

CHAPTER NO.01

INTRODUCTION

1.1 INTRODUCTION TO IMAGE STEGANOGRAPHY

Stenography is an art and science of hiding secret messages within other non-secret media, such as images, audio files, or text, to avoid detection by unauthorized individuals. The image provided illustrates the process of steganography, which involves hiding private data within a cover object and securely transmitting it to a receiver. Let's break down the workflow and explain each step in detail: The cover object is the medium used to conceal the private data. This object could be an image, audio file, video, or any digital media that can hold additional information without raising suspicion. The private data represents the sensitive information that needs to be hidden. This could include text, images, or any other form of data that the sender wishes to transmit securely. The encoder is the system or algorithm responsible for embedding the private data into the cover object. It takes both the cover object and the private data as inputs. The encoder typically uses a steganographic algorithm to modify the cover object in such a way that the changes are imperceptible to the naked eye or other analysis techniques. The result is a stego-object, which looks similar to the original cover object but contains hidden information. The key is an optional but highly recommended component in steganography. It acts as a password or cryptographic key that ensures only authorized users can decode and access the hidden data. The key is used during both the encoding and decoding processes, adding an extra layer of security.

The diagram presented in Fig 1.1 illustrates a typical Steganography Workflow. After encoding, the stego-object (the cover object with the embedded private data) is transmitted over a communication channel. This could be any medium such as the internet, a physical device, or even a public platform. The transmission channel is where the risk of interception exists, so the hidden nature of the data is crucial. On the receiving end, the decoder is used to extract the hidden data from the stego-object. The decoder requires the same key used during the encoding process to successfully retrieve the private data. Without the correct key, the decoder would be unable to distinguish the hidden information from the cover object. Secret Transmitted Data

Upon successful decoding, the hidden private data is extracted from the stego-object. This secret transmitted data is then available for use by the receiver. Cover Object (Post-Decoding): After the private data has been extracted, the cover object remains. In some cases, the cover object may be returned to its original form, while in others, it might remain slightly altered, depending on the steganographic method used [1].

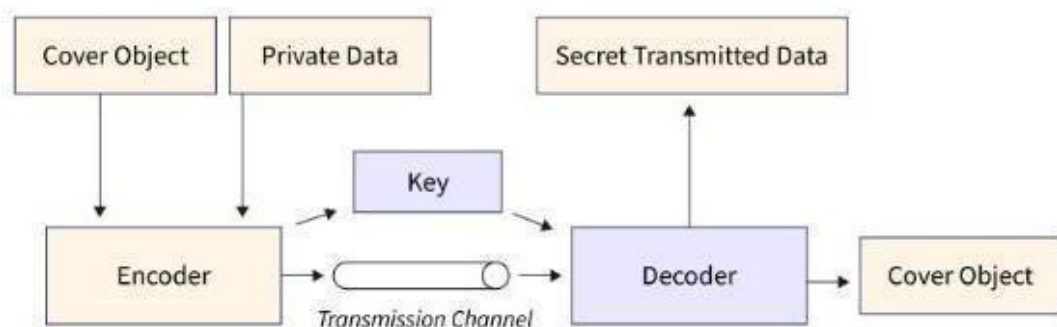


Figure 1.1: Image Steganography [31]

