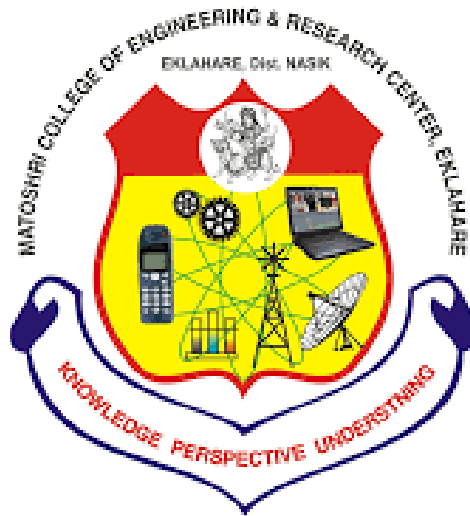


Matoshri Education Society's



**A
Project Stage I Report
on**

Image Steganography using Least Significant Bit

by

Aditya Sonar

[72153712L]

Samadhan Kardile

[72153640K]

Shashikant Kandekar

[72153636M]

Ved Malve

[72153656F]

Under the guidance of

Ms. Shilpa Adke

**Department of Computer Engineering
Matoshri College of Engineering and Research Centre,
Nashik-422105**

**SAVITRIBAI PHULE PUNE UNIVERSITY
2023-24**

CERTIFICATE



Department of Computer Engineering Matoshri College of Engineering and Research Centre, Nashik

This is to certify that
..... has suc-
cessfully completed his / her project work on
..... at
Matoshri College of Engineering and Research Centre, Nashik in partial
fulfillment of the under Graduate course B. E. Computer, in academic year
2023- 24 as prescribed by Savitribai Phule Pune University.

Ms. Shilpa Adke
Project Guide

Dr. Varsha H. Patil
Head

Dr. G. K. Kharate
Principal

External Examiner Name and Sign

ABSTRACT

Steganography is classified among the foremost methods employed in data security to conceal and safeguard confidential messages in the data transmitted. Security, especially data security, is an important requisite in today's world hence Steganography has great significance. The proposed work deals with understanding and implementation of steganography on different images using two different techniques: Least Significant Bit method (secret image is hidden using the bits at least significant level of the cover image) and Discrete Wavelet Transform method (secret image is hidden by modification of the wavelet coefficients of cover image). The image to be transmitted secretly is both encoded and decoded using these methods and a detailed analysis of the resultant images is performed using various image parameters. These experimentally obtained and compared efficiency parameters, thus, demonstrate the efficiency of the methodology proposed in the proposed work.

Keywords: Image steganography, LSB, DWT, data hiding, cover, secret, extracted, security, quality, embedded, pixels

ACKNOWLEDGMENT

First and the foremost I, express my deep sense of gratitude, sincere thanks and deep sense of Appreciation to project Guide **Ms. Shilpa Adke** and seminar coordinate **Dr. Swati Bhavsar** Department of Computer Engineering, Matoshri College of Engineering and Research Center, Nashik. Your Availability at any time throughout the Semester, valuable guidance, view, comments, critics, encouragement and support tremendously boosted this research work.

Again load and loads of thank to Head of Computer Engineering Department **Dr. Varsha H. Patil** for providing me the support we ever had. Your opinion, view, Comments and thoughts have been really helped me to improve my writing.

I like to express my sincere gratitude to Principal of Matoshri College of Engineering and Research center, **Dr.G.K.Kharate** for proving the great platform to complete the same within the schedule time.

I am also Thankful to all the faculty member, Computer Engineering Department, Matoshri College of Engineering and Research Center, Nashik for giving comments for the improvement of work, encouragement and help during completion of our thesis.

Last bit not least, I should say thanks from my bottom of heart to my family and friends for their never ending love, help and support in so many ways through all this time. Thank you so much.

Contents

Certificate	i
Abstract	ii
Acknowledgement	iii
Abbreviations	vii
1 Introduction	1
1.1 Overview	1
1.2 Motivation	2
1.3 Problem statement	3
1.4 Objectives	3
2 Literature Survey	4
2.1 Literature Survey Table	4
3 Software Specification Requirement	5
3.1 Language:	5
3.2 Operating System:	5
3.3 RAM:	5
3.4 ROM/Storage:	5
3.5 Libraries:	5
4 Design of System	6
4.1 System Architecture	6
4.1.1 System architecture has this major blocks . .	7
4.2 Data Flow Diagrams	9
4.2.1 DFD Level 0	9
4.2.2 DFD Level 1	10
4.2.3 DFD Level 2	11
4.3 UML Diagrams	12
4.3.1 Use Case Diagram	12
4.3.2 Sequence Diagram	13
4.3.3 Class Diagram	14

4.4	Modules of System	16
4.4.1	Encode Data	16
4.4.2	Decode Data	19
4.4.3	User Interface	21
5	Prototype Model of Project	23
5.1	Prototype Model of Project	23
5.1.1	Embed Data / HIdе Data	23
5.1.2	Extract Data / Unhide Data	26
5.2	Implementation	28
5.2.1	Implementation of Project	28
5.2.2	Libraries	32
5.2.3	Algorithm	34
5.2.4	Applications	35
6	Conclusion & Future Scope	37
6.1	Conclusion	37
6.2	Future Scope	38

ABBREVIATIONS

Abbreviation	Illustration/Full Forms
LSB	Least Significant Bit
DWT	Descrete Wavelength Transform
DFD	Data Flow Diagram
UML	Unified Modeling Language
2FA	Two-Factor Authentication

Chapter 1

Introduction

Introduction to Steganography, Image Steganography and Scope for Improvement

1.1 Overview

The main obstacle faced in data privacy is the need to share information while protecting personally identifiable information from hackers and other malicious attacks. The process of hiding secret information in the form of a canvassed media such that only the authorized source and destination have access to it is known as steganography. The process hides the existence of the secret information such that the communication between the two parties shall not be visible to any unauthorized person while maintaining the quality of the secret message transmitted. Spatial, frequency and temporal domains are the three categories of technical steganography.

The objectives of image steganography are capacity, intangibility, recovering ability, integrity and confidentiality. The data bits can be concealed in a single pixel of an image defines the capacity. Intangibility refers to the efficiency with which the data can be hidden in an image without it being imperceptible. Recovering ability deals with the efficiency with which the hidden image can be recovered from the steganographic image irrespective of the filtering and the compressing.

The process of image steganography has been analyzed and implemented in this proposed work. The basic skeleton of image steganography comprises of a Carrier: cover image and a Message: embedded onto the cover image. In this technique, the power of the pixel is utilized to conceal the information. Lastly, the integrity and confidentiality relate to the

secure, error-free transmission. The proposed work methodically examines and implements the Least Significant Bit (LSB) and Discrete Wavelet Transform (DWT) techniques for encrypting and decrypting secret images with respect to cover image.

In this proposed work, we will use more advanced LSB (Least Significant Bit) Technique or Algorithm for more additional security and to retrieve Images without losing its original quality of encrypted Images which will prevent it from antagonists or Steganalysts.

1.2 Motivation

Image steganography is a technique of hiding secret data within an image to protect it from unauthorized access. One of the most common methods of image steganography is the least significant bit (LSB) approach, which replaces the least significant bits of the image pixels with the bits of the secret data. However, LSB has some limitations, such as low capacity, low security, and high distortion. Therefore, many researchers have proposed various modifications and improvements to the LSB method to overcome these challenges.

The motivation for this project is to explore and implement a more advanced LSB method for image steganography that can achieve higher capacity, higher security, and lower distortion than the existing methods. The project will contribute to the field of image steganography by providing a novel and efficient method that can enhance the quality and security of the hidden data. The project will also demonstrate the practical applications of image steganography in various domains, such as communication, authentication, digital forensics, and data protection.

The project will also address some of the open issues and challenges in image steganography, such as robustness against attacks, adaptive embedding, and dynamic payload.

