SIMULATING SECURE EMAIL TRANSMISSION WITH IMAGE-BASED AES-ENCRYPTED STENOGRAPHY

By

DENVER WEKESA
ROSE WAMBUI GITHENGU
ALEX OUMA BARASA

A Project Submitted to the School of Informatics and Innovative Systems in Partial Fulfillment for the Award Bachelor of Science in Computer Security and Forensics Degree at Jaramogi Oginga Odinga University of Science and Technology

April 2025

DECLARATION AND APPROVAL

STUDENT

This project is our original work and has not been presented for an award of a diploma or conferment of a degree in any other university or institution.

Signature ……

Date ……

Wekesa Denver

I132/G/1202/21

Signature ……

Date ……

Githengu Rose Wambui

I132/0266/2020

Signature ……

Date ……

Barasa Alex Ouma

I132/G/1652/21

UNIVERSITY SUPERVISORS' APPROVAL

This research project report has been submitted for examination with my approval has a university supervisor.

Signature ……

Date ……

Dr. Joshua Agola

DEDICATION

This work is dedicated to our families whose unwavering support and encouragement have been the cornerstone of our journey. Their sacrifices and understanding during the long hours invested in this project are immeasurable. To our mentors and advisors, whose guidance has shaped my intellectual growth and enriched the quality of this work, we extend our deepest appreciation. Lastly, this work is dedicated to all individuals committed to the pursuit of knowledge and progress, as their collective contributions to the academic and professional realms inspire the continuous evolution of our understanding and capabilities.

ACKNOWLEDGEMENT

We express our sincere gratitude to the Almighty God for granting us good heath throughout the academic journey. This work was carried out at Jaramogi Oginga Odinga University of Science and Technology under the guidance and supervision of Dr.Joshua Agola, to whom we owe a special debt of appreciation. Dr. Agola's expertise, unwavering support, and insightful feedback were instrumental in shaping this project and overcoming the challenges we encountered along the way.

We also extend our thanks to our peers and faculty members at Jaramogi Oginga Odinga University of Science and Technology for their encouragement and valuable suggestions throughout this journey. Additionally, we appreciate the resources and conducive environment provided by the university, which enabled us to explore and implement our ideas effectively.

Finally, we are grateful to our families for their constant motivation and understanding, which kept us inspired and focused. This project is a culmination of collective effort, and we are deeply thankful for the support that made it possible.

ABSTRACT

As the volume of information stored and shared electronically increases, so does the need to enhance how it is secured. Email is one of the main and fastest communication tools in both daily and business life. The amount and the importance of data exchanged in emailing have been continuously increasing. This brings the security-related issues. To be able to protect the security of the information, there has been a need to develop innovative ways. Although most email servers are already including some measures, there could be some cases in which higher-level security measures would be needed. In our project, we propose an end-to-end secure emailing add-in using the steganographic method, which involves communicating secret data in an appropriate multimedia carrier, such as an image. The purpose of our project is to implement the concept of information confidentiality, integrity, and Authenticity in the email system through both steganographic and symmetric encryption techniques. This project therefore presents the combination of Least Significant Bit and Advanced Encryption Standards (AES). Python programming language libraries (Pillow, numpy, flask and pycryptodome) and, HTML, CSS, and JavaScript for user interface are used to develop the email simulation model. The result shows information can be stored, shared, and managed reliably and securely using the combined model.

Table of Contents

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

"A picture is worth a thousand words," as the saying goes, since it may provide more pictorial information than information acquired from a text for human interpretation. As a result, picture data representation, storage, and transmission must be acceptable. Information security is becoming increasingly important in data at rest and transit. Images are employed in various operations since they may contain significantly more information.

We are in an era where technology usage is growing more popular than a few years ago. This means we need to protect our digital assets from unauthorized access. The main aim is to secure information by ensuring safe communication between the sender and the receiver by adding multiple layers of security. This project is a combined approach of encryption and steganography techniques in the email simulation model to give security the utmost importance required. Encryption is divided into two types: symmetric encryption and asymmetric encryption. The most used symmetric encryption algorithms are the Advanced Encryption Standard (AES) and Data Encryption Standard (DES). Advanced Encryption Standard (AES) supports multiple variable-length keys, such as 128-bit, 192-bit, 256-bit, and 384-bit keys while Data Encryption Standard, the DES algorithm divides data into multiple 64-bit blocks at the encryption end and encrypts each block to generate a ciphertext. At the decryption end, the 64-bit ciphertext is converted into a 64-bit plaintext. The blocks are associated with each other. DES uses 16 iterative blocks to complete iteration, in which a 56-bit key is used for encryption and decryption. In our project, we use symmetric encryption, Advanced Encryption Standard (AES). Symmetric encryption achieves this by enabling the original content to be shown only after the correct key is used to decrypt the information. It prevents interception and theft of private information over networks. Encryption and steganography guarantee the confidentiality, integrity, identifiability, and non-repudiation of information.(Hambouz et al., 2019). In the sender, only the authorized person can send the secret image to the receiver. In the first stage, image steganography is processed using confidential information hidden under the cover images. The encryption method used is based on the Random Number Generator and the pixel indicator, in other words, by moving the position of the pixel in the image to encrypt the information on the transmitting side. On the receiver side, this project proposes a method to authenticate the receiver by entering a secret key to decrypt the encrypted information(Manikandan et al., 2021).

The challenge is in communication industries, where the current technology influences the storage, sharing, and management of people’s information. That is, industries employ electronic means of communication to support their mode of service and deliver reliable information. Organizations have encountered at least one data breach, which costs more than a million on average per organization.

1.1 PROBLEM STATEMENT

As the reliance on email for sharing sensitive information grows, existing security measures such as encryption are insufficient to safeguard confidential data. The increasing sophistication of cyber threats and the ease of email interception and data breaches pose significant risks to individuals and organizations. To address these challenges, this project proposes a novel approach that combines image steganography with Advanced Encryption Standards (AES) to enhance the security of email communications. By embedding encrypted sensitive data within seemingly harmless images, this method can effectively conceal the existence of the hidden information, making it difficult for unauthorized parties to detect and exploit

1.2 OBJECTIVES

1.2.1 Main Objective:

- To develop an email simulation model that uses image steganography and Advanced Encryption Standard (AES) to protect sensitive information during communication.

1.2.2 Specific Objectives:

1. To investigate data security in email communication
2. To assess data privacy in the Information Technology Industry
3. To develop an email simulation model
4. To test and evaluate email simulation model

1.3 Research Question

1. What are the key security challenges in email communication, and how can they be mitigated?
2. How does data privacy impact the Information Technology industry, and what role does secure email communication play?
3. What are the essential components required to develop a secure email simulation model?
4. How effective is the developed email simulation model in protecting sensitive information compared to traditional email security measures?

1.4 Significance of the Study

It's important to keep information sharing safe and private in the organization. If we use image steganography, it is going to hide important data inside images. This makes it extra difficult for unauthorized people to get access to the information. Organizations can use this technique to make sure that private data stays safe when it's shared or stored on computers. This is important for following privacy rules and keeping people’s trust. It also helps to stop hackers from getting hold of the information, which could be serious for both employees and organizations.

1.5 Scope

This project aims to develop a secure email system for transmitting and storing sensitive information. By combining image steganography and symmetric encryption, it will embed confidential information within images. The project will focus on image selection and preprocessing, encryption algorithm integration, steganographic technique implementation and security analysis. By addressing these aspects, we aim to enhance the confidentiality and

integrity of data, safeguarding people's privacy and enabling secure communication practices within organizations

1.6 Assumptions

Some of the assumptions on the project are as follows: first, the cover image used for steganography has sufficient quality and resolution to withstand the embedding process without significant degradation. Secondly, our chosen symmetric encryption algorithm, Simple Mail Transfer Protocol, and Internet Message Access Protocol for email communication provide adequate security to protect the embedded data. Thirdly sufficient computational resources (e.g., Central processing unit) are available to implement and execute the steganography and encryption algorithms efficiently. Moreover, the project complies with relevant data privacy regulations ensures the confidentiality of people's information, and can be incorporated with existing devices. Our project also embeds sensitive data in various media formats (e.g. images, or text). Finally, sufficient information is available for testing and evaluation the email simulation model.

Research Time

TASK	DURATION					
	OCTOBER	NOVEMBER	DECEMBER	JANUARY	FEBRUARY	MARCH
Concept paper.						
Project proposal						
Proposal defense						
Model design						
Model implementation						
Testing and training.						
Documentation	=					

CHAPTER TWO

LITERATURE REVIEW

Many researchers have worked on the security of data using cryptography, steganography, and a combination of the two techniques. There are many expertise who have carried out research on data and information security including (Sachin & Kumar, 2010), researched the GSM (Global System for Mobile Communications) network. In their research, they gave additional encryption to the GSM network by implementing Data Encryption Standard (DES), Triple Data Encryption (3DES), and Advanced Data Encryption (AES). It was discovered that 3DES is better in security performance than DES but not AES because it takes a longer time for the brute force attack to breach AES security than 3DES and DES, similarly the time taken by the brute attack to breach 3DES security is more than DES.

