# Digital Image Steganography

*A Project Report Submitted for the Partial Fulfillment of the requirements for the*
**MAJOR PROJECT**

*Of degree of*

**Bachelor of Computer Application (BCA)**

*By*


**Probal Deep Saikia**

Roll No. 17BCA008

Registration No. 17990933


Under the Guidance of

**Ponjit Borgohain**



**Department of Computer Science**

**North Lakhimpur College (Autonomous)**

**Estd. 1952**

**Khelmati – 787031, North Lakhimpur, Assam**


**2020**


**North Lakhimpur College (Autonomous)**

Estd. 1952

[*Affiliated to Dibrugarh University*]

# CERTIFICATE

This is to certify that **Probal Deep Saikia** bearing Roll No. **17BCA008** and Registration No. **17990933** has prepared the project report entitled ***Digital Image Steganography***.

The project is the result of his effort and endeavours. The project is found worthy of acceptance for the award of Bachelor of Computer Application. He worked under the supervision and guidance of **Ponjit Borgohain**.

Signature of Head of Department
(Ranjit Paul)

Date:

Department of Computer Science

North Lakhimpur College
(Autonomous)

**North Lakhimpur College (Autonomous)**

Estd. 1952

[*Affiliated to Dibrugarh University*]

# CERTIFICATE

This is to certify that **Probal Deep Saikia** bearing Roll No. **17BCA008** and Registration No. **17990933** has prepared the project report entitled ***Digital Image Steganography***. The work is done under my supervision and guidance.

The project is the result of his effort and endeavours. The project is found worthy of acceptance for the award of Bachelor of Computer Application.

Signature of Supervisor

(Ponjit Borgohain)

Date: -

Department of Computer Science

North Lakhimpur College
(Autonomous)


**North Lakhimpur College (Autonomous)**

Estd. 1952

[*Affiliated to Dibrugarh University*]

# CERTIFICATE

---

This project entitled **Digital Image Steganography** submitted by **Probal Deep Saikia**, Roll No. **17BCA008** and Registration No. **179909933** for the partial fulfilment of the requirements of the degree of Bachelor of Computer Application at North Lakhimpur College (Autonomous) has been examined by us and found to be satisfactory.

| | |
|---|---|
| Internal Examiner | External Examiner |
| Date: | Date: |

# ABSTRACT

The growth of high-speed computer network, and internet in particular has increased the ease of communication, and the use of digital formatted data. Compared to Analog media, digital media offers many different advantages such as high quality, easy editing, high reliability copy, compression, time to send it from sender to recipient etc. Therefore, Data and Information Security becomes an integral part of Data Communication. But these type of advancements in the field of data communication has increased the fear of getting data stolen at the time of transferring. In order to address this issue, numerous method of protecting that data have been evolved and one of method is steganography. Steganography plays an important role in the field of Data and information Security. Steganography is the science of embedding secret messages in such a way that no one, except the intended sender and receiver suspects that there is a secret message. It is the art of hiding the fact that a communication is taking place in plain sight. This Project is intended to develop a software application for Digital Image Steganography. Digital Image Steganography is a type of image steganography used to hide data and information inside digital images (specifically for Portable network Graphics (PNG) image format). This project makes use of The Least Significant Bit (LSB) insertion method. This method is the most common and easiest method for embedding messages in an image with high capacity. The embedding message can only be retrieved using proper techniques. For more secure approach, it also uses Cryptography techniques of both Synchronous and Asynchronous types.

# DECLARATION

I hereby, declare that the project report entitled "***Digital Image Steganography***" is an authentic work developed by me at *North Lakhimpur College (Autonomous)* for partial fulfilment of Major Project work of the degree of Bachelor of Computer Application under North Lakhimpur College (Autonomous).

I also declare that, any or all contents incorporated in this project report has not been submitted earlier either in this University or any other university/institute for any form of award of any degree or diploma.

Signature of Student

(Probal Deep Saikia)

Date: -

# ACKNOWLEDGEMENT

I, from the core of my heart, want to express my indebtedness to all persons involved in bringing out my successful completion of the project work entitled "***Digital Image Steganography***" undertaken by me as a prerequisite towards partial fulfillment for the award of Bachelor of Computer Application at North Lakhimpur College (Autonomous).

At the very outset, I would like to express my very great appreciation and thanks to my project guide Ponjit Borgohain Sir of Department of Computer Science, North Lakhimpur College. You have been a tremendous mentor for me. I would like to thank you for your edifying guidance and support for me in this Project. Your endless patience, encouragement, expert advice and above all your friendly behavior towards me have made it possible to complete this project. I would like to express my gratitude especially to Ranjit Paul Sir & Ponjit Borgohain Sir for letting me choose this project.

Last but not least, I would also like to thank to my classmates and all the members of Computer Science Department for timely help and inspiration for completion of the project.

I dedicate this project to those who believed in me, my parents and my family.

Probal Deep Saikia

# Contents

# CHAPTER 1: INTRODUCTION

## 1.1 BACKGROUND

Steganography is the practice of concealing information and data in plain sight. Digital image steganography is a type of image steganography in which data and information is embedded inside a digital image. There are various methods used for digital steganography. One such method is The Least Significant Bit (LSB) insertion is a method of digital steganography. It is the most common and easiest method for embedding message in an image. In a Coloured Image of 24 bits Color Depth, each color in a pixel is represented with 8 bits. The last bit of each color, changes to which will cause a minimum impact on the pixel is called The Least Significant bit. This helps in storing data. When using LSB method, all or some of the LSB inside the image is replaced with the bits of the message. This project is an application of digital steganography using LSB technique. To make the embedded information more secure, some Cryptography technique are also used to encrypt and decrypt the message.

## 1.2 OBJECTIVES

The aim of this project is to develop an Application Software, with simple and user friendly interface to hide and retrieve message inside any Image file (Specifically for Portable Network Graphics (.PNG) files) with a support for range of Devices and Operating Systems.

## 1.3 PURPOSE AND APPLICABILITY

### 1.3.1 PURPOSE

The purpose of this project is to limit unauthorized access and provide better security during message transmission.

### 1.3.2 APPLICABILITY

This project is applicable to, but not limited to, the following areas.

- Secretive Data transmission and Confidential communication
- Military and law enforcement agencies.
- Systems with Access control for digital content distribution
- Media Database Systems

## 1.4 ACHIEVEMENTS

After working on this Major Project:

- It provided me with the opportunity to demonstrate a wide range of Java skills and knowledge.
- It helped me gain knowledge of Data and Information Security.
- It helped me develop Stenographical and Crypto graphical skills.
- It helped me gain more experience in software development and distribution.
- It helped me gain to increase experience for future projects

# CHAPTER 2: SURVEY OF TECHNOLOGIES

## 2.1 OPEN SOURCE SOFTWARE:

Open-source software (OSS) is a type of computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose. Open-source software may be developed in a collaborative public manner. According to scientists who studied it, open-source software is a prominent example of open collaboration. The term is often written without a hyphen as "open source software".

## 2.2 USED TECHNOLOGIES:

### 2.2.1 JAVA AS PROGRAMMING LANGUAGE:

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

### 2.2.2 JCOMMANDER AS COMMAND LINE PARSER:

JCommander, created by Cédric Beust, is an annotation-based library for parsing command-line parameters. It can reduce the effort of building command-line applications and help us provide a good user experience for them. With JCommander, we can offload tricky tasks such as parsing, validation, and type conversions, to allow us to focus on our application logic.

### 2.2.3 JAVAFX AS GUI FRAMEWORK:

JavaFX is a software platform for creating and delivering desktop applications, as well as rich Internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE, but both will be included for the foreseeable future. JavaFX has support for desktop computers and web browsers on Microsoft Windows, Linux, and macOS.

#### 2.2.3.1 WHY SHOULD WE USE JAVAFX?

- JavaFX can be styled with CSS, whereas Swing cannot be.

- JavaFX makes programmers' life easy by offering "JavaFX scene builder" with which programmers can create GUI controls like buttons via drag and drop.
- JavaFX supports Java Lambda expressions in action-triggering events unlike Swing.
- JavaFX has built-in data visualization such as 2D charts

### 2.2.4 INTELLIJ IDEA AS IDE:

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

#### 2.2.4.1 WHY SHOULD WE USE INTELLIJ IDEA?

- **Deep intelligence** : After IntelliJ IDEA has indexed your source code, it offers a blazing fast
  And intelligent experience by giving relevant suggestions in every context: instant and clever code completion, on-the-fly code analysis, and reliable refactoring tools.

- **Out-of-the-box experience**: Mission-critical tools such as integrated version control systems and a wide variety of supported languages and frameworks are all to hand — no plugin hassle included.

- **Smart code completion**: While the basic completion suggests names of classes, methods, fields, and keywords within the visibility scope, the smart completion suggests only those types that are expected in the current context.

- **Productivity booster**: The IDE predicts your needs and automates the tedious and repetitive development tasks, so you can stay focused on the big picture.