1.3 Problem statement

To improve the capacity and robustness of image steganography, researchers have proposed advanced least significant bit (LSB) methods that use secret map techniques, 3D chaotic maps, and hash functions. However, there is a need for further research to evaluate the performance of these advanced LSB methods and compare them with other image steganography techniques. The research should also focus on developing new advanced LSB methods that can overcome the limitations of the existing methods and improve the security and efficiency of image steganography.

1.4 Objectives

The objectives of the system are

- To evaluate the performance of advanced LSB methods and compare them with other image steganography techniques
- To develop new advanced LSB methods that can overcome the limitations of the existing methods and improve the security and efficiency of image steganography
- To hide the secret message or image inside the cover image using the LSB technique.
- To ensure that the hidden message is undetectable by steganalysis attacks.
- To test and validate the results.

Chapter 2

Literature Survey

This chapter discuss brief literature regarding the project. Literature survey is mainly used to identify information relevant to the project work and know impact of it within the project area. It defines as till yet how many surveys have been done knowledge of latest technology and implementation designs.

2.1 Literature Survey Table

Author(s)	Title	Year	Journal/Conference	Methodology & Key Findings
Fridrich et al.	Steganalysis of LSB Matching in Images	2009	Proceedings of SPIE - Media Forensics and Security	Investigated the vulnerabilities of LSB steganography and proposed steganalysis techniques to detect LSB-based steganography.
Zhang et al.	A survey of Image Steganography Techniques	2010	IEEE Transactions on Information Forensics and Security	Reviewed various steganographic techniques, including LSB-based methods, and discussed their advantages and limitations.

Table 2.1: Literature Survey Table

Explanation of literature survey in detail

Chapter 3

Software Specification Requirement

As Image Steganography do not required high system Requirements to work on any system.

Here are some minimum system requirements are mentioned below to run our application on any system seamlessly.

3.1 Language:

Python 3.11

3.2 Operating System:

Windows 10/11 or Ubuntu 22.01

3.3 RAM:

Minimum 4GB

3.4 ROM/Storage:

Minimum 100GB

3.5 Libraries:

OpenCV, Pillow, BitArray, NumPy, Scikit-Image, Matplotlib, Stegano,

Chapter 4

Design of System

This chapter introduces architecture of the system and module of the system. It also contain the registration, verification, authentication process and functioning of the system. DFD and UML diagram are explained in this chapter.

4.1 System Architecture

Figure 4.1 shows a typical example of a steganography system. In such a system, there are always two main parts. The encoder hides a secret message in a cover image. The output of this process is a stego-image. The decoder retrieves the information from the received stego-image by using predefined rules based on an implicit agreement between the sender and the receiver, including the key and the stego algorithm

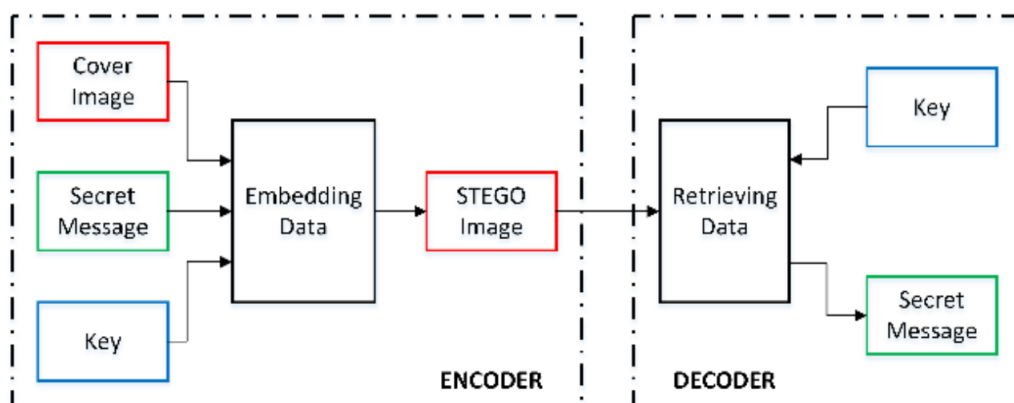


Figure 4.1: System Architecture

There are several approaches to concealing a message inside an image so that any changes in the original image are undetectable and insensible, including least significant bit (LSB) insertion, masking & filtering, and transformation.

In the LSB insertion method, the LSB planes of the cover image are altered by the bits of the message, but conform to particular rules. The quality of the output stego-image is maintained at the same level as the original. Carrying out this technique requires using an image lossless compression format so that the hidden data will not get lost, which usually happens with lossy compression. The most common image format used in this case is the 24-bit color image. The masking & filtering method is usually done on 24-bit and grey-scale images. The message is embedded in the substantial fields of the image so that they are well mixed. This method resembles paper watermarks, in which the information is scattered throughout the cover image.

4.1.1 System architecture has this major blocks

1. Cover Image
2. Public and Private Keys
3. Secret Message
4. Stego Image

Cover Image

In image steganography, a cover image refers to the original image that conceals a secret message. This technique involves embedding information, such as text or another image, within the pixels of the cover image in a way that appears imperceptible to the human eye. The primary goal of using a cover image is to hide the existence of the hidden data while ensuring that the cover image remains visually intact. The process typically involves altering the least significant bits of the pixel values in the cover image to encode the hidden message. Cover images can be any image file, and their choice is crucial in maintaining the effectiveness of the steganographic method, as they should possess characteristics that can mask the presence of hidden data to maintain secrecy and minimize suspicion.

Public and Private Keys

In image steganography, public and private keys are not typically used in the same manner as in traditional cryptographic systems. Instead, the concept of keys is more closely associated with the process of encryption and decryption. The public key is not commonly used, as steganographic methods typically rely on concealing data within an image without the need for a key to recover it. Instead, the focus is on the technique used to embed the data and the algorithm or method employed to extract the hidden information. The security often depends on the strength of the embedding technique itself rather than keys, making it a distinct approach from conventional encryption systems that rely heavily on public and private key pairs.

Secret Message

In image steganography, a secret message is the confidential information that one seeks to hide within an image, typically for covert communication or data protection. This message can be in the form of text, another image, or any digital data. The primary objective is to embed this secret message within the pixel values of the cover image in such a way that it remains imperceptible to the human eye, preserving the appearance of the original image. Various steganographic techniques are employed to encode the secret message into the cover image, often by subtly altering the least significant bits of pixel values. The secret message can only be extracted with the knowledge of the specific steganographic method or algorithm used, ensuring that unauthorized parties cannot easily discern or access the hidden data.

Stego Image

In image steganography, a stego image is the result of embedding a secret message into a cover image using a steganographic technique. It serves as the vessel or carrier of the hidden information. The stego image appears to be an ordinary image to the human eye, preserving the visual characteristics of the original cover image while concealing the secret message within its pixel data. The process typically involves altering the least significant bits of the pixel values in the cover image to encode the hidden data. The stego image is generated by this modification, and it is crucial that it appears unaltered to avoid suspicion. The secret message can later be extracted from the stego image using the appropriate steganographic method or key, ensuring the confidentiality of the concealed information.

4.2 Data Flow Diagrams

A data flow diagram(DFD) is a graphical representation of the flow of data through information system, modeling it's process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail ,which can later be elaborated.

The Data Flow Diagram (DFD) for the proposed system can be decomposed into three levels such as level 0, level 1 and level 2.

4.2.1 DFD Level 0

The above diagram represents level 0 data flow diagram of our proposed model of steganography using Least Significant Bit Algorithm. The proposed model accepts the input of cover image (original image) from the entity of user interface. The application also uses key for encrypting. For the sake of simplicity, the above figure shows the protected user text message (encrypted) as the obvious outcome of the proposed system.

Considering the overall system architecture and the real time implementation it can be found that the overall system specification and the real time application can be achieved only when all the integrating components are functioning properly.