Even before computers, people used steganography to keep their messages safe. The ancient Greeks and Romans were good at this. They used things like invisible ink, secret compartments, and hidden messages to send important information without anyone else knowing. During World War II, both the good guys and the bad guys used steganography to share secret plans. They hid tiny messages in pictures. It was like a secret game of hide-and-seek, but with really important information secure.

Image steganography has emerged as one of the most widely studied and applied forms of steganography in the digital age. Numerous algorithms and methodologies have been proposed for embedding hidden messages within digital images while preserving visual quality. Techniques such as least significant bits (LSB) substitution, LSB matching, and adaptive steganography have been extensively researched for their effectiveness and security properties. Researchers have also explored the vulnerabilities of image steganography to steganalysis techniques, leading to the development of more robust and covert methods.

(Singh & Singh, n.d.)The Least Significant Bit (LSB) steganography is one such technique in which the least significant bit of the image is replaced with a data bit. As this method is vulnerable to steganalysis to make it more secure we encrypt the raw data before embedding it in the image. Though the encryption process increases the time complexity, but at the same time provides higher security. This approach is very simple. In this method, the least significant bits of some or all of the bytes inside an image are replaced with a bit of the secret message. The LSB embedding approach has become the basis of many techniques that hide messages within multimedia carrier data. LSB embedding may even be applied in particular data domains – for example, embedding a hidden message into the color values of RGB bitmap data, or the frequency coefficients of a JPEG image. LSB embedding can also be applied to a variety of data formats and types. Therefore, LSB embedding is one of the most important steganography techniques in use today.

In photography and computing, Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only two colors, black (also called binary images). Grayscale images have many shades of gray in between. Grayscale images are often the

result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases, they are monochromatic proper when only a given frequency is captured.

Integrating encryption and steganography techniques offers a cooperative approach to data security and privacy. By encrypting sensitive data before embedding, it within digital media using steganography, the study aims to enhance the confidentiality and integrity of protected communication channels. Various studies have investigated the feasibility and efficacy of combining encryption and steganography methods, exploring factors such as capacity, robustness, and computational complexity.

(Apoorva Arora, 2015) In their research work, they stated that the AES algorithm was selected in the final shortlisting of encryption algorithms because it provides the best encryption security. The conclusion was obtained that the AES encryption algorithm is considered the fastest one among the other options. To perform the steganography in the third objective, which is used to embed an image (secret object) into an image (cover object), Several papers have studied for the selection of the best steganography technique for the development of our security model.] Steganography is a security mechanism which is used to hide the message into another object which may be a text or image.

(Thomas & Panchami, 2015), worked on the security of messages transferred through Short Message Service (SMS). The study used the blowfish algorithm for the encryption and decryption of text messages being sent through SMS from one mobile phone to another. It was concluded that the blowfish algorithm is fast and saves the battery of the mobile phone.

There are lots of encryption algorithms available in cryptography. A study was conducted on symmetric key algorithms and asymmetric key algorithms. It is found that symmetric key algorithms run faster than asymmetric key algorithms such as RSA and the memory requirement of symmetric algorithms is lesser than asymmetric encryption algorithms. The performance of asymmetric algorithms is relatively low as compared to symmetric key encryption. Most asymmetric algorithms depend on the properties of hard problems in mathematics. Asymmetric encryption works slower as compared to symmetric encryption.

The comparisons of symmetric key algorithms like DES, 3DES, and AES are performed. Based on key size and security the supremacy is for the AES algorithm over DES and Triple DES. The function of the AES algorithm provides a high level of security to encrypt the plaintext data. Also, the AES algorithm runs faster than DES and 3DES symmetric key encryption algorithms.

AES uses a 128-bit block length and supports 128-bit, 192-bit, 256-bit, and 384-bit key lengths. It supports different platforms. A 128-bit key can provide sufficient security and takes less time to process than longer keys. To date, the AES does not have any serious weaknesses. DES can still be used due to the production of a large number of fast DES chips. However, AES will gradually replace the DES and 3DES to enhance security and efficiency.

Considering modern standardization AES is the current encryption standard recommended by the National Institute of Standard and Technology (NIST) and widely adopted, while DES is deprecated. AES is designed to meet the demand of modern encryption needs and is expected to remain secure shortly. Additionally, AES has simpler key management and benefits from hardware acceleration in modern processors.

(Noida International University, Department of Computer Science and Engineering, NCR Delhi Noida, India et al., 2019) The paperwork concentrates on the process of steganalysis, which is the art of detecting hidden messages within files, applications, and limitations of Steganography. It presents a deliberation on diverse steganography images file formats like JPEG (Joint Photographic Experts Group*)*, BMP(Bitmap), and PNG (Portable Network Graphics) along with color models which is a system for creating a whole range of colors from the basic colors for image formats like CMYK cyan- magenta- yellow -black model and RGB model which consists of three basic colors red, green and blue. A modified inspection of the existing models is performed, based on parameters like stego image perceptibility, technical resources, and security facets. The inspection shows that JPEG (DCT/DWT) algorithms are more resistant to attacks and offer higher security. BMP spatial domain techniques have higher capacity but are more vulnerable to detection whereas the PNG palette is reliable for small-sized data but less suitable for large amounts of data. Accordingly, the Bitmap format is apt for high-capacity requirements. To effectively hide a secret message, one must consider the type of message (text, image, audio), the desired capacity (how much data to hide), the required security level, and the type of cover image (format and color model) so that it disallows the captivation of the attacker.

Email systems use Simple Mail Transfer Protocols which use port 25 for mail transport while POP (Post Office Protocol, port 993 for security purposes) and IMAP (Internet Message Access Protocols, port 995 for security purposes) to retrieve mail. POP downloads emails to a single device and deletes them from the server, while IMAP keeps emails on the server, allowing access and synchronization across multiple devices. IMAP offers more flexibility and organization, making it the preferred choice for most users, especially those who access their email from multiple devices.

CHAPTER THREE

3.0 METHODOLOGY

Due to the rapid present-day growth of communication methods, the preservation of data during transmission has become a pressing concern. It is crucial that data exchanged between parties be protected against unauthorized access or alteration. The primary goal of secure data transmission is to protect data from unauthorized access, tampering or eavesdropping during its transit from one user to another. Secure data transmission can be achieved by keeping the confidentiality, integrity and authenticity of the data. To protect, data confidentiality, integrity and authenticity techniques like cryptography, steganography etc. can be applied.

To protect the information from unauthorized access a certain method is designed and applied on a data. This section gives detailed understanding of the algorithm designed and in-depth knowledge of the mechanisms used to implement the same. The proposed methodology works on two important concepts which altogether helps to improve the data security. The concepts include Symmetric Encryption and Image Steganography. This method is mainly divided into two different layers where each layer plays its important role and provide extra security. To support the same methodology and to perform Encoding and Decoding operations using this methodology, Simulation is designed using Python programming language (libraries Pillow, Flask, numpy, and pycryptodome) and, HTML, CSS, and JavaScript for frontend.

First Layer is Symmetric Encryption algorithm. This layer uses Advanced Encryption Standard algorithm also known as AES to encrypt the message and convert the plain text message into cipher text which cannot be understood without decrypting. Output of this layer is given as an input to the next layer. The next and final layer will accept the output of the first layer as input and embed it in a cover image using LSB to generate an output in image form called a stega image such that only authorized users can extract the image and read the sent mail.

CHAPTER FOUR

4.0 MODEL DESIGN AND IMPLEMENTATION

Our Steganographic Email simulation model is an email system designed to solve the Information Security vulnerabilities associated with eavesdropping, phishing and tapping of messages on transit. We came up with these models to address these challenges by encrypting sender's mail and on top embedding the encrypted mail in a cover image before sending it using AES encryption and LSB steganography. This ensures that only the sender and the receiver have the access to the message using a similar key for encryption and decryption thus safeguarding the integrity of information sent.

4.1 How Does AES Work?

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block size of 128 bits. It converts these individual blocks using 128, 192, and 256-bit keys. Once it encrypts these blocks, it joins them to form the ciphertext.

It is based on a substitution-permutation network, it consists of a series of linked operations, including replacing inputs with specific outputs (substitutions) and others involving bit shuffling (permutations).

Since a single block is 16 bytes, a 4x4 matrix holds the data in a single block, with each cell holding a single byte of information.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

The matrix shown in the table above is known as a state array. Similarly, the initial key is expanded into (n+1) keys, with n being the number of rounds followed in the encryption process. So, for a 128-bit key, the number of rounds is 16, with the number of keys to be generated being 10+1, which is a total of 11 keys.

