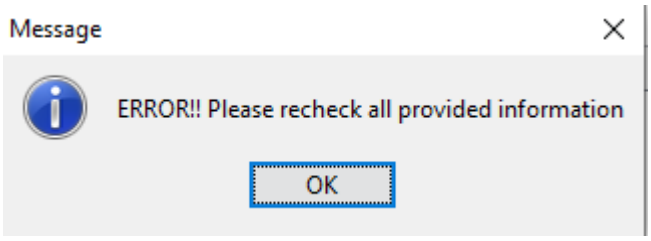
Screenshot 3: "Encrypt and Embed" screen



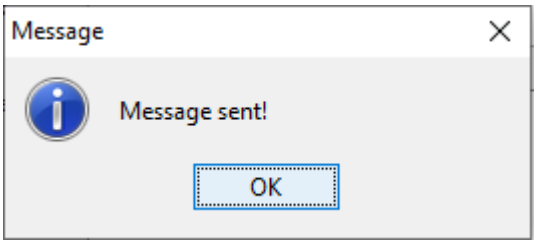
Screenshot 4: "Extract and Decrypt" screen



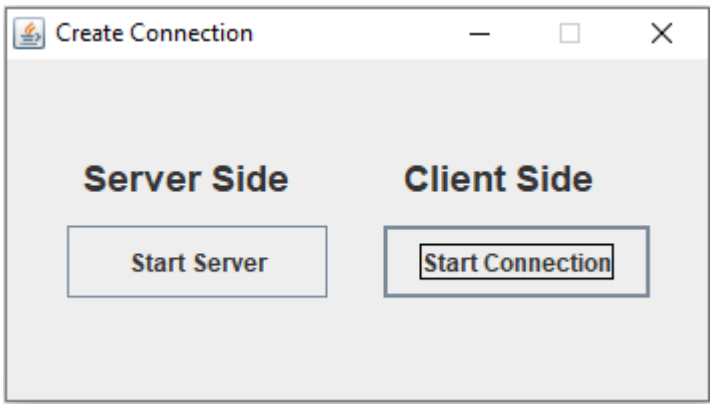
Screenshot 5: Email window



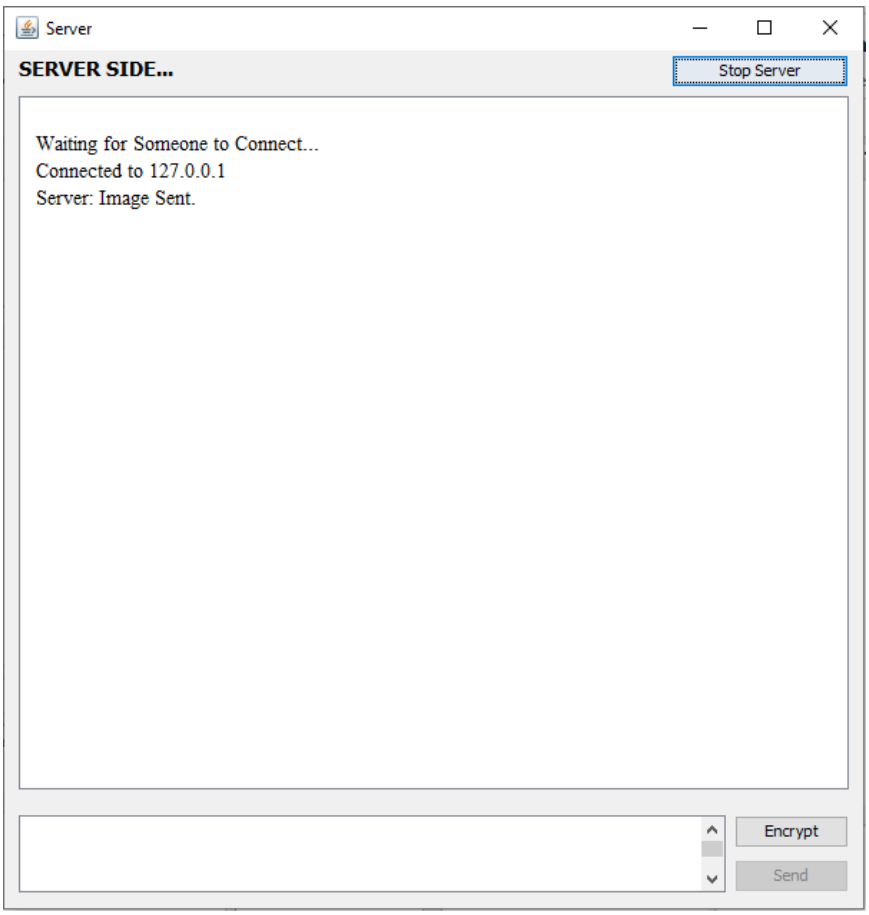
Screenshot 6: Incorrect emailing credentials