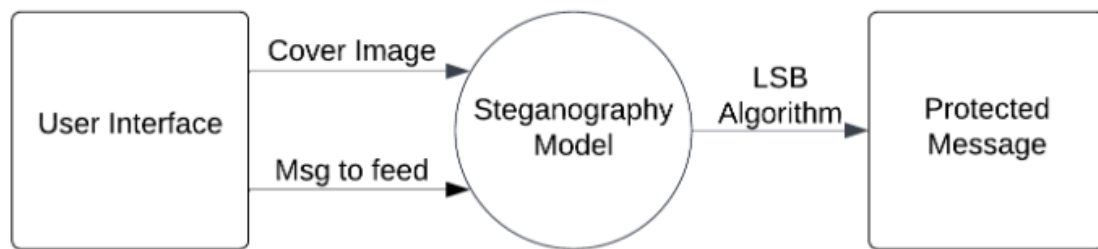


Figure 4.2: DFD Level 0

4.2.2 DFD Level 1

The above diagram represents level 1 data flow diagram of our proposed model of steganography using Least Significant Bit Algorithm. So from 4.3, it can be seen that the main process in level 0 is generically classified as two sub-process e.g. Message Embedding and Message Extraction, where the internal processing using Inverse Wavelet Transform and Genetic Algorithm will lead to design a robust application against RS-analysis. It embeds the secret message in the cover media (e.g. image, audio, video, etc.) to hide the existence of the message.

To resist to RS analysis, the influence on the correlation of pixels needs to be compensated. The compensation may be bringing out by adjusting other bit planes. The proposed design presents a new LSB algorithm approach in order to find the best position for data embedding and also optimize the quality of the steganographic image using Least Significant Bit.

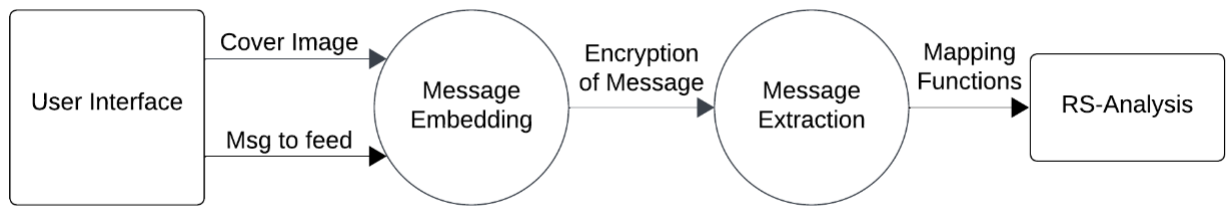


Figure 4.3: DFD Level 1

4.2.3 DFD Level 2

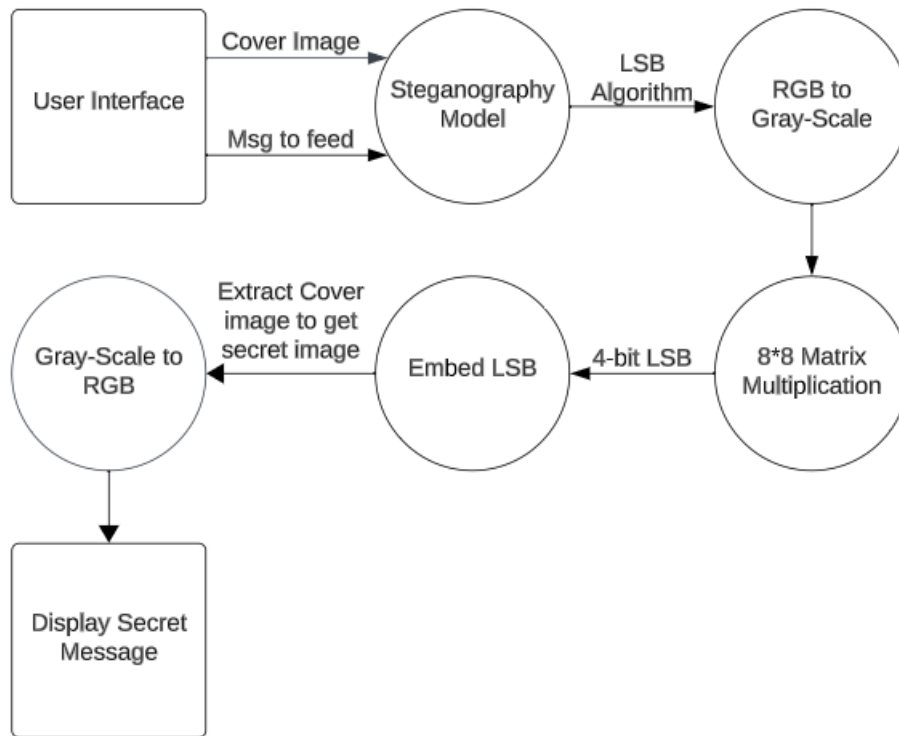


Figure 4.4: DFD Level 2

The above diagram represents level 2 diagram shows that the cover image uses its least significant bits (LSB) store the most significant bits (MSB) of the hidden message image. Furthermore, it is to be noted that pixels of an image are stored in bit form and the intensity of the image is stored in 8 bits(1 byte) per pixel and 24 bits (8 bits * 3(red,green,blue)) per pixel for grayscale and colour image respectively.

Hence, the least significant bit modification during changes in the colour or intensity of a pixel is often not detected by the human eye. Thus, psycho-redundancy acts as an advantage for storing information in the LSB. Along with the advantages, it should also be taken into consideration that the method may not exhibit robustness against noise and may incur losses during compression. This technique is not resilient to images which undergo cropping, rotation or various filters.

4.3 UML Diagrams

4.3.1 Use Case Diagram

A use case diagram for Image Steganography using Least Significant Bit (LSB) encoding illustrates the key interactions between actors and the system components. The primary use cases include "Input Cover Image," where a user provides the image to be used as a cover for steganography, "Input Secret Image," where another user inputs the image to be hidden within the cover image, "Public Key," representing the mechanism to encrypt data for secure transmission, "Private Key," for decryption purposes, and "Encryption" and "Decryption" use cases, denoting the processes of embedding the secret image into the cover image and extracting the secret image from the stego image, respectively. These use cases demonstrate the fundamental functionalities and interactions within the image steganography system, which involve users and encryption/decryption processes.

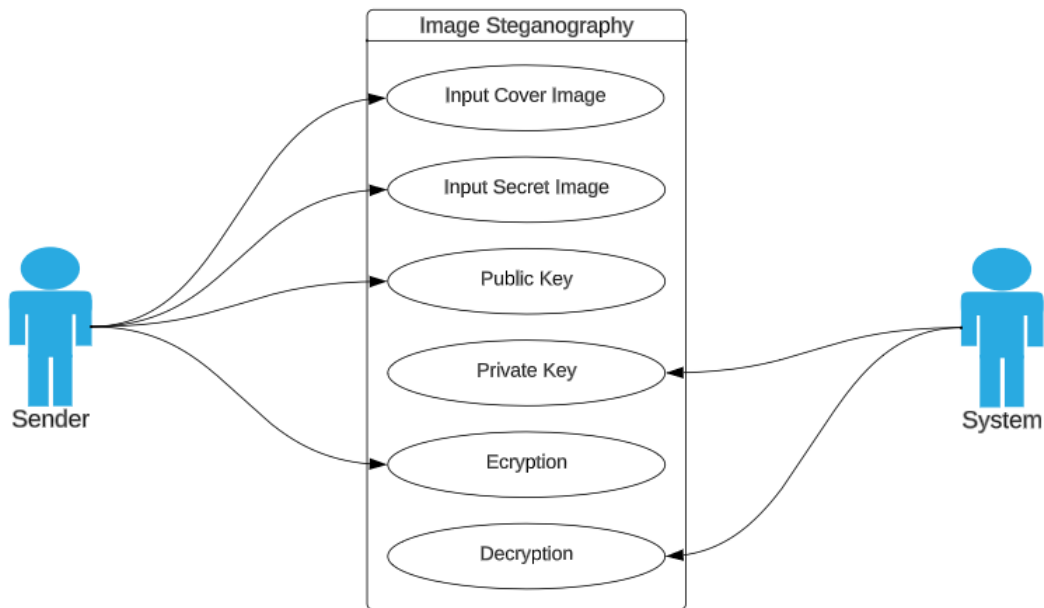


Figure 4.5: Use Case Diagram

4.3.2 Sequence Diagram

Sequence diagram and collaboration diagram are call INTERACTION DIAGRAM. An interaction diagram shows an interaction , consisting of set of object and their relationship including the message that may be dispatch among them

sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. Figure 4.6 shows a Sequence Diagram.