# CHAPTER 3: REQUIREMENTS AND ANALYSIS

## 3.1 PROBLEM DEFINITION

The problem is to implement a Steganography function which can embed encrypted messages in any image file and can retrieve and decrypt the message. This problem can be further divided into two problems, Encoding and Decoding,

1) Encoding: This can be divided into two problems. First, the message to be embedded should be encrypted using encryption algorithms. Second, the encrypted message should be converted to binary and inserted in the LSBs of the pixels the cover image bit by bit. Also the information should be stored in such a way that when the Decoding part can identify the LSBs which are used to store data.
2) Decoding: This can further be divided into three problems. First, this function should identify if a given image a stego image or not if it is a stego image than the next parts kicks in. Second, The LSBs which were used for storing data are extracted and converted into original encrypted message format. Third, the encrypted message should be decrypted.

## 3.2 REQUIREMENTS SPECIFICATION

This is the description of the system to be developed.

- User should be able select the cover image.
- User should be able to enter the Message to be embedded.
- User should be able to select encryption techniques to use from a list.
- The system should give user information on encoding Success or Failure message.
- User should be able to select any stego image processed from the same system.
- With right authentication system should retrieve and decrypt and show the message that was hidden inside the stego image.

## 3.3 DESIGN REQUIREMENTS

- The system should have user-friendly GUI with two options Encode, Decode.
- The interface should be easy for everyone.

## 3.4 SOFTWARE AND HARDWARE REQUIREMENTS

### 3.4.1 SOFTWARE REQUIREMENTS:

- Operating System: Microsoft Windows 7/8/10 (32-bit or 64-bit).

- JAVA: Open JDK 11 or later
- OpenFX 11 or later
- Gluon Scene Builder
- Jcommander library 1.78 or later
- Jfoenix library 9.0.10 or later
- Jetbrains annotations 16.0.2 or later
- IntelliJ IDEA 19.3.3 or later
- Git 2.18 or later

### 3.4.2 HARDWARE REQUIREMENTS:

#### 3.4.2.1 FOR INTELLIJ IDEA:

According to IntelliJ Idea's Wikipedia page: https://en.wikipedia.org/wiki/IntelliJ_IDEA

| | |
|---|---|
| **RAM** | 2 GB RAM minimum, 8 GB RAM recommended |
| **Disk space** | 2.5 GB and another 1 GB for caches minimum, solid-state drive with at least 5 GB of free space recommended |
| **JDK Version** | Add support for Java 14 |
| **JRE Version** | JRE 11 |
| **Screen resolution** | 1024×768 minimum screen resolution. 1920×1080 is a recommended screen resolution. |

#### 3.4.2.2 FOR JAVA:

According to java official website https://java.com/en/download/help/sysreq.html

| | |
|---|---|
| **RAM**: | 128 MB minimum. |
| **Disk space:** | 124 MB minimum, for JRE; +2 MB for Java Update. |

#### 3.4.2.3 FOR JAVAFX:

| | |
|---|---|
| **Memory** | 1 GB of RAM (2 GB recommended) |
| **Disk Space** | 150 MB minimum of free disk space |
| **Screen Resolution** | 1024x768 pixels (1280x800 recommended) with 16-bit video card |

## 3.5 CONCEPTUAL MODELS

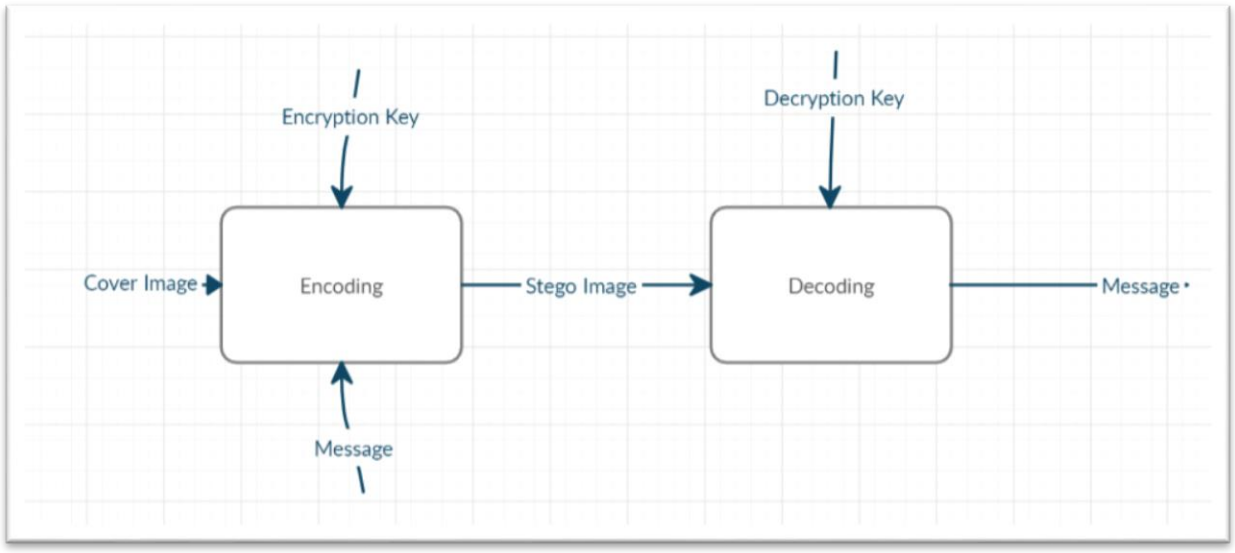



Fig. Conceptual Model

# CHAPTER 4: SYSTEM DESIGN

## 4.1 BASIC MODULES

### 4.1.1 ENCODER:

This module will be used to insert the given message in the selected cover image. At first it will check if the capacity of the give image is less than the size of the message. And if the capacity is enough, then it will generate a variable as a header containing the of message length, Encryption type used, the color channels, and the number of LSB. The header will also contain a special pattern which will be used to detect if the image has been processed by the same encoder. This header will be of fixed size and will not be encrypted. It will be inserted in the cover image bit by bit to some LSBs in some predefined color channels. After that the message will be converted to binary and inserted in cover image.

### 4.1.2. DECODER:

This module will first decode the header which was encoded in the cover file and matched the special pattern with the one stored. If it matches, then the selected image is a stego image processed with the encoder. Than the decoder will proceed further and check the LSBs and Color channels used of encoding. After that the message will be decoded.

