Implementing Modified AES(AES+Blowfish) In Steganography Application

Mini Project Report

Submitted By

Ajay Dileep (Reg. No. SJC22CC003)
Allwin Nebu (Reg. No. SJC22CC010)
Alvin Boby Mathew (Reg. No. SJC22CC011)
Mathew Somy (Reg. No. SJC22CC039)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the degree

of

Bachelor of Technology

in

Computer Science and Engineering (Cyber Security)

Department of Computer Science and Engineering (Cyber Security)

St. Joseph's College of Engineering & Technology
Palai - 686 579

April 2025

St. Joseph's College of Engineering and Technology, Palai

Department of Computer Science and Engineering (Cyber Security)

CERTIFICATE

This is to certify that the Mini-project report entitled **Implementing Modified AES(AES+Blowfish) In Steganography Application** submitted by **Ajay Dileep (SJC22CC003)**, **Allwin Nebu (SJC22CC010)**, **Alvin Boby Mathew (SJC22CC011)**, **Mathew Somy (SJC22CC039)**, Sixth Semester B.Tech Computer Science and Engineering (Cyber Security) students, is a bonafide record of the Mini-Project done by them, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **B.Tech Computer Science and Engineering (Cyber Security)** of **APJ Abdul Kalam Technological University, Kerala**.

Prof. Devina Vinod	**Prof. Sabarinath G. Pillai**	**Dr. Sruthy S**
Supervisor	Mini Project Coordinator	Head
Dept. of CC	Dept. of CC	Dept. of CC

Acknowledgment

I wish to record my indebtedness and thankfulness to all who helped me prepare this Mini Project Report titled **Implementing Modified AES(AES+Blowfish) In Steganography Application** and present it in a satisfactory way.

I would like to convey my special gratitude to **Dr. V.P. Devassia**, Principal, SJCET, Palai,for the moral support he provided. I express my sincere thankfulness to **Dr. Sruthy S**, Head of the department, Department of Computer Science and Engineering (Cyber Security) for her co-operation and valuable suggestions. Also I express my sincere thanks to Mini project co-ordinator **Prof. Sabarinath G. Pillai** for his helpful feedback and timely assistance.

I am especially thankful to my guide **Prof. Devina Vinod**, Assistant Professor, Department of Computer Science and Engineering (Cyber Security) for giving me valuable suggestions and critical inputs in the preparation of this report.

I also extend my thanks to all my friends who helped me by giving motivation. Their smallest piece of advice was really valuable. Once again I convey my gratitude to all those persons who had directly or indirectly influenced my work as a whole.

Ajay Dileep
Allwin Nebu
Alvin Boby Mathew
Mathew Somy
St. Joseph's College of Engineering & Technology
Palai - 686 579.

Abstract

StegApp is an advanced full-stack steganography solution designed to securely embed and extract encrypted data within image files. The application integrates state-of-the-art encryption algorithms, including AES and Blowfish, to ensure robust data security. The frontend is developed using CustomTkinter in Python, providing an intuitive and user-friendly interface that facilitates the embedding and extraction of various data types, such as text, audio, and images, within carrier images. Key features of the frontend include real-time progress monitoring, encryption key management, and streamlined file selection tools, making the steganographic process accessible to users with varying levels of technical expertise. The backend, implemented in Python, optimizes image processing through the use of OpenCV and NumPy, while GPU acceleration via CUDA and multi-threading techniques significantly enhance the performance of large-scale data embedding and extraction operations. By combining advanced encryption, high-performance image manipulation, and user-friendly design, StegApp offers a secure, efficient, and scalable solution for digital data protection and secure communication. This work demonstrates the effective application of modern cryptographic techniques and parallel processing in the field of steganography, offering both security and efficiency for sensitive data storage within digital media.

Keywords— Steganography, Data Encryption, AES, Blowfish, GPU Acceleration, CUDA, CustomTkinter, Image Processing, Secure Communication.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

StegApp is a modern steganography and data security application designed to safeguard sensitive information by embedding encrypted data within images. In an era of increasing digital surveillance and data vulnerability, StegApp provides a discreet and robust method for secure communication.

The project features a CustomTkinter-based frontend for an intuitive user experience and a Python-powered backend leveraging AES and Blowfish encryption to protect hidden data. OpenCV and NumPy ensure efficient image processing, while CUDA-accelerated GPU computation optimizes performance for large-scale data embedding and extraction. Users can securely hide text, audio, and image files within carrier images, with real-time progress tracking and encryption key management for added security.

Key features include strong encryption, GPU-optimized steganographic processing, multi-threaded execution for faster operations, and a modern, user-friendly interface. StegApp bridges the gap between security and usability, offering a powerful solution for covert data transmission and secure digital communication.

1.1 Problem Statement

In today's digital era, secure communication and data protection have become critical concerns. Traditional encryption methods, while effective, often draw attention to the presence of sensitive data, making them susceptible to interception. Encryption techniques, such as AES and RSA, secure data transmission but do not hide its existence. Consequently, adversaries can easily detect and target encrypted content, potentially compromising its confidentiality.

Organizations and individuals face challenges in securely transmitting confidential information without raising suspicion. Existing methods, such as password-protected files or cryptographic hashing, provide data security but fail to offer covert transmission capabilities. Furthermore, surveillance and cyber threats have increased, necessitating more advanced techniques to ensure data privacy and secure communication.

Steganography, the practice of concealing information within non-suspicious media, provides a viable solution to these challenges. However, existing tools either sacrifice security for simplicity or are too complex for regular use, limiting their practicality. Many conventional steganographic tools have significant drawbacks, including limited payload capacity, vulnerability to steganalysis, and lack of proper encryption mechanisms.

StegApp addresses these limitations by offering a secure, high-performance steganography solution with a user-friendly interface. By integrating AES and Blowfish encryption, CUDA-accelerated processing, and multi-threaded execution, StegApp ensures efficient and secure data hiding and extraction. The application is designed for both novice and expert users, providing a seamless and effective approach to digital steganography.

1.2 Core Functionalities and Features

1.2.1 Data Hiding and Extraction

StegApp embeds and retrieves data within carrier images or audio files using LSB-based (Least Significant Bit) steganography. This technique ensures minimal perceptual distortion while maximizing payload capacity and maintaining the integrity of the carrier file. The application supports multiple formats, including PNG, BMP, and WAV, ensuring broad usability. Advanced image processing techniques, such as histogram equalization and perceptual masking, enhance security against forensic analysis.

1.2.2 Hybrid Encryption

StegApp employs a hybrid encryption model combining AES and Blowfish encryption. AES provides fixed key lengths of 128, 192, or 256 bits, ensuring strong security, while Blowfish offers variable key lengths and fast encryption, making it efficient for large datasets. By encrypting data before embedding, StegApp ensures that even if steganographic content is detected, it remains unreadable without the correct decryption key.

1.2.3 GPU Acceleration

CUDA-based GPU acceleration significantly speeds up embedding and extraction processes by enabling parallel execution of multiple operations. For large-scale applications, this provides substantial performance improvements, allowing efficient handling of high-resolution images and large audio files.

1.2.4 Error Handling and Validation

To ensure reliability, StegApp incorporates advanced error-handling mechanisms. Carrier file integrity verification prevents embedding in corrupted files, while encryption key compatibility checks ensure proper encryption and decryption. Capacity management prevents data overflow, and embedded data validation detects corruption. Additionally, adaptive redundancy encoding enhances error recovery, providing robust error detection and minimizing the risk of data loss.

1.2.5 User Interface

StegApp features a modern GUI built using CustomTkinter. It provides real-time status updates, progress tracking, and dynamic input fields for selecting carrier files and encryption keys. The responsive design adapts to different screen resolutions, offering dark mode and customizable themes. Keyboard shortcuts and accessibility features further enhance usability, making the application intuitive for both novice and expert users.

1.3 Implementation Details and Optimizations

1.3.1 Adaptive Image Resizing

StegApp dynamically adjusts carrier image dimensions to accommodate different data sizes while maintaining quality. This optimization ensures minimal visual distortion and efficient utilization of available storage capacity.

1.3.2 Multi-Threading for Parallel Processing

To improve efficiency, StegApp employs multi-threading, distributing processing tasks across multiple CPU cores. This enhances performance during encryption, file I/O operations, and large-scale data embedding while maintaining application responsiveness.

1.3.3 Secure Key Management with SHA-256

Encryption keys in StegApp are processed using SHA-256 hashing, ensuring enhanced security and preventing weak keys from compromising data protection. This approach reduces vulnerability to brute-force attacks and strengthens overall encryption reliability.

1.3.4 Optimized Memory Usage

Memory-mapped file handling and buffered reading/writing optimize memory usage, enabling efficient processing of large media files. This technique prevents excessive RAM consumption, ensuring smooth performance even for high-resolution images and large audio files.

1.4 Applications and Future Scalability

StegApp supports various applications, including secure messaging, confidential communication, and intellectual property protection through digital watermarking. It facilitates covert transmission of sensitive data in high-security environments and enables secure storage within images and audio files. Additionally, it proves useful in forensic investigations requiring hidden data storage and plays a role in anti-piracy measures for multimedia content distribution.

1.5 Objectives and Scope

1.5.1 Objectives

The primary objective of StegApp is to develop a highly secure, efficient, and scalable steganographic application that allows users to embed and extract hidden data within digital media files. The system ensures confidentiality, data integrity, and high-speed processing by integrating advanced cryptographic algorithms, optimized steganographic techniques, GPU acceleration, and a user-friendly graphical interface.

StegApp is designed to address a wide range of use cases, including covert communication, intellectual property protection, forensic investigations, and secure digital storage. The system aims to balance security, performance, and usability, providing a robust solution for modern steganography.

Confidential and Undetectable Data Embedding

A core objective of StegApp is to ensure secure, imperceptible, and resilient data embedding. The system minimizes the risk of detection through various advanced steganographic techniques.

Least Significant Bit (LSB) modification is employed to embed data within the least significant bits of image pixels or audio waveforms, ensuring that visual and auditory properties remain unchanged. This method is further enhanced by adaptive embedding techniques that dynamically select pixel regions or audio segments based on entropy analysis. This approach ensures that steganalysis-based detection methods cannot easily identify hidden data.

To further protect against statistical anomalies, StegApp integrates histogram equalization and perceptual masking techniques, which modify image or audio properties in a way that conceals the presence of embedded data. Multi-layered redundancy encoding is also applied, improving robustness by embedding error-correcting codes to ensure data integrity even in lossy transformations such as image compression or audio noise interference.

Security is further reinforced through AES-256 and Blowfish encryption, encrypting the embedded data before encoding it within the carrier medium. Key-based data extraction mechanisms ensure that only authorized users possessing the correct decryption key can retrieve hidden information, preventing unauthorized access.

High-Performance Processing and Optimization

Efficiency and speed are critical components of StegApp, particularly when handling high-resolution images and large audio files. The system is designed to achieve real-time embedding and extraction through multiple optimization strategies.

CUDA-based GPU acceleration is utilized to parallelize pixel and waveform operations, significantly reducing execution time compared to CPU-only processing. This is further complemented by multi-threaded execution, enabling concurrent processing of encryption, embedding, and file input/output operations to prevent bottlenecks.

