ABSTRACT

This project presents an Image Steganography Software developed using Python, enabling secure data hiding within image files. The software offers encryption and decryption functionalities using Triple DES taking a randomly generated secret key, allowing users to embed secret information into images or extract it when needed. Utilizing techniques such as binary conversion, pixel value manipulation, and file handling, the software ensures the concealed data remains imperceptible to the human eye. The encoding process modifies RGB pixel values to store binary data, while a ninth pixel acts as a marker to control the continuation or termination of the encoding process. The encoded image appears visually identical to the original, ensuring data confidentiality without arousing suspicion. The decryption process reverses the encoding, accurately retrieving the embedded information. This project highlights the seamless combination of cryptographic methods and image processing to enable secure information exchange.

Contents

List of Figures

Chapter 1

Introduction

With the increasing need for secure communication and confidential data exchange, conventional encryption methods often attract unwanted attention. To address this challenge, this project presents an Image Steganography Software that enables seamless and covert data transmission by embedding encrypted information within digital images. By leveraging image processing techniques and cryptographic principles, the software ensures that sensitive information remains hidden from unauthorized access while appearing visually indistinguishable from the original image.

The software integrates multiple layers of security, including encryption through randomly generated secret keys and advanced steganographic encoding mechanisms. It modifies RGB pixel values to store binary data while employing a ninth pixel as a control marker to regulate the encoding process. This method guarantees that the hidden data remains imperceptible to human vision, preserving the image's original appearance. The system also includes a decryption module, allowing users to accurately retrieve embedded information when required.

Beyond simple data concealment, this project explores the intersection of cryptography and steganography to provide a robust framework for secure information exchange. By minimizing the risk of detection and interception, this software offers a practical and efficient solution for maintaining confidentiality in digital communication.

The following sections will delve into the technical implementation, security mechanisms,

and potential applications of this software, highlighting its effectiveness.

Chapter 2

Existing Methods

2.1 Review of Literature

The literature review section of this report aims to provide an overview of existing studies on image steganography and hybrid encryption techniques for secure data hiding.

2.1.1 Image Steganography Using LSB and Hybrid Encryption Algorithms [1]

The paper "Image Steganography Using LSB and Hybrid Encryption Algorithms" (MDPI, Journal of Applied Sciences, Volume 13/Issue 21, May Alanzy et al.) explores an advanced steganographic method that combines the Least Significant Bit (LSB) technique with hybrid cryptographic approaches. The authors propose an encryption scheme that integrates symmetric and asymmetric encryption algorithms to enhance data security before embedding it into an image.

The study demonstrates that applying a hybrid encryption technique significantly improves security compared to standalone encryption or traditional LSB steganography. The proposed method ensures that even if an attacker extracts hidden data, the encryption layer makes it unreadable without the correct decryption key. The paper also discusses various evaluation metrics, such as Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error

(MSE), to assess the imperceptibility of the stego image. The results indicate minimal visual distortion while maintaining strong encryption, making this approach highly effective for secure communication.

2.1.2 Hybrid Cryptosystem Algorithm Using Vigenere Cipher and Base64 with LSB Steganography [2]

The study "Hybrid Cryptosystem Algorithm Vigenere Cipher and Base64 for Text Message Security Utilizing Least Significant Bit (LSB) Steganography as Insert into Image" (Journal of Artificial Intelligence and Engineering Applications, Vol.2, No.3, RIH Nasution et al.) presents a steganographic method that leverages both encryption and encoding techniques before embedding data into an image. The researchers employ the Vigenere cipher, a classical polyalphabetic encryption method, alongside Base64 encoding to transform text data before applying LSB steganography.

The primary advantage of this approach lies in the additional security layers provided by encryption and encoding. Base64 ensures that the textual data is converted into a binary-compatible format, while the Vigenere cipher obfuscates the content, making unauthorized extraction more difficult. The study evaluates security and imperceptibility through visual quality assessments and encryption strength analysis. The experimental results confirm that the integration of multiple security techniques enhances the robustness of the steganographic system.

2.1.3 Image Steganography: A Review of Recent Advances [3]

The paper "Image Steganography: A Review of Recent Advances" (IEEE Access, Vol.9, N. Subramanian et al.) provides a comprehensive survey of modern developments in image steganography. The review categorizes different techniques, including spatial domain methods like LSB substitution, transform domain techniques such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), and deep learning-based approaches for automated steganographic encoding and detection.

A key focus of this study is the trade-off between security, payload capacity, and imperceptibility. The authors discuss how traditional methods such as LSB substitution remain popular due to their simplicity, but they also highlight vulnerabilities like susceptibility to statistical attacks. The paper explores recent advancements in hybrid methods, where encryption techniques are integrated to improve security. Additionally, the study reviews deep learning models that can enhance steganographic encoding by learning optimal pixel modifications, reducing detectability by steganalysis tools. The findings suggest that future research should focus on developing adaptive steganographic systems that dynamically adjust encoding strategies based on the cover image characteristics.

2.2 MOTIVATION

A review of these studies highlights the growing need for enhanced security in image steganography through encryption. Traditional LSB-based methods are easy to implement but vulnerable to detection and extraction attacks. The integration of hybrid encryption schemes, as seen in the reviewed studies, strengthens the security of hidden data by adding an extra layer of protection.

However, gaps remain in ensuring a balance between security and efficiency. Many existing methods increase computational complexity, making real-time applications challenging. Additionally, most techniques focus solely on text-based data hiding without exploring multi-format payloads such as audio or video.

Motivated by these gaps, this project aims to develop an image steganography software that utilizes Triple DES encryption for secure data embedding. By combining encryption with LSB steganography and implementing a controlled marker mechanism, the system will enhance both security and data retrieval accuracy while maintaining imperceptibility. The proposed solution seeks to provide an efficient, scalable, and robust approach for secure information exchange.

Chapter 3

Problem Statement and Objectives

In the digital age, secure communication is essential to protect sensitive data from unauthorized access and cyber threats. Traditional encryption methods, while effective, often make the presence of encrypted information obvious, potentially drawing unwanted attention. To address this, steganography—hiding information within multimedia files—has emerged as a crucial technique for covert communication.

This project, StegaCrypt, combines encryption and steganography to enhance data security. By leveraging the Triple DES encryption algorithm alongside the Least Significant Bit (LSB) steganographic method, the system ensures that hidden messages remain both encrypted and imperceptible within image files. A randomly generated secret key further strengthens security, allowing only authorized recipients to extract and decrypt the embedded data.