## 4.2 CONTEXT DIAGRAM

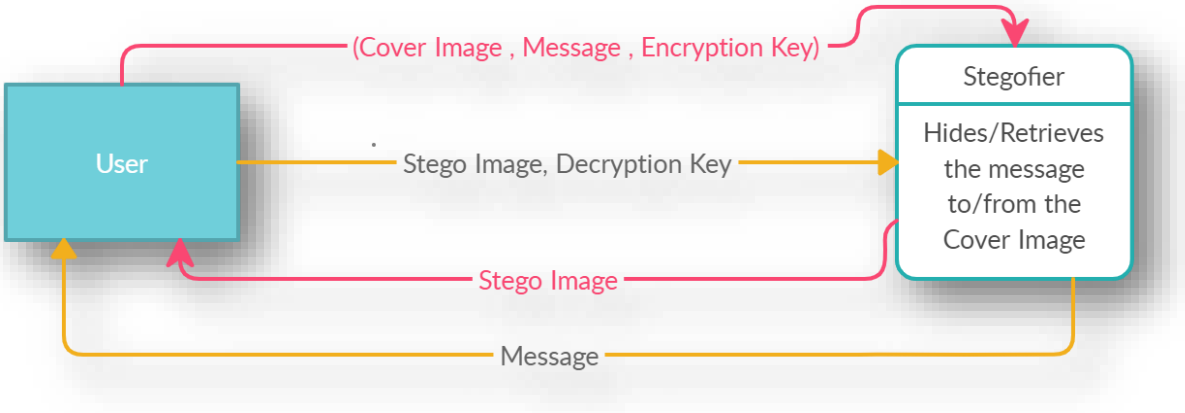



Fig. Context Diagram or Level – 0 Data Flow

## 4.3 LEVEL -1 DATA FLOW DIAGRAM

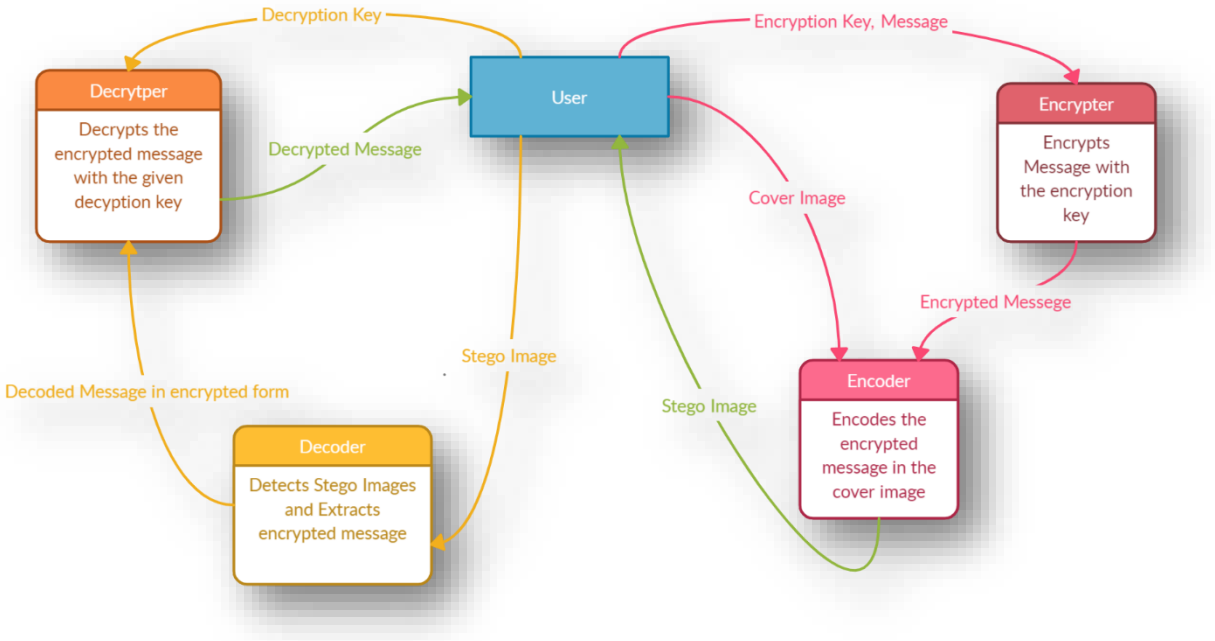



Fig. Level -1 Data Flow Diagram

## 4.4 FLOWCHARTS

### 4.4.1 FLOW CHART FOR LSB INSERTION ALGORITHM

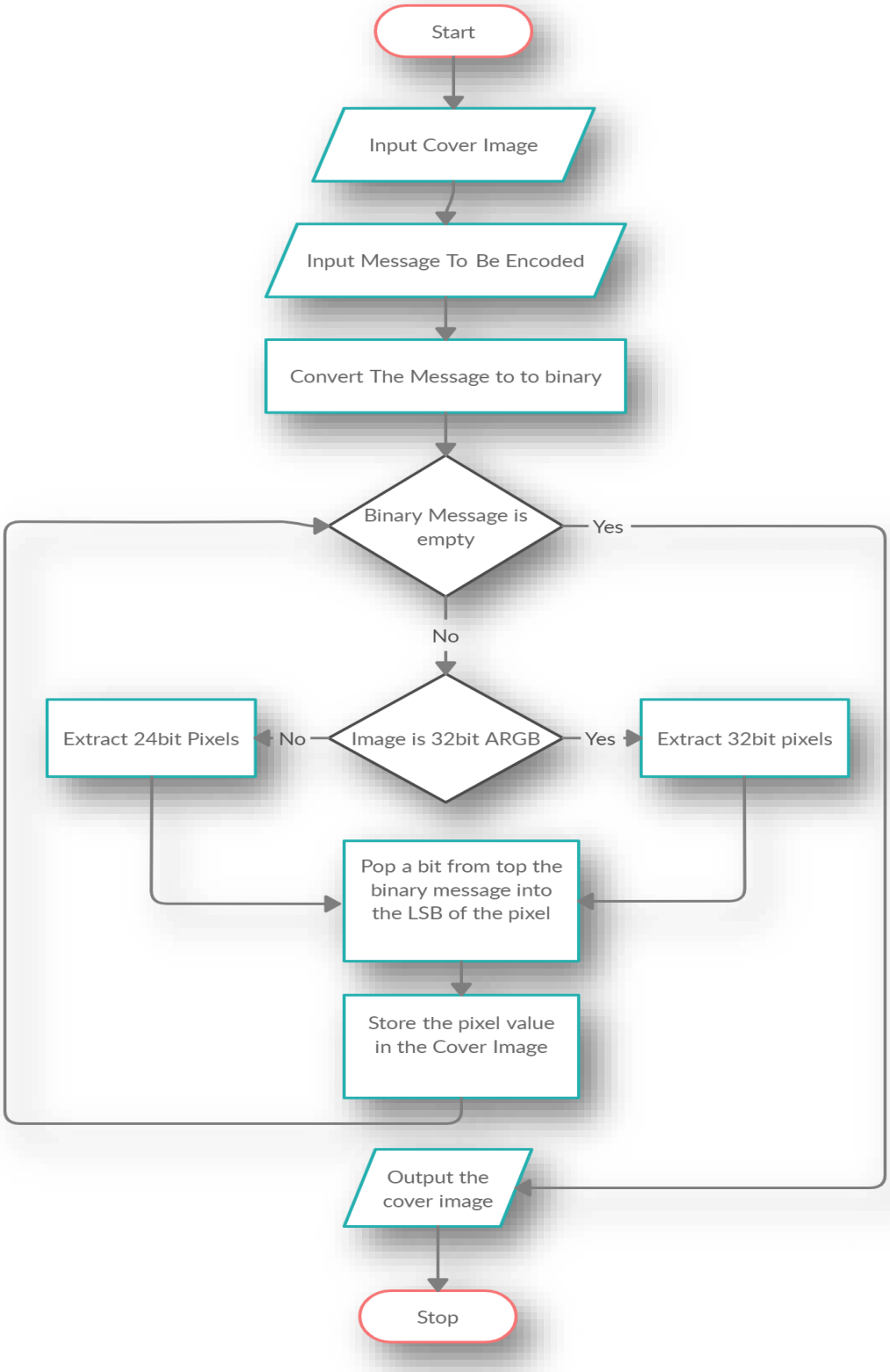



Fig. Flow Chart for LSB insertion algorithm

### 4.4.2 FLOWCHART FOR LSB EXTRACTION ALGORITHM

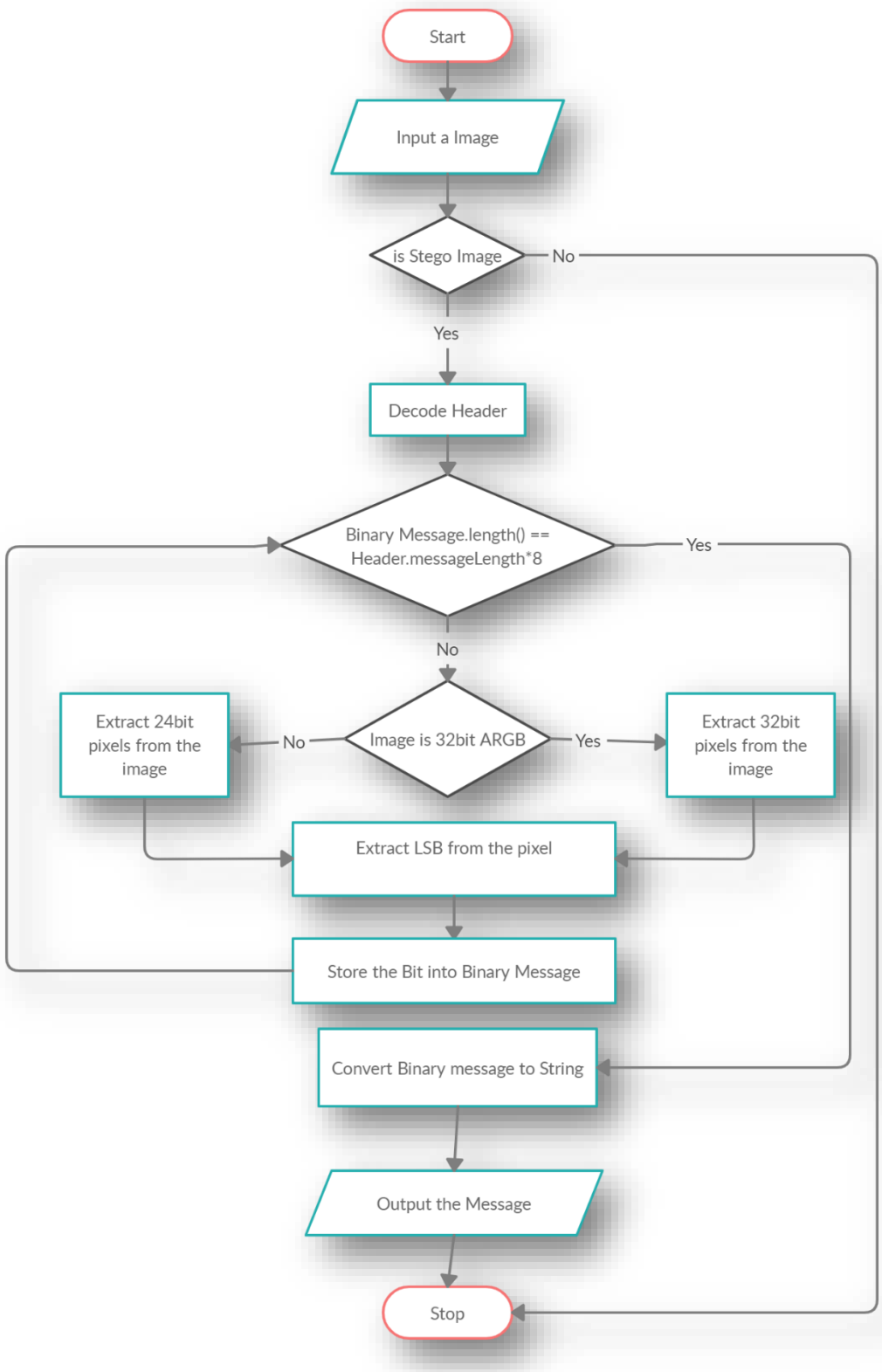