Memory-mapped I/O techniques optimize file handling, reducing RAM usage while processing large files to ensure efficient resource allocation. Adaptive compression techniques minimize the size of embedded payloads while maintaining the quality of the original media files.

Dynamic resource allocation mechanisms distribute workload between CPU and GPU cores based on the available system resources, optimizing performance

across different hardware configurations. These optimizations enable StegApp to support large-scale applications where speed and efficiency are paramount.

User-Friendly Interface and Accessibility

While StegApp integrates advanced cryptographic and steganographic techniques, it is designed to be user-friendly and accessible to a broad audience. The CustomTkinter-based graphical interface ensures that users, regardless of technical expertise, can securely embed and extract data with ease.

The modern graphical interface includes a drag-and-drop file selection feature that simplifies the process of choosing carrier files and payloads. Users receive real-time feedback on the progress of embedding and extraction processes, including estimated completion time and system resource usage.

Secure key management tools are integrated to allow users to generate, store, and retrieve encryption keys safely. The interface also includes accessibility features such as customizable themes, font size adjustments, and keyboard shortcuts.

Automated error handling mechanisms detect potential issues in real time and provide detailed log reports for troubleshooting. These features ensure that even users with limited technical knowledge can utilize StegApp effectively.

Scalability and Future Development

StegApp is designed with a modular and scalable architecture, allowing for seamless integration of future enhancements. Several planned expansions will enhance the platform's capabilities.

One of the primary future developments includes support for video steganography, enabling users to embed data within video files using motion vector analysis and frame-level encoding techniques. AI-assisted steganography will also be explored, leveraging machine learning models to optimize data concealment techniques and improve resistance against automated detection.

To enhance security further, quantum-resistant cryptography will be integrated, utilizing lattice-based encryption methods to protect against future quantum computing threats. Cloud-based secure storage solutions will be implemented, allowing users to store and retrieve steganographic data remotely in a fully encrypted environment.

Additional features under consideration include cross-platform compatibility to support macOS, iOS, and Android devices, as well as steganographic watermarking for copyright protection. These enhancements will ensure that StegApp remains a cutting-edge tool for secure digital communication.

1.5.2 Scope of Development

StegApp's development encompasses multiple core areas, including frontend and backend implementation, encryption security, performance optimization, and future scalability.

Frontend Development

The frontend of StegApp is designed to provide an intuitive user experience. The graphical user interface is built using CustomTkinter, featuring a modern and interactive design that allows users to perform steganographic operations with ease.

The interface supports real-time status updates, displaying progress bars, estimated processing time, and system resource utilization. It includes user configuration options that allow individuals to customize encryption strength, steganographic techniques, and performance settings according to their needs.

The secure key handling interface provides mechanisms for generating, storing, and retrieving encryption keys safely. This ensures that encryption keys are managed securely, preventing unauthorized access.

Backend Processing and Algorithmic Efficiency

The backend is optimized for fast and secure steganographic operations. It integrates OpenCV and NumPy for efficient image processing, allowing pixel data manipulation with minimal computational overhead.

Multi-threading support enables parallel execution of key tasks, including encryption, data embedding, and file input/output operations. This significantly improves processing speed and reduces latency.

Adaptive steganographic techniques are implemented to intelligently select optimal embedding locations within the carrier medium, enhancing security while maintaining imperceptibility.

Robust file handling mechanisms, including memory-mapped I/O, are incorporated to ensure efficient processing of large media files without excessive memory consumption.

Security Mechanisms and Cryptographic Implementation

StegApp employs multiple layers of security to protect embedded data from unauthorized access and tampering.

AES-256 and Blowfish encryption algorithms provide strong data protection, ensuring that hidden information remains confidential. SHA-256 hashing is used

for secure key verification, preventing unauthorized users from accessing or modifying steganographic content.

To further strengthen security, two-factor authentication is planned for future versions, adding an additional layer of protection when decrypting embedded data.

Application Areas and Use Cases

StegApp is designed to serve a diverse range of applications and industries.

In secure digital communication, it enables covert data exchange between individuals who require private messaging channels. It is also valuable for intellectual property protection, allowing content creators to embed copyright metadata within their digital media files.

Forensic investigations benefit from StegApp's capabilities, as it allows law enforcement and cybersecurity professionals to conceal and retrieve sensitive evidence without raising suspicion.

The system also supports anti-piracy measures by embedding unique identifiers within media files, enabling copyright holders to track unauthorized distribution.

Cloud-based secure storage is another important application area, allowing businesses and individuals to store sensitive information in encrypted formats within cloud environments.

As a multi-purpose steganographic solution, StegApp is designed to support a wide range of use cases, ensuring that secure data embedding and extraction can be performed reliably and efficiently.

Chapter 2

Literature Review

2.1 Digital Image Steganography: Principles, Algorithms, and Applica- tions (2018) Johnson et al

Johnson et al. conducted a comprehensive study on various steganographic techniques, particularly focusing on Least Significant Bit (LSB) methods. Their research significantly influenced StegApp's choice of LSB for embedding data in image and audio files. They provided an in-depth analysis of the advantages and limitations of LSB techniques, which serve as the foundation for modern steganographic applications. article

LSB-based Data Hiding

Modifies the least significant bits of pixel values in an image or sample values in an audio file. Enables data embedding without significant perceptible changes. A simple yet effective technique for hiding information while preserving the original structure.

- **Capacity and Imperceptibility Trade-offs**

Balances between embedding capacity and visual imperceptibility. Overloading LSBs can make steganographic changes detectable. StegApp adopts an adaptive embedding strategy: Assesses image properties dynamically. Adjusts embedding intensity accordingly.

- **Vulnerabilities to Statistical Attacks**

LSB substitution is predictable and susceptible to steganalysis (e.g., histogram analysis, machine learning detection). To enhance security, StegApp integrates AES and Blowfish encryption before embedding. Ensures robust data protection, making extraction significantly more difficult even if the stego-medium is compromised.

Johnson et al. also explored how LSB methods apply to audio and video files, demonstrating that each media type requires unique adaptations. Their work highlighted that noise distribution in different formats influences embedding efficiency and detectability. StegApp builds upon these insights, supporting multi-format steganography and paving the way for future enhancements like video-based data hiding.

1. Efficient Cryptographic Techniques for Secure Data Transmission (2020) Kumar and Singh

   Kumar and Singh analyzed AES and Blowfish encryption algorithms, providing critical insights into hybrid encryption methods, influencing StegApp's security model. Their comparative study on encryption efficiency and security provided a strong basis for selecting an encryption algorithm suitable for a steganographic application.

   - **AES for strong, standardized encryption:** AES provides robust encryption with fixed key lengths (128, 192, or 256 bits), ensuring strong resistance to brute-force attacks. This standardization has made AES one of the most widely used encryption techniques in secure communications.
   - **Blowfish for faster encryption with variable-length keys:** Blowfish is efficient for high-speed encryption, offering flexibility in security levels with key lengths between 32 to 448 bits. It is particularly useful in applications requiring rapid processing without significant computational overhead.
   - **Comparative analysis of encryption efficiency:** Their study highlighted that AES excels in security-intensive applications, whereas Blowfish offers faster encryption speeds, leading to StegApp's adoption of a hybrid approach. By leveraging both algorithms, StegApp ensures a balance between security and performance, making it an optimal choice for real-time steganographic applications.

2.2 GPU-Accelerated Steganography: Enhancing Performance (2021) Lee et al.

Lee et al. investigated the application of GPU acceleration for steganographic operations, demonstrating significant efficiency improvements. Their work emphasized the importance of parallel computing in reducing processing time for complex data embedding and extraction tasks.

- **Parallel processing for faster data embedding:** Utilizing GPU-based parallel computing speeds up steganographic operations by processing multiple pixels simultaneously. This dramatically reduces latency, making large-scale steganography feasible in real-time applications.
- **CUDA optimization techniques:** CUDA optimizations such as memory coalescing and shared memory usage significantly improve execution time. By efficiently managing memory transactions and leveraging GPU architecture, StegApp maximizes performance without sacrificing quality.
- **Handling large-scale media efficiently:** GPU acceleration facilitates the processing of high-resolution images and large audio files, making StegApp highly scalable. The research also suggested techniques to minimize computational bottlenecks, ensuring seamless performance across different hardware configurations.

2. **"Steganography and Cryptography: Hybrid Approaches" (2022) Tan and Wei**

   Tan and Wei examined hybrid techniques combining steganography and cryptography to enhance data security. Their findings underscored the importance of integrating multiple security mechanisms to counteract emerging threats in digital communication.

   - **Enhanced resistance to steganalysis and brute-force attacks:** The study demonstrated that combining encryption with steganography strengthens security by obfuscating hidden data. Encrypted payloads ensure that even if steganalysis detects the presence of hidden data, extracting meaningful information remains infeasible without the decryption key.
   - **Integration of encryption and steganography:** The research emphasized optimized encryption-steganography workflows, inspiring StegApp's efficient encryption pipeline. By implementing encryption be-

fore embedding, StegApp ensures that its hidden data is resistant to both direct and indirect attacks.

- **Practical use cases in secure communication:** Hybrid methods are vital for confidential messaging, file transfers, and covert data transmission, aligning with StegApp's intended applications. The study provided examples of real-world implementations, demonstrating the effectiveness of combining encryption and steganography for secure communication.

2.3 Detection and Countermeasures of Steganographic Attacks (2023) Zhang et al.

Zhang et al. explored detection techniques used in steganalysis and proposed countermeasures to strengthen security. Their research outlined the increasing sophistication of steganalytic methods, prompting the need for more advanced countermeasures.

- **Statistical analysis of media files:** Techniques such as histogram analysis can reveal steganographic modifications, necessitating adaptive embedding strategies. StegApp employs methods like random bit flipping and noise addition to mitigate detection risks.
- **AI-based steganalysis:** Advanced deep learning models detect hidden data with high accuracy, pushing StegApp to integrate randomized embedding strategies and noise simulation to counteract detection. Zhang et al. demonstrated that convolutional neural networks (CNNs) can effectively recognize steganographic patterns, necessitating continuous evolution in hiding techniques.
- **Countermeasures for secure steganography:** Zhang et al. proposed encryption, dynamic embedding, and payload dispersal as essential countermeasures, all of which are implemented in StegApp. The research also recommended utilizing multi-layer security, a strategy StegApp incorporates by embedding encrypted segments across multiple areas of an image or audio file to avoid clustering that may trigger detection.

Chapter 3

Finalization of Target Requirements

3.1 Core Functionalities and Features

StegApp is designed with a wide range of functionalities to enhance usability, security, and efficiency. The following sections elaborate on the key features of the application:

3.1.1 Data Hiding and Extraction

The primary function of StegApp is to embed and retrieve data within carrier images or audio files. The application employs LSB-based (Least Significant Bit) steganography, ensuring minimal perceptual distortion in the carrier media. By optimizing bit-level embedding techniques, StegApp maximizes payload capacity while maintaining the integrity and quality of the carrier file.

StegApp offers multi-format support, allowing users to work with different file types such as PNG, BMP, and WAV. The embedding process ensures that the modifications in the media file remain imperceptible, making it highly resistant to visual detection. Furthermore, the extraction mechanism is designed to retrieve hidden data efficiently without altering the original carrier file.

To improve resilience against detection, StegApp incorporates randomized embedding techniques, making statistical analysis more challenging for adversaries. Additionally, it offers adaptive payload management, adjusting embedding density based on image noise levels to reduce detectability.