3.1 Problem Statement

Traditional methods of data security and confidentiality often fall short when it comes to ensuring secure and inconspicuous transmission of sensitive information. Conventional encryption methods alone, while robust, may still attract attention when transmitting encrypted data. Similarly, basic steganographic techniques may not offer the desired level of security against advanced steganalysis attacks.

This project, StegaCrypt, addresses these gaps by integrating Triple DES encryption with steganography to provide a multi-layered approach to secure data concealment. The system ensures that sensitive information is not only encrypted but also embedded within images using the Least Significant Bit (LSB) technique, making it nearly imperceptible to unauthorized entities. A randomly generated secret key enhances the security, ensuring that only intended recipients can retrieve and decrypt the hidden message. The goal is to create a user-friendly and efficient system for secure communication, preventing unauthorized access and data breaches.

3.2 OBJECTIVE

The primary objective of this project is to develop a single-phase, fully functional image steganography system integrated with encryption and decryption functionalities using Triple DES. The key objectives include:

- Secure Encryption Framework: Implement Triple DES encryption to ensure robust data security before embedding messages into images.
- Steganographic Data Concealment: Utilize the Least Significant Bit (LSB) technique for embedding encrypted messages within image files while maintaining imperceptibility.
- User Authentication and Access Control: Develop a registration and authentication system to restrict access to authorized users only.
- Secret Key Generation and Management: Generate a unique random key for each encryption process and ensure its secure storage and retrieval for decryption.
- User-Friendly Interface: Provide an intuitive interface for both the sender and receiver, guiding them through the encryption, embedding, decryption, and retrieval processes.
- Incident Logging and Verification: Archive and log encrypted image files for future reference and validation, allowing users to verify the integrity of their hidden messages.

By integrating these objectives, StegaCrypt ensures a secure, efficient, and practical solution for digital communication privacy, minimizing the risks associated with unauthorized access and data exposure.

Chapter 4

Design and Implementation

This section looks into the various aspects of the design and implementation of the project. Design guidelines that should be used to make the system effective and user friendly are addressed first. Subsequently the essential design of the system including the softwares used for implementation of the same are looked into in detail.

4.1 Design Principles and Guidelines

The design principles and guidelines for StegaCrypt, the Image Steganography Software, focus on ensuring secure, efficient, and user-friendly data embedding and retrieval within image files. The key principles include:

- User-Centered Design: User-Centered Design: The software should provide a simple, intuitive interface that allows users to easily embed and extract hidden data without requiring extensive technical knowledge. A streamlined workflow will enhance usability for both novice and advanced users.

- Robust Security and Encryption: Implementing Triple DES encryption ensures that hidden data remains confidential, even if the steganographic image is intercepted. The system should use a randomly generated secret key, making it resilient against brute-force attacks.

- Imperceptibility: The embedded data should not create visible distortions in the image. The Least Significant Bit (LSB) technique, along with intelligent pixel selection, will ensure that the modified image remains visually identical to the original.
- Efficiency and Performance Optimization: The encoding and decoding processes should be optimized to handle large image files efficiently. The system should ensure minimal computational overhead while maintaining the integrity of the embedded data.
- Scalability and Adaptability: The software should be scalable, allowing future enhancements such as support for different image formats (e.g., PNG, JPEG, BMP) and potential integration with cloud-based storage for encrypted stego-images.
- Error Resilience and Data Integrity: The retrieval process must be highly reliable, ensuring that even if minor distortions occur (e.g., compression or resizing), the hidden data remains retrievable. Implementing redundancy checks and error correction techniques can enhance data integrity.

These principles collectively ensure that StegaCrypt is secure, efficient, and accessible, offering a powerful solution for covert communication and data protection.

4.2 SOFTWARE REQUIREMENT SPECIFICATION

4.2.1 Introduction

Purpose

The purpose of this project is to develop StegaCrypt, a secure image steganography software that enables users to hide and extract encrypted data within image files. By integrating Triple DES encryption with steganography techniques, the software ensures that sensitive information remains secure and undetectable, providing a robust solution for confidential communication and data protection.

Intended Audience

StegaCrypt is designed for individuals and organizations that require secure communication and covert data storage.

Project Scope

StegaCrypt provides a user-friendly, efficient, and secure method to embed encrypted data inside image files. The software ensures that:

- Data remains imperceptible within the image, preserving its quality and integrity.
- Triple DES encryption is applied to the hidden message before embedding, ensuring confidentiality even if the stego-image is intercepted.
- A randomly generated secret key is used for encryption and decryption, making brute-force attacks significantly more difficult.
- The system allows extraction and decryption of hidden data only when the correct key is provided.
- The software supports multiple image formats (JPEG, PNG, BMP) for flexibility.
- A graphical user interface (GUI) simplifies the embedding and extraction processes.
- The system maintains low computational overhead, ensuring fast and efficient performance.

4.2.2 Overall Description

Product Perspective

StegaCrypt is a standalone desktop application designed for secure image-based data hiding. It combines steganography techniques with Triple DES encryption to provide a dual-layer security mechanism. The system allows users to input a message, encrypt it with a randomly generated secret key, and embed it into an image file.

During the retrieval process, the application extracts the hidden data and decrypts it using the secret key. The software is designed to work locally, ensuring data privacy without reliance on external servers.

Product Functions

- Data Embedding: Hides encrypted text or files within an image while preserving its visual quality.
- Triple DES Encryption: Secures the hidden message with a randomly generated key, preventing unauthorized access.
- Data Extraction and Decryption: Retrieves and decrypts the hidden message only when the correct key is provided.
- Format Support: Supports common image formats such as PNG, JPEG, and BMP for maximum usability.

Product Features

- **Least Significant Bit (LSB) Steganography:** Uses the LSB technique to embed data within pixel values, ensuring imperceptibility.
- **Triple DES Encryption:** Encrypts messages before embedding them in images, adding an extra security layer.
- **Random Key Generation:** Generates a unique encryption key for every message, preventing brute-force decryption.
- **Password-Based Access Control:** Requires a user-provided key to extract and decrypt hidden messages.
- **Image Integrity Preservation:** Ensures that the visual quality of the image remains unchanged after embedding.

- **Real-Time Processing:** Provides fast encoding and decoding, ensuring efficiency for users handling large image files.
- **Stego-Image Analysis Protection:** Implements countermeasures against steganalysis attacks by minimizing detectable patterns in image modifications.