Fig. Flow Chart for LSB extraction algorithm

### 4.4.3 FLOWCHART FOR AES 256 ALGORITHM

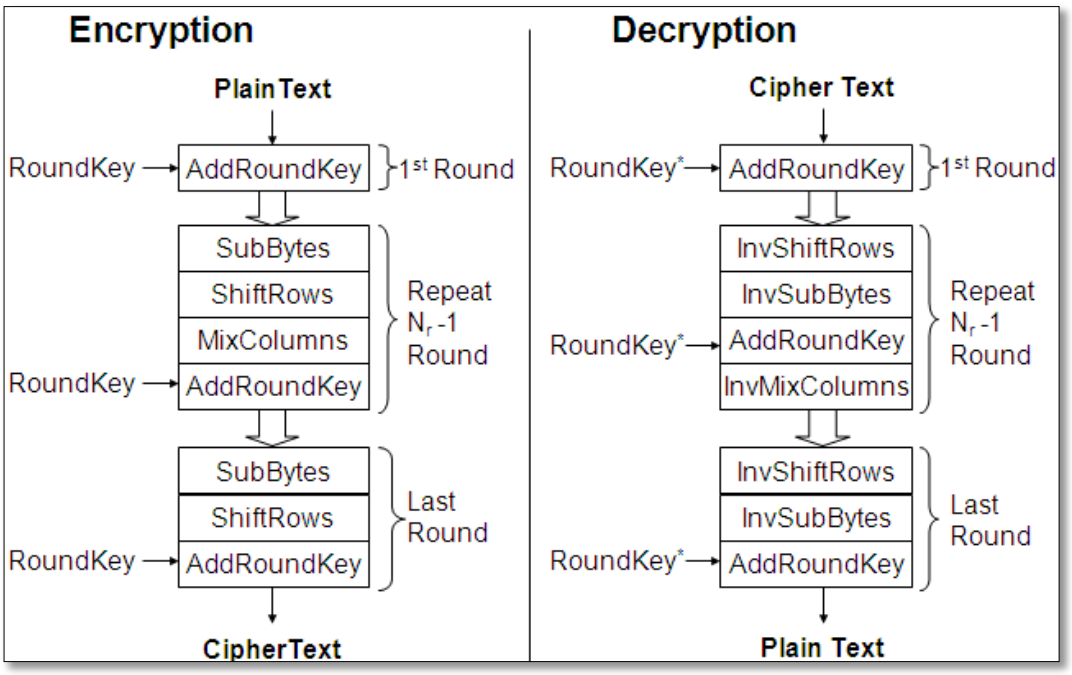



Fig. Flow Chart for AES 256 Algorithm

### 4.4.4 FLOWCHART FOR TRIPLE DES ALGORITHM

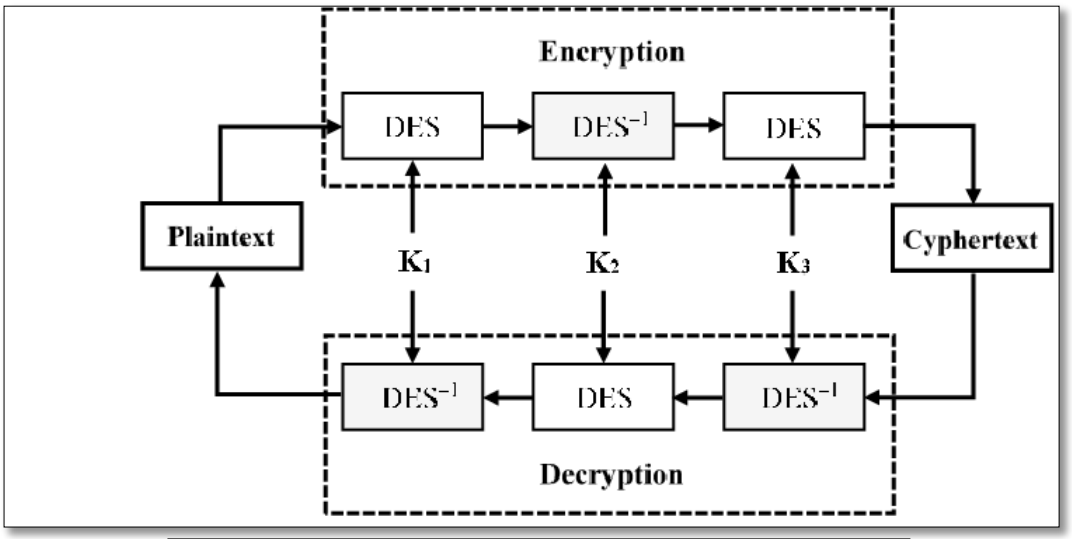



Fig. Flowchart for Triple DES algorithm

### 4.4.5 FLOWCHART FOR RSA ALGORITHM

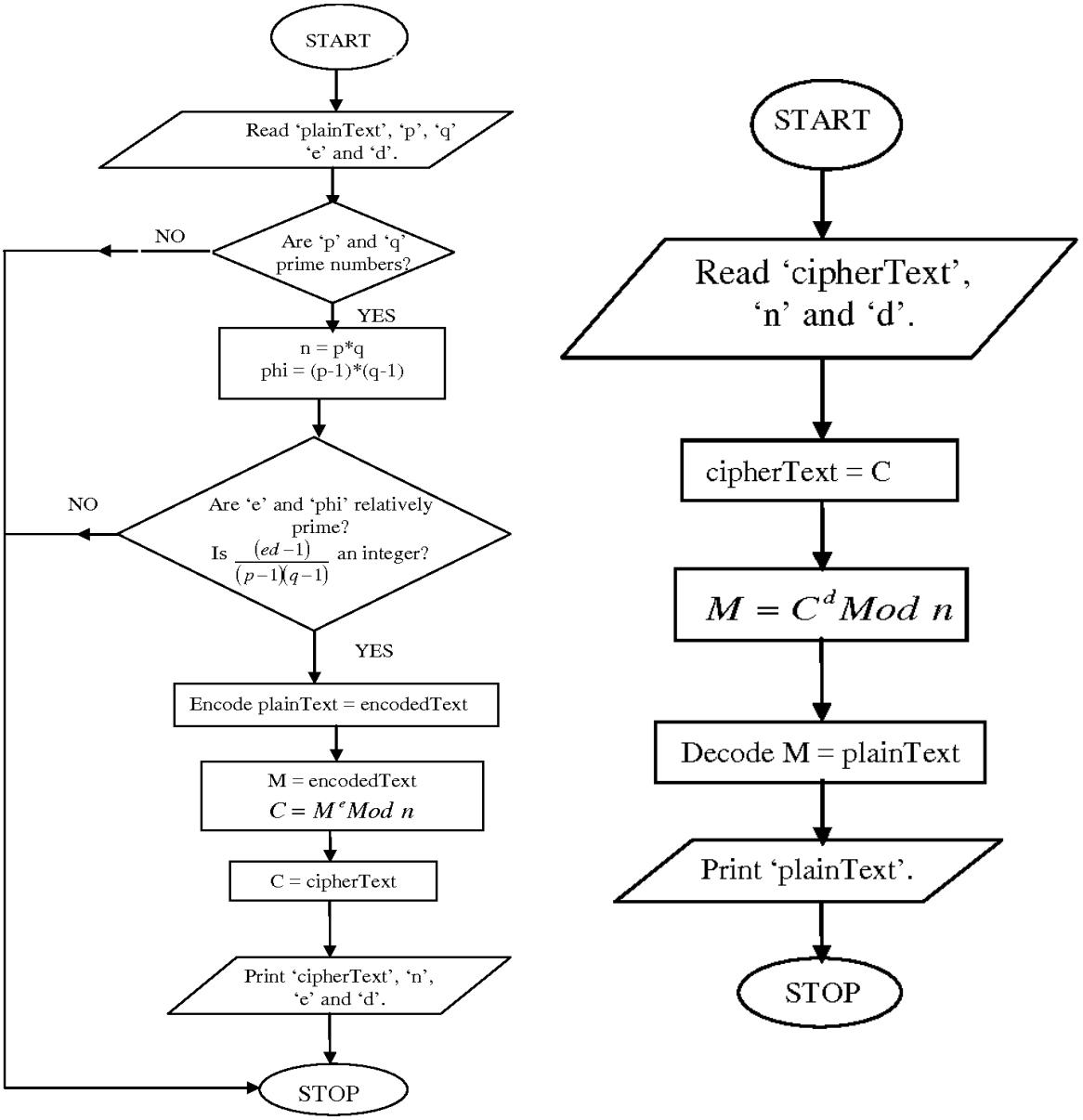



Fig. Flowchart for RSA algorithm (Left: Encryption, Right: Decryption)