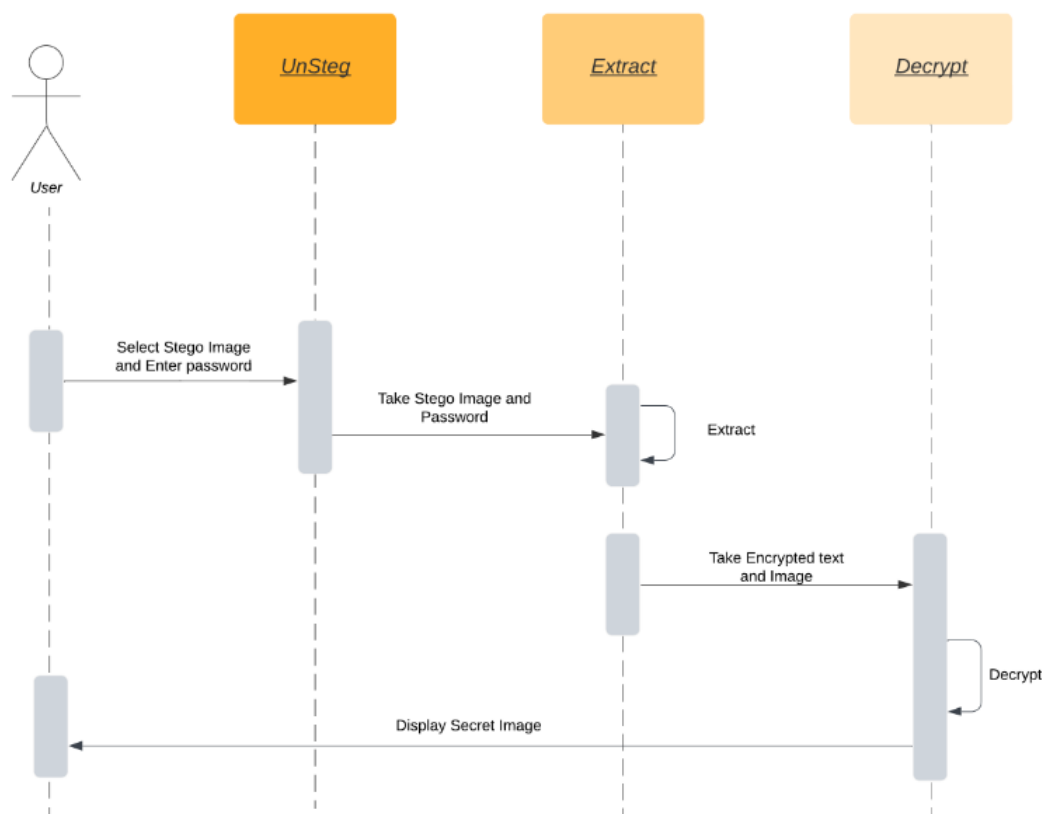


Figure 4.6: Sequence Diagram

4.3.3 Class Diagram

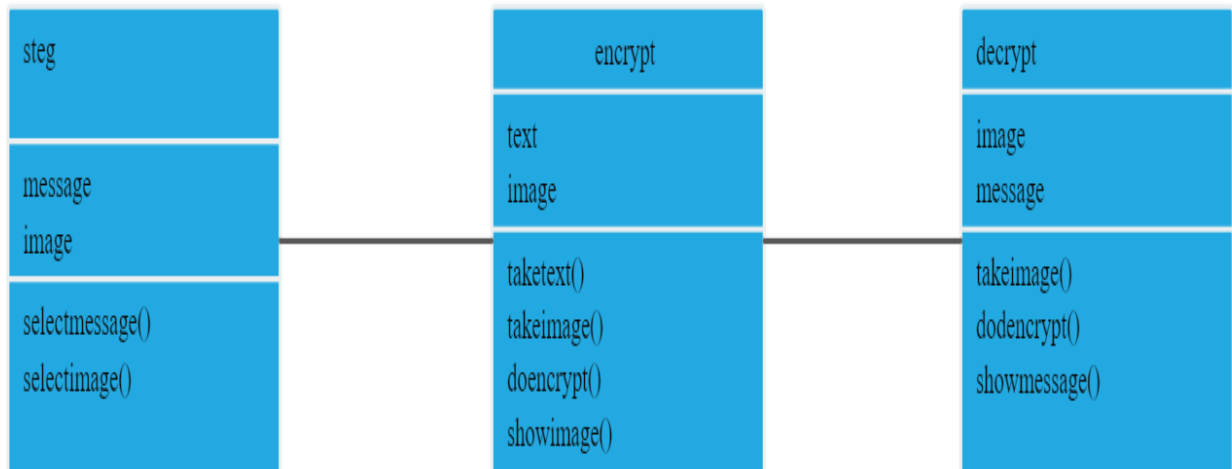


Figure 4.7: Class Diagram

In the context of Image Steganography using LSB encoding, a class diagram can be created to represent the key classes and their relationships. Here's an explanation of the class diagram based on the provided class methods:

1. Class "steg":

Methods:

- **selectmessage()**: This method is responsible for selecting a message or text that the user wants to hide within an image.
- **selectimage()**: This method allows the user to choose an image that will be used as a cover for steganography.

Class "encrypt":

(a) Methods:

- **taketext()**: This method is used to take the selected text or message.
- **takeimage()**: This method is used to take the chosen image as a cover for encryption.

- **doencrypt()**: This method performs the process of embedding the selected text within the chosen image, creating the stego image.
- **showimage()**: This method displays or presents the stego image with the hidden message.

Class "decrypt":

(a) Methods:

- **takeimage()**: This method is used to take the stego image containing the hidden message.
- **doencrypt()**: This method performs the process of extracting the hidden message from the stego image.
- **showmessage()**: This method displays or presents the extracted message.

In this class diagram, "steg," "encrypt," and "decrypt" represent the three main classes involved in the steganography process. The methods within each class describe the specific functionalities of these classes in terms of selecting, encrypting, and decrypting messages within images. The relationships between these classes may involve method calls, but they are not explicitly specified in the provided information. The class diagram captures the essential components and their interactions in the Image Steganography system using LSB encoding.

4.4 Modules of System

4.4.1 Encode Data

The "Encode Data" module in image steganography plays a pivotal role in concealing sensitive information within digital images. Through a meticulous process of embedding, it subtly integrates secret data into a chosen cover image, ensuring that the alterations remain imperceptible to the human eye. This covert operation allows for secure data transmission or storage within a seemingly innocent image. The outcome is a stego image that appears virtually identical to the original cover image, preserving its visual quality while safeguarding concealed data. In essence, the "Encode Data" module facilitates hidden communication and confidential data protection through the art of digital camouflage.

Function Of Module:

The "Encode Data" module in image steganography is responsible for embedding hidden information, often referred to as the "payload" or "secret data," into a cover image. This process is carried out in a manner that is imperceptible to the human eye, ensuring the visual quality of the image is maintained while the data is hidden within it. The main goal is to facilitate secure and covert communication or data storage.