Steps to be followed in AES

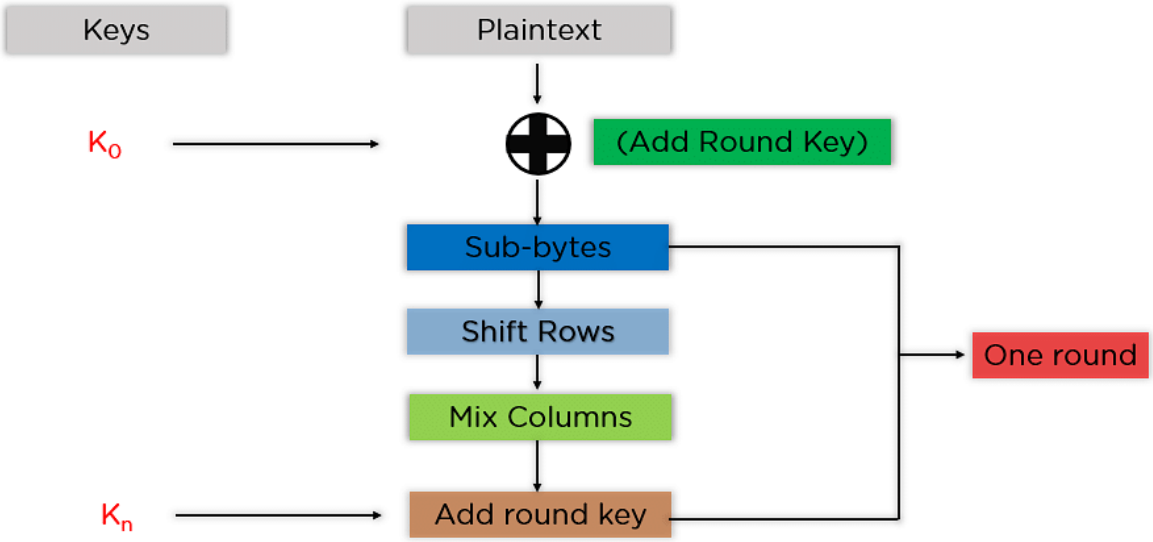
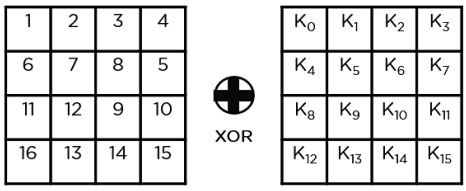


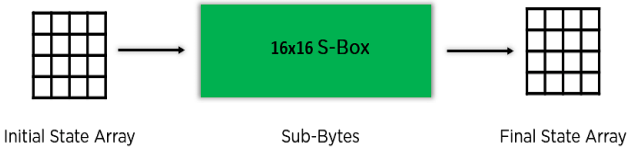
Figure 1: Steps followed in AES

The mentioned steps are to be followed for every block sequentially. It joins them to form the final ciphertext after successfully encrypting the individual blocks. The steps are as follows:

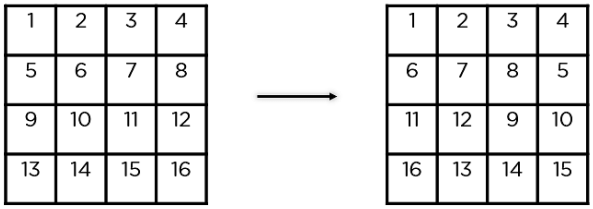
- Add Round Key: You pass the block data stored in the state array through an XOR function with the first key generated (K0). It passes the resultant state array on as input to the next step.



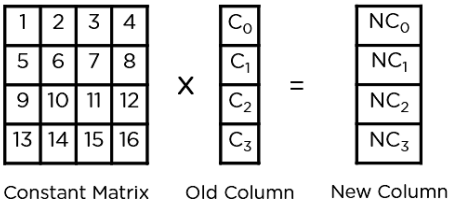
- Sub-Bytes: Byte-level substitution with the aid of an S-box. The relevant value from the S-box is substituted for every byte in the state matrix



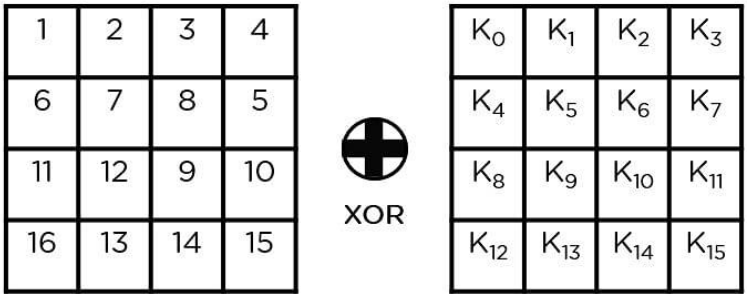
- Shift Rows: Diffusion is created by shifting rows of the current state matrix to the left. The first row is left unaltered, followed by a one-byte shift in the second row, a twobyte shift in the third row, and a three-byte shift in the fourth row.



- Mix Columns: The state matrix's individual columns are blended via a matrix multiplication process. Diffusion is created, and the cryptographic strength is increased



- Add Round Key: In this stage state matrix is XORed with the round key which is assigned for the current round.



4.1.2 DECRYPTION PROCESS

AES decryption process is somewhat similar to the encryption however the operations are performed in a reversed manner.

Decryption process mainly involves following stages:

• Adding Round Key:

In this stage the XOR operation is performed on the cipher text with the last round key.

• Inverse Shift Rows:

In this stage the rows which were shifted to the left side during encryption

process are shifted back in opposite direction to reverse the encryption effect.

• Inverse Sub Bytes:

Each byte of state matrix was replaced by a corresponding byte from the s-box

during encryption process same process is reversed in this stage the bytes of sbox are replaced with corresponding value.

• Final Round of Decryption:

The first "Add Round Key" operation, which was applied to the plaintext during

encryption, is reversed in this stage. The result of the "Add Round Key"

operation is the original plaintext, which has been successfully decrypted,

represented in the state matrix

4.1.3 Steganography process using Least Significant Bit

Steganography is an additional and important method for enhancing the security of the data transmission procedure. The basic idea of steganography is to accomplish covert communication, in which the existence of the confidential information is concealed from anyone who is not supposed to receive it. The purpose of steganography is to make it difficult for unauthorized individuals to access the concealed message by concealing its existence.

This method mainly takes benefit of the fact that micro or small changes in the pixels least significant bit is rarely visible to the human vision. In Digital images pixels are mainly represented or consists set of color channels like Red, Green and Blue in RGB image. Each colors intensity is presented using certain number of bits. LSB is the rightmost bit of the binary representation of color values.

The fundamental principle of LSB steganography is to swap out the LSB of the cover image's pixel values with the bits of the hidden message. This alteration typically goes undetected since the LSB has the least impact on the image's aesthetic appearance. Let’s see how the encoding and decoding process works with LSB in steganography along with its code implementation.

Embedding of the data into cover Image using LSB

Hiding the data into cover image consists of multiple steps and each step is very important as

per security concerns. These steps are as following:

1. Preparing the Message:

Generally, the message which is to be embedded into Cover image is in text or binary format. So, each bit of the message needs to be represented as equivalent to binary.

2. Pixel Selection:

In this step the sequence of pixels is chosen from the cover image where the data will be hidden. The selected pixel should be big enough to accommodate the complete message.

3. Embedding:

The embedding procedure substitutes bits from the secret message for the least significant bits of each pixel's color channel values (Red, Green, and Blue) for each frame in the chosen sequence.

For example, let’s suppose if the least significant bit of pixel’s color values is (130,199,50) for RGB respectively and the message bit is 10000101 so it changes the value as follows

130 in binary is 10000010 so adding the message bit to its LSB will give 10000011 which makes small change from 130 to 131.

199 in binary is 11000111 so adding the message bit to its LSB will give 11000110 which is 198 in decimal.

Same with 50 which is 00110010 in binary and adding message bit will produce 00110011 that is 51 in decimal.

Same process of modifying the least bit of the pixel is repeated for each bit in the secret message sequence and entire message is accommodated in a cover image.

4. End marking:

To ensure decoding process fully smooth and quick end market could be added to the encoded data which helps to identify the message is complete.

4.1.4 Extracting Data from cover image using LSB

Once the data is embedded into a Cover image, to extract the hidden data same methodology is applied but in a reverse manner and this methodology consists of different steps which are as follows.

Choose Pixels: In this step the pixels of image which was modified during the embedding process are identified. Which helps to locate the part of an image where data is stored.

Retrieve LSB: Once the pixels are selected in a sequence then least significant bit of each color from each pixel is extracted and then the last bit of each color is isolated. Once all the last bits are extracted then they are rearranged in such a manner to reconstruct the hidden binary data.