## 4.5 GUI DESIGN:

### 4.5.1 ENCODER LAYOUT DESIGN:

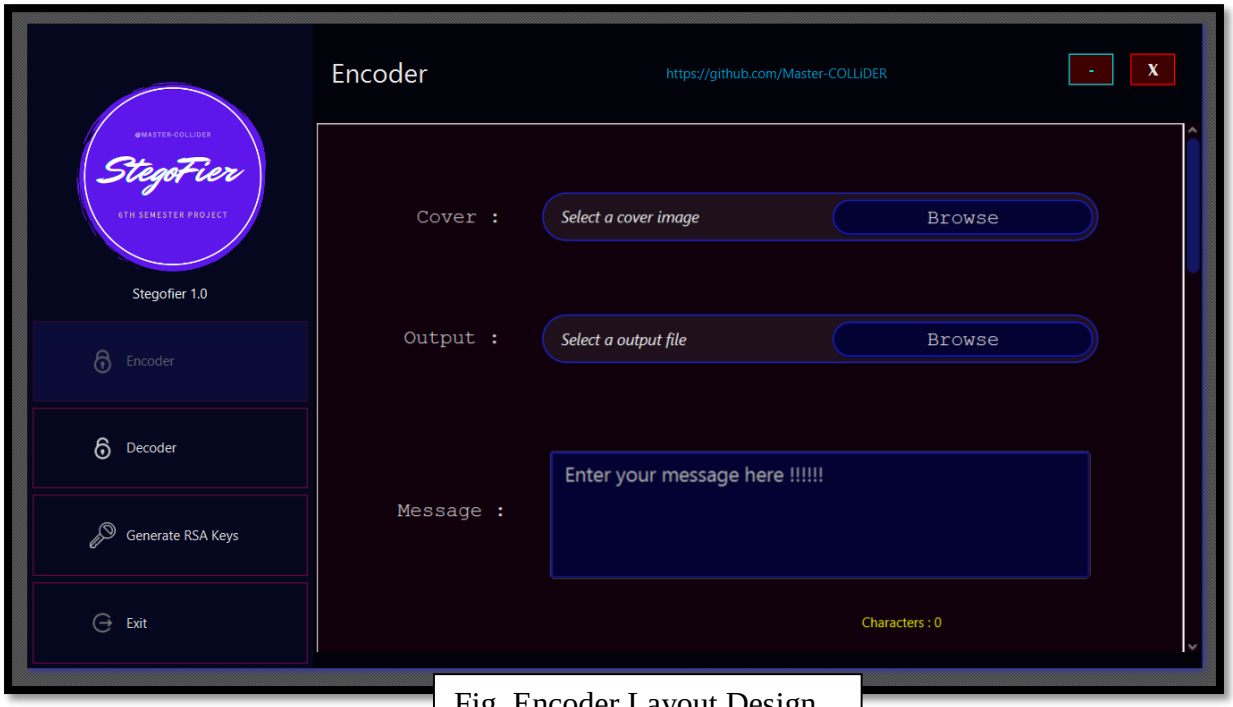



Fig. Encoder Layout Design

### 4.5.2 DECODER LAYOUT DESIGN:

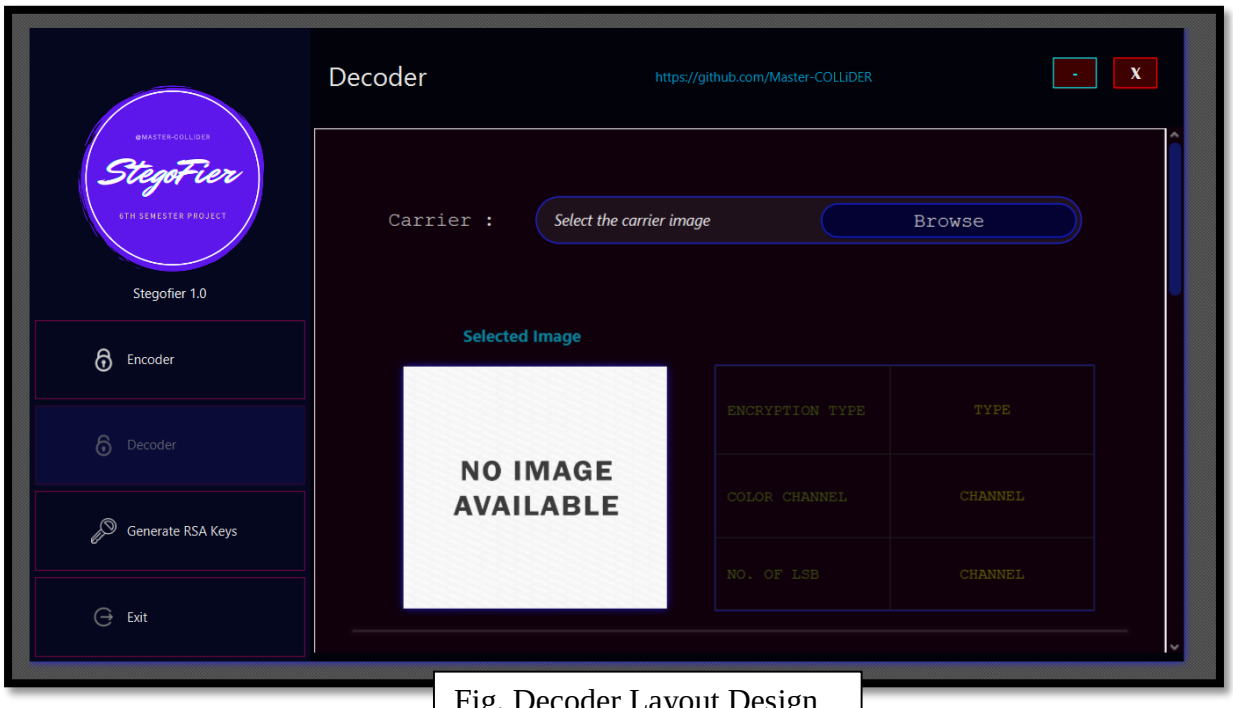



Fig. Decoder Layout Design

## 4.6 DESIGN PHASE CONCLUSION

Hence we can conclude that the design phase of the SIS give us the information of all the processes Used in the project and their relation.

# CHAPTER 5: IMPLEMENTATION AND TESTING

## 5.1 IMPLEMENTATION APPROACHES

### 5.1.1 IMPLEMENTATION OF GRAPHICAL USER INTERFACE (GUI):

GUI windows of the systems are managed as an object of class named “Stage”. Each Stage can preview multiple views but only one at a time. The first Stage that is shown is called the Primary Stage. The views in JavaFx are called “Scene”. Each scene can have multiple child elements such as, Button, Text Box, Labels, Checkbox, and Radio Buttons etc. The scenes can also be inflated with an “fxml” layout made with scene builder. Each Fxml layout has its own java controller class controlling the functions of the fxml elements.

The implementation process is done in the following steps,

- The splash_screen.fxml is the first layout that is shown for a certain period of time after executing the application.
- After the Splash Screen is closed the Encoder_Layout.fxml is loaded in the PrimaryStage.
- In each this layout there is a sidebar which contain a button bar to load other views (Decoder_Layout.fxml, RSA_Key_Pair_Generator.fxml).
- The main purpose of the encoder layout is to make is easy for users to encode messages in a cover image.
- The encoder layout also contains a function for the encryption of code before encoding.
- The decoder layout is used to extract and display the messages from a stego image.
- The user can also decrypt the message if encrypted in decoder layout.
- The RSA key pair generator layout is used to generate RSA key pairs and use it to encrypt and decrypt messages using RSA encryption algorithm.

### 5.1.2 IMPLEMENTATION OF COMMAND LINE INTERFACE (CLI):

The Command line interface and the Command line parameters are parsed using a java library named “Jcommander”. The CLI is for the advance users. It is very powerful and faster than the graphical interface. The CLI can be used with the help of commands and options given in the below Picture.

Picture: Command Line Interface

```
r "StegofierFX.jar"  -h
Usage: Stegofier [options] [command] [command options]
  Options:
    -cli, --command-line-interface
      Open StegofierFx in Command line Interface
      Default: false
    -h, --help
      Displays help information
  Commands:
    encode
      Usage: encode [options]
        Options:
          -cc, --color-channel,
            Color Channels  (1 - 7) - R: 1, G: 2, B :3, RG: 4, GB: 5, RB: 6,
            RGB: 7
            Default: 1
        * -c, --cover
            Absolute path to the Cover Image which will store data (Must be
            PNG or BMP).
          -ET, --encryption-type
            Encryption type (1 - 3): 1 - AES256, 2 - TripleDES, 3 - RSA
            Default: 0
          -h, --help
            Displays help information
          -m, --message
            Message to be stored
          -lsb, --no-of-lsb
            Number of least significant bits: ranges 0 - 7
            Default: 0
          -o, --output
            File name for the output Image.
          -p, --password
            Password for encryption
          -pbk, --public-key
            File name for the public key (Required during encryption if -ET=3
            (RSA) is selected)

    decode
      Usage: decode [options]
        Options:
        * -c, --cover
            Absolute path to the Cover Image which stores the information
            (Must be PNG or BMP).
          -h, --help
            Displays help information
          -o, --output
            File name for the output extracted data
          -p, --password
            Password for decryption
          -pvtk, --private-key
            File name for the private key (Required during decryption if -ET =
            3 (RSA) was used as encryption algorithm)

    generate
      Usage: generate [options]
        Options:
        * -d, --destination_directory
            Absolute path to directory which will store the generated Keys
          -h, --help
            Displays help information
```

## 5.2 TESTING APPROACH

Testing of the software can be done in any machine using JRE 11 or later and JavaFX 11 or later.