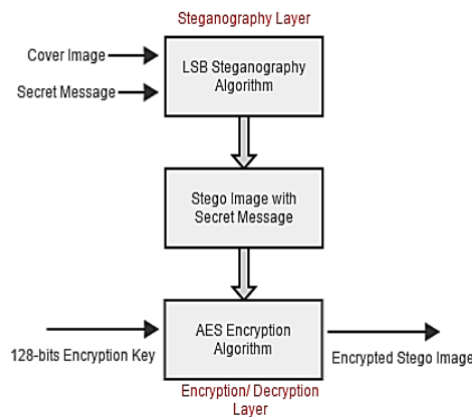


Figure 4.8: Encode Data Flowchart

Input

1. Cover Image: The input image, which can be any digital image (e.g., JPEG, PNG), acts as a container for the hidden data.
2. Secret Data: The data that needs to be concealed within the cover image. This can be any type of data, such as text, files, or even other images.
3. Steganography Algorithm: A specific steganographic technique or algorithm, which determines how the secret data will be embedded in the cover image. Common methods include LSB (Least Significant Bit) substitution and frequency domain techniques.

Process

1. Pre-processing: The cover image is loaded into the module, and if necessary, any necessary preprocessing (such as resizing or format conversion) is performed to ensure it is in a suitable format for steganography.
2. Data Transformation: The secret data is processed to make it suitable for embedding. For example, if the data is in text form, it may be converted into binary or another suitable format. If the secret data is larger than the cover image can accommodate, data compression or encryption may be applied.
3. Embedding: The steganography algorithm is applied to embed the processed secret data into the cover image. The algorithm decides where and how to hide the data, ensuring that it is distributed throughout the image to avoid suspicion. The most common approach is to alter the least significant bits of the pixel values in the cover image, as these changes are less likely to be noticed.
4. Post-processing: After embedding the secret data, the stego image is generated. Post-processing may involve any necessary adjustments to ensure that the stego image looks visually similar to the original cover image. This can include techniques to reduce any noticeable artifacts or distortions.

Output

1. Stego Image: The output of the "Encode Data" module is the stego image. This is the cover image with the secret data embedded in it. The stego image should ideally appear visually similar to the original cover image.

Working

The "Encode Data" module works by making subtle alterations to the cover image so that the hidden data can be later extracted without being conspicuous. Here's an overview of how it works:

1. Pixel Modification: In the encoding process, the module identifies specific locations within the cover image, typically at the pixel level, where it can safely change the pixel values without making a noticeable visual impact.
2. Secret Data Embedding: The module then embeds the secret data into these chosen locations. For instance, it might modify the least significant bits of the color channels in the cover image to store the binary representation of the secret data.
3. Stego Image Creation: The result is the stego image, which looks nearly identical to the original cover image but now contains the hidden data.

4.4.2 Decode Data

The "Decode Data" module within the realm of image steganography serves as the key to unlocking concealed information. Its purpose is to meticulously analyze stego images, identifying and extracting hidden data without compromising the visual integrity of the cover image. By employing specialized decoding algorithms, it reverses the encoding process carried out by its counterpart, the "Encode Data" module. This enables the confidential or covert information to be retrieved, making the module an essential tool for secure data extraction and clandestine communication. Ultimately, the "Decode Data" module safeguards the hidden information, ensuring it can be unveiled while maintaining the veil of visual secrecy within the stego image.

Function Of Module:

The "Decode Data" module in image steganography is responsible for extracting hidden or encoded information from a stego image. Its primary purpose is to reverse the process carried out by the "Encode Data" module, revealing the concealed data without visibly altering the cover image.

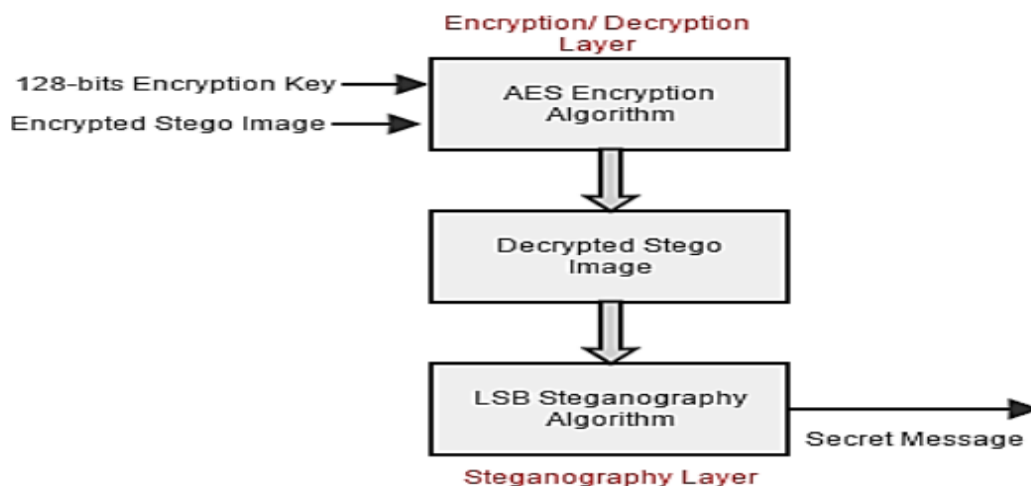


Figure 4.9: Decode Data Flowchart

Input

1. Stego Image: The stego image that contains the hidden data, which is the result of the "Encode Data" module's operation.

Process

1. **Stego Image Loading:** The stego image is loaded into the "Decode Data" module as input.
2. **Decoding Algorithm:** The module employs a specific decoding algorithm or technique to identify and extract the hidden data from the stego image. This algorithm should be compatible with the encoding algorithm used to create the stego image.
3. **Data Extraction:** The decoding algorithm analyzes the stego image to locate and extract the hidden data. It reverses the encoding process by identifying the specific locations where the data was embedded and extracts it in the appropriate format.
4. **Post-processing (Optional):** Depending on the steganographic technique and any post-processing applied during encoding, further processing may be required to convert the extracted data into its original format. This could include data decompression or decryption.

Output

1. **Secret Data:** The extracted secret data that was concealed within the stego image. This can be any type of data, such as text, files, or images

Working

The "Decode Data" module operates by carefully examining the stego image and applying the appropriate decoding algorithm to reveal the concealed information. Here's how it works:

1. **Stego Image Analysis:** The module begins by analyzing the stego image, looking for the hidden data. It uses the decoding algorithm to identify the locations or patterns within the image where the data is concealed.
2. **Data Extraction:** Once the hidden data is identified, the module extracts it from the stego image. This process typically involves reversing the encoding operations.
3. **Data Presentation:** The extracted data is then presented as the output of the "Decode Data" module. Depending on the encoding method and the nature of the concealed data.

4.4.3 User Interface

The User Interface (UI) is a crucial component of any software system, serving as the intermediary between users and the underlying technology. It offers a visual and interactive platform through which users can communicate with the software, providing input, configuring settings, and receiving feedback. The UI's primary function is to make complex systems user-friendly and accessible, allowing individuals to interact with technology in a way that is intuitive and efficient.

Function Of Module:

The "User Interface" module in image steganography provides the means for users to interact with the steganography system. Its primary purpose is to facilitate user control and input, allowing users to specify the cover image, secret data, steganographic parameters, and to view the results of the encoding and decoding processes. It acts as the front-end of the steganography system, making it user-friendly and accessible.

Input

1. Cover Image: Users input the digital image they want to use as the container for hidden data.
2. Secret Data: Users provide the data they wish to hide within the cover image.
3. Encoding Parameters: Users may specify steganographic parameters, such as the encoding algorithm, data compression settings, or encryption keys.
4. Decoding Parameters: For the decoding process, users may input any necessary parameters, such as the decoding algorithm or decryption keys.

Process

1. Input Gathering: The User Interface module collects user-provided information, including the cover image and secret data. Users may also set various encoding and decoding parameters based on their requirements.
2. Encoding Control: In the encoding phase, the User Interface communicates the chosen cover image, secret data, and parameters to

the "Encode Data" module. It may also provide progress updates and completion notifications to the user.

3. **Decoding Control:** During the decoding process, the User Interface passes the stego image and any necessary parameters to the "Decode Data" module. Like the encoding phase, it communicates progress and completion status to the user.
4. **Output Presentation:** The User Interface displays the results of the encoding and decoding processes to the user. It may show the stego image after encoding and the extracted data after decoding. Additionally, it allows users to save or export these results as needed.