Steganography and cryptography are both vital techniques in the realm of information security, but they operate with distinct objectives. As illustrated in Table 1.1, steganography primarily aims to conceal the very existence of a message. It achieves this by embedding the message within another seemingly innocuous file such as an image, video, or audio file so that the message blends seamlessly into its carrier. This makes it virtually undetectable to anyone who is unaware of its presence, as there is no overt indication that a message is hidden. For example, a simple photograph may contain hidden data without raising any suspicion, ensuring that unintended observers have no reason to investigate further. In contrast, cryptography transforms the message into an unreadable format using encryption techniques. This means that even if the message is intercepted by someone during transmission, it remains incomprehensible without the proper decryption key. Cryptography protects the content by scrambling it into ciphertext, which can only be decoded by an authorized recipient with access to the correct key. Therefore, the focus of cryptography is on safeguarding the confidentiality of the information, even when its presence is known. Although steganography and cryptography differ in their approaches, combining these two techniques can offer enhanced security. By using steganography to hide the presence of a message and cryptography to encrypt its contents, the message becomes not only invisible but also unreadable without proper authorization. This dual layer of protection ensures that even if someone suspects the existence of hidden information, they still face the challenge of decrypting it. Consequently, integrating both methods can significantly bolster the protection of sensitive communications against unauthorized access.

Table No 1.1 : The Difference Between Steganography And Cryptography

Aspect	Steganography	Cryptography
Objective	Hide the existence of secret information within a cover medium (e.g., image, audio) without raising suspicion.	Securely transform and protect the content of the message from unauthorized access.
Main Purpose	Concealment and secrecy of the message itself.	Data protection, confidentiality, and security.
Visibility	Conceals the presence of the message.	Typically, the presence of encrypted data is evident, but the content is protected and unreadable without the key.
Key Role	Usually not directly related to steganography.	Essential for both encryption and decryption processes.
Technique	Embeds secret data within a cover medium (e.g., LSB encoding in images).	Mathematically transforms the original data into ciphertext using an algorithm and a secret key.
Reversibility	Can be reversible (original data can be extracted).	Often reversible (with the correct decryption key), but can also be designed for one-way functions (e.g., hashing).
Security Focus	Focuses on hiding the message's existence.	Focuses on securing the content of the message against unauthorized access or tampering.
Authentication	Typically does not provide authentication of the sender or recipient.	Can incorporate authentication through digital signatures.

The goal is to avoid drawing any attention to the fact that communication is happening, allowing the hidden message to go unnoticed by unintended observers. In contrast, cryptography transforms a message into an unreadable format through encryption. While cryptography does not hide the presence of the message, it ensures that even if the message is intercepted, it cannot be understood without the correct decryption key.

Illustrates the basic workflow of encryption and decryption, a fundamental process in securing digital communication. Let's delve into the working flow of this diagram and explain each step in detail. Plain Text: The process begins with plain text, which refers to the original, encrypted data. This could be any form of readable information, such as a message, document, or file, that needs to be secured before transmission. Encryption: Encryption is the process of converting plain text into an unreadable format, known as cipher text. This is achieved using an encryption algorithm, which systematically alters the data based on a specific encryption key. The purpose of encryption is to protect the content of the plain text from unauthorized access by making it unintelligible to anyone who does not possess the correct key.

Encryption Key: The encryption key is a crucial component in the encryption process. It is a piece of information—often a string of characters—used by the encryption algorithm to transform the plain text into cipher text. The security of the encryption largely depends on the strength and secrecy of this key. Without the correct key, decryption the cipher text back into readable plain text would be extremely difficult or practically impossible. Cipher Text: After encryption, the plain text is transformed into cipher text, which is the encrypted unreadable form of the original data. The cipher text can be safely transmitted over insecure channels, such as the internet or other communication networks, without revealing the original information to unauthorized parties. Transmission: The cipher text is transmitted through a communication channel, which could be any medium such as email, messaging systems, or file transfers. During transmission, the security of the data is maintained because the information remains encrypted and unreadable to anyone who might intercept it. Decryption: Decryption is the reverse process of encryption, where the cipher text is converted back into its original, readable form—plain text. This is done using a decryption algorithm that applies the corresponding decryption key to the cipher text. The decryption process restores the data to its original state, making it accessible to the intended recipient as shown in figure 1.2.