### 5.2.1 UNIT TESTING

In unit testing we tested the individual units of the source code

#### 5.2.1.1 ENCODER TESTING IN GUI

| Case no. | Test Case | Input | Expected Result | Output | Test Result |
|---|---|---|---|---|---|
| **1.** | Trying to encode a message in a selected image. | A cover image path,<br>A output image path,<br>A message | System should display encoding successful alert. | Encoding successful Alert | Success |
| **2.** | Trying to<br>Select images other than PNG while trying to select the cover image | Selecting a file | The system should not allow to select any image formats other than PNG. | No effect | Success |
| **3.** | Trying to start encoding without selecting a cover image, message or output file name. | Pressing the Start encode button | Start encode button should be disabled and not allow clicks | No effect | success |

#### 5.2.1.2 DECODER TESTING IN GUI

| Case no. | Test Case | Input | Expected Result | Output | Test Result |
|---|---|---|---|---|---|
| **1.** | Decoding a message from a stego image | A stego image | System should display Decoding successful Alert and the hidden message should be displayed inside the decoder window | Decoding Successful Alert,<br><br>Hidden message in the decoder window | success |
| **2.** | Trying to decode a non stego image | A non stego image file | System should not allow to select the input image and display a error message in the decoder window | Image is not a stego image. | success |
| **3.** | Trying to select file other than an image in place of the | A non-image file | The system should not allow to select any | No effect | success |

| | | | | | |
|---|---|---|---|---|---|
| | carrier image file | | image formats other than PNG. | | |

#### 5.2.1.3 COMMAND LINE MODULE TESTING

| Case no. | Test Case | Input | Expected Result | Output | Test Result |
|---|---|---|---|---|---|
| **1.** | Trying command encode without any required options | /encode | It should display the usage list in the terminal | Usage List | success |
| **2.** | Trying command encode with all the required options | /encode –c "cover.png" –m "Message" –o "output.png" | The system should output encoding successful output in the terminal | Encoding Successful | success |
| **3.** | Trying command decode with all the required options | /decode –c "carrier" | The message hidden inside the stego image should be displayed | Message | success |

### 5.2.2 INTEGRATED TESTING

In integrated testing we combined all the individual units of the source code and tested.

| Case no. | Test Case | Input | Expected Result | Output | Test Result |
|---|---|---|---|---|---|
| **1.** | Trying to open the GUI window by double clicking the run.bat file | Double Mouse Left Click in the run.bat file | The GUI window should appear. | The GUI window appeared | Success |
| **2.** | Trying to open the application in Command Line | /run –cli | The CLI menu should appear | The CLI menu appeared | Success |
| **3.** | Generating the RSA key pairs using the Command line parameters. | /run –generate –d "destination directory" | A pair of RSA keys should be generated in the given destination directory | The key pairs generated | Success |

### 5.2.3 ENCRYPTION ALGORITHMS TESTING

### 5.2.3.1 AES-256 ENCRYPTION ALGORITHM

| AES-256 ENCRYPTION ALGORITHM | |
|---|---|
| Maximum Input Text Size | Unlimited |
| Maximum Output Text Size | Let Message Length be len.<br><br>If ( length <13)<br>Then Encrypted Text Size = 40;<br>Else<br><br>Max Encrypted Text Size =<br>$$24 + \left(\frac{len - 5}{8}\right) + \left(\frac{\frac{len - 5}{8}}{3}\right) * 4 + 16$$ |
| Encryption Type | Single-Key |
| Minimum Password Length | 1 |

### 5.2.3.2 TRIPLE DES ALGORITHM

| **TRIPLE DES ENCRYPTION ALGORITHM** | |
|---|---|
| Maximum Input Text Size | Unlimited |
| Maximum Output Text Size | Let Message Length be len<br><br>If len <=8;<br>Maximum Encrypted Text Size = 12;<br>else<br>If len < 16;<br>Maximum Encrypted Text Size = 24<br>else<br>Then Maximum Output Text Size =<br>$$\left(\left(\left(\frac{len}{8}\right) * 12 + 12\right) - \left(\frac{len}{24}\right) * 4\right)$$ |

| | |
|---|---|
| Encryption Type | Single-Key |
| Minimum Password Length | 24 Characters |

5.2.3.3 RSA ENCRYPTION ALGORITHM

| RSA ENCRYPTION ALGORITHM | |
|---|---|
| **Maximum Input Text Size** | 117 Bytes |
| **Maximum Output Text Size** | Fixed, 172 Bytes |
| **Encryption Type** | Multi-Key (Public Key, Private Key) |
| **Minimum Password Length** | ---------- |

## 5.3 TEST REPORTS

While testing the system, we tested the GUI for the modules encoder and decoder. In Encoder, when we tried to select an image format that is not a PNG, any other files other than PNG were not available, which was a success. Also in Encoder, when we tried to start encoding before giving valid input, the start encoding button would be disables and will not allow inputs. In Decoder, we tried selecting a non/non-stego image files and the decoder window would handle the exception very nicely by detecting the image format and the special pattern encoded inside a stego image. All the test results in the Encoder and Decoder GUI testing came successful. We saw that it was not possible to exploit the encoder or the decoder with invalid inputs. We also tested the command line module by giving invalid or no inputs, but the system would handle it very nicely. After analysing the above test results we can say that the system is good to go for deployment.

# CHAPTER 6: CONCLUSIONS

## 6.1 CONCLUSION

In present world, with the growth of high-speed computer network, and internet in particular has increased the ease of communication, and the use of digital formatter data. Compared to Analog media, digital media offers many different advantages. The time to send data from one to another is lesser day by day. It is so easier as well as faster to transfer the data than ever. So, many individuals and business people, as well as the Government use to transfer confidential and important data using the internet. Therefore, data and information security is becoming an integral part of Data Communication.

The goal of this project is to make application software with the idea of steganography. This software is can be used to hide the secretive data inside a high-resolution image. The image can be transferred to the receiver and he/she can decode the message with the same software. Using this process no one will suspect that a secret communication is taking place in plain sight.

Hence, we can conclude that this software is probably ideal for those who wants enhanced privacy during communication. It can be used by military, to transfer confidential information. It can be used by Businesses to transfer documents. It can be used by companies to control the access for digital contents distribution. It can be used by users with Privacy and anonymity is a concern on the internet. The GUI provided in the software is so simple that anyone can learn to use it. The software will enhance the privacy of the user much more than ever.

## 6.2 LIMITATIONS OF THE SYSTEM

The software in great in all the ways, but it still has some limitations,

- The stego image generated by this software can be detected by using Steganalysis.
- Only PNG with 24bit or 32bit color depth can be used with this software.
- Only the systems running Java 11 or later and JavaFx 11 or later can run this software.
- The software can be used to make an innocent looking image hiding a deadly terrorist plan.