User Characteristics

StegaCrypt is designed for both technical and non-technical users. The application requires minimal expertise to operate, with an intuitive interface that guides users through the data embedding and extraction processes.

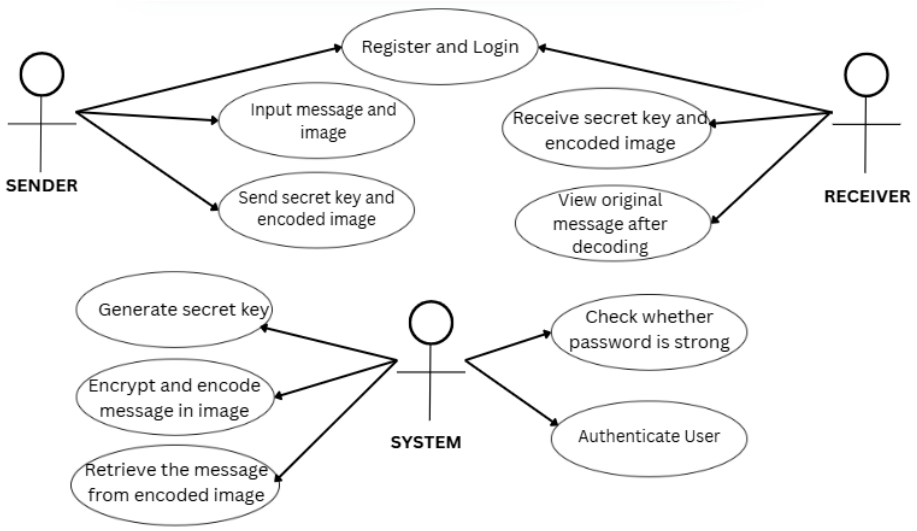

Figure 4.1: Use Case Diagram

Constraints

- Data Capacity Limit: The amount of data that can be embedded is limited by the image's size and format. Larger images can store more data without perceptible changes.
- Encryption Processing Time: While Triple DES ensures strong encryption, it may introduce slight processing delays, especially for large text files.

- Image Compression Risks: Lossy compression formats (e.g., JPEG) can potentially alter embedded data, making PNG or BMP more suitable for secure storage.

Assumptions and Dependencies

- Assumes that the software runs on systems with sufficient processing power to handle encryption and decryption efficiently.
- Assumes that users will not apply heavy post-processing (e.g., compression, resizing) to stego-images, as this could affect data extraction reliability.

Operating Environment

- Hardware: A standard modern dual-core or quad-core CPU (e.g., Intel i3/i5 or AMD Ryzen 3/5) processor. Minimum 500MB free space for storing images and encrypted data. A standard HD screen (720p or higher) for clear visualization.
- Software: StegaCrypt is developed using Python and utilizes PyCryptodome for implementing Triple DES encryption with a randomly generated secret key. OpenCV and Pillow are used for image processing, allowing seamless embedding and extraction of encrypted messages within images. The user interface is designed with Tkinter or PyQt, ensuring ease of use for encryption, decryption, and steganography functionalities. The software is cross-platform, supporting Windows, Linux, and macOS, and requires Python 3.8 or higher to ensure compatibility and optimal performance. Additional dependencies include NumPy for handling binary data efficiently.
- Platform: StegaCrypt is a cross-platform application, compatible with Windows, Linux, and macOS, ensuring accessibility for a wide range of users. It runs on any system that supports Python 3.8 or higher and requires minimal computational resources, making it suitable for both personal and professional use.

Design and Implementation Constraints

- Data Capacity Limitations: The amount of data that can be embedded is limited by the size and format of the carrier image. PNG and BMP formats are preferred to avoid data loss due to lossy compression in JPEG.
- Encryption Overhead: The use of Triple DES encryption increases processing time, requiring optimized algorithms for fast execution.
- Steganographic Detection Risks: The software must minimize visible changes in the image to reduce the risk of detection by steganalysis tools.
- System Security: Encrypted messages should be impossible to recover without the correct key. Key management is crucial.
- User Accessibility: The system must provide a simple GUI for non-technical users while maintaining advanced functionalities for security experts.
- File Size and Performance: Embedding too much data can increase file size, potentially making images suspicious.The system should optimize hidden data distribution across pixels to balance capacity and imperceptibility.

4.2.3 System Features

Functional Requirements

- Data Embedding: Allows users to hide encrypted text or small files inside images using LSB-based steganography.
- Triple DES Encryption: Encrypts the data before embedding, ensuring dual-layer security.
- Stego-Image Generation: Produces an image that looks unchanged but contains hidden encrypted information.

- Data Extraction Decryption: Retrieves and decrypts the hidden message using the correct secret key.
- Integrity Verification: Checks whether an image has been altered after embedding, ensuring reliable extraction.
- Error Handling Mechanism: Detects corrupted or tampered images, preventing unauthorized modifications.

Non-Functional Requirements

- Performance: The system must embed and extract data efficiently without causing noticeable lag.
- Scalability: Should handle multiple image formats and varying message sizes.
- Reliability: Ensures accurate retrieval of hidden data without distortion.
- Security: Uses strong encryption (Triple DES) to prevent unauthorized access.
- Usability: Provides a user-friendly interface while allowing expert users to configure advanced settings.

4.2.4 External Interface Requirements

User Interface Requirements

- Allows users to select an image, enter a message, and embed data securely.
- Provides a simple decryption panel where users input the file where the key is stored and the encrypted image.

Hardware Interface Requirements

- Processor: Must support efficient cryptographic processing for encryption/decryption tasks.

- Storage: Requires adequate disk space to store original and stego-images securely.
- Memory: Sufficient RAM to process high-resolution images without performance degradation.

Software Interface Requirements

- Encryption Module: Uses PyCryptodome for implementing Triple DES encryption.
- Image Processing: OpenCV and PIL for manipulating pixel values during embedding/extraction. NumPy for efficient array operations during LSB manipulation.
- File Handling: Supports jpeg, png files for storing hidden messages securely.
- GUI Framework: PyQt or Tkinter for creating an interactive user interface.

4.3 SOFTWARE DESIGN DOCUMENT

4.3.1 Introduction

Purpose

The purpose of this document is to outline the design principles and architecture of the StegaCrypt system. This document serves as a guide for developers, stakeholders, and security professionals, offering detailed insights into the system's design structure, encryption mechanisms, and data flow.