Termination Marker: During the encoding process termination marker or end marker was added to signal the end of the secret massage. This step involves detecting this end marker once the marker is found it indicates that entire data is extracted successfully

Reconstruction: Once all the bits are collected and end marker is detected this step starts processing. In this the sequence of bits is grouped into bytes and then these bytes are reconstructed into original data format

Below attached figures 1,2 and 3 represents the flow of simulating the secure email transmission with images-based AES Encryption Steganography

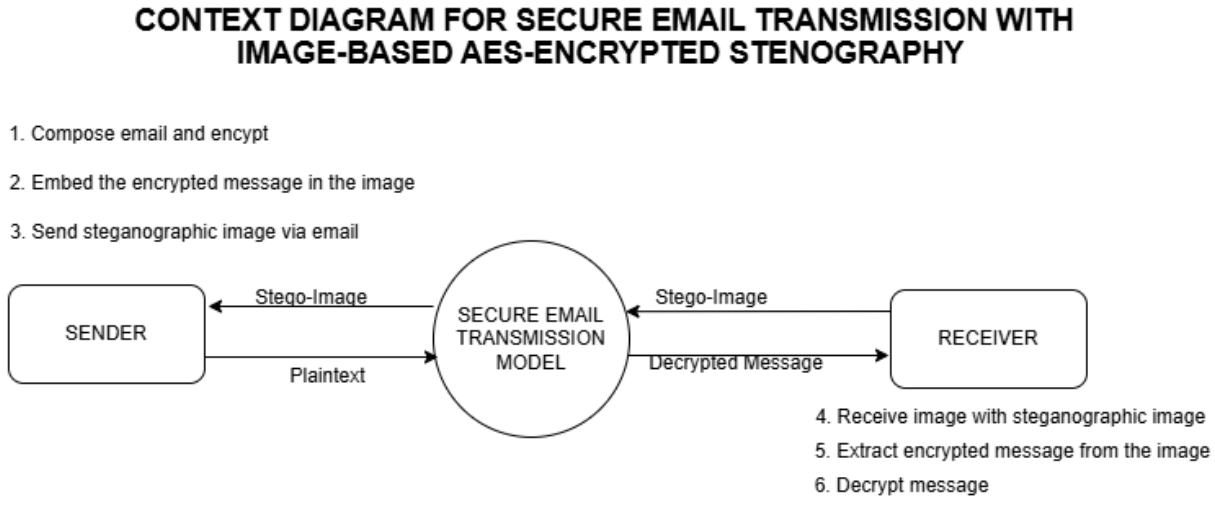


Figure 2 Context Diagram

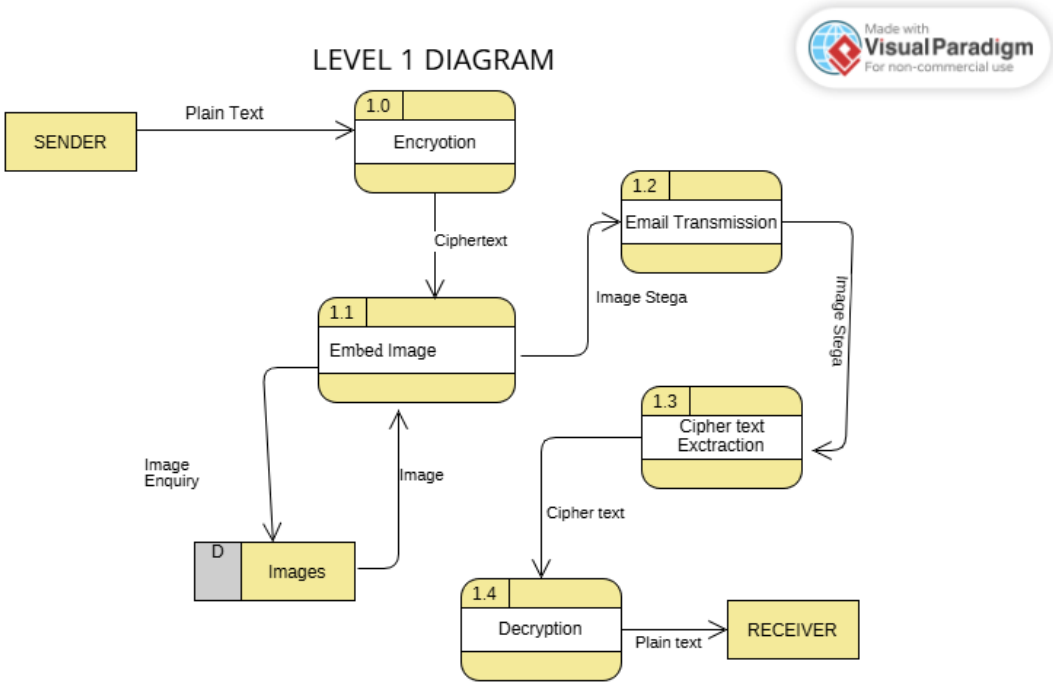


Figure 3 Level 1 Diagram

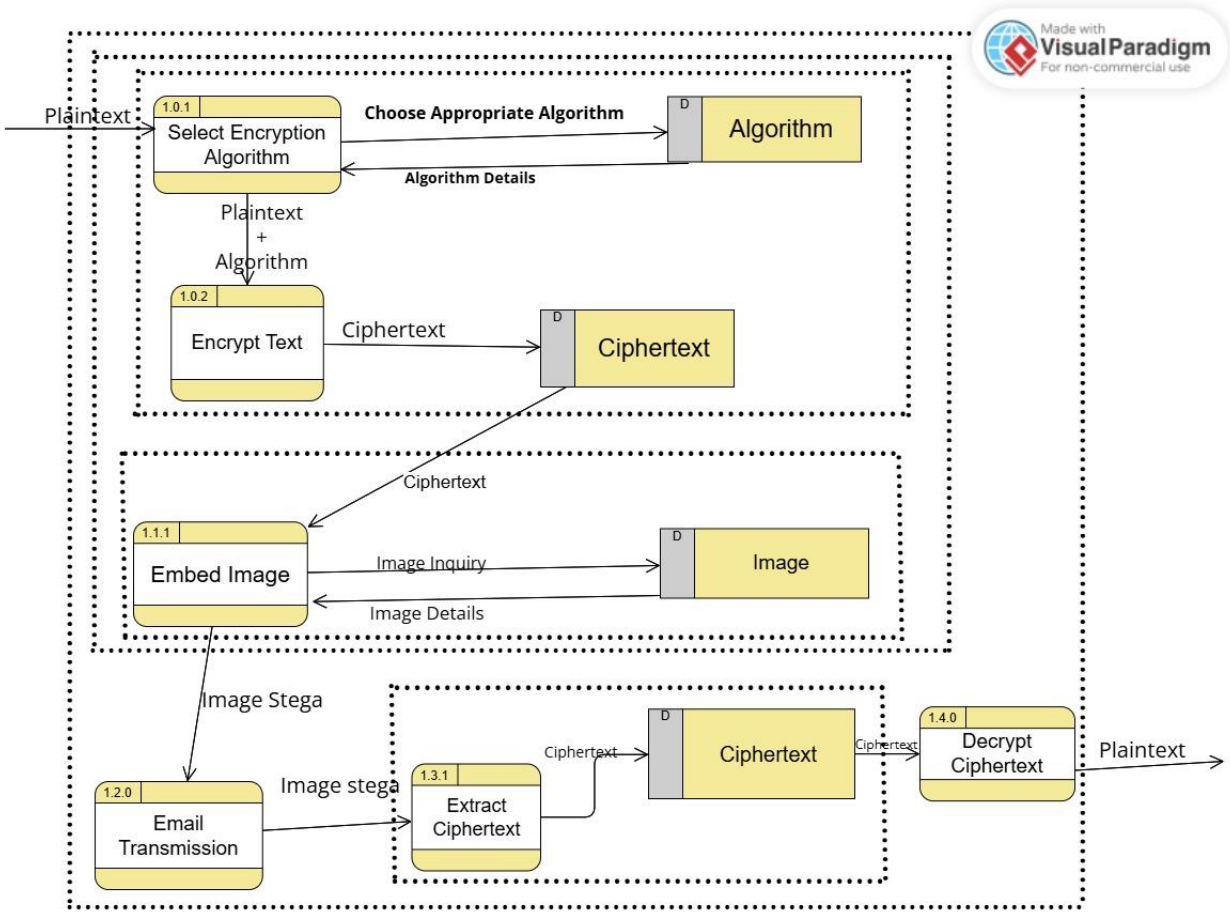


Figure 4 Level 2 Diagram

Database Design

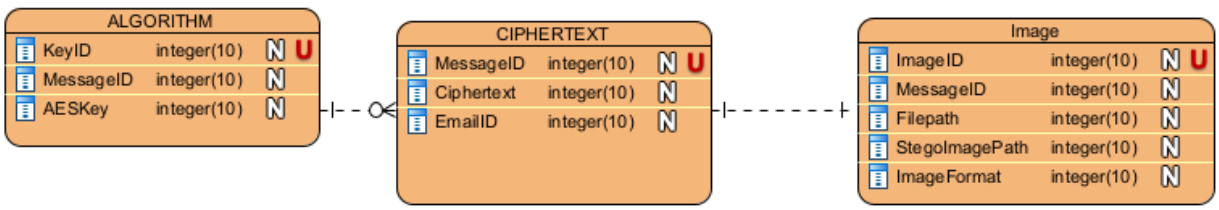


Figure 5: Entity Relational Diagram

PSEUDOCODE

Pseudocode for Secure Email Transmission System

1. Encryption Process

Plaintext

BEGIN

FUNCTION Encrypt(plaintext):