Figure 1.2 Illustrates the critical role of the decryption key in the decryption process, showing how it is used by a decryption algorithm to convert ciphertext back into its original, readable form, known as plain text. In encryption systems, the nature of the decryption key varies depending on the type of encryption being used. In symmetric encryption, the decryption key is identical to the encryption key. This means the same key that was used to encrypt the data is also used to decrypt it. The simplicity of using one key for both processes is efficient, but it requires secure management of the key, as both the sender and recipient must have access to the same secret key. On the other hand, asymmetric encryption operates differently. It uses two keys: a public encryption key and a private decryption key. These keys are mathematically related, but knowing the encryption key does not directly reveal the decryption key. In this system, the public key can be freely shared for encrypting data, while the private key remains secret and is used for decryption. The security of the decryption process, regardless of the type of encryption, hinges on keeping the decryption key confidential. If the decryption key is compromised, unauthorized users can gain access to the encrypted data. Once the decryption process is completed, the ciphertext is transformed back into plain text, returning the data to its original, human-readable format. The recipient can then access and use the information exactly as it was before encryption. This restored plain text contains all the original meaning and structure, allowing the intended recipient to fully interact with the data, ensuring secure and private communication between the sender and recipient [2].

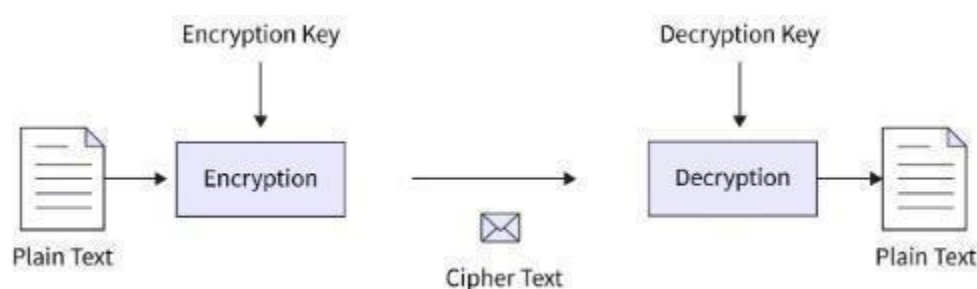


Figure 1.2: Cryptography [32]

1.2 Image Stenography Basics:

Steganography is both an art and a science that revolves around the concealment of secret messages within seemingly harmless, non-secret media like images, audio files, or text. Its primary goal is to keep the existence of the hidden message undetectable to unauthorized individuals. When steganography is applied to images, the process typically involves two fundamental operations: embedding and extraction. Embedding is the act of incorporating the secret message into the image by making subtle alterations to its pixel values. These changes are crafted in such a way that they are imperceptible to the human eye, meaning that the image looks nearly identical to its original form. The challenge here lies in embedding the hidden information while maintaining the image's visual integrity, ensuring that the modifications do not raise any suspicion or affect the quality of the image. The second operation, extraction, involves retrieving the hidden message from the altered image. This process is essentially the reverse of embedding, where the modifications made during embedding are decoded to reveal the secret message. Extraction must be performed precisely, as any errors in this process could result in the loss or corruption of the hidden information. To ensure successful retrieval, the original embedding procedure must be reversible without compromising the data's integrity. For image steganography to be effective, it requires a careful balance between concealment and data preservation.

The alterations to the image must be subtle enough to avoid detection by unauthorized individuals, yet significant enough to allow the secret message to remain intact and recoverable [3].

1.3 Encoding and Decoding Functions:

The core functionality of the application revolves around the encoding and decoding of messages within digital images as shown in figure 1.3. This functionality must be robust, reliable, and easy to use. Key considerations include [4].

The encoding process involves embedding a secret message into an image, modifying the image's pixels or bits in a way that is imperceptible to the naked eye. The application should offer a simple interface for users to select an image, input their message, and, if necessary, apply a secret key for added security. During this process, it is crucial to minimize any visible alterations to the image so that the embedding does not raise suspicion. The algorithm used for encoding must be efficient, embedding the message in such a way that it maintains the image's original quality while ensuring the hidden data can be easily retrieved later. Additionally, the system should support a variety of image formats (such as PNG, JPEG, and BMP) and handle different sizes of secret messages.