Output

1. **Stego Image:** After encoding, users can view and save the stego image, which contains the concealed data.
2. **Extracted Data:** After decoding, users can see and save the extracted hidden data

Working

The "User Interface" module serves as the bridge between the user and the underlying steganography system. Here's how it works:

1. **User Interaction:** Users interact with the User Interface, providing the cover image, secret data, and any relevant parameters for encoding and decoding.
2. **Communication with Encoding Module:** When the user initiates the encoding process, the User Interface passes the user's inputs to the "Encode Data" module. It also monitors the progress of the encoding operation.
3. **Communication with Decoding Module:** In the decoding phase, the User Interface forwards the stego image and any necessary decoding parameters to the "Decode Data" module. It tracks the progress of the decoding process.
4. **Results Presentation:** The User Interface displays the results to the user, showing the stego image generated during encoding and the extracted data obtained during decoding. Users can interact with these results and save them as necessary.

Chapter 5

Prototype Model of Project

5.1 Prototype Model of Project

A prototype model is a development approach that involves creating an initial, simplified version of a software application or system to demonstrate its functionality and gather feedback from stakeholders. This early-stage representation helps developers and project teams validate requirements, design concepts, and user interface elements. Prototypes are typically quick to build and can be either throwaway or evolved into the final product, depending on project needs. They serve as a crucial tool for refining and enhancing the software's design and functionality, reducing development risks, and ensuring that the end product aligns with user expectations and business objectives.

5.1.1 Embed Data / HId e Data

In the above 5.1 shows that, Our prototype model includes a user-friendly interface that allows users to select the image they want to embed data into, specify the secret message or file to hide, and set encryption parameters for added security. The LSB method we employ ensures that alterations to the image are virtually indistinguishable to the human eye, guaranteeing the confidentiality of the embedded information. Additionally, the project features robust data extraction functionality, enabling users to retrieve concealed data from encoded images, thus providing a comprehensive solution for secure data transmission and storage through steganography techniques.

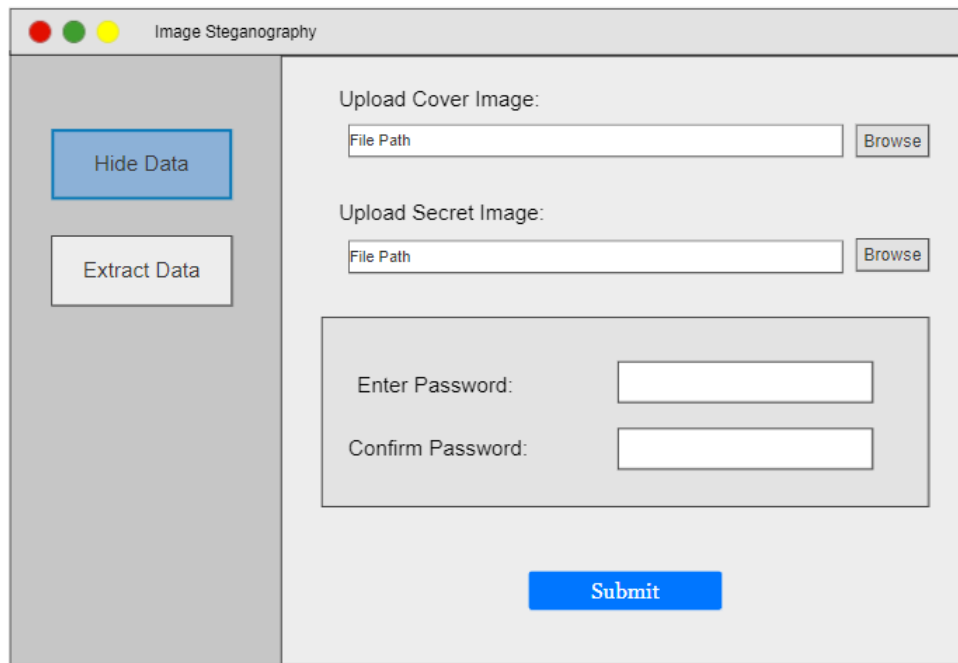


Figure 5.1: Embed Data

1. Upload Cover Image:

- This field is designed to give users the flexibility to select the image they want to use as the 'cover' for hiding their secret message. The cover image is the visual container in which the secret data will be embedded.
- Users can click a button or drag-and-drop an image file from their local storage. The system accepts various image formats, such as JPEG, PNG, BMP, etc.
- The selected image will serve as the canvas for embedding the secret information.

2. Upload Secret Message:

- In this field, users can upload the file or enter the text they want to conceal within the cover image. It is a versatile feature that accommodates different types of data.
- Users can upload a text file, a document, an image, or any other file format. The system will automatically convert and hide the data within the cover image's binary data.
- The text input field is also available for users who prefer to enter a secret message manually.

3. Enter Password:

- To enhance the security of the embedded data, our prototype model includes a password field.
- Users can create a unique password of their choice. This password will be required when extracting the concealed message from the image.
- The password adds an extra layer of protection to the embedded data, making it inaccessible to unauthorized individuals.

4. Confirm Password:

- In this field, users are prompted to re-enter the password they created in the previous step.
- Confirming the password ensures that users have not made any typing errors and will be able to successfully access the concealed data later.

With these four fields, our "Embed Data" module offers a comprehensive and user-friendly solution for image steganography. Users can select a cover image, upload their secret message, set a password, and confirm the password to ensure that their sensitive data remains secure and hidden within the image. This approach provides a powerful tool for covert communication and data protection through steganography techniques.

5.1.2 Extract Data / Unhide Data

The prototype model of our Image Steganography project features an equally robust "Extract Data" module that complements the data embedding process. This module is designed to seamlessly retrieve concealed information from steganographically modified images, which have been encoded using the Least Significant Bit (LSB) technique.

The "Extract Data" module includes an intuitive user interface, allowing users to upload the steganographic image they want to decode. To ensure data security, users are prompted to enter the password that was set during the data embedding process. The system then intelligently extracts the hidden information, whether it's a secret message or a concealed file, while safeguarding the integrity and confidentiality of the data.

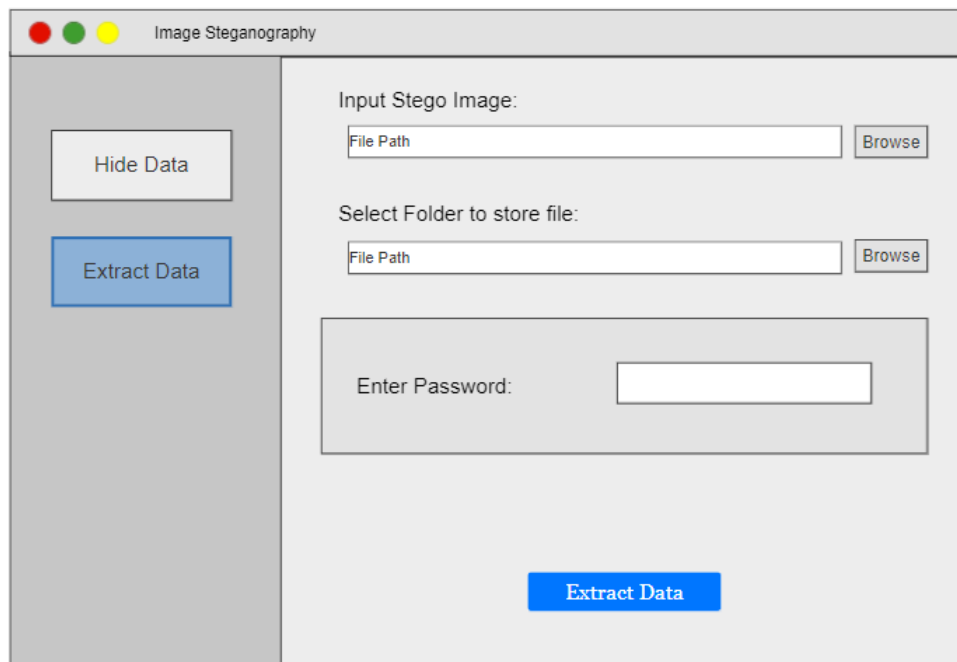
The image shows a software window titled "Image Steganography" with standard macOS window controls (red, green, yellow buttons). On the left side, there is a vertical sidebar with two buttons: "Hide Data" (light gray) and "Extract Data" (blue). The main area of the window is divided into sections. The top section is labeled "Input Stego Image:" and contains a "File Path" text input field followed by a "Browse" button. Below this is another section labeled "Select Folder to store file:" with another "File Path" text input field and a "Browse" button. Further down, there is a section labeled "Enter Password:" with a text input field. At the bottom center of the main area, there is a large blue button labeled "Extract Data".