Scope

The StegaCrypt project aims to provide a secure and efficient steganography-based encryption system that allows users to hide sensitive information within image files. Using advanced cryptographic techniques, such as Triple DES encryption combined with pixel manipulation, StegaCrypt ensures that confidential messages remain undetectable. The system enables both encryption and decryption of hidden messages, providing a user-friendly interface for secure communication. By enhancing data security and privacy, this project seeks to offer a

robust and accessible steganographic tool for personal and professional use.

4.3.2 System Overview

StegaCrypt is an advanced steganography and encryption platform designed to securely hide sensitive messages within image files. By integrating cryptographic encryption and pixel-based steganographic techniques, StegaCrypt ensures that hidden data remains imperceptible while maintaining the integrity of the image. This solution enables users to securely encode and extract messages, providing an additional layer of security for confidential communication.

At its core, StegaCrypt utilizes an input processing system that accepts text messages and images. The text is first encrypted using Triple DES, ensuring strong security before embedding. The system generates a random secret key for encryption, which is essential for the decryption process. The encrypted data is then converted to binary format and hidden within the pixel values of the selected image. Least Significant Bit (LSB) modification is used to embed the encrypted data without significantly altering the image's appearance.

For decryption, StegaCrypt extracts the hidden binary data from the modified pixel values. The extracted data is then decrypted using Triple DES with the same secret key that was randomly generated during encryption. To ensure secure communication, the user must share both the encoded image and the secret key with the intended recipient. The recipient can then upload these two components into the system to retrieve the hidden message.

To enhance usability, StegaCrypt provides an intuitive user interface where users can easily upload images, encrypt messages, and retrieve hidden data. Additionally, the system incorporates error detection mechanisms to verify the integrity of hidden messages and prevent data loss during extraction. It also supports various image formats (PNG, BMP, and JPEG) for flexibility and broad compatibility.

Designed for efficiency and scalability, StegaCrypt maintains a lightweight architecture that can run on low-resource systems while ensuring fast encryption and decryption pro-

cesses. Its layered structure allows for future enhancements, such as additional encryption algorithms and improved steganographic techniques. By combining cryptographic security with image-based concealment, StegaCrypt provides a reliable tool for secure and discreet communication, ensuring that only authorized recipients with the correct key can access the hidden information.

4.3.3 System Architecture

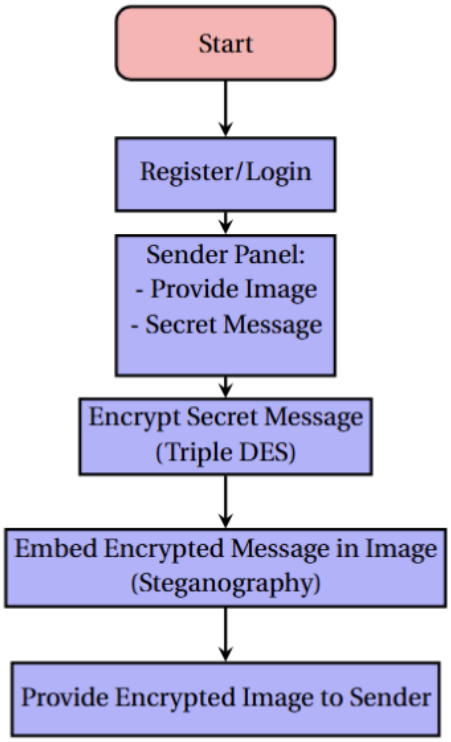

Figure 4.2: Sender's Workflow

User Registration and Authentication:

- Users create an account by registering their details, including a username and password.
- Authentication ensures secure access for both senders and receivers.

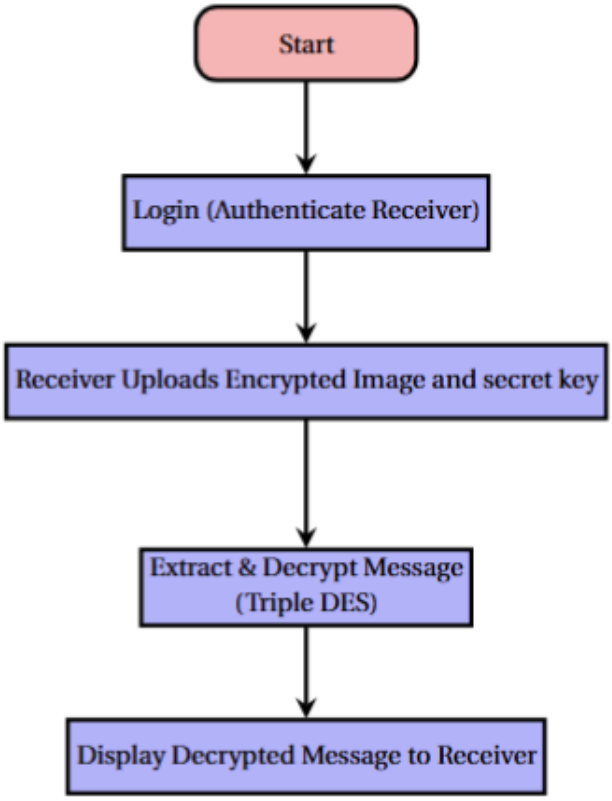

Figure 4.3: Receiver's Workflow

Input Data Collection:

- **Sender:** Uploads an image and provides the secret message to be hidden.
- **Receiver:** Uploads the encrypted image and the secret key to retrieve the hidden message.

Encryption and Embedding (Sender's Workflow):

- The sender's secret message is encrypted using the Triple DES encryption algorithm with a unique randomly generated key which is stored on the system once created.
- The encrypted message is embedded into the image using the Least Significant Bit (LSB) steganography technique.

- The system generates a modified image containing the hidden message, which the sender can download and share.

Verification and Decryption (Receiver's Workflow):

- The receiver uploads the encrypted image to the platform.
- The receiver also uploads the secret key generated.
- The encrypted message is extracted from the uploaded image and decrypted using Triple DES.
- The original secret message is displayed securely to the receiver.

User Interface and Interactivity:

- Interactive widgets guide users through the upload, embedding, and decryption processes.
- Visual feedback is provided to confirm successful encryption and decryption.

Output Delivery:

- **Sender:** Receives the encrypted image for sharing with the receiver.
- **Receiver:** Views the original message securely after decryption.

4.3.4 Data Design

StegaCrypt employs a secure SQLite database architecture for user authentication and credential management. The system implements enterprise-grade security measures to protect sensitive user data while maintaining optimal performance for steganography operations.