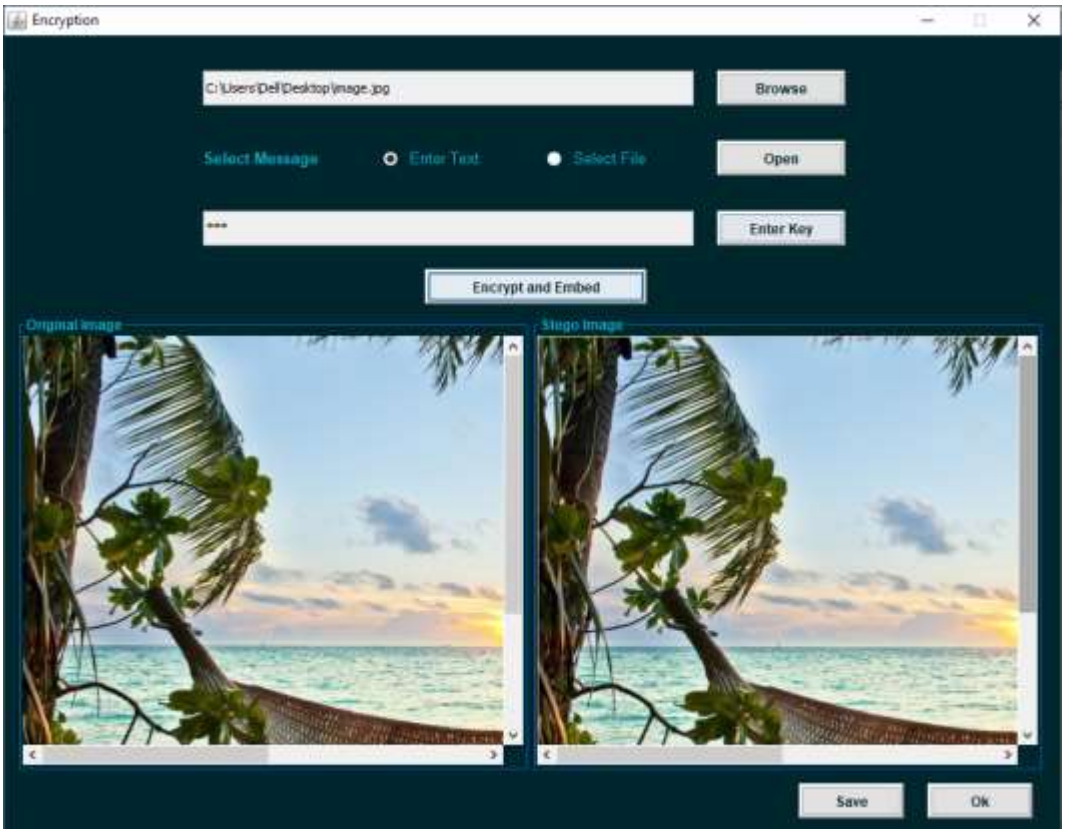
Screenshot 7: Email sent successfully



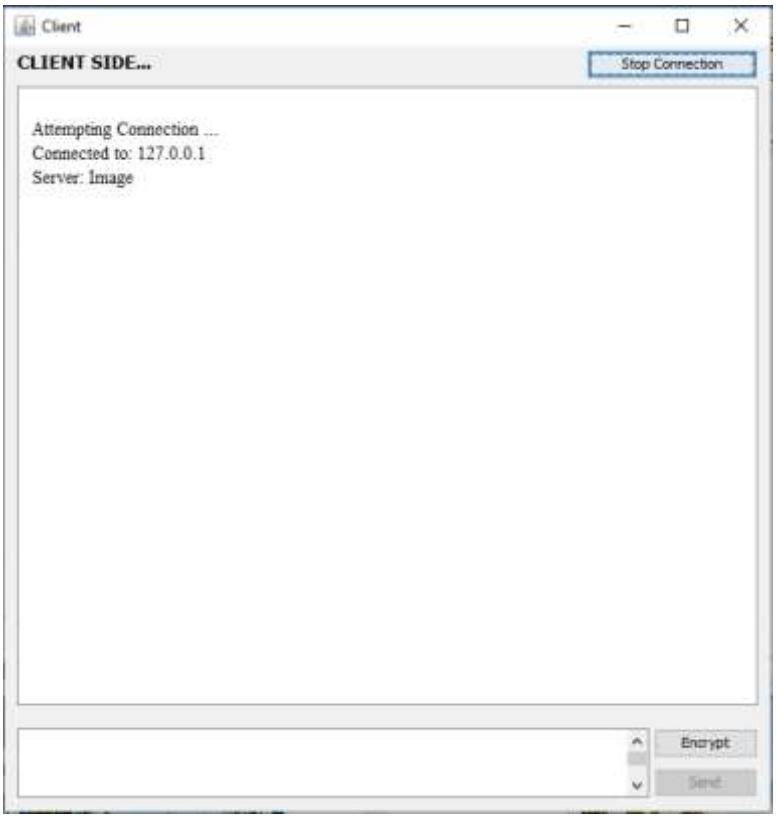
Screenshot 8: Client/Server mode: User Interface



Screenshot 9: Server side

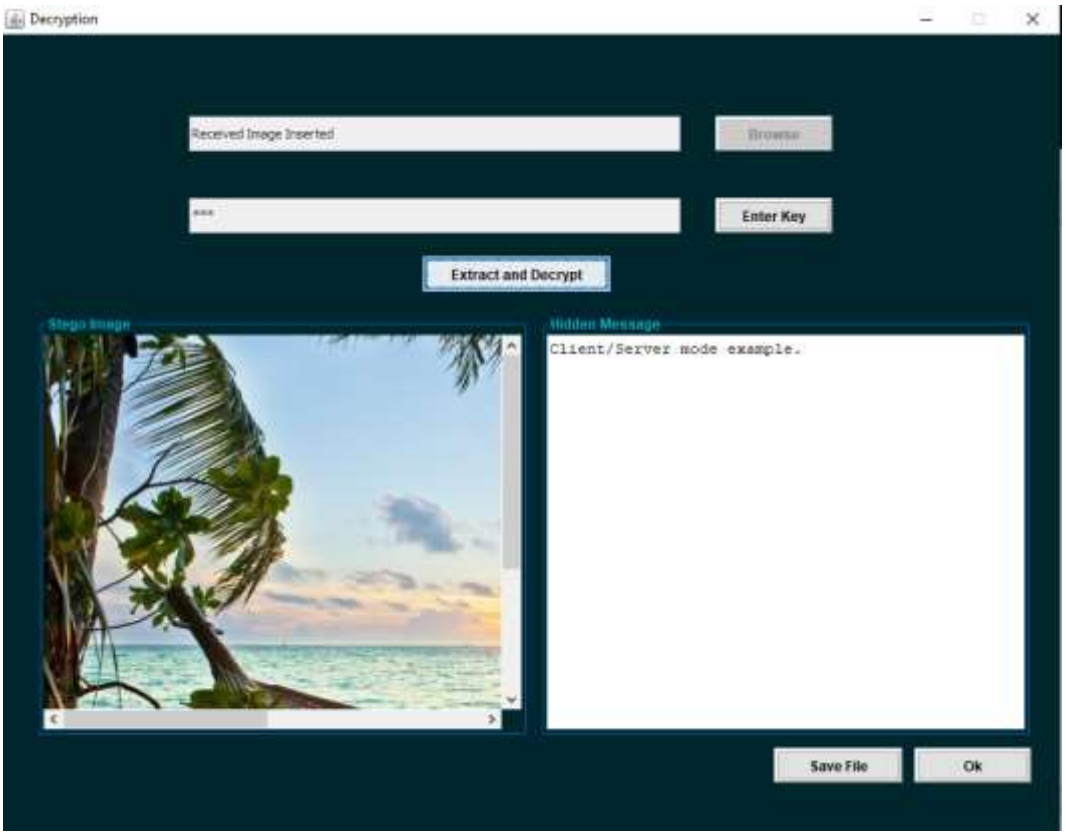


Screenshot 10: Encryption window at Sender (server) side



Screenshot 11: Client side

Screenshot 12: Automatic image viewer at receiver (client) side



Screenshot 13: Hidden message extraction at receiver side

5.3 Analysis

An online histogram generator tool has been used for input and output images' analysis purpose. A histogram is a graphical representation of a frequency distribution. It is the most commonly used graph to show frequency distributions. An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance. An image histogram displays pixel value distribution within an image.