Figure 5.2: Enter Caption

Our prototype model's "Extract Data" module is equipped with three key fields to facilitate the retrieval of hidden information from steganographically modified images, encoded using the LSB technique. Here's an in-depth look at each field:

1. Upload Stego Image:

- This field allows users to select the steganographic image from which they want to extract concealed data. The stego image is the one that has been previously encoded with hidden information.

- Users can upload the stego image by clicking a button or dragging and dropping the file. The system accepts various image formats, ensuring flexibility in data extraction.

2. Select Folder to Store File:

- In this field, users can specify the destination folder where the extracted data will be saved. This feature enables users to keep their retrieved information organized and easily accessible.
- By selecting a specific folder, users can efficiently manage the extracted data and access it at their convenience.

3. Enter Password:

- To ensure the security of the extraction process, our prototype model incorporates a password field.
- Users must input the correct password that was set during the data embedding process. This password acts as a crucial authentication step, guaranteeing that only authorized users can access the concealed data.

With these three fields, our "Extract Data" module provides a seamless and secure solution for extracting hidden information from steganographic images. Users can select the stego image, specify where the extracted data will be stored, and verify their identity through the password, ensuring the confidentiality and integrity of the concealed data throughout the extraction process.

5.2 Implementation

5.2.1 Implementation of Project

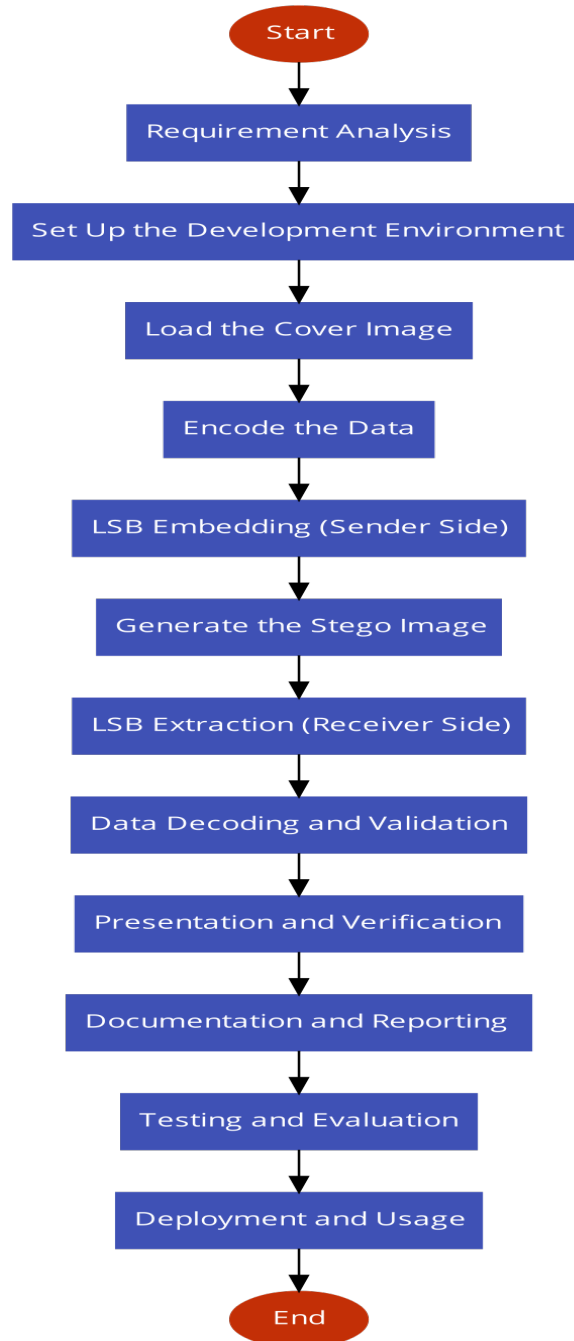


Figure 5.3: Implementation of Model

Step 1: Requirement Analysis

- Define the project's goals and requirements.
- Determine the type of data to be hidden (e.g., text or binary).
- Select the cover image in which the data will be hidden.
- Decide the level of security and capacity for data hiding.

Step 2: Set Up the Development Environment

- Install and configure the necessary development tools and libraries, including Python, OpenCV, Pillow, BitArray, NumPy, Scikit-Image, Matplotlib, and Stegano.
- Create a working directory to organize project files.

Step 3: Load the Cover Image

- Use OpenCV or Pillow to load the chosen cover image into your project.
- Check the image format and dimensions to ensure compatibility.

Step 4: Encode the Data

- Convert the text or binary data to be hidden into a format suitable for embedding.
- If needed, apply any encryption or compression to the data to enhance security and reduce size.

Step 5: LSB Embedding (Sender Side)

- Iterate through the pixels of the cover image.
- For each pixel, modify the least significant bit(s) to embed a portion of the data.
- Ensure that the embedded data doesn't significantly alter the image's appearance to avoid detection.

Step 6: Generate the Stego Image

- After embedding all the data, save the modified image as the stego image.
- Document the stego image's location and format.

Step 7: LSB Extraction (Receiver Side)

- Load the stego image using OpenCV or Pillow.
- Iterate through the stego image's pixels, extracting the LSBs to retrieve the hidden data.

Step 8: Data Decoding and Validation

- Decode the extracted data to its original format (text or binary).
- If encryption or compression was applied, decrypt and decompress the data.
- Ensure the integrity and authenticity of the received data by checking for any errors or tampering.

Step 9: Presentation and Verification

- Present the extracted data to the receiver/user.
- Verify that the hidden data matches the original data.
- Assess the effectiveness of the LSB embedding by comparing the stego image with the original cover image for visual differences.

Step 10: Documentation and Reporting

- Create documentation that includes the project's objectives, methodology, and implementation details.
- Include code comments and explanations for future reference.
- Prepare a report summarizing the project's outcomes and any challenges encountered.

Step 11: Testing and Evaluation

- Test the steganography system with different types of cover images and data.
- Evaluate the system's capacity, security, and resistance to detection.
- Analyze the performance of the project and identify areas for improvement.

Step 12: Deployment and Usage

- If necessary, deploy the steganography system for specific applications.
- Educate users on how to hide and extract data using the system securely.

Throughout the implementation process, it's important to follow best practices in coding, documentation, and project management to ensure the success of your Image Steganography using LSB project.

5.2.2 Libraries

1. OpenCV (Open Source Computer Vision Library):

- OpenCV is a widely used computer vision library that provides a comprehensive set of tools for image processing and computer vision tasks.
- It supports image loading, manipulation, and conversion, making it a go-to choice for working with images in various formats.
- OpenCV offers numerous functions for color space transformations, image enhancement, and filtering, all of which are essential for image steganography.

2. Pillow (PIL Fork):

- Pillow is a user-friendly image processing library that extends Python Imaging Library (PIL) capabilities.
- It is primarily used for image input/output, basic image manipulation, and format conversions.
- Pillow is particularly useful for handling image formats not natively supported by OpenCV, making it an important part of the steganography project.

3. BitArray:

- BitArray is a Python module that facilitates the efficient manipulation and management of binary data.
- It is valuable for encoding and decoding binary messages in the LSB (Least Significant Bit) algorithm, enabling you to easily work with the individual bits of data.

4. NumPy:

- NumPy is a fundamental Python library for scientific computing and data manipulation.
- It provides support for arrays and matrices, making it instrumental in handling numerical data.
- For your project, NumPy can be used to work with image pixel data and perform mathematical operations on the image arrays.