    SELECT encryption_algorithm = AES

    GENERATE encryption_key

    APPLY encryption_algorithm(plaintext, encryption_key) → ciphertext

    RETURN ciphertext, encryption_key

END

2. Image Steganography (Embedding Ciphertext)

Plaintext

BEGIN

FUNCTION EmbedCiphertext(ciphertext, cover_image):

    OPEN cover_image

    EMBED ciphertext into cover_image

    SAVE stego_image

    RETURN stego_image

END

3. Email Transmission

Plaintext

BEGIN

FUNCTION SendEmail(stego_image, recipient_email):

    ATTACH stego_image

    SEND email to recipient_email

```
    IF delivery_successful:
        RETURN "Email Sent Successfully"
    ELSE:
        RETURN "Email Delivery Failed"
END
```

4. Ciphertext Extraction from Stego Image

```
Plaintext
BRGIN
FUNCTION ExtractCiphertext(stego_image):
    OPEN stego_image
    EXTRACT ciphertext from stego_image
    RETURN ciphertext
END
```

5. Decryption Process

```
Plaintext
BEGIN
FUNCTION Decrypt(ciphertext, encryption_key):
    APPLY decryption_algorithm(ciphertext, encryption_key) → plaintext
    RETURN plaintext
END
```

CHAPTER FIVE

5.1 TESTING AND VALIDATING THE PROPOSED MODEL

This chapter outlines the testing and validation processes undertaken to ensure the secure email simulation model meets its objectives of providing confidentiality, integrity, and authenticity in email communication. The model, which integrates Advanced Encryption Standard (AES) encryption and Least Significant Bit (LSB) steganography, was rigorously evaluated through unit testing, integration testing and system testing. Below, we describe how these tasks were performed, the strategies employed, the test plans and procedures for critical functions, the tools utilized, and the results obtained.

5.1 How Testing and Validation Tasks Were Performed

Testing and validation were conducted systematically to confirm that the email simulation model functions as intended and securely protects sensitive information. The process involved:

- **Unit Testing**: Each module (AES encryption/decryption, LSB embedding/extraction, and email transmission) was tested independently to verify individual functionality.
- **Integration Testing**: The interactions between modules were assessed to ensure they operate cohesively when combined.
- **System Testing**: The complete system was tested end-to-end to simulate real-world email communication scenarios, validating overall performance and security.
- **Validation**: The system's ability to maintain confidentiality (data hidden and encrypted), integrity (data unchanged during transit), and authenticity (accessible only to authorized users with the correct key) was confirmed through comparative analysis of inputs and outputs.

Testing was performed in a controlled environment using predefined test data, including sample plaintext messages ("Confidential Data"), cover images (270x148 PNG files), and 128-bit AES keys. Validation involved checking that the original message could be accurately recovered after decryption and extraction, with no detectable traces of hidden data in the stego-image for unauthorized parties.

5.2 Plans and Strategies Used in Integration Testing and System Testing

5.2.1 Unit Testing

- **Plan:** Isolate and test each component (AES encryption/decryption, LSB steganography, email simulation) to ensure it performs its specific task correctly.
- **Strategy:**

Used test cases with known inputs and expected outputs (e.g., encrypting "Good morning, this is our testing phase and implementation, thanks" and verifying the ciphertext).

Included edge cases such as empty inputs, oversized messages, and invalid keys to test robustness.

Employed automated scripts to repeatedly execute functions and log results for consistency.

5.2.2 Integration Testing

- **Plan**: Verify that the AES encryption, LSB steganography, and email transmission modules integrate seamlessly.
- **Strategy**:

Adopted a bottom-up approach: First tested AES encryption/decryption, then added LSB embedding/extraction, and finally integrated email transmission.

Conducted incremental tests to isolate integration issues (e.g., embedding encrypted data into an image, then extracting it before transmission).

Simulated error conditions (e.g., mismatched keys) to ensure proper error handling and recovery.

5.2.3 System Testing

- **Plan**: Evaluate the entire system's performance and security in a simulated email environment.
- **Strategy**:

Performed end-to-end tests with sender and receiver roles to replicate real-world usage.

Varied test conditions (e.g., different message lengths, image resolutions) to assess scalability and reliability.

Conducted security tests to attempt unauthorized data extraction (e.g., without the AES key) and analyzed stego-images for detectability using statistical methods.

5.3 Test Plans and Procedures for Testing Critical Functions

5.3.1 Test Plan

The test plan focused on validating the critical functions: AES encryption/decryption, LSB embedding/extraction, and email transmission. The objectives were to ensure data security and system reliability, with the following details:

- **Objective**: Confirm that the system protects sensitive information during transmission.
- **Scope**: Test AES encryption/decryption, LSB steganography, and email delivery processes.
- **Resources**: Python environment with libraries (Pillow, NumPy, Flask, PyCryptodome), sample cover images, text messages, and a simulated email setup.
- **Success Criteria**:

- Encrypted data is unreadable without the correct key.
- Stego-images exhibit no visible changes compared to cover images.
- Transmitted messages are accurately recovered by the recipient.

5.3.2 Test Procedures

Test Procedure 1: AES Encryption/Decryption

- **Input**: Plaintext ("Secure Email Test"), 128-bit key ("abcdef1234567890").
- **Steps**:
  1. Encrypt the plaintext using AES with the specified key.
  2. Record the resulting ciphertext.
  3. Decrypt the ciphertext using the same key.
  4. Compare the decrypted output with the original plaintext.
- **Expected Output**: Decrypted message matches "Secure Email Test".
- **Pass Condition**: Exact match with no data corruption.

Test Procedure 2: LSB Embedding/Extraction

- **Input**: Cover image (e.g., "cover.png", 270x148), encrypted ciphertext from Procedure 1.
- **Steps**:
  1. Embed the ciphertext into the cover image using LSB steganography.
  2. Save the resulting stego-image as "stego.png".
  3. Extract the ciphertext from "stego.png".
  4. Compare the extracted ciphertext with the original input.
- **Expected Output**: Extracted ciphertext matches the original; stego-image visually identical to cover image.
- **Pass Condition**: Accurate data recovery and no perceptible image degradation.

Test Procedure 3: Email Transmission

- **Input**: Stego-image ("stego.png"), recipient email (simulated as "alex2019ouma@gmail.com").
- **Steps**:
  1. Attach the stego-image to an email via the Flask-based simulation.
  2. Send the email to the recipient.

3. Retrieve the email and download the stego-image.
4. Extract and decrypt the message using Procedure 2 and Procedure 1 (in reverse).

- **Expected Output**: Receiver recovers "Secure Email Test" from the stego-image.
- **Pass Condition**: Email delivered successfully with intact stego-image.

5.4 Test Tools Used

The following tools were employed to test and validate the model:

- **Python**: The primary language for implementing and testing the system.
  - **Pillow**: Handled image loading, manipulation, and saving for steganography tests.
  - **NumPy**: Supported pixel-level operations and data array management during LSB embedding/extraction.
  - **PyCryptodome**: Provided AES encryption/decryption functions with 128-bit key support.
  - **Flask**: Facilitated the web-based email simulation interface for transmission testing.
- **HTML/CSS/JavaScript**: Enabled the front-end interface for user interaction and result visualization.
- **unittest (Python)**: Automated unit tests for individual functions (e.g., test_encrypt_decrypt(), test_embed_extract()).
- **StegExpose**: A hypothetical steganalysis tool used to test the detectability of hidden data in stego-images (optional, based on availability).
- **Image Viewer**: Manual inspection tool (e.g., Windows Photos) to visually compare cover and stego-images.
- **MySQL.connector** is a driver that allows applications to interact with the MySQL database. When using XAMP, which comes with MariaDB (a MySQL-compatible database).

5.5 Test Results

The testing process yielded successful outcomes across all phases, confirming the model's effectiveness. Below are the summarized results.

5.5.1 Unit Test Results

- **AES Encryption/Decryption**: Successfully encrypted and decrypted test messages with no data loss.