3.1.2 Hybrid Encryption

To enhance security, StegApp employs a hybrid encryption model that combines AES and Blowfish encryption algorithms. AES is a standardized encryption algorithm offering fixed key lengths of 128, 192, or 256 bits, ensuring strong security. Blowfish, known for its efficiency, provides variable key lengths and fast encryption, making it ideal for handling large amounts of data.

This hybrid approach ensures that even if steganographic content is detected, it remains unreadable without the correct decryption key. By encrypting data before embedding, StegApp adds an additional layer of protection, preventing unauthorized access even in the case of steganographic detection.

To further strengthen security, StegApp implements salting and key derivation functions (KDFs) to enhance password-based encryption. The use of PBKDF2 and bcrypt ensures that brute-force attacks remain computationally expensive, adding a significant security advantage.

3.1.3 GPU Acceleration

A key feature of StegApp is CUDA-based GPU acceleration, which significantly speeds up the embedding and extraction processes. Unlike CPU-based methods that process image pixels or audio samples sequentially, GPU acceleration enables parallel execution of multiple operations.

For large-scale applications, GPU-based processing provides significant performance improvements. The integration of CUDA technology allows the application to handle high-resolution images and large audio files efficiently. Users can experience a substantial reduction in processing time, making the tool highly scalable and practical for real-world usage.

Additionally, the use of OpenCL support extends compatibility across different GPU architectures, making StegApp usable on a broader range of hardware configurations.

3.1.4 Error Handling and Validation

To ensure reliability, StegApp incorporates advanced error-handling mechanisms that prevent corruption of embedded data. Validation checks are performed at multiple stages, including carrier file integrity verification to prevent embedding in corrupted files, encryption key compatibility checks to ensure proper encryption and decryption, and capacity management to prevent data overflow and avoid carrier distortion. The system also includes embedded data validation to detect potential corruption during retrieval. These validation steps ensure robust error detection and

provide meaningful feedback to users, minimizing the risk of data loss or embedding failures. Additionally, logging mechanisms are implemented to track errors and provide debugging insights.

3.1.5 User Interface

StegApp features a modern graphical user interface built using CustomTkinter, designed for both functionality and ease of use. It includes real-time status updates on embedding and extraction processes, progress tracking to monitor long-running tasks, and dynamic input fields for selecting carrier files, encryption keys, and data files. The interface is responsive, ensuring seamless usability across different screen resolutions. . Additional features such as dark mode and customizable themes further enhance the user experience.

3.2 Implementation Details and Optimizations

3.2.1 Adaptive Image Resizing

To accommodate different data sizes while maintaining image quality, StegApp dynamically adjusts carrier image dimensions when necessary. This prevents embedding failures due to insufficient space while ensuring minimal visual distortion. The application automatically analyzes the available storage capacity within the carrier file and determines whether resizing or an alternative carrier is needed.

3.2.2 Multi-Threading for Parallel Processing

In addition to GPU acceleration, StegApp employs multi-threading to distribute processing tasks across multiple CPU cores. This improves efficiency during encryption, file I/O operations, and large-scale data embedding while keeping the application responsive. By using parallel execution techniques, StegApp reduces processing delays and ensures real-time performance improvements.

3.2.3 Secure Key Management with SHA-256

StegApp processes encryption keys using SHA-256 hashing, ensuring enhanced security and preventing weak keys from compromising data protection. The use of SHA-256 reduces the likelihood of brute-force decryption attempts, reinforcing overall system security. Additionally, StegApp implements secure key derivation functions (KDFs) to strengthen encryption key management.

3.2.4 Optimized Memory Usage

By implementing memory-mapped file handling and buffered reading/writing operations, StegApp optimizes memory usage. This ensures efficient handling of large media files without excessive RAM consumption. The application dynamically allocates resources based on the file size, preventing memory overflows and improving performance.

3.2.5 Adaptive Steganographic Techniques

To counter steganalysis attacks, StegApp incorporates noise simulation and randomized bit embedding strategies. These methods enhance the stealthiness of hidden data, making it harder for AI-based or statistical analysis techniques to detect modifications.

Advanced techniques such as edge-based embedding and frequency-domain transformations are also under consideration to further strengthen resistance against steganalysis methods.

3.3 Applications and Future Scalability

StegApp is designed for a wide range of applications, including secure messaging and confidential communication, protection of intellectual property by embedding metadata within digital media, and covert transmission of sensitive information in high-security environments. It also facilitates secure storage of hidden data within images and audio files, enables digital watermarking for copyright protection, and supports secure forensic investigations where hidden data storage is necessary.

To ensure long-term usability, StegApp follows a modular design that allows for future scalability. Planned enhancements include support for video steganography by expanding the application to embed data within video files, leveraging motion vectors for increased storage capacity. AI-powered steganographic encoding will be introduced using machine learning techniques to dynamically optimize data hiding strategies, improving undetectability against modern analysis techniques. Additionally, cloud-based secure storage integration will enable encrypted steganographic file storage and retrieval, allowing secure data access from multiple devices.

Chapter 4

Design

StegApp System Architecture

4.1 Algorithm

Data Hiding in an Image Using Encryption

Input: Image I, Data D, Key K
Output: Stego-image I_{stego}
$K_{\text{derived}} \leftarrow$ SHA-256(K) {Derive the key from the master key K}
$D_{\text{enc}} \leftarrow$ AES(D, K_{derived}) {Encrypt data D using AES}
$D_{\text{enc}} \leftarrow$ Blowfish$(D_{\text{enc}}, K_{\text{derived}})$ {Encrypt the data using Blowfish}
$B \leftarrow$ Binary(D_{enc}) {Convert encrypted data to binary}
for each pixel P_i in image I **do**
  Extract the color components (R_i, G_i, B_i) of P_i
  Embed one bit of B into the least significant bit of each color component R_i, G_i, B_i
end for
Save the modified image as I_{stego}

4.2 Entities and Their Attributes

The entities in the ER diagram represent the major components involved in encryption, data hiding, and processing. Each entity has specific attributes and functions that contribute to the overall system workflow.

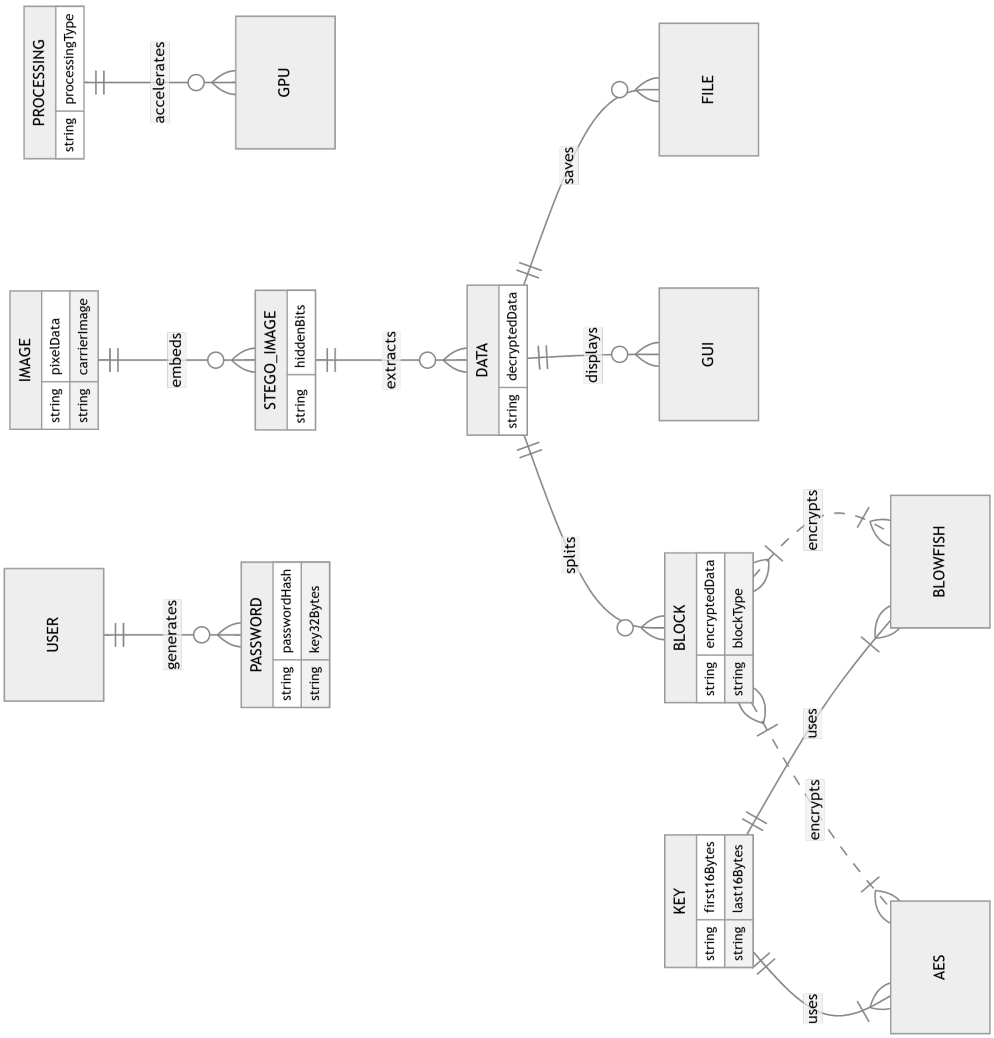

Figure 4.1: ER diagram

4.2.1 User

The user is the primary entity interacting with the system. They provide input in the form of a password, which serves as the foundation for encryption and security. The password is used to generate cryptographic keys, ensuring secure data protection and controlled access to hidden information.

4.2.2 Password

The password entity plays a crucial role in security by storing both the hashed password and a 32-byte cryptographic key. It has two primary attributes: the password hash, which is a securely hashed version of the user's input, and the 32-byte key, which is derived from the password and is essential for encryption and decryption processes.

4.2.3 Key

The key is derived from the user-provided password and is divided into two segments. The first 16 bytes of the key are allocated for AES encryption, while the last 16 bytes are reserved for Blowfish encryption. This structured approach ensures an effective hybrid encryption mechanism. The key entity is directly involved in the encryption process, enabling secure data transformation.

4.2.4 AES and Blowfish

AES and Blowfish are the two encryption algorithms utilized in the system. Each algorithm encrypts blocks of data using different portions of the key. AES operates on the first 16 bytes of the key, whereas Blowfish makes use of the last 16 bytes. This hybrid encryption model enhances security by combining the strengths of both algorithms, making it more resilient to cryptographic attacks.

4.2.5 Block

A block represents a segment of data that undergoes encryption. Each block has two fundamental attributes: encrypted data, which contains the transformed secure data, and block type, which specifies whether the block has been encrypted using AES or Blowfish. By organizing data into blocks, the system ensures structured and efficient encryption and decryption.

4.2.6 Image

The image entity acts as the carrier medium for steganographic embedding. It holds raw pixel data and serves as the host for encrypted information. The primary attributes of this entity include pixel data, which contains the original image content, and carrier image, which stores the modified version of the image with embedded hidden data. This process ensures that the encryption remains imperceptible to external observers.