The tool basically creates 256 bins for each color (red, green, blue) and greyscale (luma) intensity. The number of bins is shown on the horizontal axis. The tool then loops through every image pixel and counts the occurrence of each intensity. The counts of occurrences in each bin are then displayed on vertical axis. Counts for each pixel intensity are normalized to range 0 to 255 before they are displayed on the graph.

Example:

Figure 5-2 Cover Image

Figure 5-1 Stego Image

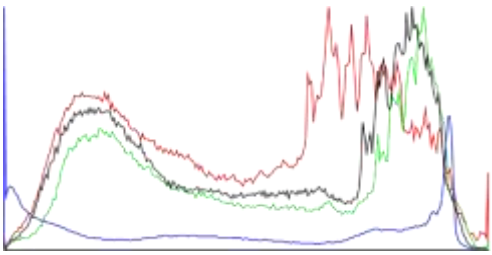


Figure 5-3 Cover Image Histogram

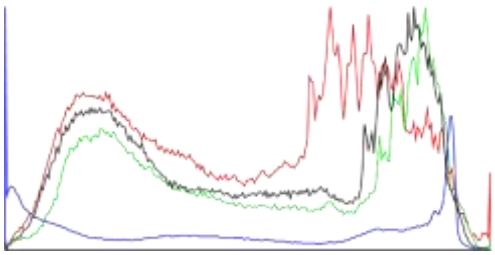


Figure 5-4 Stego Image Histogram

Chapter 6. Conclusion and Future Enhancement

6.1 Conclusion

The final product of the project is a system that can take a cover image file, hidden message and a secret key as input and provide a stego-image as output as well as can extract the secret message hidden in the stego-image when provided with a key and the corresponding stego image. On sender's side, the hidden message is first encrypted with AES technique with the help of provided secret key and the encrypted text is then embedded onto the cover image file to produce stego-medium. The embedding process follows image's LSB replacement algorithm. On receiver's side the process includes extraction of the encrypted message then its decryption with original key for obtaining the original message. During extraction of hidden cipher text, the bits at LSB are extracted and finally through a series of steps, converted to the encrypted text.

6.2 Future Enhancement

If needed, the system will be open for future enhancements. The future enhancements might include audio/video steganography with improved algorithms.

References

[1] S. Singh, Literature Review On Digital Image Steganography and Cryptography Algorithms, 2015.

[2] A. Soni, J. Jain and R. Roshan, "Image steganography using discrete fractional Fourier transform," *Intelligent Systems and Signal Processing (ISSP),* 2013.

[3] Akhtar, N.; Johri, P.; Khan, S., "Enhancing the Security and Quality of LSB Based Image Steganography," *Computational Intelligence and Communication Networks (CICN),* 2013.

[4] K. Joshi and R. Yadav, "A New LSB-S Image Steganography Method Blend with Cryptography for Secret Communication," in *Third International Conference on Image Infonnation Processing*, 2015.

[5] S. Sugathan, "An Improved LSB Embedding Technique for Image Steganography," in *2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2016.

[6] L. Xiaoqin, L. Wei, C. Xiuxin, Z. Xiaoli and D. Zhengang, "Application of the Advanced Encryption Standard and DM642 in the image transmission system," in *7th International Conference on Computer Science & Education (ICCSE)*, Melbourne, 2012.

[7] M. R. Garg, "Comparison Of Lsb & Msb Based Steganography In Gray-Scale Images," *International Journal of Engineering Research and Technology (IJERT)*, vol. 1, no. 8, 2012.

[8] E. Walia, P. Jain and Navdeep, "“An Analysis of LSB & DCT based Steganography," *Global Journal of Computer Science and Technology,* vol. 10, no. 1, 2010.