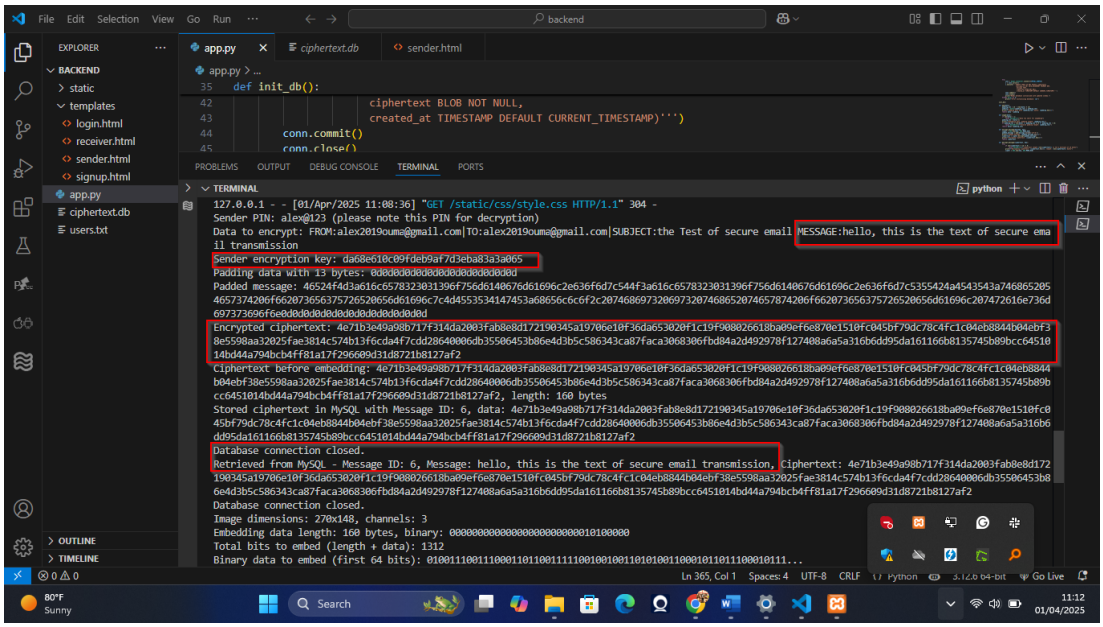


Figure 6: AES encryption/Decryption

- **LSB Embedding/Extraction**: Embedded data was accurately extracted, with stego-images showing no visible changes.

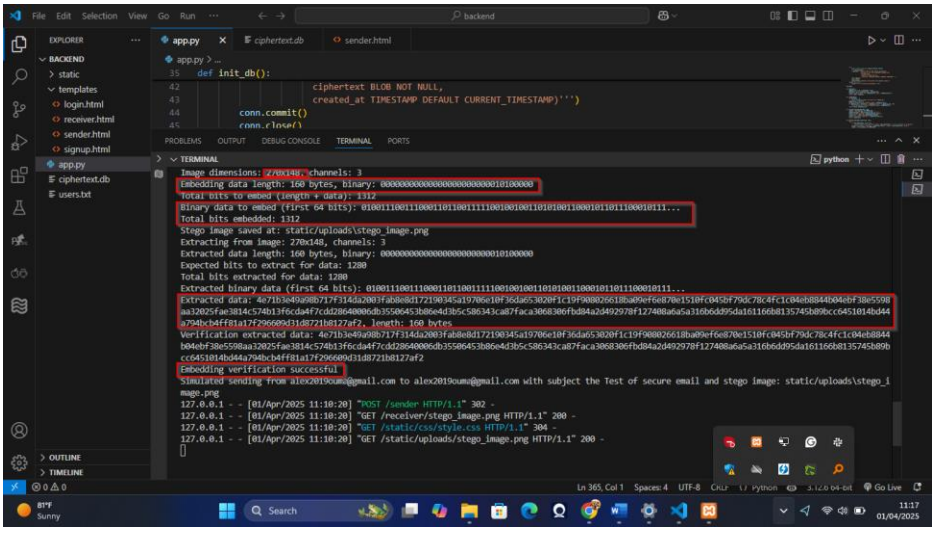


Figure 7: LSB embedding/ Extraction

- **Databases:** plaintext and ciphertext is stored accurately without being lost.

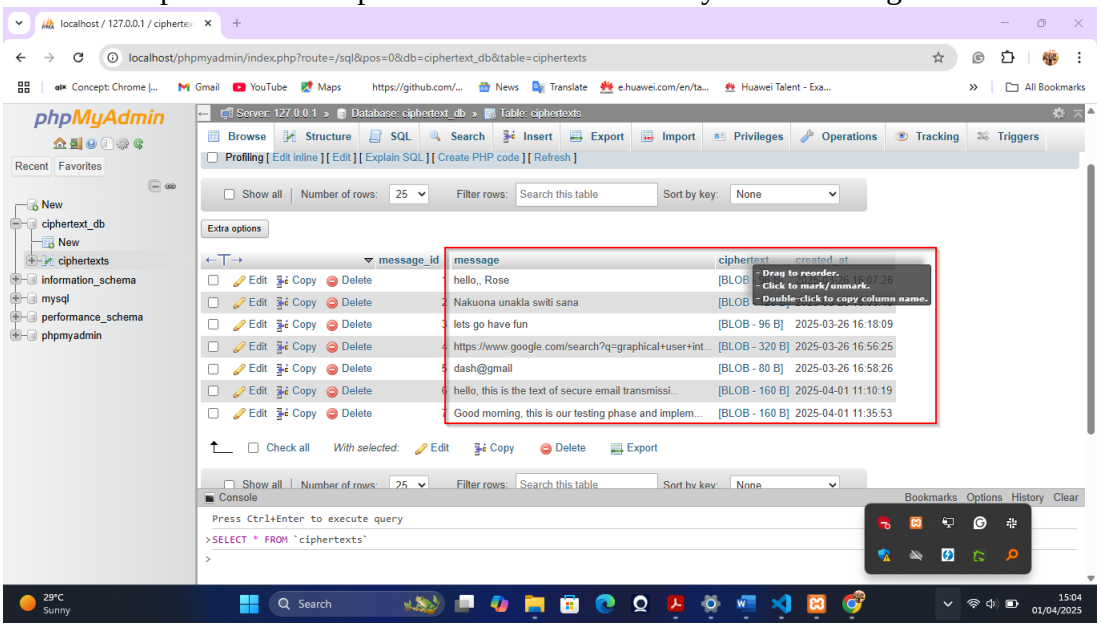


Figure 8: Database storage of Ciphertext

5.5.2 Integration Test Results

- **Encryption + Steganography**: The encrypted message was successfully embedded and extracted from the stego-image.

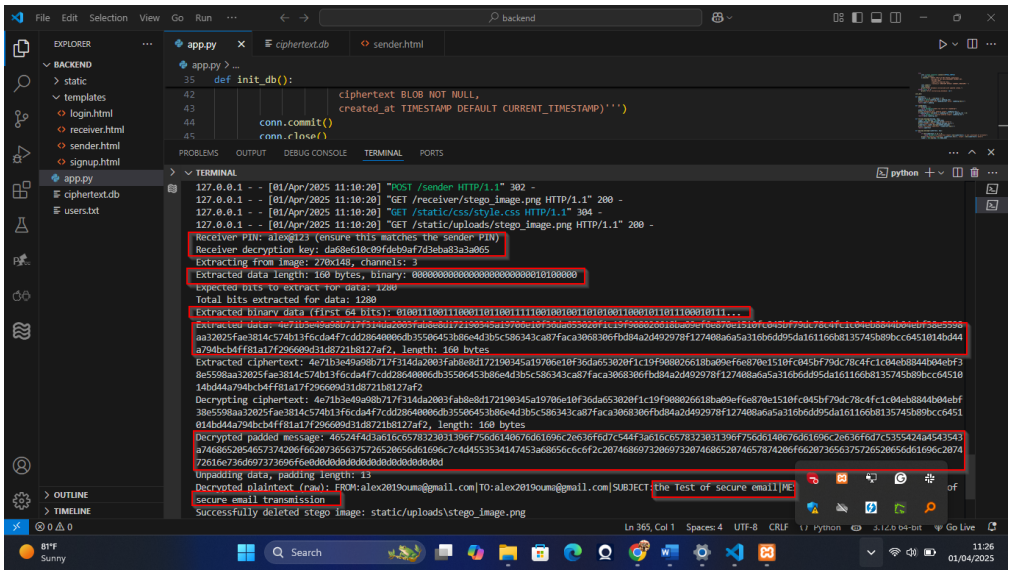


Figure 9: Encryption & Steganography

5.5.3 System Test Results

- **End-to-End Email Transmission**: The stego-image was sent, received, and decoded correctly.
- **Sender**