4.2.7 Stego Image

The stego image is the modified version of the carrier image, containing encrypted hidden data. It has a crucial attribute, hidden bits, which stores the embedded information. The stego image enables covert data transmission, ensuring secure communication and confidentiality. Additionally, the system provides mechanisms for extracting the hidden data when needed.

4.2.8 Data

The data entity represents the decrypted output retrieved from the stego image. It contains a key attribute, decrypted data, which stores the original message after successful decryption. The data entity plays a crucial role in splitting encrypted data into blocks, displaying the decrypted information in the graphical user interface, and saving the recovered message into a file for later use.

4.2.9 Graphical User Interface (GUI)

The GUI provides an interactive platform for users to interact with the system. It allows users to view decrypted data, select files for embedding, and monitor the encryption and extraction processes. Designed with accessibility in mind, the interface ensures ease of use even for those unfamiliar with cryptographic and steganographic techniques.

4.2.10 File

The file entity is responsible for storing decrypted data. Once the hidden data is extracted and decrypted, users can choose to save it into a file for future reference. This ensures persistence and easy retrieval of information.

4.2.11 Processing

Processing represents different computational modes used in encryption and steganography. The processing entity has an important attribute, processing type, which determines whether the system is utilizing GPU acceleration or CPU-based execution. If a compatible GPU is detected, processing is optimized to enhance performance and reduce computational overhead.

4.2.12 GPU Acceleration

The GPU entity facilitates accelerated encryption and steganographic embedding. When available, GPU acceleration significantly improves processing speed by parallelizing encryption and data embedding tasks. If no GPU is detected, the system defaults to CPU execution while maintaining security and efficiency.

4.3 Relationships and Workflow

The relationships between these entities define the flow of data through the system. Below is a detailed breakdown of how different components interact.

4.3.1 Password Generation

The user generates a password, which undergoes hashing to enhance security. This hashed password is then used to derive a cryptographic key. The key serves as the foundation for encryption, ensuring that only authorized users can decrypt the hidden information.

4.3.2 Encryption Process

Once the key is derived, it is split into two segments. The first 16 bytes are utilized for AES encryption, while the last 16 bytes are allocated for Blowfish encryption. The data is then divided into blocks, with AES encrypting odd-numbered blocks and Blowfish encrypting even-numbered blocks. These encrypted blocks are combined to form a secure dataset, ensuring robust data protection.

4.3.3 Steganographic Embedding

The encrypted data is embedded into an image using Least Significant Bit (LSB) steganography. This method ensures that the modifications to the image are

imperceptible to the human eye. The resulting image, known as the stego image, acts as the carrier for hidden data, enabling secure covert communication.

4.3.4 Data Extraction and Decryption

When the hidden information needs to be retrieved, the system extracts the embedded bits from the stego image. These bits are then reassembled into encrypted blocks, which undergo decryption using the hybrid AES-Blowfish model. The decrypted data is then made available for further use.

4.3.5 Displaying and Saving Output

The decrypted data is presented in the graphical user interface, providing users with a visual representation of the retrieved information. Additionally, users can choose to save the decrypted data into a file for future reference, ensuring secure storage.

4.4 Security Considerations

StegApp employs multiple security measures to ensure data integrity and confidentiality. The hybrid encryption model enhances security by utilizing both AES and Blowfish, making decryption more complex for unauthorized users. The LSB steganography technique ensures that the hidden data remains imperceptible, preventing detection. Furthermore, GPU acceleration speeds up encryption and embedding without compromising security. To maintain data integrity, the system performs validation checks after decryption to ensure that the extracted information remains unaltered.

4.5 Potential Enhancements

Future enhancements to StegApp include implementing AI-based steganalysis resistance techniques to improve detection avoidance. The integration of post-quantum cryptographic algorithms will further enhance security against potential future threats posed by quantum computing. Additionally, multi-factor authentication will be introduced to strengthen access control, ensuring that only authorized users can retrieve hidden information.

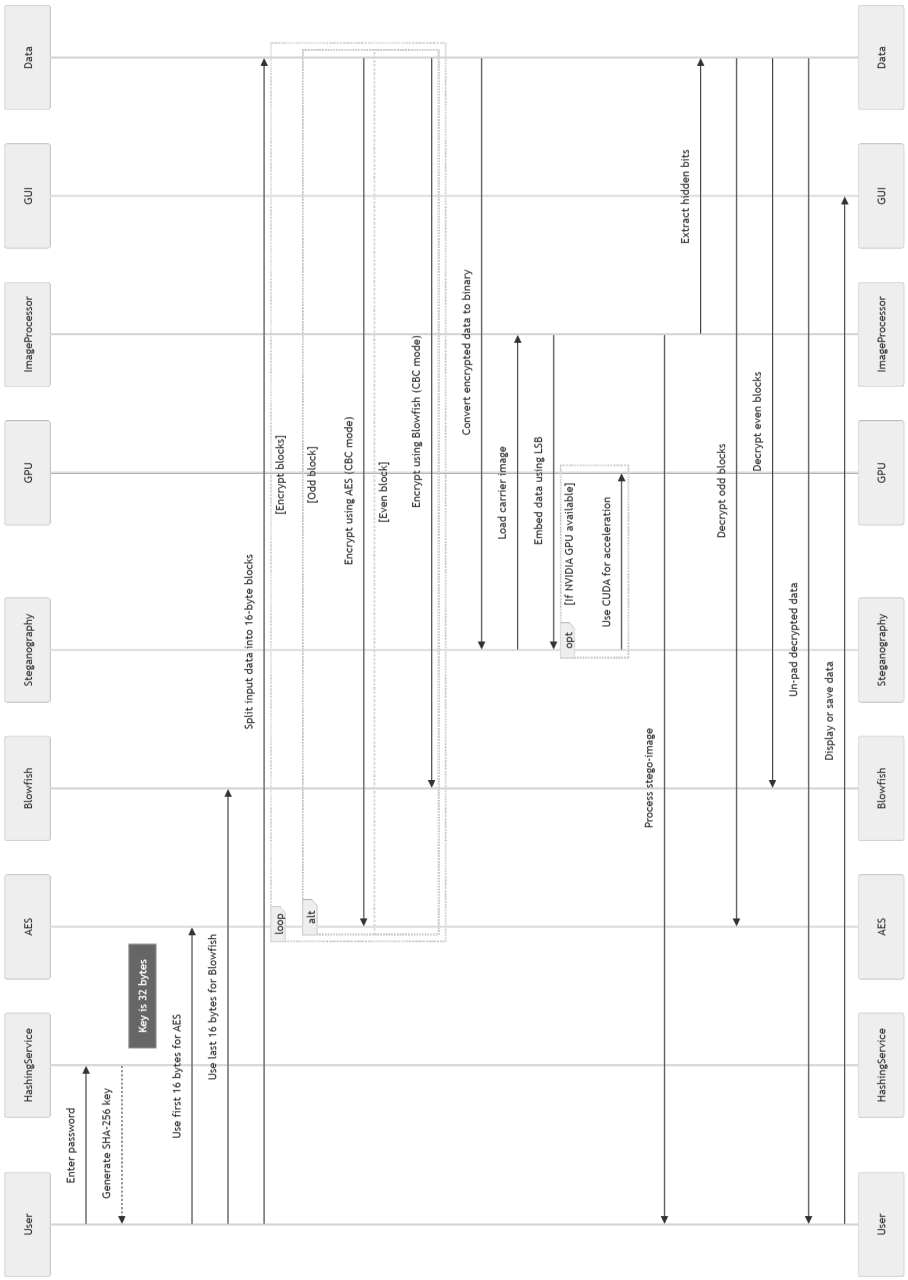

Figure 4.2: Sequence Diagram

4.6 System Components and Workflow

The sequence diagram divides the process into multiple stages, each of which is essential for ensuring robust security and performance.

4.6.1 User Input and Key Generation

The process begins when the user enters a password, which plays a crucial role in securing hidden data. The password is then sent to the hashing service, where it undergoes a cryptographic hash function using the SHA-256 algorithm. This function generates a secure, one-way hash value that cannot be reversed, producing a 256-bit (32-byte) output. The resulting hash serves as a derived key for encryption. To facilitate a hybrid encryption approach, the 32-byte hash is divided into two segments: the first 16 bytes are designated for AES encryption, while the last 16 bytes are allocated for Blowfish encryption. This division enables the system to leverage the strengths of both encryption algorithms, enhancing security and performance.

4.6.2 Data Encryption Process

Once the encryption key is generated, the system proceeds with encrypting the input data to ensure its security before embedding it into an image. The encryption process is structured to optimize both security and efficiency by implementing a hybrid encryption approach that utilizes Advanced Encryption Standard (AES) and Blowfish.

Before encryption begins, the input data is divided into fixed-size 16-byte blocks, allowing for streamlined processing and compatibility with block cipher encryption methods. This ensures that even large datasets can be efficiently encrypted while maintaining strong security guarantees. The hybrid encryption strategy alternates between two cryptographic algorithms: odd-numbered blocks are encrypted using AES in Cipher Block Chaining (CBC) mode, while even-numbered blocks undergo Blowfish encryption in CBC mode. This alternating encryption method enhances resistance to cryptanalysis while balancing security and performance.

AES is a symmetric-key encryption algorithm known for its high security and resistance to brute-force attacks. It follows a substitution-permutation network (SPN) structure, applying multiple rounds of transformation, including substitution, permutation, and key mixing. In this system, AES operates in CBC mode, which utilizes an initialization vector (IV) to prevent identical plaintext blocks from producing identical ciphertexts. The key for AES encryption is derived

from the first half of the hashed user password, ensuring strong cryptographic security.
Blowfish is another symmetric-key block cipher valued for its speed and efficiency, particularly in low-memory environments. Unlike AES, it uses a Feistel network structure, dividing each data block into two halves and applying key-dependent transformations over multiple rounds. Blowfish is also used in CBC mode to enhance security, ensuring distinct ciphertexts for identical plaintext inputs. The encryption key for Blowfish is derived from the second half of the hashed password, maintaining a balanced distribution of cryptographic strength. The combination of AES and Blowfish in a hybrid encryption approach provides a strong balance between security and computational efficiency. Using two encryption methods increases resistance to attacks, as an adversary would need to break both encryption schemes. This approach also ensures computational efficiency by leveraging the speed of Blowfish to compensate for the higher computational demands of AES.
By integrating AES and Blowfish encryption in a structured manner, the system guarantees a high level of data protection before embedding it into a steganographic carrier. This hybrid approach enhances confidentiality while optimizing processing speed for real-world applications.

4.6.3 Steganographic Embedding of Encrypted Data

Once encryption is complete, the encrypted data is concealed within an image using Least Significant Bit (LSB) steganography. The first step in this process involves converting the encrypted data into binary format. A carrier image is then loaded, serving as the medium for data embedding. LSB steganography is applied by modifying the least significant bits of pixel values to store encrypted bits. These alterations remain imperceptible to the human eye, ensuring that the presence of hidden data is not easily detectable. To further enhance security, the system employs advanced embedding techniques such as randomized pixel selection, which prevents predictable data distribution, and multi-layer embedding, which disperses data across different components of the image. These techniques increase the resilience of the steganographic method against detection by analysis tools.