5. Scikit-Image (scikit-image):

- Scikit-Image is an image processing library built on top of NumPy and SciPy.
- It offers a comprehensive set of functions for image analysis, manipulation, and enhancement.
- Scikit-Image is valuable for performing various image processing tasks required in steganography, such as noise reduction, filtering, and feature extraction.

6. Matplotlib:

- Matplotlib is a popular data visualization library in Python.
- It provides a wide range of tools for creating visualizations, including charts and graphs, which can be used to visualize images before and after the LSB (Least Significant Bit) embedding process.
- This can be helpful for monitoring and verifying the results of the steganographic procedure.

7. Stegano:

- Stegano is a Python library specifically designed for steganography purposes.
- It offers a range of steganographic techniques, including the LSB-based hiding, making it a valuable resource for your project.
- Stegano simplifies the process of hiding and extracting data from images, providing a high-level interface for steganographic operations.

5.2.3 Algorithm

- **Least Significant Bit (LSB):**

1. **Input Cover Medium and Message:** Choose a cover medium, such as an image, and the message you want to hide within it.
2. **Convert Cover Medium to Binary Representation:** Convert the pixel values of the cover medium into their binary representations. The number of bits used for each channel (e.g., Red, Green, Blue in the case of a color image) depends on the image's color depth. Common color depths are 8 bits per channel (24 bits per pixel) or 24 bits per channel (32 bits per pixel).
3. **Prepare the Message for Embedding:** Convert the message you want to hide into a binary format. Each character or symbol in the message should correspond to a specific binary sequence.
4. **Determine the Number of LSBs to Replace:** Choose how many least significant bits you want to replace in each color channel for each pixel. This determines the capacity for message embedding.
5. **Embed the Message:** For each pixel in the cover medium, replace the specified number of least significant bits in each color channel with the corresponding bits from the message. Continue this process until the entire message is embedded.
6. **Record Changes (Optional):** If you want to extract the hidden message later, you must keep a record of the LSBs you replaced in each pixel and their corresponding positions in the message.
7. **Output the Steganographic Image:** The steganographic image is the result of the embedding process. It should appear visually similar to the original cover medium.
8. **Extraction of Hidden Message (Optional):** To extract the hidden message, you must reverse the process by reading the least significant bits of the color channels in the steganographic image. Re-assemble these bits into binary groups, and then convert them back into the original message using the same encoding scheme.
9. **Display or Use the Extracted Message:** Once the message is extracted, it can be displayed, decrypted, or used for its intended purpose.

10. **Quality Control (Optional):** Assess the quality of the steganographic image in terms of perceptual similarity to the original cover medium. Balancing message capacity and image quality is important in LSB steganography.
11. **Security Considerations:** Keep in mind that LSB steganography is not highly secure, and the hidden message may be vulnerable to detection by steganalysis techniques. If strong security is needed, consider using more advanced steganographic methods.

These steps outline the basic process of the LSB steganography algorithm. The specific implementation details may vary depending on the programming language and tools you use.

5.2.4 Applications

Here are some applications of image steganography using LSB in detail:

1. **Secure Communication:** One of the primary applications of image steganography is in secure communication. You can embed a secret message within an innocuous cover image, such as a photograph, and then share this image with the intended recipient. The recipient can then extract the hidden message using the LSB extraction process, provided they have the original cover image.
2. **Copyright Protection:** LSB steganography can be used to embed copyright information or ownership details within an image. This helps in proving ownership and deterring unauthorized use or distribution of copyrighted content. Watermarking is a common application of steganography in this context.
3. **Data Hiding in Medical Images:** In the medical field, steganography is used to hide patient information within medical images like X-rays, MRIs, or CT scans. This helps in protecting patient privacy while retaining necessary data for reference.
4. **Authentication:** Steganography can be used for authentication purposes. For example, a digital signature or a cryptographic hash of an image can be embedded within the image using LSB steganography. This allows anyone to verify the authenticity and integrity of the image by extracting and checking the embedded data.

5. **Covert Communication in Restricted Environments:** In scenarios where internet access or communication is restricted or monitored, individuals can use LSB steganography to hide sensitive information in seemingly harmless images. This can be especially useful in oppressive or surveillance-heavy environments.
6. **Invisible Watermarking:** LSB steganography can be used to embed invisible watermarks in images, which are imperceptible to the human eye but can be detected using specialized software. This is useful for proving the authenticity of images, especially in the field of digital forensics and image tampering detection.
7. **Geotagging and Location-Based Data:** Geotagging information (GPS coordinates) can be embedded within images using LSB steganography. This is useful for associating location data with images in a way that is not immediately obvious to viewers.
8. **Hidden Data Exchange:** Individuals or organizations can use image steganography to exchange confidential data in a covert manner, such as in corporate espionage or intelligence operations.
9. **Hiding Encryption Keys:** LSB steganography can be used to hide encryption keys within images, making it challenging for unauthorized individuals to access the keys necessary to decrypt sensitive data.
10. **Digital Art and Creative Expression:** Some artists use steganography to embed hidden messages, stories, or additional layers of meaning within their artwork, adding an extra dimension to their creative expression.

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

In conclusion, the project on "Image Steganography using Least Significant Bit" holds great promise as a means to conceal information within digital images while maintaining their visual integrity. Although the implementation is pending, the potential applications and implications of this project are significant.

The use of the Least Significant Bit (LSB) method in image steganography is an efficient and straightforward technique that allows for the hiding of data within the least significant bits of image pixels. By altering these bits, we can embed information into the image that is often imperceptible to the human eye. This technique not only offers a high level of concealment but also a low chance of detection, making it a valuable tool for various applications, such as secure communication, copyright protection, and data hiding.

6.2 Future Scope

The future scope for image steganography using LSB (Least Significant Bit) holds several opportunities and challenges. While LSB steganography has been in use for some time, there are evolving trends and areas where it can continue to be relevant:

1. **Quantum Steganography:** With the emergence of quantum computing, the field of quantum steganography may gain traction. This involves hiding information within quantum states, offering potential security advantages.
2. **Advanced Security and Authentication:** As cybersecurity becomes increasingly important, the application of LSB steganography can expand into advanced authentication and secure communication systems. This may involve embedding biometric data or multifactor authentication information in images.
3. **Digital Forensics and Image Tampering Detection:** The use of LSB steganography in image forensics will continue to grow. As more images are shared and manipulated on the internet, there is a need for improved techniques to detect tampering and verify the authenticity of images.
4. **Blockchain and Distributed Ledger Technologies:** Integrating image steganography with blockchain technology can be a potential area for development. Storing references to steganographic data in a blockchain can enhance the security and verifiability of hidden information.
5. **Legal and Ethical Considerations:** As the use of steganography techniques evolves, so do legal and ethical concerns. The future may bring about the need for regulations and standards governing the use of image steganography in various applications.

References

1. Srushti S Yadahalli, Shambhavi Rege, Dr. Reena Sonkusare “Implementation and analysis of image steganography using Least Significant Bit and Discrete Wavelet Transform techniques”. IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1.
2. Vikas Verma, Poonam, Rishma Chawla “An Enhanced Least Significant Bit Steganography Method Using Midpoint Circle Approach”. International Conference on Communication and Signal Processing, April 3-5, 2014, India.
3. Ako Muhammad Abdullah, Roza Hikmat Hama Aziz “New Approaches to Encrypt and Decrypt Data in Image using Cryptography and Steganography Algorithm”. International Journal of Computer Applications (0975 – 8887) .
4. Quang Do Vinh, Insoo Koo* , Member, KIICE ”FPGA Implementation of LSB-based Steganography”. Journal of Information and Communication Convergence Engineering.
5. Riad jabri, Boran Ibrahim and Hadi Al-Zoubi, ”Information Hiding: A Generic Approach”. Journal of Computer Science 5 (12): 930-936, 2009 ISSN 1549-3636 © 2009 Science Publications.