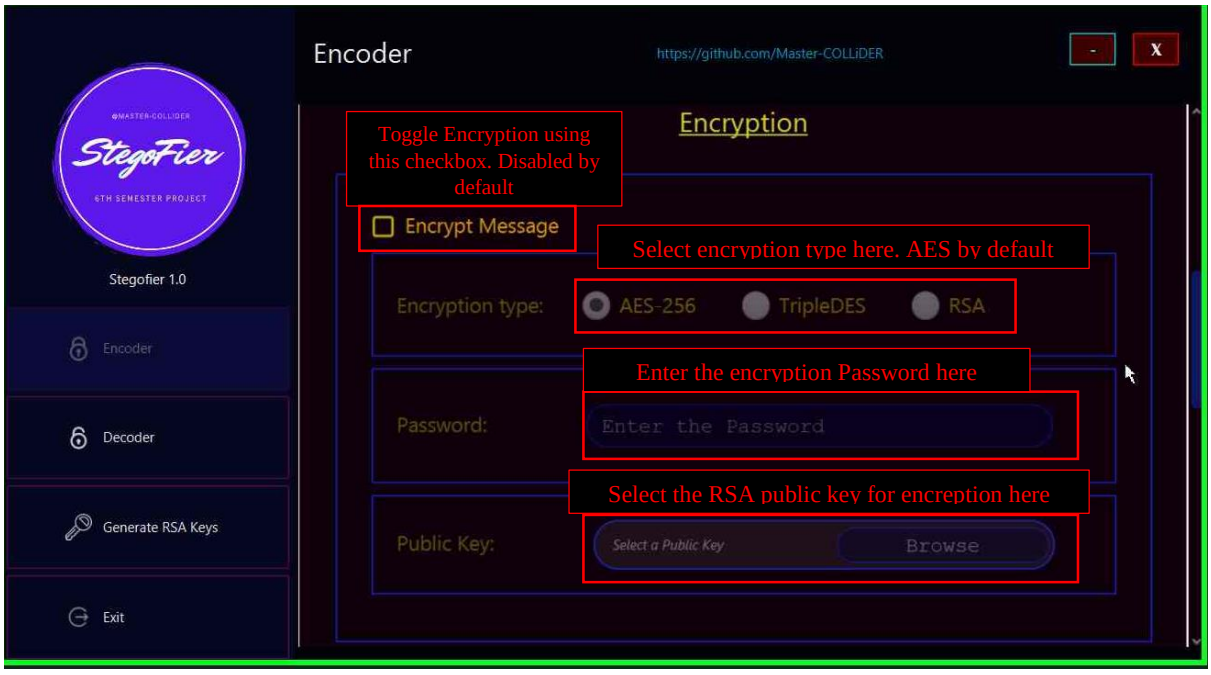
## 6.3 FUTURE SCOPE OF THE PROJECT

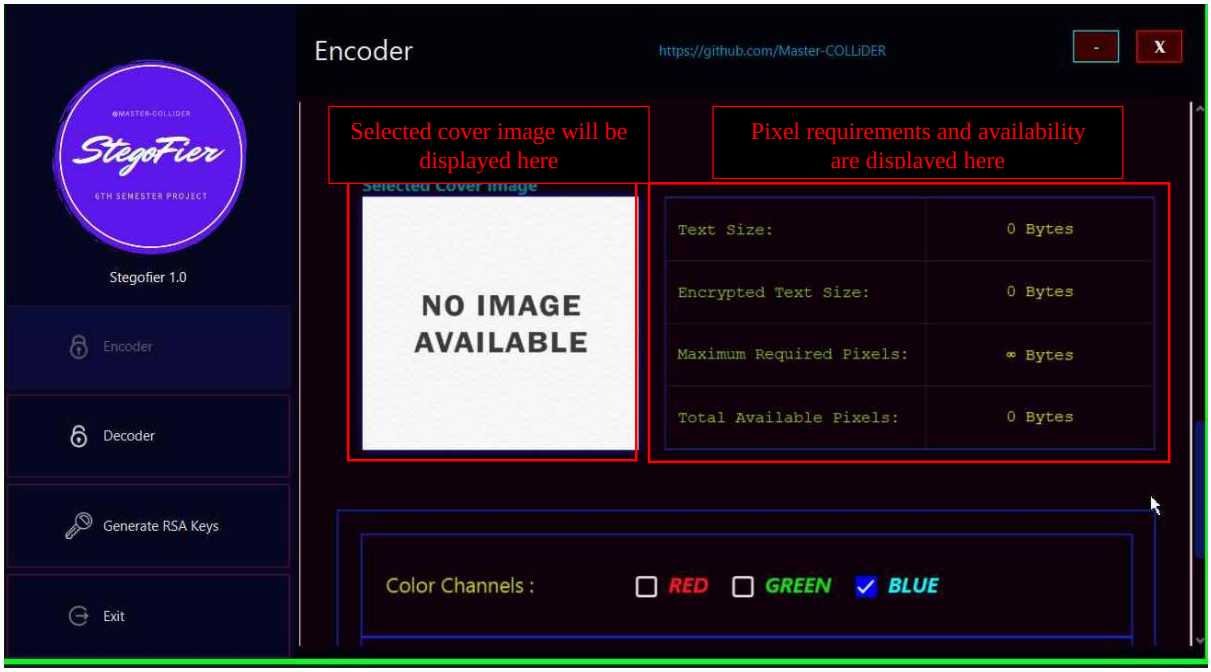
In this project the work is limited to only a small part of digital steganography. There is a lot more research and development that can be done. Some of them are,

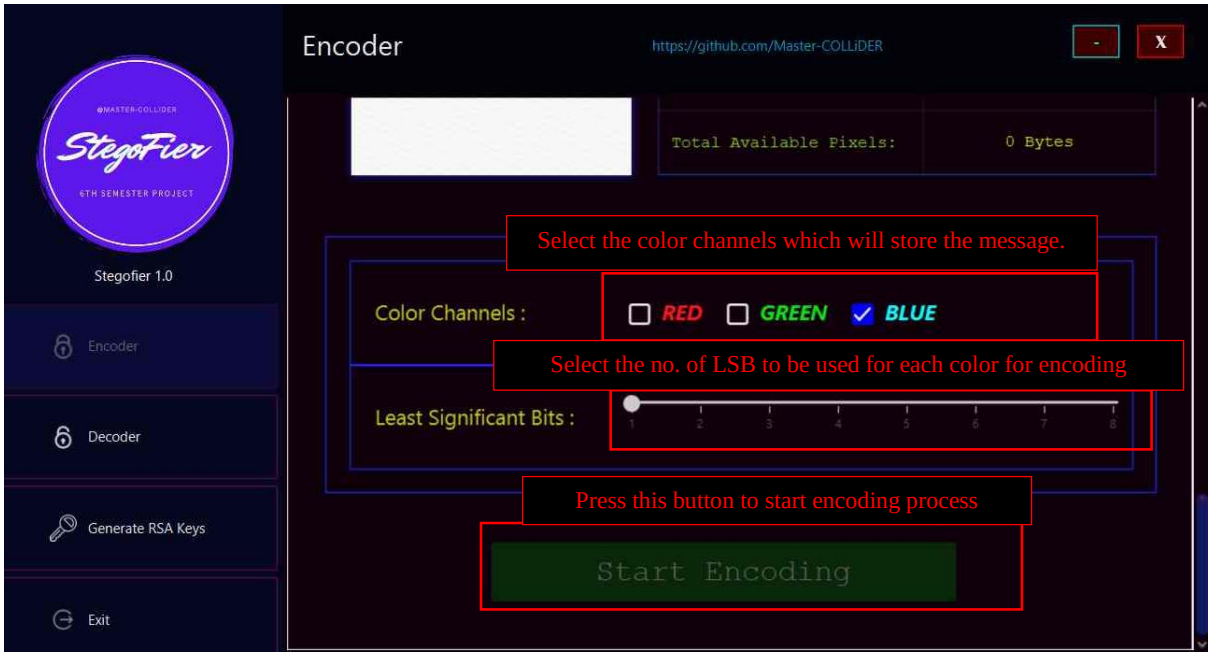
- **Adding the support for more image file formats**: We have seen that this software only supports 24bit and 32bit PNG file formats.
- **Adding the support for Audio file:** This software can only process specific image file format.
- **Adding more encoding algorithm other than LSB insertion**.
- **Finding a Measure for anti-steganography**: The stego image generate with this software can be easily detected by using Steganalysis.

# ANNEXURE-I: USER DOCUMENTATION

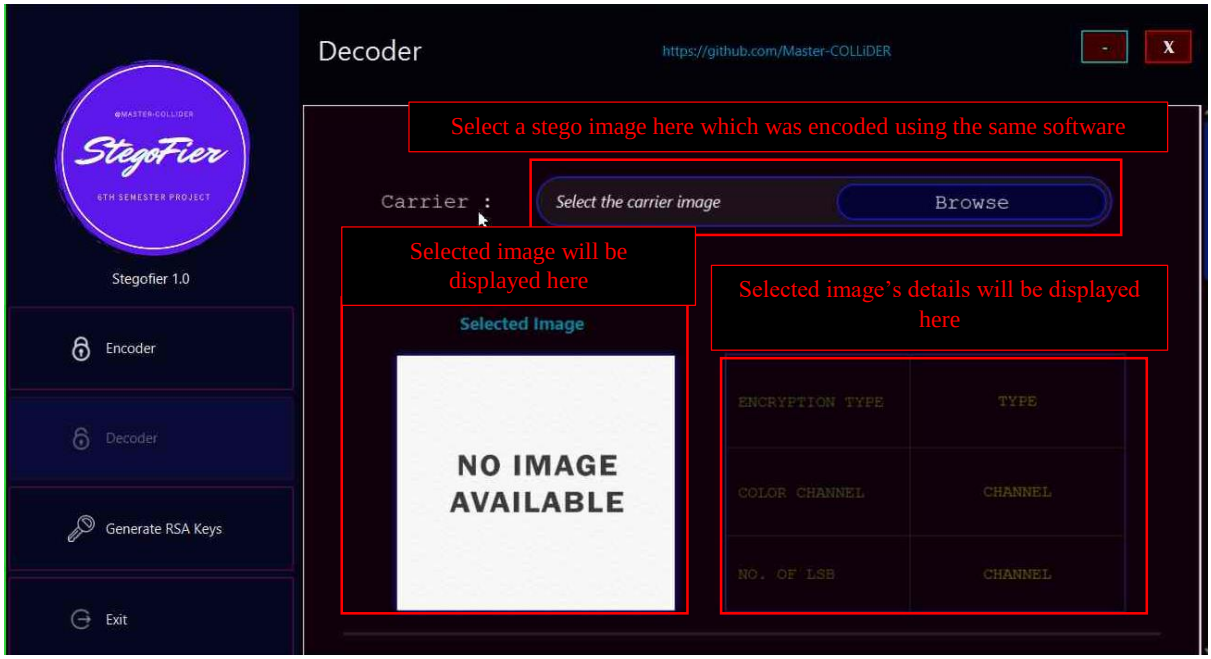
## ENCODER WINDOW:

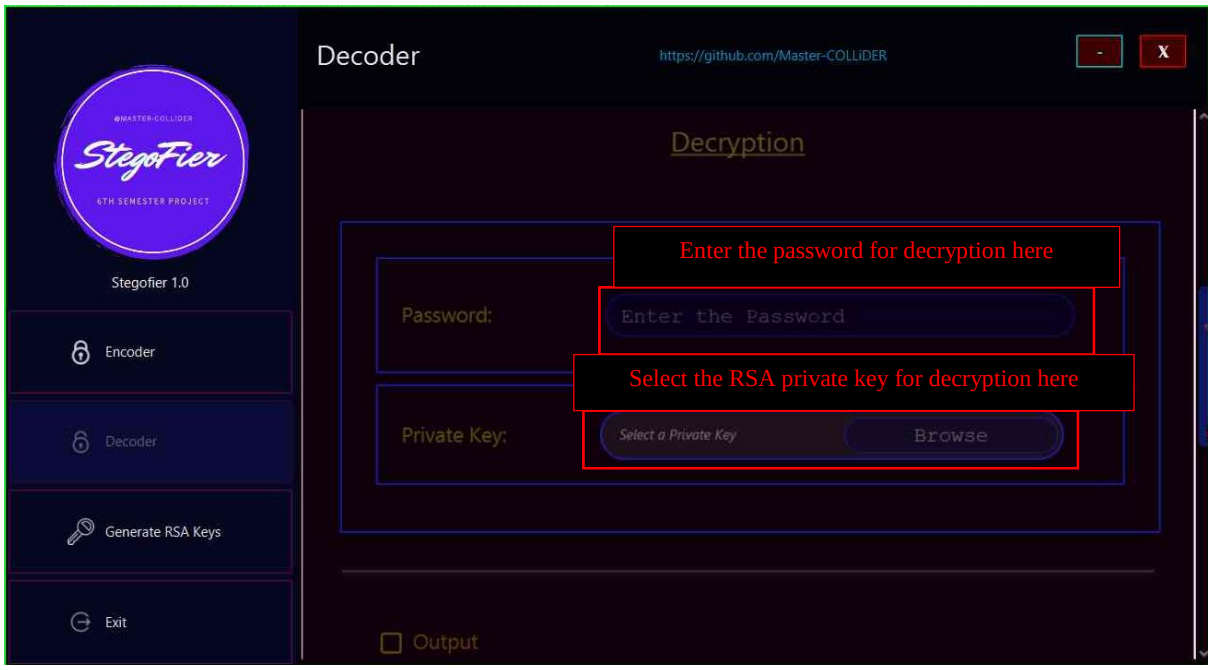



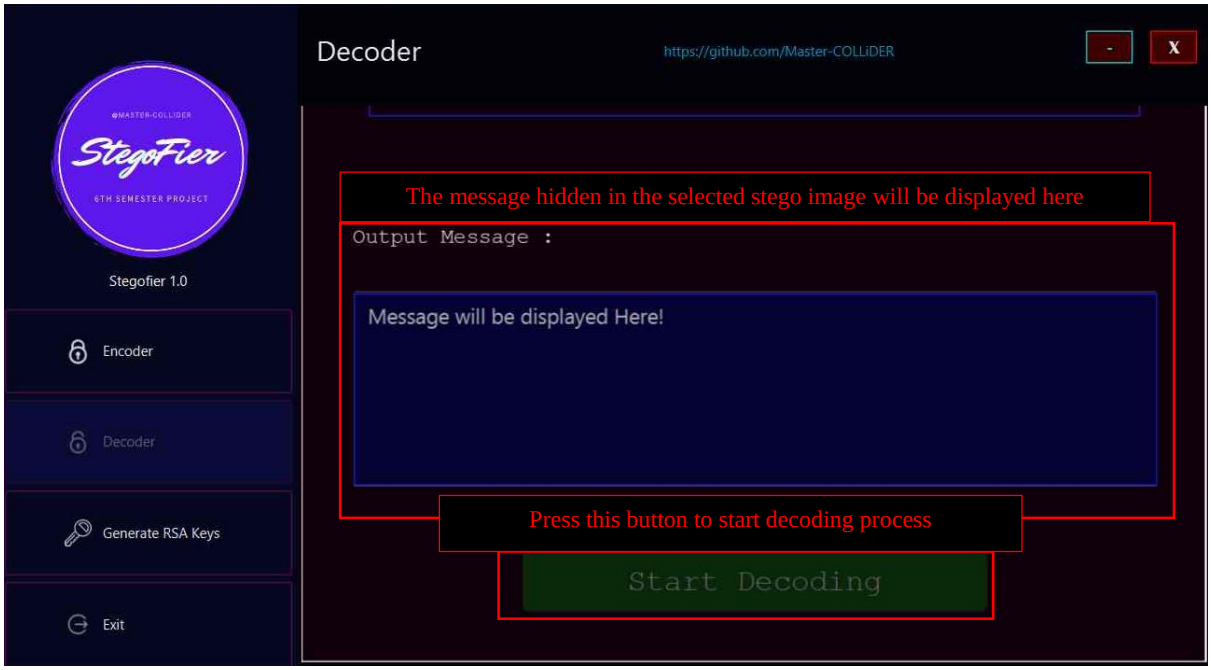

Encoder
https://github.com/Master-COLLiDER
Selected cover image will be displayed here
Pixel requirements and availability are displayed here
NO IMAGE AVAILABLE
Text Size: 0 Bytes
Encrypted Text Size: 0 Bytes
Maximum Required Pixels: ∞ Bytes
Total Available Pixels: 0 Bytes
Color Channels : RED GREEN BLUE
Stegofier 1.0
Encoder
Decoder
Generate RSA Keys
Exit


Encoder
https://github.com/Master-COLLiDER
Total Available Pixels: 0 Bytes
Select the color channels which will store the message.
Color Channels : RED GREEN BLUE
Select the no. of LSB to be used for each color for encoding
Least Significant Bits :
Press this button to start encoding process
Start Encoding
Stegofier 1.0
Encoder
Decoder
Generate RSA Keys
Exit

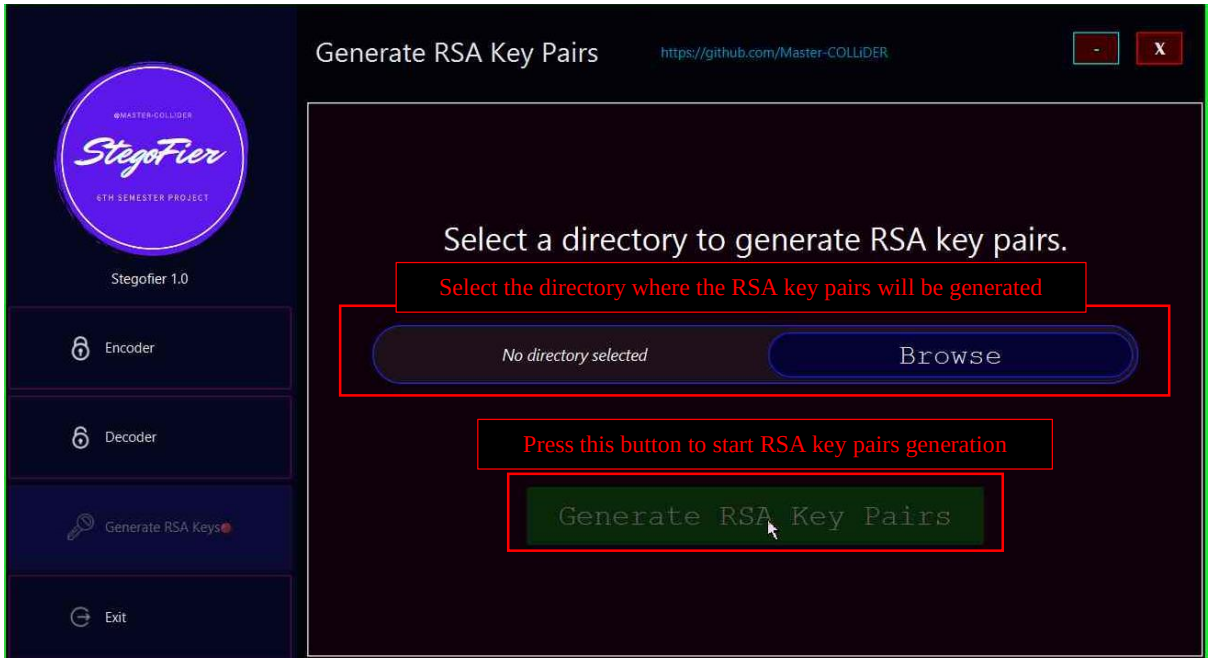
## DECODER WINDOW:






## RSA KEY GENERATOR WINDOW:



# BIBLIOGRAPHY AND REFERENCES

## BIBLIOGRAPHY:


BOOKS:

- ***Fundamentals of Software Engineering*** (5th Edition) by Rajib Mall.

  (Accessed from June 25 2019 to November 18 2020)

WEBSITES:

- *Stack Overflow Questions on Java*:

  https://stackoverflow.com/questions/tagged/java

  (Accessed from August 26 2019 to October 16 2020)

- *Stack Overflow Questions on JavaFx :*

  https://stackoverflow.com/questions/tagged/javaFx

  (Accessed from February 26 2020 to October 16 2020)

- *Jcommander full JavaDocs:*

  https://jcommander.org/

  (Accessed from June 2 2020 to October 16 2020)

- *JFoenix Github Link:*

  https://github.com/jfoenixadmin/JFoenix

  (Accessed from March 3 2020 to October 16 2020)


## REFERENCES:


WEBSITES:

- *Triple DES in JAVA*:

  https://gist.github.com/joeolaoye/6afc88a474aa086036c65b15abcf0947

  (Accessed from March 9 2020 to October 16 2020)

- *AES 256 in JAVA*:
  https://howtodoinjava.com/java/java-security/aes-256-encryption-decryption/
  (Accessed from January 20 2020 to October 16 2020)
- *RSA in JAVA*:

  https://www.devglan.com/java8/rsa-encryption-decryption-java

  (Accessed from March 10 2020 to October 16 2020)