4.6.4 GPU Acceleration Using CUDA

To optimize performance, the system checks for the presence of an NVIDIA CUDA-compatible GPU. If such a GPU is detected, CUDA acceleration is applied to both the encryption and steganographic embedding processes, signif-

icantly reducing processing time. In the absence of a compatible GPU, the system defaults to CPU-based execution. While this ensures broad compatibility across different hardware configurations, it may result in slower processing speeds. GPU acceleration is particularly advantageous for handling large files and high-resolution images, making the system more efficient for users dealing with substantial data volumes.

4.6.5 Extracting Hidden Data and Decryption

Data retrieval begins with the processing of the stego image to extract hidden bits. The extracted bits are then reconstructed into binary encrypted data, which is subsequently divided into 16-byte blocks. Decryption follows a reverse hybrid approach, in which odd-numbered blocks are decrypted using AES, while even-numbered blocks undergo Blowfish decryption. After decryption, the data is unpadded to restore its original format, ensuring that the retrieved content matches the initial input data before encryption.

4.6.6 Final Output and User Interaction

Upon successful decryption, the decrypted data is displayed in the graphical user interface (GUI). Users are given the option to view the output directly or save it as a file for future use. This flexibility enhances usability, allowing individuals to choose the most suitable method for accessing their recovered data.

4.7 Security Enhancements

To further reinforce security, the system integrates additional protective measures. Hash-based integrity checks verify that the decrypted data matches the original input, ensuring data consistency. Error correction codes (ECC) are implemented to detect and rectify minor errors in extracted data, enhancing reliability. Additionally, advanced obfuscation techniques are used to prevent detection by steganalysis tools, further increasing the system's resistance to external threats.

4.8 Performance Considerations

Several factors influence the overall performance of the system. The complexity of the encryption process is carefully balanced to ensure both security and efficiency. The size and resolution of the carrier image also play a crucial role, as larger images provide greater data embedding capacity but require more processing time.

4.9 Potential Future Improvements

To remain at the forefront of secure steganographic applications, future versions of StegApp may introduce several significant enhancements. As cybersecurity threats continue to evolve, it is crucial to adopt forward-looking security measures that ensure resilience against emerging attack vectors. One of the most promising directions for improvement is the implementation of quantum-safe encryption techniques. With advancements in quantum computing, traditional cryptographic algorithms may become vulnerable to quantum attacks, potentially compromising encrypted data. By integrating post-quantum cryptographic algorithms, StegApp can ensure long-term security against quantum adversaries, making it future-proof against this technological shift.

Another area of enhancement is the development of AI-based steganalysis resistance mechanisms. Steganalysis techniques are continuously evolving, with artificial intelligence playing a major role in detecting hidden data within digital media. To counteract this, machine learning models can be leveraged to dynamically adjust data embedding strategies, making steganographic concealment more adaptive and undetectable. AI-driven obfuscation techniques can further enhance the system's ability to evade detection from automated steganalysis tools.

Furthermore, cloud integration can be introduced to allow for secure remote storage and retrieval of encrypted steganographic files. With the increasing adoption of cloud-based solutions, users can benefit from seamless access to their protected data across multiple devices while ensuring end-to-end encryption during transmission and storage. Implementing decentralized encryption mechanisms within cloud storage could provide an additional layer of security, ensuring that even cloud service providers cannot access the hidden data.

Additionally, StegApp can incorporate adaptive encryption algorithms that dynamically select optimal encryption techniques based on the sensitivity of the data being processed. Different data types and user requirements may necessitate varying levels of security and computational efficiency. By utilizing adaptive cryptographic frameworks, the system can intelligently determine the most suitable encryption method, balancing security strength and processing performance. This approach would ensure that highly sensitive information receives the strongest encryption, while less critical data can be processed with lighter encryption techniques to optimize efficiency.

By integrating these advancements, StegApp aims to enhance its security, efficiency, and usability, positioning itself as a cutting-edge steganographic solution capable of withstanding evolving cybersecurity threats.

StegApp Encryption and Steganography Workflow

The diagram illustrates the encryption, steganographic embedding, GPU acceleration, and decryption processes in StegApp. It ensures secure and efficient data concealment by integrating hybrid encryption, CUDA-based acceleration, and Least Significant Bit (LSB) steganography. The workflow is divided into three primary stages: key generation and encryption, data embedding, and decryption with data extraction.

The encryption process begins with secure key generation and management. The user enters a password, which undergoes a cryptographic transformation to enhance security. The password is first hashed using SHA-256, a cryptographic hash function that generates a fixed-length 256-bit hash, ensuring that the original password remains secure. This hash is then used to derive a 32-byte encryption key, which is split into two parts: the first 16 bytes are designated for AES (Advanced Encryption Standard) encryption, while the last 16 bytes are allocated for Blowfish encryption. This dual-key approach allows for a hybrid encryption mechanism, leveraging the strengths of both algorithms. The input data intended for concealment is divided into 16-byte blocks to facilitate efficient encryption and decryption. Odd-numbered blocks are encrypted using AES in Cipher Block Chaining (CBC) mode, while even-numbered blocks undergo Blowfish encryption in the same mode. AES is widely recognized for its high level of security, utilizing a complex substitution-permutation network structure to resist brute-force attacks. Blowfish, in contrast, is valued for its high-speed performance and efficiency in low-memory environments. By integrating these two encryption techniques, the system balances strong security with computational efficiency.

Following encryption, the encrypted data undergoes a steganographic embedding process to ensure covert and imperceptible storage within a carrier image. The encrypted data is first converted into binary format, allowing it to be mapped onto the least significant bits of image pixels. A carrier image is then loaded, serving as the medium for hidden data storage. Using LSB steganography, the encrypted bits are embedded by modifying the least significant bit of selected pixel values. This subtle alteration ensures that the carrier image remains visually unchanged, making it extremely difficult for adversaries to detect the presence of hidden information. To further enhance the security and efficiency of the embedding process, the system incorporates advanced techniques such as randomized pixel selection and multi-layer embedding, distributing encrypted data across different image components.

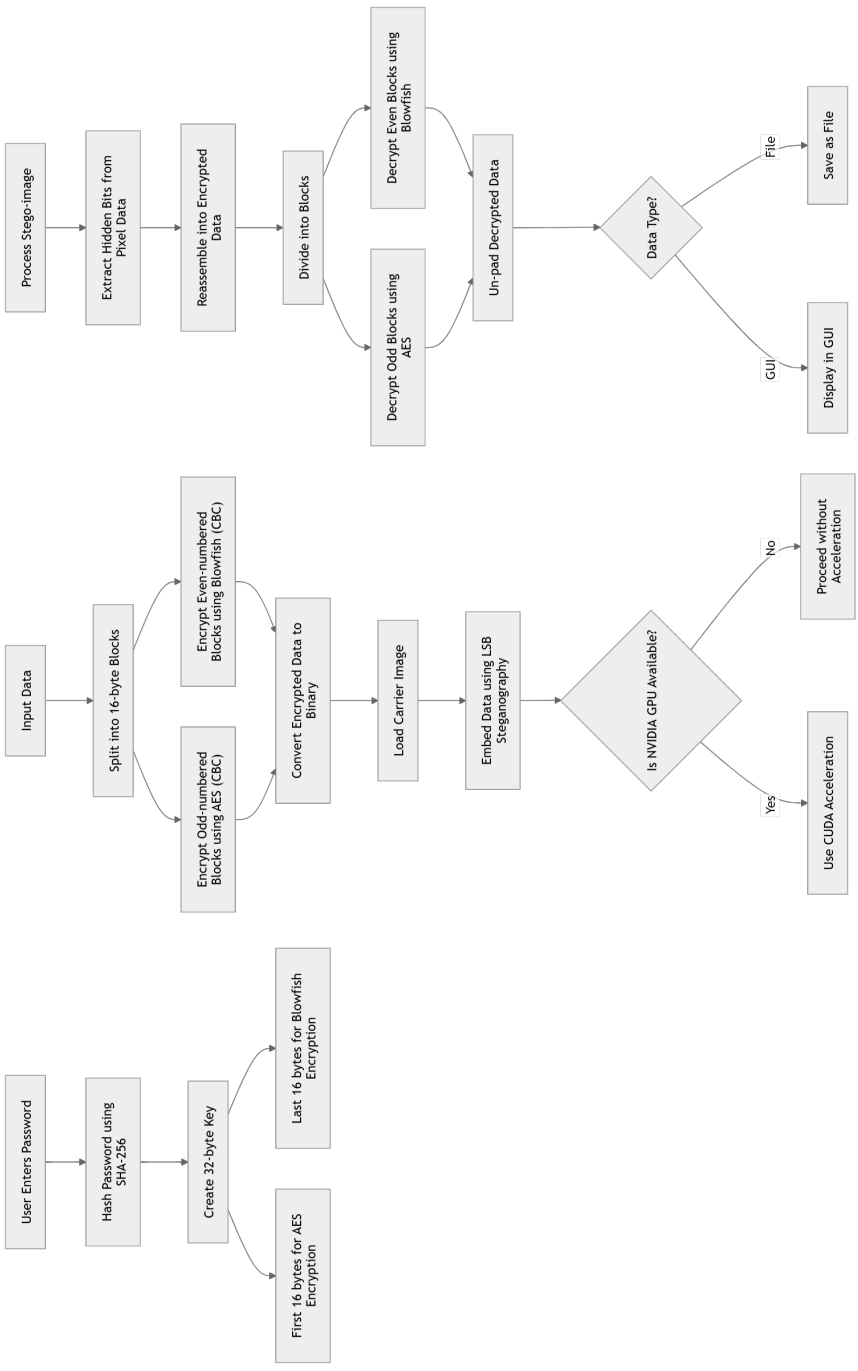

Figure 4.3: Flow Chart

To optimize computational performance, the system checks for the availability of a CUDA-enabled NVIDIA GPU. If a compatible GPU is detected, CUDA acceleration is applied to the encryption, embedding, and retrieval processes, significantly reducing processing time. GPU-based parallel processing offloads computational tasks from the CPU, enabling the system to handle large datasets and high-resolution images with minimal latency. In cases where no CUDA-compatible GPU is available, the system defaults to CPU-based operations, ensuring compatibility across different hardware configurations while maintaining functionality.

Once the encrypted data has been securely embedded within the carrier image, the decryption and extraction process allows for its recovery. The system first processes the stego-image to extract the hidden information by analyzing modified pixel values. The extracted bits are then reconstructed into their original binary encrypted form, reversing the embedding process. To ensure accurate decryption, the system divides the reconstructed encrypted data into 16-byte blocks, following the same structure used during encryption. Decryption occurs using the same hybrid approach: odd-numbered blocks are decrypted with AES, while even-numbered blocks are decrypted with Blowfish. Once the decryption process is complete, the system performs an un-padding operation to restore the data to its original form before encryption.

The final stage of the workflow involves presenting the decrypted data to the user. If the extracted data is intended for immediate access, it is displayed within the graphical user interface (GUI), allowing users to view or interact with the retrieved content. Alternatively, if the decrypted data is meant for later use, it can be saved as a file for secure storage. This ensures that users have flexible options for managing their confidential data while maintaining the highest standards of security and efficiency.

Chapter 5

System Implementation

5.1 Technologies Used

5.1.1 Software requirement

1.**Python:** Ensure you have Python 3.x installed. Python 3.8 or higher is recommended.