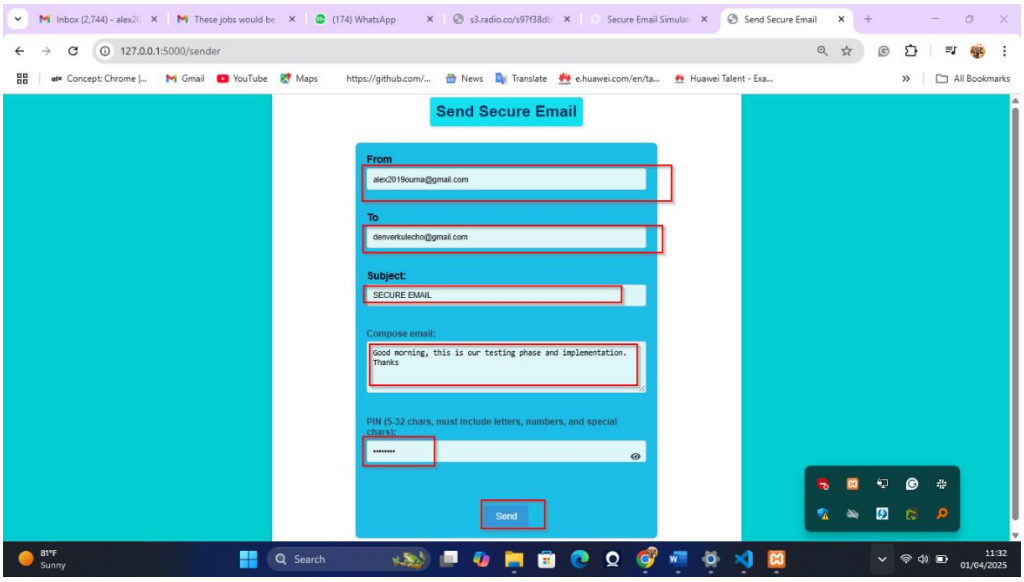


Figure 10: Sende's Dashboard

- **Receiver**

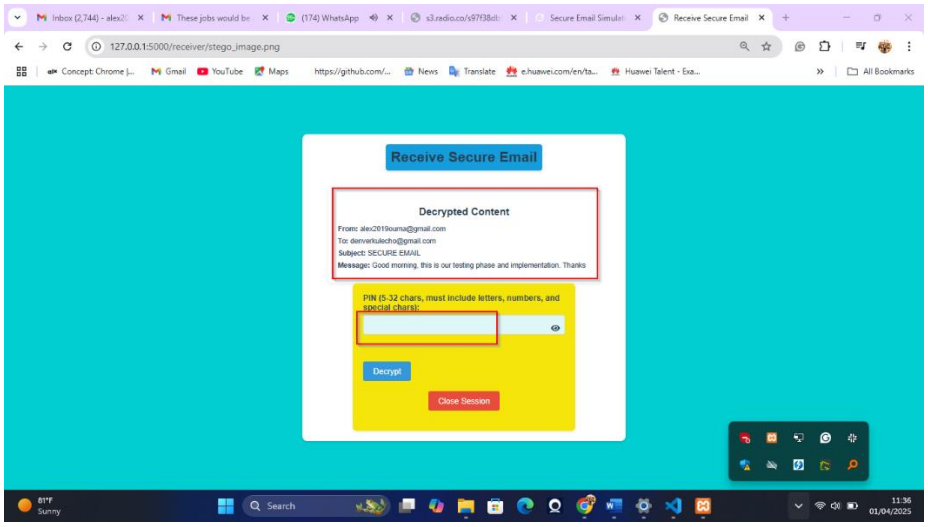


Figure 11: Receiver's Dashboard

- **Visual Comparison**: Cover and stego-images were indistinguishable to the naked eye.
    - Left: "cover.png"
    - Right: "stego.png"

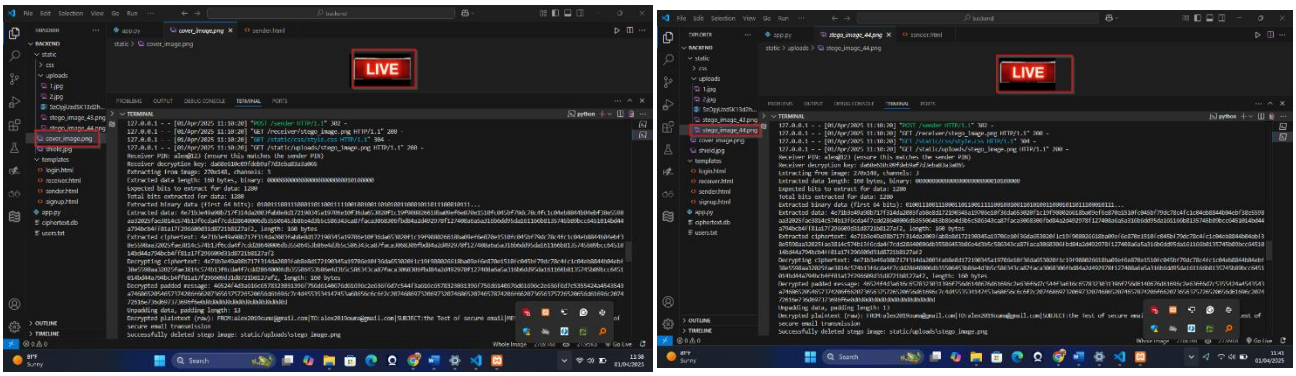


Figure 12: Cover image & stego image

6.0 CHAPTER SIX: CONCLUSION, FINDINGS AND RECOMMENDATION

6.1 CONCLUSION

Simulating secure email transmission using image-based AES encryption and steganography provides an effective way to protect sensitive information during digital communication. By combining the strong encryption capabilities of AES (Advanced Encryption Standard) with the covert nature of steganography, this approach ensures that both the confidentiality and integrity of the message are maintained. The use of an image to carry the encrypted message adds an additional layer of security, making it difficult for unauthorized parties to detect the presence of hidden data. This method significantly reduces the risk of interception or eavesdropping, providing enhanced privacy for email communication.

However, while this method can be highly effective for protecting messages, it is important to note that its security relies on the secrecy of the AES key and the effectiveness of the steganographic method used to hide the data. If the steganography technique is poorly executed or the AES key is compromised, the system may lose its protective value.

6.2 FINDINGS

As email communication continues to be a critical means of transmitting sensitive information, securing it against unauthorized access is crucial. The combination of AES encryption and steganography provides a robust solution for protecting email messages. AES encryption ensures that the content of the message remains unreadable to anyone without the correct key, while steganography hides the encrypted data within an image, adding an additional layer of concealment. The findings below explore the strengths, weaknesses, and security considerations associated with this approach, highlighting how effectively AES encryption and steganography work together in securing email communication.

1. **Encryption (AES)**
   AES is a secure and efficient symmetric encryption algorithm, widely recognized for its ability to provide confidentiality and integrity.The choice of AES key length (128, 192, or 256 bits) significantly impacts both the security and performance of the encryption, with longer keys offering more protection but requiring more computational resources.

2. Steganography

Steganography, particularly using LSB (Least Significant Bit) methods, can effectively hide the ciphertext inside an image without noticeable alterations to the image's visual appearance.The choice of image is important; larger images provide more space for embedding data, making it easier to hide more complex messages.

3. Security of the Key

The system's security is only as strong as the secrecy of the AES key. If the key is compromised, the encrypted message can easily be decrypted.Secure transmission of the key (e.g., via a different secure channel) is crucial to maintaining the integrity of the system.

4. Practicality and Secrecy

Embedding encrypted data within an image adds a layer of "secrecy" to the message transmission, making it harder to detect that the image contains sensitive information.Email systems do not inherently provide strong security, so combining encryption with steganography can substantially improve the overall protection of data in transit.

5. Limitations

Steganography is not foolproof; some detection techniques (e.g., statistical analysis of image pixel data) may reveal hidden information. Large image files or poor-quality steganographic embedding can result in a noticeable degradation of image quality.

6.3 RECOMMENDATION

Building upon the findings, the following recommendations are proposed to optimize the use of AES encryption and steganography for secure email transmission. These recommendations aim to improve the security, practicality, and efficiency of this approach by focusing on aspects like encryption key management, choosing appropriate image formats, and enhancing the resistance to steganographic detection. The goal is to ensure that this method remains both effective and resilient against emerging threats, enabling secure communication in increasingly complex digital environments.

1. **Ensure Strong Key Management**

   Use secure methods for exchanging and managing the AES key. Never send the key with the encrypted message. Consider using a secure key exchange protocol (e.g., Diffie-Hellman or public-key cryptography) to securely share the key between sender and recipient.

2. **Use High-Quality Images**

   To ensure that the steganographic process does not degrade the image quality, use high-resolution, lossless image formats (e.g., PNG or BMP). Larger images provide more room for embedding ciphertext without significantly altering the image's appearance.