Database Overview

The system utilizes a single-table relational database design with the following security features:

users			
int	id	PK	AUTOINCREMENT
varchar	username		UNIQUE
varchar	password		BCRYPT HASHED

Figure 4.4: Users Table

- Military-grade password hashing using bcrypt algorithm
- Automatic salting with 128-bit cryptographically secure salts
- Configurable hash complexity (default: 12 rounds)
- Fixed-length hash storage (60-character format)
- Local database encryption at rest

Security Implementation

The authentication system implements multiple layers of protection:

- **Password Storage**: Uses bcrypt's adaptive hash function ($2b$ format) with:
  - Unique per-user salt generation
  - Computational cost hardening against brute-force attacks
  - Built-in protection against rainbow table attacks

- **Database Operations**:
  - Parameterized queries to prevent SQL injection
  - Transactional integrity for all write operations
  - Automatic indexing on username field for fast lookups
- **Data Validation**:
  - Username uniqueness enforcement at database level
  - Null constraints on all fields Input sanitization before database operations

4.4 STRENGTHS AND WEAKNESSES

4.4.1 Strengths

- Secure Encryption and Concealment: StegaCrypt ensures high-level security by encrypting messages with Triple DES before embedding them into images, making it extremely difficult for unauthorized users to access the hidden data.
- Random Secret Key Generation: The system generates a unique secret key for each encryption process, ensuring that only intended recipients can decrypt and retrieve the concealed message.
- Stealthy Data Hiding: By using Least Significant Bit (LSB) modification, StegaCrypt embeds encrypted messages within image pixels with minimal distortion, making detection nearly impossible.
- Cross-Platform Compatibility: The system supports various image formats (PNG, BMP, JPEG) and can run on different operating systems, enhancing its usability across multiple devices.
- User-Friendly Interface: With an intuitive design, users can easily upload images, encrypt messages, and extract hidden data without requiring advanced technical skills.

4.4.2 Weaknesses

- Key Dependency for Decryption: If the randomly generated secret key is lost or misplaced, the encrypted message cannot be recovered, making key management crucial for users.
- Limited Data Capacity: The amount of data that can be hidden depends on the image size—larger messages require higher-resolution images, which may not always be practical.
- Potential for Image Distortion: Although LSB-based steganography minimizes noticeable changes, excessive data embedding may slightly alter image quality, making it detectable in some cases.
- Processing Time for Large Files: Encrypting and embedding messages in high-resolution images may take longer, affecting performance on low-end devices.
- Security Risks if Key is Shared Insecurely: If the secret key is shared over an unsecure channel, an attacker may intercept it, compromising the confidentiality of the hidden message.

Chapter 5

Results and Discussion

5.1 Introduction

This section presents the results of StegaCrypt, an advanced steganography and encryption system designed for securely embedding encrypted messages into images. The system ensures confidential and undetectable data transmission by combining Triple DES encryption with Least Significant Bit (LSB) steganography. Users can encode sensitive information within an image and share it along with a randomly generated secret key, allowing the recipient to retrieve the hidden message upon decryption.

StegaCrypt enhances data security and privacy by providing an intuitive platform for users to securely conceal information without raising suspicion. The following sections outline the programming languages, frameworks, and libraries used, along with details on the deployment, user interface, and system performance in terms of encryption speed and image quality retention.

5.2 Technical Details

5.2.1 Languages and Libraries

- Python – Core language for backend processing and encryption.

- PIL (Pillow) – Library for image processing, used for modifying pixel values.
- PyCryptodome – Implements Triple DES encryption for secure message encoding.
- NumPy – Assists with efficient image and data manipulation.
- Tkinter – Used to create a user-friendly desktop GUI.

5.2.2 Frameworks

- SQLite – Lightweight database for storing encryption logs, user data, and message history.
- Tkinter – Framework for building a simple, intuitive GUI for easy interaction with StegaCrypt.

5.2.3 Cryptographic and Steganographic Techniques

- Triple DES (3DES) – Encrypts messages before embedding them into images, ensuring high security.
- Least Significant Bit (LSB) Steganography – Modifies the least significant bits of image pixels to store encrypted data without noticeable visual changes.

5.3 USER INTERFACE

The StegaCrypt system is built as a desktop application with a graphical user interface designed for secure image steganography operations. The interface provides intuitive navigation between authentication, encoding, and decoding functionalities.

5.3.1 Desktop Application

The application comprises the following key interfaces:

Login Page

The Login Page serves as the entry point to the StegaCrypt system, featuring secure authentication with username and password verification. The interface includes:

- Username and password input fields with secure password masking
- Real-time validation with error messages for incorrect credentials
- Secure password storage using bcrypt hashing
- Registration button for new users

The authentication system queries the SQLite database to verify credentials, ensuring only authorized users can access the steganography features.

Registration Page

New users create accounts through this interface, which enforces:

- Unique username validation (prevents duplicate accounts)
- Strong password requirements (minimum 8 characters with special symbols)
- Secure password hashing before database storage
- Immediate feedback for registration errors

Main Dashboard

After successful login, users access the central hub offering:

- Clear navigation to encoding and decoding functions
- System overview explaining steganography capabilities
- Secure logout functionality
- Responsive design with consistent color scheme (#093545 background)

Encoding Interface

The encoding page enables users to hide secret messages in images using:

- Triple DES encryption for message security (24-byte keys)
- Image selection dialog with PNG/JPG support
- Message input field with multi-line support
- Automatic key generation and desktop storage
- Visual confirmation of successful encoding

The system employs LSB (Least Significant Bit) steganography to embed encrypted messages without perceptible image quality loss.