2.**Libraries and Dependencies:** The following Python libraries are required:

–customtkinter: For GUI (`pip install customtkinter`)

–pycryptodome: For encryption (`pip install pycryptodome`)

–Pillow: For image processing (`pip install pillow`)

–numpy: For numerical operations (`pip install numpy`)

–opencv-python: For advanced image processing (`pip install opencv-python`)

–soundfile: For audio processing (`pip install soundfile`)

–scipy: For signal processing (`pip install scipy`)

–concurrent.futures: For multithreading (part of Python's standard library)

–mmap, asyncio, aiofiles: For asynchronous file handling (`pip install aiofiles`)

–psutil: For system resource management (`pip install psutil`)

–wave: For handling audio files (standard library)

5.1.2 Hardware requirement

(a)**GPU Acceleration:** NVIDIA GPU with CUDA support is recommended for enhanced performance.

(b)**Memory (RAM):** Minimum 4GB RAM required (8GB or more recommended for better performance).

(c)**Processor:** Multi-core CPU recommended for efficient parallel processing.

(d)**Disk Space:** Sufficient disk space to store large media files and encrypted outputs.

5.1.3 Security requirements

(a)**Data Encryption:** All hidden data must be encrypted using secure cryptographic algorithms like AES (Advanced Encryption Standard) and Blowfish to prevent unauthorized access.

(b)**Secure Key Management:**

∗The encryption key must be strong (minimum 16 characters, including uppercase, lowercase, numbers, and special symbols).

∗The application should not store the encryption keys locally or in plain text.

(c)Data Validation:

∗Validate user inputs to prevent malicious data injection.

∗Ensure the integrity and authenticity of hidden data before extraction.

(d)**Secure Communication:** If the application supports remote access or data transfer, implement secure communication protocols (e.g., HTTPS, TLS) to prevent data interception.

(e)**Access Control:** Limit access to sensitive features, such as encryption and decryption, to authorized users only.

(f)**Error Handling:** Properly handle errors to avoid information leaks that could reveal encryption keys or sensitive data.

(g)**File Security:** Ensure that files used for hiding or extracting data are protected from unauthorized modifications.

5.1.4 API Integration

The API integration for the project relies on key libraries that enhance functionality and performance. The **PIL (Python Imaging Library)** is used extensively for image processing tasks, allowing seamless image manipulation, format conversion, and pixel-level data embedding necessary for steganographic applications. Additionally, **SoundFile and Wave** are crucial for handling audio data, enabling reading and writing of WAV files while ensuring high-fidelity data embedding in audio signals.

5.1.5 Development Environment

The development environment is structured to support efficient coding, debugging, and collaboration. **Python 3.x** serves as the primary programming language due to its extensive support for scientific computing and cryptographic libraries, making it ideal for implementing advanced steganographic and encryption techniques. **Visual Studio Code** is the chosen integrated development environment (IDE), providing powerful debugging tools, extension support, and seamless version control integration. To manage collaborative development and

track changes effectively, **GitHub** is used as the version control system, ensuring efficient code management, documentation, and teamwork throughout the project's lifecycle.

5.1.6 GPU Acceleration

To optimize performance for large-scale data embedding and extraction, the project leverages GPU acceleration. **CUDA and PyCUDA** enable efficient parallel computing, significantly speeding up the steganographic process. **Cupy** further enhances GPU-based computations, optimizing resource utilization for high-resolution media processing. These technologies allow the system to handle complex operations with reduced latency, ensuring scalability and responsiveness in steganographic applications.

5.1.7 API Integration

The API integration for the project relies on key libraries that enhance functionality and performance. The **PIL (Python Imaging Library)** is used extensively for image processing tasks, allowing seamless image manipulation, format conversion, and pixel-level data embedding necessary for steganographic applications. Additionally, **SoundFile and Wave** are crucial for handling audio data, enabling reading and writing of WAV files while ensuring high-fidelity data embedding in audio signals.

5.1.8 Development Environment

The development environment is structured to support efficient coding, debugging, and collaboration. **Python 3.x** serves as the primary programming language due to its extensive support for scientific computing and cryptographic libraries, making it ideal for implementing advanced steganographic and encryption techniques. **Visual Studio Code** is the chosen integrated development environment (IDE), providing powerful debugging tools, extension support, and seamless version control integration. To manage collaborative development and

track changes effectively, **GitHub** is used as the version control system, ensuring efficient code management, documentation, and teamwork throughout the project's lifecycle.

WHY AES+ BLOWFISH?

Feature	AES	Blowfish	AES + Blowfish
Security	High	High	Enhanced
Speed	Moderate	Fast	Optimized
Key Size	128, 192, 256-bit	32-448-bit	Variable
Complexity	High	Moderate	Balanced

Table 5.1: Comparison of AES, Blowfish, and Hybrid AES + Blowfish

STEGAPP VS OTHER APPLICATIONS

Feature	STEGAPP	Other Steganography Applications
Security Level	High	Medium / Low
Steganography Technique	Advanced	Basic / Moderate
Encryption Support	Yes	No / Yes
Speed	Fast	Moderate / Slow
User-Friendliness	High	Low / Medium

Table 5.2: Comparison between STEGAPP and other steganography applications

Steganographic Analysis Results

SL NO	Image/Audio	MSE	PSNR	Remarks
1.	Original audio-extracted audio	0		MSE low - good steganography Small change (not noticeable)
2.	Original image-stegano image	9.44	38.38 dB	MSE low - good steganography Small change (not noticeable)
3.	Original image-enlarged image	8.28	38.95 dB	MSE low - good steganography Small change (not noticeable)

Table 5.3: MSE and PSNR Comparison for Steganographic Analysis

Chapter 6

Results and Discussions

The StegApp project successfully integrates encryption, steganography, and GPU acceleration to provide a secure and efficient data-hiding solution. It enables hybrid encryption using AES and Blowfish, ensuring strong data protection. The system utilizes Least Significant Bit (LSB) steganography for secure embedding, preserving the carrier image's visual integrity while concealing encrypted data.

Users benefit from a seamless workflow that includes password-based authentication, encrypted data embedding, and efficient extraction with decryption. The CUDA-based GPU acceleration significantly enhances performance, reducing processing time for encryption, embedding, and retrieval, making the system scalable for large data and high-resolution images.

The implementation ensures robust security through multiple layers of encryption while maintaining minimal distortion in stego-images. StegApp effectively resists statistical and visual steganalysis, reinforcing its reliability for secure communication. Additionally, users can retrieve hidden information securely and efficiently without compromising the integrity of the original image.

The system enhances privacy, usability, and protection against data interception and unauthorized access. Future improvements may include AI-based steganalysis resistance, adaptive encryption techniques, and integration with cloud storage for enhanced scalability and security.