3. **Improve Steganographic Methods**

   While LSB steganography is effective, consider exploring more advanced steganographic techniques that provide higher levels of security and resistance to detection, such as spreading the embedded data across multiple pixels or using frequency-domain steganography.

4. **Monitor and Mitigate Detection**

   Be aware that steganographic methods can sometimes be detected through advanced statistical analysis. Implement additional countermeasures, such as randomization of embedding locations or encryption of the image itself before embedding the ciphertext, to reduce the likelihood of detection.

5. **Regularly Update Security Protocols**

   Keep AES key management and encryption protocols updated. Ensure that cryptographic standards are regularly reviewed and follow the latest best practices to defend against emerging threats or vulnerabilities.

6. **Educate Users on Security Best Practices**

   Train users to recognize the importance of using strong encryption and steganography in protecting their messages. Emphasize the importance of safe key transmission, securing their devices, and the risks associated with sharing sensitive information over unsecured channels like regular email.

APPENDIX
Libraries used

```
from flask import Flask, request, render_template, send_file, redirect, url_for, flash
import os
from PIL import Image
import numpy as np
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import hashlib
import re
import base64
import mysql.connector
import uuid
from mysql.connector import Error
from datetime import datetime
```

Symmetric encryption (AES)

Encryption

```
def encrypt_message(message, key):
    cipher = AES.new(key, AES.MODE_ECB)
    padded_message = pad(message.encode('utf-8'))
    print(f"Padded message: {padded_message.hex()}")
    ciphertext = cipher.encrypt(padded_message)
    print(f"Encrypted ciphertext: {ciphertext.hex()}")
    return ciphertext
```

Decryption

```
def decrypt_message(ciphertext, key):
    try:
        if len(ciphertext) % 16 != 0:
            raise ValueError(f"Ciphertext length {len(ciphertext)} is not a multiple of 16 bytes")
        print(f"Decrypting ciphertext: {ciphertext.hex()}, length: {len(ciphertext)} bytes")
        cipher = AES.new(key, AES.MODE_ECB)
        padded_message = cipher.decrypt(ciphertext)
        print(f"Decrypted padded message: {padded_message.hex()}")
        plaintext = unpad(padded_message).decode('utf-8')
        print(f"Decrypted plaintext (raw): {plaintext}")
        return plaintext
    except ValueError as ve:
        print(f"Decryption error (ValueError): {ve}")
        return None
    except UnicodeDecodeError as ude:
        print(f"Decryption error (UnicodeDecodeError): {ude}")
        return None
    except Exception as e:
        print(f"Decryption error (Other): {e}")
        return None
```

Steganography (LSB)

Embedding message to an image

```
def embed_message(image_path, data):
    try:
        img = Image.open(image_path).convert('RGB')
        pixels = np.array(img)
        height, width, channels = pixels.shape
        print(f"Image dimensions: {width}x{height}, channels: {channels}")

        binary_data = ''.join(format(byte, '08b') for byte in data)
        data_length = len(data)
        length_binary = format(data_length, '032b')
        print(f"Embedding data length: {data_length} bytes, binary: {length_binary}")

        payload = length_binary + binary_data
        total_bits = len(payload)
        print(f"Total bits to embed (length + data): {total_bits}")
        print(f"Binary data to embed (first 64 bits): {binary_data[:64]}...")

        if total_bits > width * height * 3:
            raise ValueError(f"Data too large for image: {total_bits} bits vs {width * height * 3} available bits")

        bit_idx = 0
        for i in range(height):
            for j in range(width):
                for k in range(3):
                    if bit_idx < len(payload):
                        pixels[i, j, k] = (pixels[i, j, k] & ~1) | int(payload[bit_idx])
```

```
                        bit_idx += 1
                    else:
                        break

        print(f"Total bits embedded: {bit_idx}")
        stego_img = Image.fromarray(pixels)
        output_path = os.path.join(app.config['UPLOAD_FOLDER'], 'stego_image.png')
        stego_img.save(output_path, 'PNG')
        print(f"Stego image saved at: {output_path}")

        extracted = extract_message(output_path)
        print(f"Verification extracted data: {extracted.hex() if extracted else 'None'}")
        if extracted != data:
            print(f"Embedding verification failed! Embedded: {data.hex()}, Extracted: {extracted.hex() if extracted else '
        else:
            print("Embedding verification successful")

        return output_path
    except Exception as e:
        print(f"Embedding error: {e}")
        return None
```

Extraction of a message from an image

```python
def extract_message(stego_image_path):
    try:
        if not os.path.exists(stego_image_path):
            print(f"Image not found: {stego_image_path}")
            return None
        img = Image.open(stego_image_path).convert('RGB')
        pixels = np.array(img)
        height, width, channels = pixels.shape
        print(f"Extracting from image: {width}x{height}, channels: {channels}")

        binary_message = ''
        bit_idx = 0
        for i in range(height):
            for j in range(width):
                for k in range(3):
                    bit = (pixels[i, j, k] & 1)
                    binary_message += str(bit)
                    bit_idx += 1
                if bit_idx >= 32:
                    break
            if bit_idx >= 32:
                break

        length_binary = binary_message[:32]
        data_length = int(length_binary, 2)
        print(f"Extracted data length: {data_length} bytes, binary: {length_binary}")
```

```python
total_bits = data_length * 8
print(f"Expected bits to extract for data: {total_bits}")

binary_message = ''
bit_idx = 0
for i in range(height):
    for j in range(width):
        for k in range(3):
            if bit_idx < 32 + total_bits:
                bit = (pixels[i, j, k] & 1)
                if bit_idx >= 32:
                    binary_message += str(bit)
                bit_idx += 1
            else:
                break
        if bit_idx >= 32 + total_bits:
            break
    if bit_idx >= 32 + total_bits:
        break

print(f"Total bits extracted for data: {len(binary_message)}")
print(f"Extracted binary data (first 64 bits): {binary_message[:64]}...")
if len(binary_message) != total_bits:
    print(f"Error: Expected {total_bits} bits, but extracted {len(binary_message)} bits")

byte_message = [binary_message[i:i+8] for i in range(0, len(binary_message), 8)]
```

```python
        print(f"Extracted data: {data.hex()}, length: {len(data)} bytes")
        return data
    except Exception as e:
        print(f"Extraction error: {e}")
        return None
```

The image used to test (PNG format)

REFERENCES

Apoorva Arora, P. R. (2015). Image Security System using Encryption and Steganography. *International Journal of Innovative Research in Science, Engineering and Technology*, *04*(06), 3860–3869. https://doi.org/10.15680/IJIRSET.2015.0406008

Hambouz, A., Shaheen, Y., Manna, A., Al-Fayoumi, M., & Tedmori, S. (2019). Achieving Data Integrity and Confidentiality Using Image Steganography and Hashing Techniques. *2019 2nd International Conference on New Trends in Computing Sciences (ICTCS)*, 1–6. https://doi.org/10.1109/ICTCS.2019.8923060

Kulecho, D., & Barasa, A. O. (n.d.). *TITLE: IMAGE STEGANOGRAPHY USING THE ADVANCED ENCRYPTION STANDARD (AES) IN HEALTHCARE SYSTEMS*.

Manikandan, T., Muruganandham, A., Babuji, R., Nandalal, V., & Mazher Iqbal, J. (2021). Secure E-Health using Images Steganography. *Journal of Physics: Conference Series*, *1917*(1), 012016. https://doi.org/10.1088/1742-6596/1917/1/012016

Noida International University, Department of Computer Science and Engineering, NCR Delhi Noida, India, Sajid Ansari, A., Sajid Mohammadi, M., & Tanvir Parvez, M. (2019). A Comparative Study of Recent Steganography Techniques for Multiple Image Formats. *International Journal of Computer Network and Information Security*, *11*(1), 11–25. https://doi.org/10.5815/ijcnis.2019.01.02

Sachin, M., & Kumar, D. (2010). *Implementation and Analysis of AES, DES and Triple DES on GSM Network*.

Senior Professor, Computer Science and Engineering department in Galgotias University, Gautam Budhh Nagar, U. P, Sindhu, R., Singh, P., & Student B. Tech, Computer Science and Engineering department in Galgotias University, Gautam Budhh Nagar, U.P. (2020).

Information Hiding using Steganography. *International Journal of Engineering and Advanced Technology*, *9*(4), 1549–1554. https://doi.org/10.35940/ijeat.D8760.049420

Singh, A. K., & Singh, J. (n.d.). Steganography in Images Using LSB Technique. *International Journal of Latest Trends in Engineering and Technology*.

Thomas, M., & Panchami, V. (2015). An encryption protocol for end-to-end secure transmission of SMS. *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, 1–6. https://doi.org/10.1109/ICCPCT.2015.7159471

(Wambui R, Kulecho D & Barasa A.)