Decoding Interface

This page allows extraction of hidden messages with:

- Encrypted image selection
- Key file upload functionality
- Real-time decryption using Triple DES
- Clean display of extracted messages
- Error handling for invalid keys or corrupted images

Operation Results Page

After successful encoding, users see:

- Side-by-side comparison of original and encoded images to save the steganographic image
- Navigation options to return to encoding or switch to decoding

All interfaces maintain consistent styling with:

- Dark theme (#093545 background) for reduced eye strain
- Responsive layout (862x519 fixed window size)
- Clear typography hierarchy (Poppins and Lexend Deca fonts)

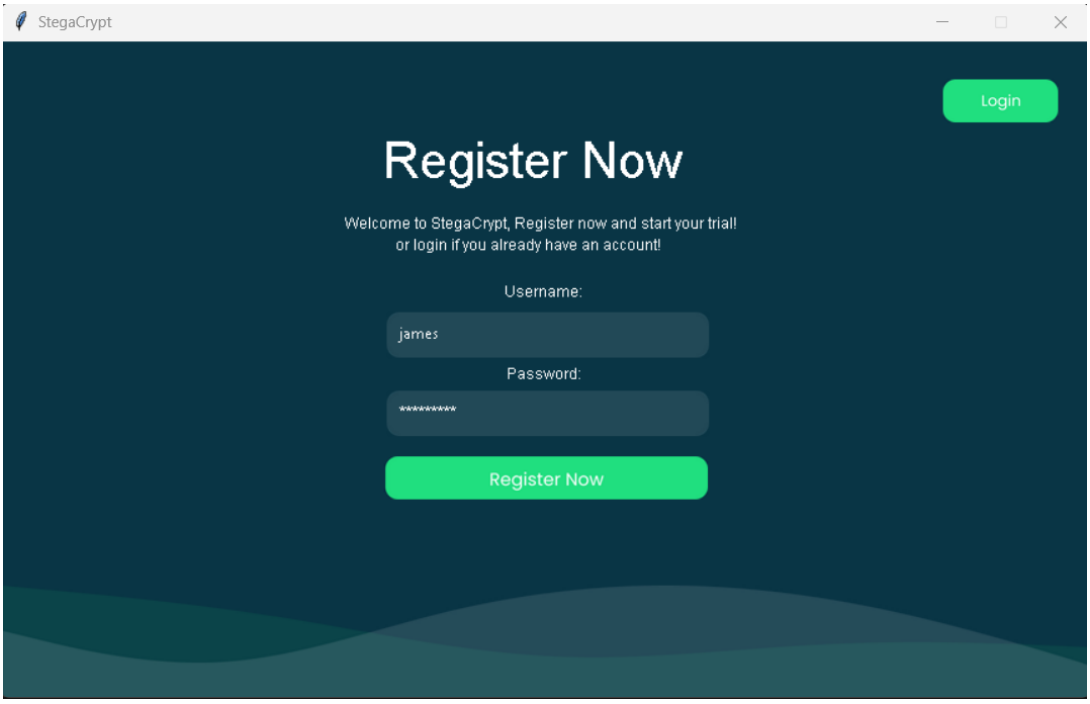

Figure 5.1: Resgistration Interface

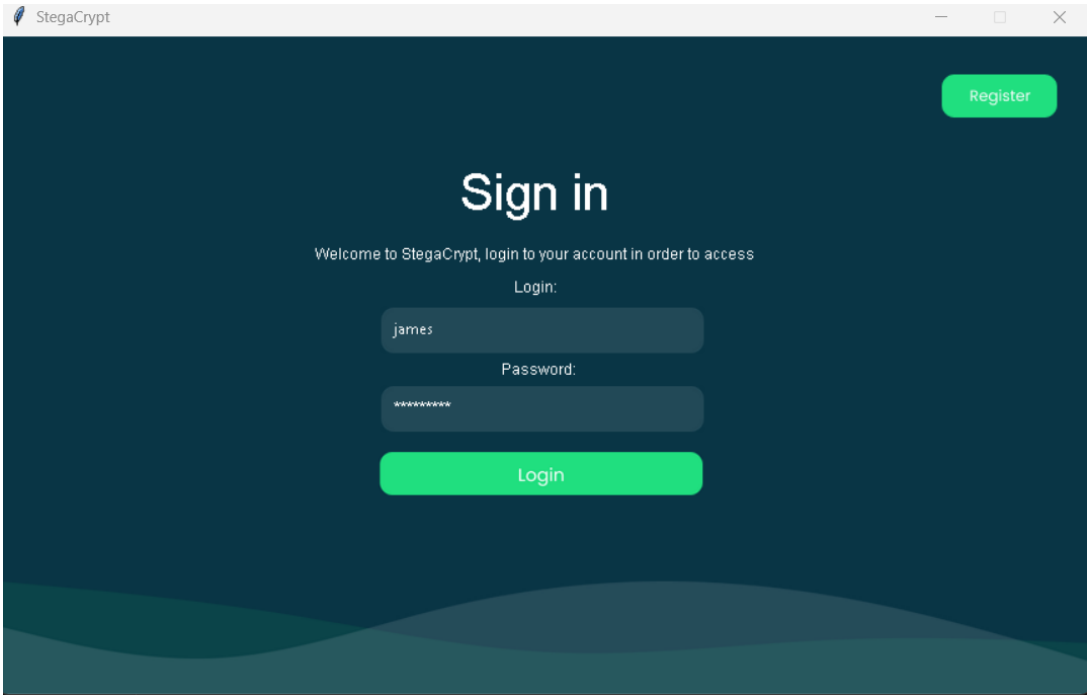

Figure 5.2: Login Interface

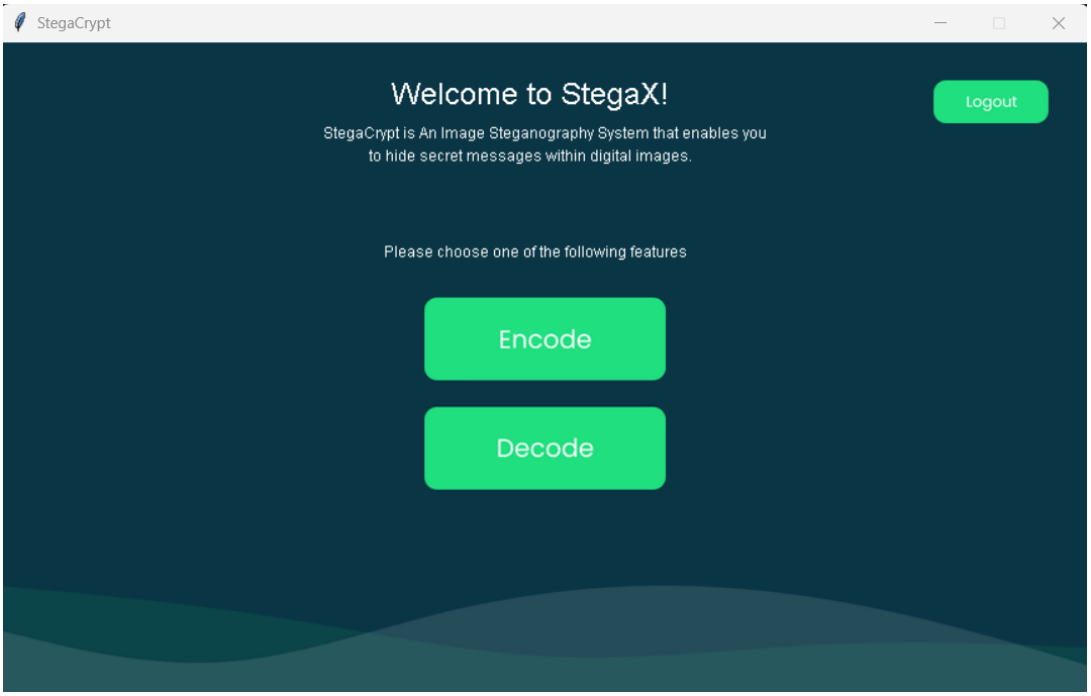

Figure 5.3: Home Interface

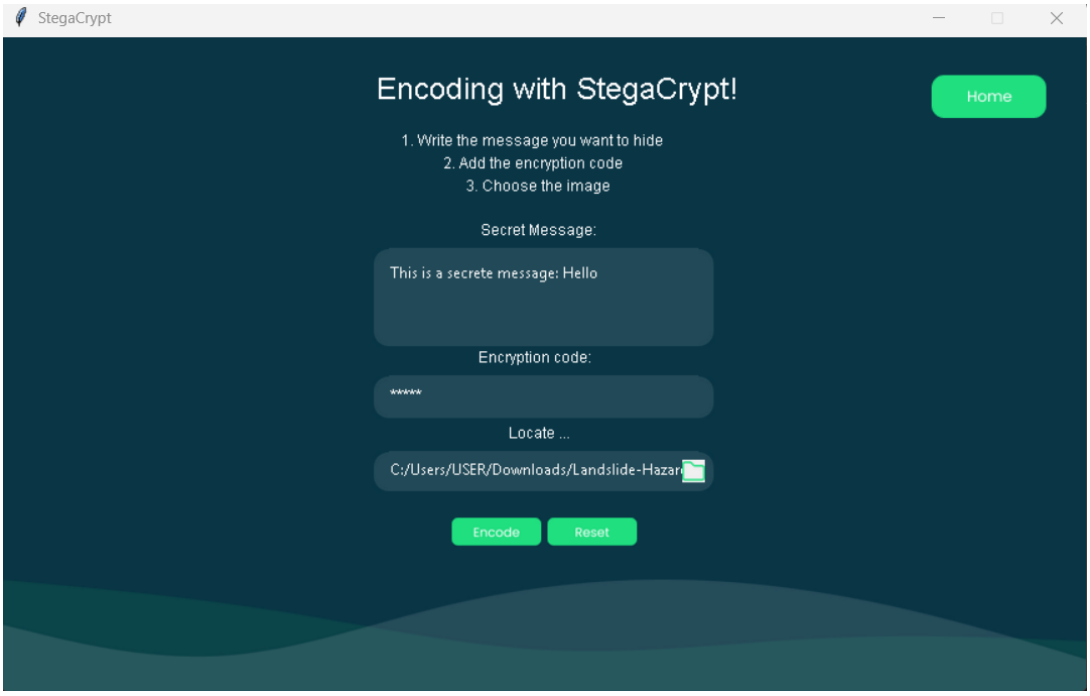

Figure 5.4: Encode Interface

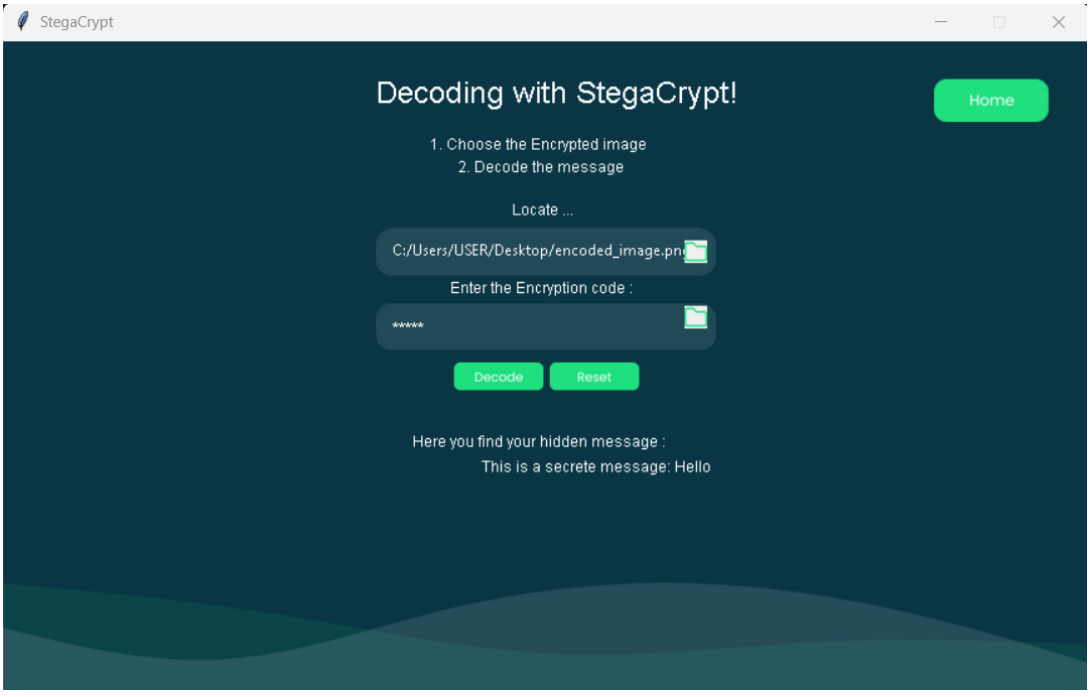

Figure 5.5: Decode Interface

5.3.2 File Transfer System

The StegaCrypt system includes a secure file transfer mechanism for sharing encoded images and encryption keys between users. This client-server implementation operates over a local network connection.