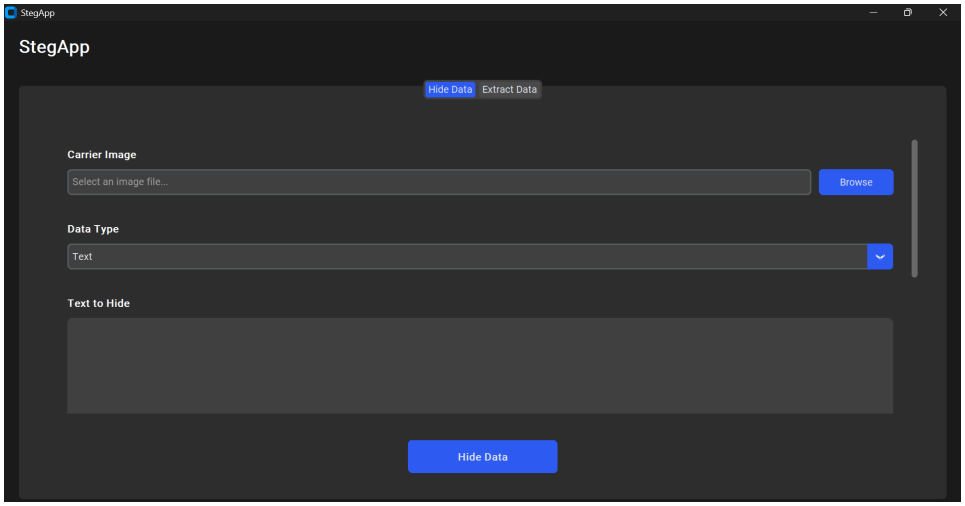

Figure 6.1: Front Page

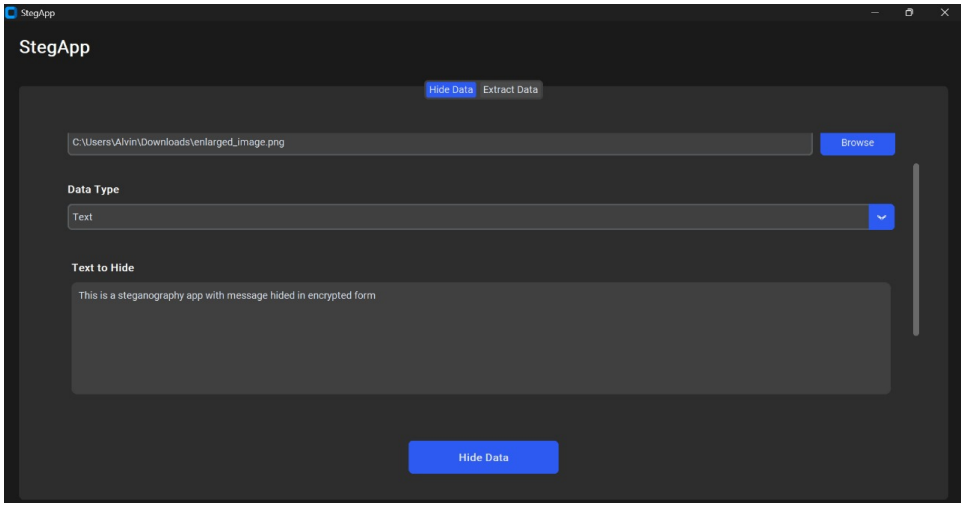

Figure 6.2: Text Hiding

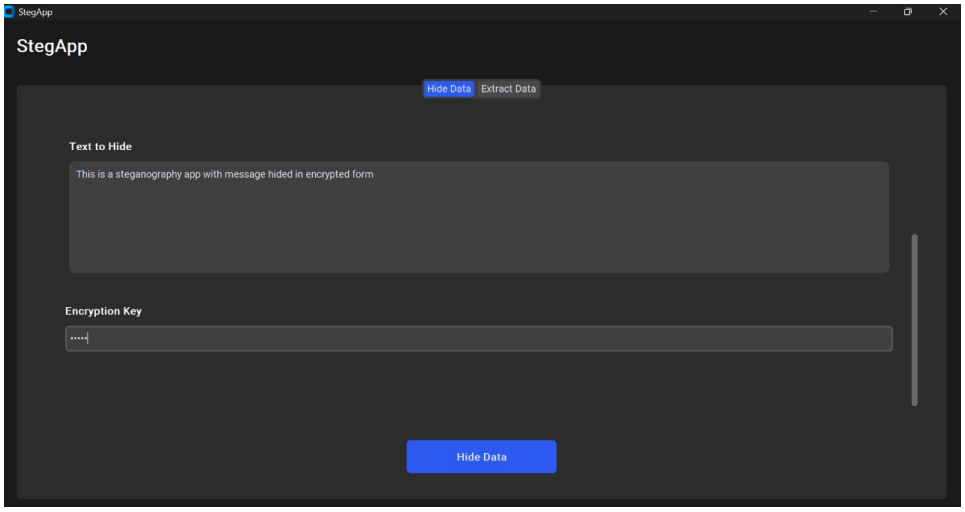

Figure 6.3: Text Entering

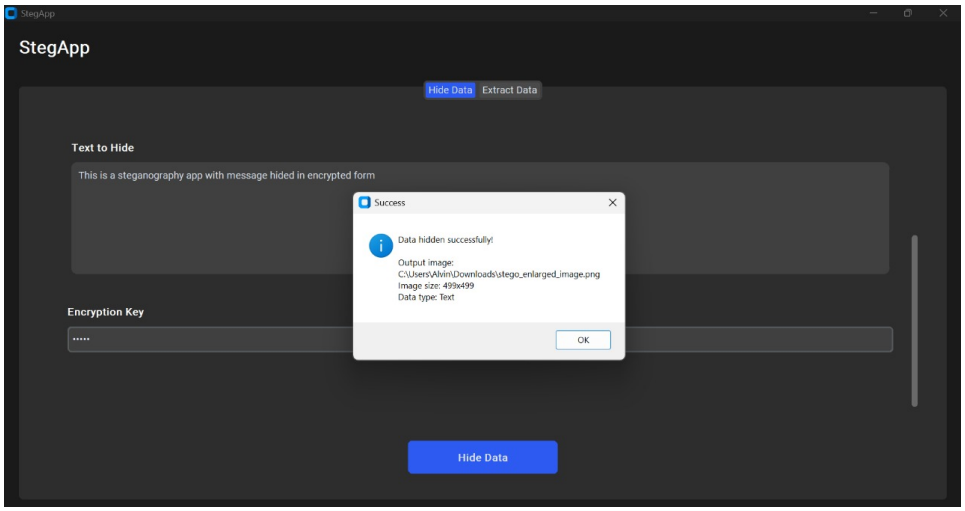

Figure 6.4: Data Hidden Successfully

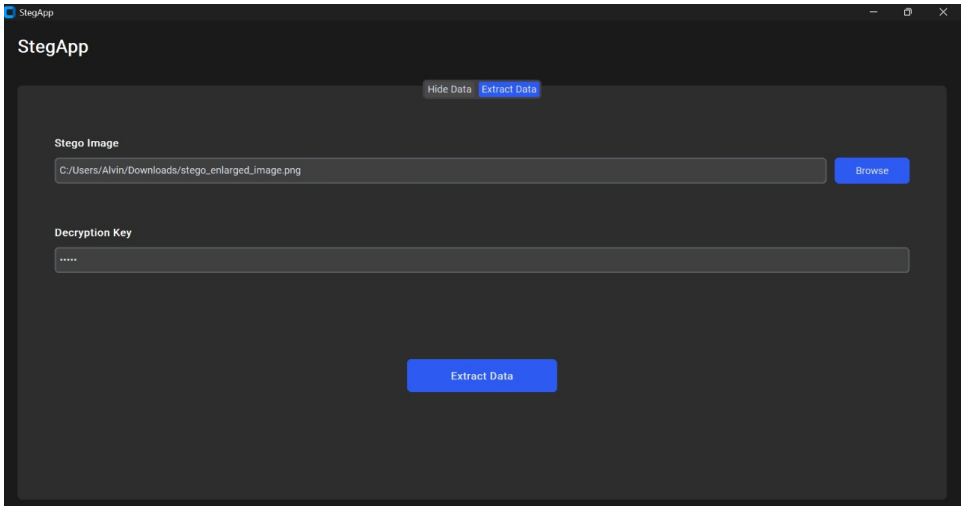

Figure 6.5: Data Extraction

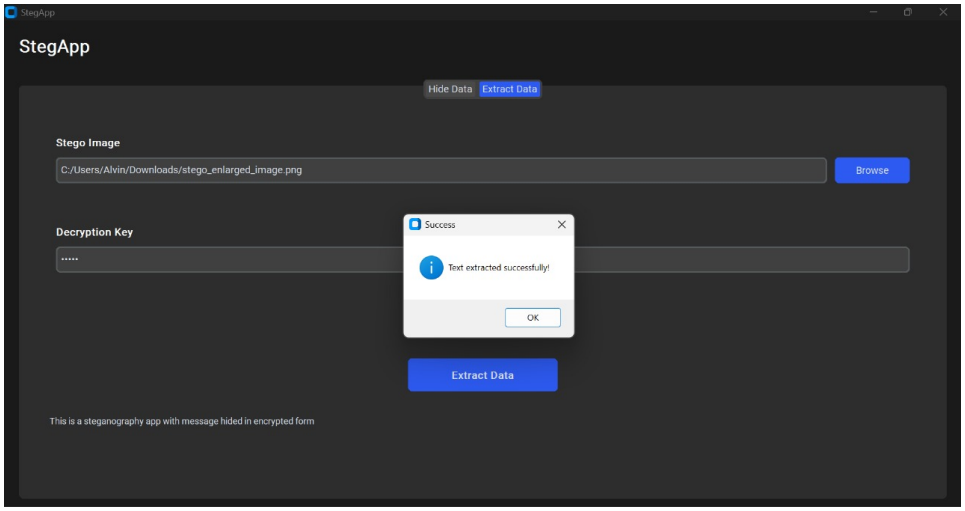

Figure 6.6: Extracted Successfully

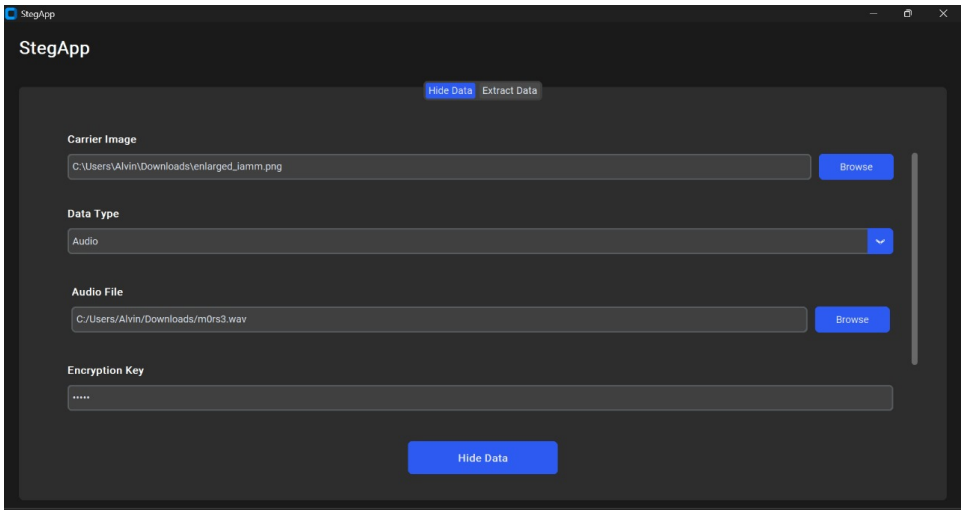

Figure 6.7: Audio Hiding

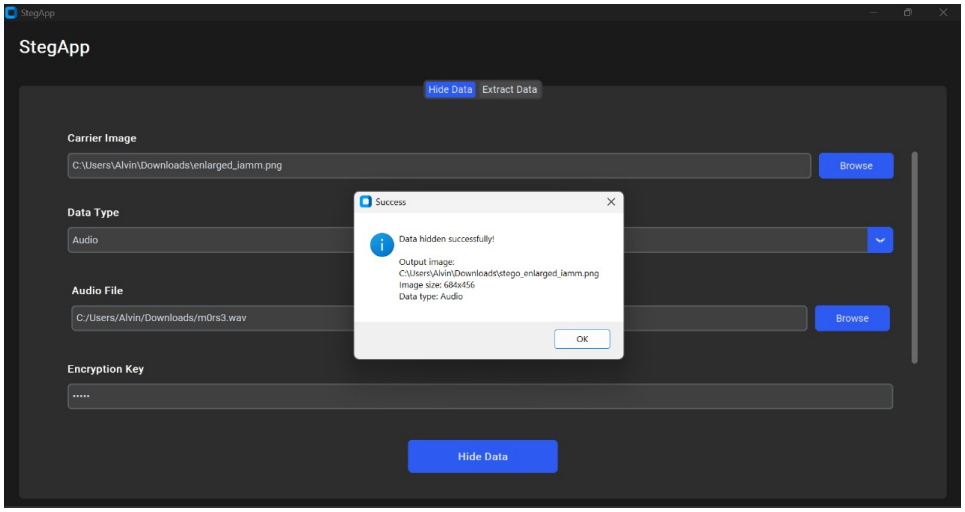

Figure 6.8: Data Hidden Successfully

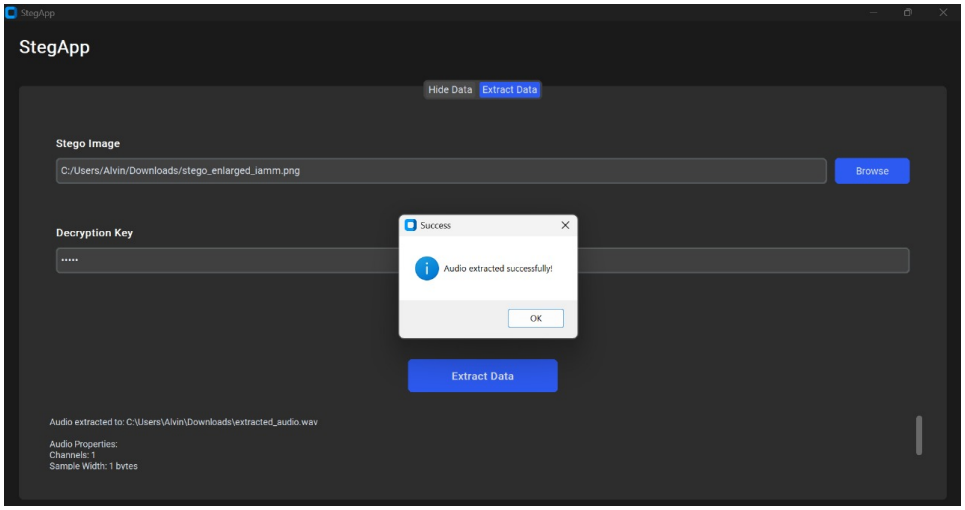

Figure 6.9: Audio Added Successfully

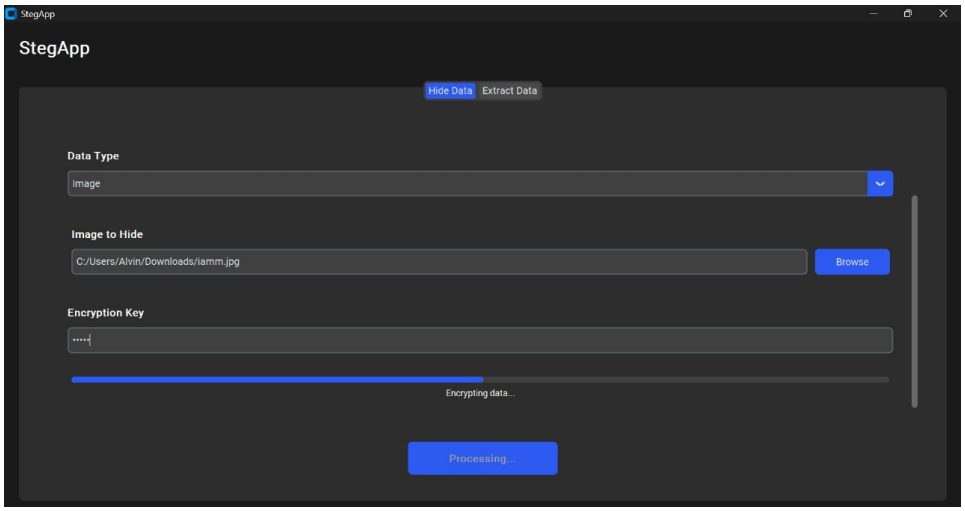

Figure 6.10: Image Encrypting

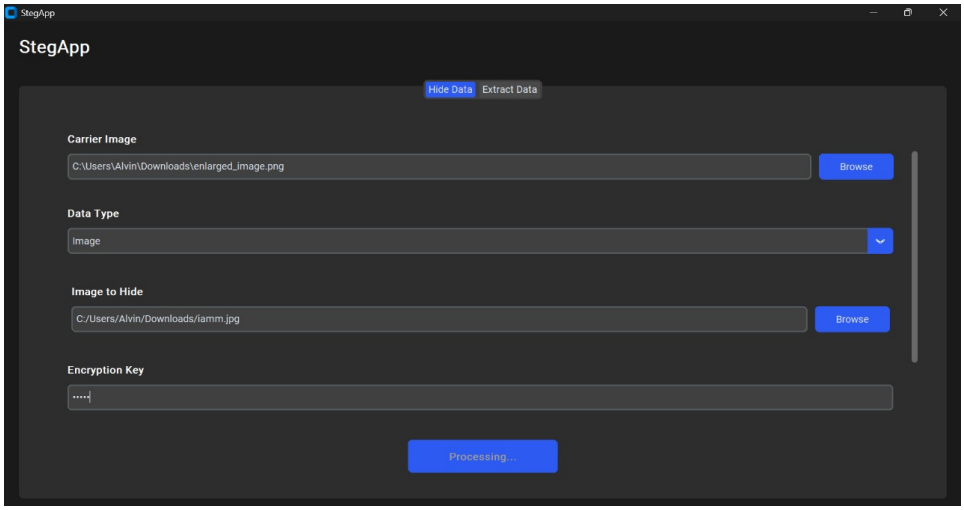

Figure 6.11: Image Hiding

Chapter 7

Conclusions and Future Scope for Improvement

7.1 Conclusions

The Mini Project has successfully achieved its primary objectives of providing a secure, efficient, and user-friendly steganography tool for data hiding and extraction. By integrating advanced encryption methods such as AES and Blowfish with robust steganographic techniques, the application ensures data confidentiality, integrity, and imperceptibility. The combination of cryptographic security and steganographic encoding makes the system highly resistant to unauthorized access and detection.

One of the key achievements of the Mini Project is the successful implementation of CUDA-based GPU acceleration, which significantly improves processing efficiency. By utilizing parallel computing, the system reduces processing time when embedding and extracting data from large media files, thereby enhancing overall performance. The project also features a modern, intuitive GUI built using CustomTkinter, ensuring ease of use and accessibility for both technical and non-technical users. The GUI provides seamless interaction, allowing users to select files, configure encryption settings, and monitor real-time progress during data embedding and extraction.

The security of the Mini Project is reinforced through hybrid encryption, where AES (Advanced Encryption Standard) and Blowfish work in combination to protect hidden data against brute-force attacks and cryptographic vulnerabilities. This dual-layer encryption mechanism enhances the overall robustness of the system, ensuring that even if one encryption algorithm is compromised, the second layer provides an additional barrier to unauthorized decryption.

Another major strength of the Mini Project lies in its use of Least Significant Bit (LSB) steganography, which ensures that embedded data remains undetectable to conventional steganalysis techniques. The system employs adaptive embedding techniques, which dynamically select the most suitable regions within media files to minimize distortions and statistical anomalies. This ensures that the hidden information is not easily detectable, even when analyzed using advanced forensic tools.

Additionally, robust error-handling mechanisms have been incorporated to maintain data accuracy and integrity throughout the embedding and extraction processes. The system performs integrity checks to verify that the retrieved data matches the original embedded content, ensuring reliability in real-world applications.

7.2 Future Scope for Improvement

While the Mini Project has successfully met its initial objectives, there are several potential areas for further enhancement and expansion. Future improvements can focus on extending the system's capabilities, optimizing its performance, and integrating cutting-edge security techniques to adapt to emerging cybersecurity challenges.

7.2.1 Expansion of Supported Media Types

Currently, the Mini Project supports data embedding in images and audio files. Future developments can include:

- Video Steganography: By embedding data within video frames or motion vectors, users can hide larger payloads without compromising video quality. - Text Steganography: Implementing methods such

as font-based encoding, whitespace manipulation, or character substitution can enable secure text-based communication. - Document Steganography: Embedding hidden messages within PDF, DOCX, or other document formats can provide additional security layers for confidential communications.

7.2.2 Advanced Cryptographic Enhancements

Future improvements can incorporate advanced encryption techniques to further strengthen data security:

- Hybrid RSA-AES Encryption: By combining asymmetric RSA encryption with symmetric AES encryption, users can enhance the security of key exchanges and data confidentiality. - Quantum-Resistant Cryptography: Implementing lattice-based encryption techniques can ensure long-term security against quantum computing attacks. - Homomorphic Encryption: Allowing computations to be performed on encrypted data without decrypting it first, enabling secure cloud-based data processing.

7.2.3 Performance Optimization and Scalability

To enhance performance and scalability, several optimizations can be considered:

- Enhanced GPU Utilization: Optimizing CUDA-based parallelism to handle even larger datasets efficiently. - Asynchronous Data Processing: Implementing asynchronous processing pipelines can improve responsiveness and reduce execution time. - Memory Optimization Techniques: Using memory-mapped I/O and efficient data compression to minimize resource consumption.

7.2.4 Cross-Platform Compatibility and Accessibility

To increase user reach, the Mini Project can be expanded to support multiple platforms:

- Cross-Platform Support: Developing compatibility for Windows, macOS, and Linux. - Mobile App Development: Creating an Android and iOS version to enable steganographic data hiding and retrieval on

mobile devices. - Web-Based Interface: Implementing a cloud-based web interface where users can securely upload files, apply encryption, and download steganographic media.

7.2.5 Integration of AI and Machine Learning

Artificial Intelligence and Machine Learning techniques can be leveraged to enhance steganographic efficiency and security:

- AI-Assisted Data Concealment: Using deep learning to identify optimal embedding locations for improved imperceptibility. - Automated Steganalysis Detection: Implementing machine learning models that can analyze media files for potential steganographic threats, enhancing cybersecurity applications. - Adaptive Steganography: Developing self-learning algorithms that adjust embedding patterns based on forensic analysis to counteract steganalysis techniques.

7.2.6 Secure Network-Based Communication

Another area of expansion includes integrating network-based steganography for real-time data exchange:

- Encrypted Peer-to-Peer Communication: Developing secure communication channels where users can exchange steganographic messages over encrypted networks. - Tor and VPN Integration: Enhancing anonymity by enabling steganographic messages to be sent through secure tunnels. - Steganographic Messaging Protocols: Implementing hidden communication protocols that leverage TCP/IP steganography to transmit concealed data within network packets.

7.2.7 Enterprise-Level Security Features

To make the Mini Project suitable for enterprise applications, advanced security features can be introduced:

- Multi-User Authentication: Implementing role-based access controls (RBAC) to restrict access to authorized users. - Audit Logging and Activity Monitoring: Keeping logs of all steganographic operations

for forensic and compliance purposes. - Blockchain for Key Management: Utilizing blockchain technology to securely store and track encryption keys in an immutable ledger.

7.2.8 Forensic and Anti-Forensic Enhancements

To improve resilience against forensic detection and steganalysis, additional techniques can be implemented:

- Multi-Layered Steganography: Spreading encrypted data across multiple carrier files to enhance security. - Steganographic Watermarking: Embedding unique identifiers to authenticate ownership and prevent unauthorized modifications. - Adaptive Noise Injection: Introducing controlled noise patterns to mask embedded data from forensic analysis.

7.2.9 Cloud-Based Storage and API Integration

Cloud-based storage and API integration can make the Mini Project more versatile and accessible:

- Secure Cloud Storage: Allowing users to store encrypted steganographic files in secure cloud environments. - RESTful API Development: Providing an API for third-party developers to integrate steganographic capabilities into their own applications. - Blockchain-Based Storage: Leveraging decentralized storage solutions to enhance data security and privacy.

7.2.10 Enhanced Usability and User Experience

Improvements to the user interface and user experience can make the Mini Project more intuitive:

- Automated Key Management: Introducing a built-in key vault with biometric authentication for secure key storage. - Interactive Visualization Tools: Enhancing the GUI with real-time graphical representations of encryption and embedding progress. - Context-Aware Help System: Implementing AI-driven assistance to provide real-time guidance on steganographic operations.

7.3 Final Thoughts

With a strong foundation in cryptographic security, steganographic techniques, and high-performance computing, the Mini Project demonstrates significant potential for real-world applications. The advancements outlined above can transform the system into a comprehensive, multi-functional steganography platform suitable for personal, enterprise, and forensic applications. By continually refining its encryption methods, optimizing computational performance, and integrating emerging technologies, the Mini Project can evolve into a state-of-the-art solution for secure digital communication and data protection.

Bibliography

[1]P. Sharma, *"A New Image Encryption using Modified AES Algorithm and its Comparison with AES," International Journal of Engineering Research & Technology (IJERT)*, vol. 9, issue 08, August 2020. [Online]. Available: http://www.ijert.org, IJERTV91S080083.

[2]S. Kumar and P. Singh, *"Efficient Cryptographic Techniques for Secure Data Transmission," IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1234-1245, 2020. doi: 10.1109/TIFS.2020.2975284.

[3]Y. Lee, A. Kim, and D. Park, *"GPU-Accelerated Steganography: Enhancing Performance," IEEE Access*, vol. 9, pp. 54321-54335, 2021. doi: 10.1109/ACCESS.2021.3057892.

[4]H. Tan and X. Wei, *"Steganography and Cryptography: Hybrid Approaches," Journal of Information Security and Applications*, vol. 66, pp. 102-119, 2022. doi: 10.1016/j.jisa.2022.102119.

[5]Q. Zhang, M. Li, and L. Zhou, *"Detection and Countermeasures of Steganographic Attacks," Computers & Security*, vol. 120, pp. 91-108, 2023. doi: 10.1016/j.cose.2023.102345.

[6]R. Patel and J. Roy, *"Advanced Techniques in Digital Steganography," IEEE Transactions on Multimedia*, vol. 25, pp. 4567-4580, 2023. doi: 10.1109/TMM.2023.3156789.

[7]D. Artz, *"Digital steganography: hiding data within data,"* in IEEE Internet Computing, vol. 5, no. 3, pp. 75-80, May-June 2001, doi: 10.1109/4236.935180.

Department of Computer Science and Engineering
(Cyber Security)
St. Joseph's College of Engineering & Technology,Palai
Palai - 686 579