5.3.2.1 Client Implementation

The client component features:

- Socket-based communication using TCP/IP protocol
- Continuous file transfer mode until explicit exit
- Robust file validation before transmission
- Secure filename extraction using `os.path.basename()`
- Chunked file transfer (1024-byte blocks) for large files
- Clear EOF signaling using `b"EOF"` marker

Key client operations:

1. Establishes connection to server at `127.0.0.1:12345`
2. Prompts user for file path input
3. Validates file existence before transmission
4. Sends filename as metadata first
5. Transmits file contents in binary mode
6. Terminates connection on "exit" command

5.3.2.2 Server Implementation

The server component provides:

- Secure file reception with directory validation
- Protection against directory traversal attacks
- Proper EOF handling for complete file transfers
- Dedicated save folder (`ReceivedFiles`)
- Connection management for multiple transfers

Server workflow:

1. Creates listening socket on port 12345
2. Validates and creates save directory if needed
3. Processes incoming files with proper EOF detection
4. Handles empty filename cases gracefully
5. Maintains connection until "DONE" signal received

5.3.2.3 Usage Example

Typical workflow for steganography operations:

1. User encodes message into image using GUI
2. System generates 3DES key and encoded image
3. Client transfers both files to recipient's server
4. Recipient decodes using received files

The system ensures reliable transfer of:

- Encoded PNG images (typically `encoded_image.png`)
- Encryption key files (typically `3des_key.txt`)

```
PS C:\Users\USER\Documents\GitHub\ImageSteganographySystem> python client.py
Sender:
Connected to server at 127.0.0.1:12345
Enter file path to send (or type 'exit' to quit): C:\Users\USER\Desktop\3des_key.txt
Sent: C:\Users\USER\Desktop\3des_key.txt
Enter file path to send (or type 'exit' to quit): C:\Users\USER\Desktop\encoded_image.png
Sent: C:\Users\USER\Desktop\encoded_image.png
Enter file path to send (or type 'exit' to quit): exit
All files sent successfully.
```

Figure 5.6: Sender

```
PS C:\Users\USER\Documents\GitHub\ImageSteganographySystem> python server.py
Receiver:
Server listening on 127.0.0.1:12345...
Connection established with ('127.0.0.1', 50951)
Receiving file: 3des_key.txt
Received: 3des_key.txt -> Saved as: C:\Users\USER\Documents\ReceivedFiles\3des_key.txt
Receiving file: encoded_image.png
Received: encoded_image.png -> Saved as: C:\Users\USER\Documents\ReceivedFiles\encoded_image.png
All files received. Closing connection.
```

Figure 5.7: Receiver

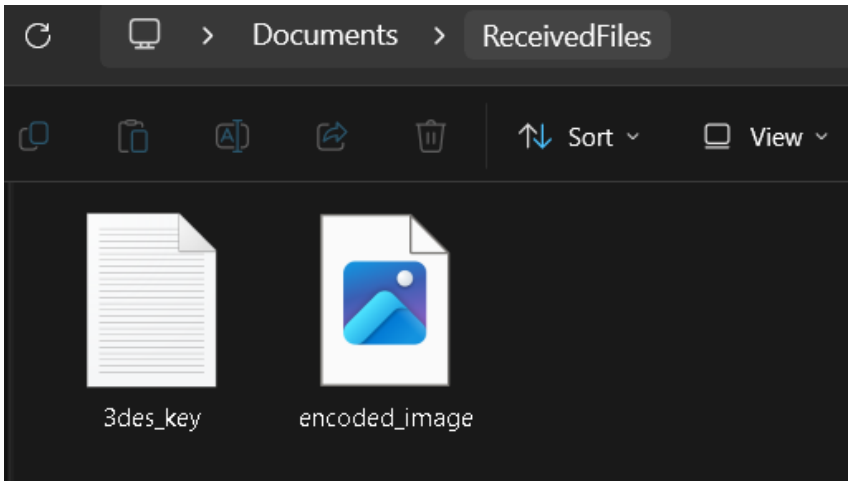

Figure 5.8: Received files

Chapter 6

Conclusion and Future Scope

The successful completion of the StegaCrypt project demonstrates the achievement of its primary objectives—secure data transmission and access control through the integration of steganography and encryption techniques. By leveraging Least Significant Bit (LSB) steganography alongside Triple DES encryption, the system ensures a robust and user-friendly framework for securely embedding and retrieving hidden data within images. The developed solution enables seamless encryption, embedding, and decryption workflows, thereby offering a reliable method for concealing sensitive information within digital media.

The implementation results validate the effectiveness of StegaCrypt in securely transmitting hidden messages while maintaining the imperceptibility of the embedded data. The project highlights the potential of steganography as an additional layer of security in digital communications, reinforcing its importance in secure data exchange applications.

The future scope of the application can be expanded as follows:

- Remote Database Integration: Implementing cloud-based storage for encrypted images and keys would enhance accessibility and scalability, allowing users to retrieve hidden messages from any location securely.

- Support for Multiple File Formats: Extending the steganography function to support data hiding in formats beyond images, such as audio (MP3, WAV), video (MP4), and text files, would increase versatility and application scope.

- Enhanced Encryption Mechanisms: Integrating more advanced cryptographic techniques such as AES or RSA alongside Triple DES could improve security and resilience against brute-force attacks.
- Improved User Interface and Experience: Developing a more intuitive and visually appealing user interface, including real-time previews and progress indicators, could enhance user experience.
- Mobile Application Development: Creating a mobile-friendly version of StegaCrypt would increase usability, making it more accessible to a wider range of users.
- Blockchain-based Security Enhancements: Implementing blockchain technology for key management and authentication could further reinforce the security and integrity of the system.
- AI-driven Steganalysis Prevention: Exploring adversarial techniques to counteract steganalysis attacks and improve the system's resistance to unauthorized detection of embedded data.

References

[1] Alanzy, May, Razan Alomrani, Bashayer Alqarni, and Saad Almutairi. 2023. "Image Steganography Using LSB and Hybrid Encryption Algorithms" Applied Sciences 13, no. 21: 11771. https://doi.org/10.3390/app132111771

[2] Nasution, Raja Imanda Hakim, Achmad Fauzi, and Husnul Khair. 2023. “Hybrid Cryptosystem Algorithm Vigenere Cipher and Base64 for Text Message Security Utilizing Least Significant Bit (LSB) Steganography As Insert into Image”. Journal of Artificial Intelligence and Engineering Applications (JAIEA) 2 (3):89-98. https://doi.org/10.59934/jaiea.v2i3.201.

[3] N. Subramanian, O. Elharrouss, S. Al-Maadeed and A. Bouridane, "Image Steganography: A Review of the Recent Advances," in IEEE Access, vol. 9, pp. 23409-23423, 2021, doi: 10.1109/ACCESS.2021.3053998.