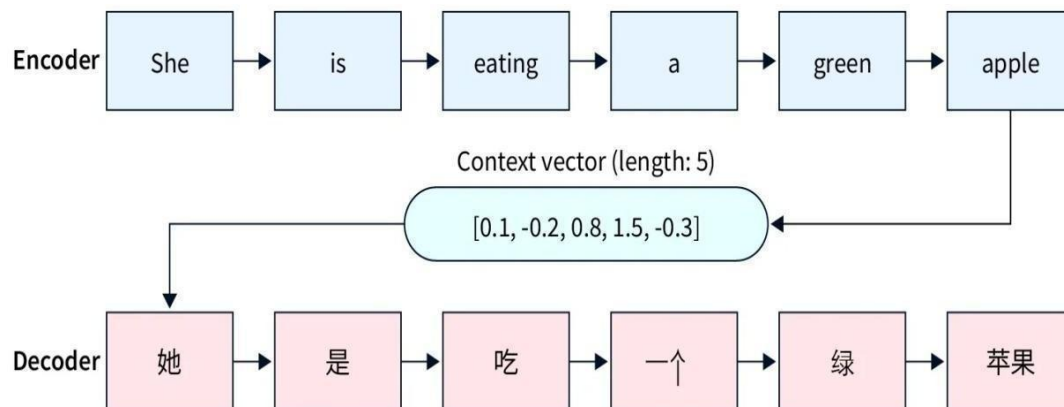


Figure 1.3: Encoding and Decoding [33]

1.3.1 Image Selection:

The application should allow users to select images from their local file system, supporting a wide variety of formats such as JPEG, PNG, and BMP. By accommodating these commonly used image formats, the application ensures flexibility, allowing users to work with the format that best suits their needs, whether it's for higher quality images (like PNG) or smaller file sizes (like JPEG). This compatibility with different formats broadens the application's appeal and usability across diverse scenarios, including cases where file size, quality, or format-specific requirements are important. To enhance the user experience, the application should implement a drag-and-drop functionality for selecting images. This feature allows users to easily drag an image from a folder or desktop and drop it directly into the application's interface, simplifying the process and making it more intuitive. Drag-and-drop functionality not only improves efficiency but also makes the image selection process more user-friendly, particularly for users who prefer quick and direct interactions with their files.

1.3.2 Message Input:

The application should feature a text box where users can easily enter the message they wish to encode within an image. This text box should be designed with sufficient flexibility to handle messages of various lengths, ensuring that users can input both short and moderately long messages without any difficulty. For ease of use, the text box should have a clear character limit indicator, informing users if their message is approaching the maximum allowable length based on the image's capacity to store the hidden data. To accommodate users who may wish to encode longer messages, the application should also provide an option to upload a message file instead of manually typing into the text box. This file upload feature could support text-based files such as `.txt` or `.docx`, enabling users to work with more extensive data without being constrained.

1.3.3 Encoding Options:

Offer various encoding techniques, such as Least Significant Bit (LSB) modification, Discrete Cosine Transform (DCT), or Spread Spectrum methods. Figure 1.4 shows and Allow users to select their preferred technique based on their needs and the image type. As mention in Table No 1.2 Provide settings for adjusting encoding parameters, such as the depth of modification or the strength of encoding, to balance between data capacity and image quality.

1.3.4 Decoding Options:

Figure 1.4 shows and Implement a straightforward process for decoding hidden messages. Users should be able to select an image from which they want to extract the hidden message and initiate the decoding process with minimal effort. If encryption is used, provide options for entering the decryption key or password as mention in Table No 1.2. Ensure that the decoding process handles various encryption methods and integrates seamlessly with the decryption process [5].

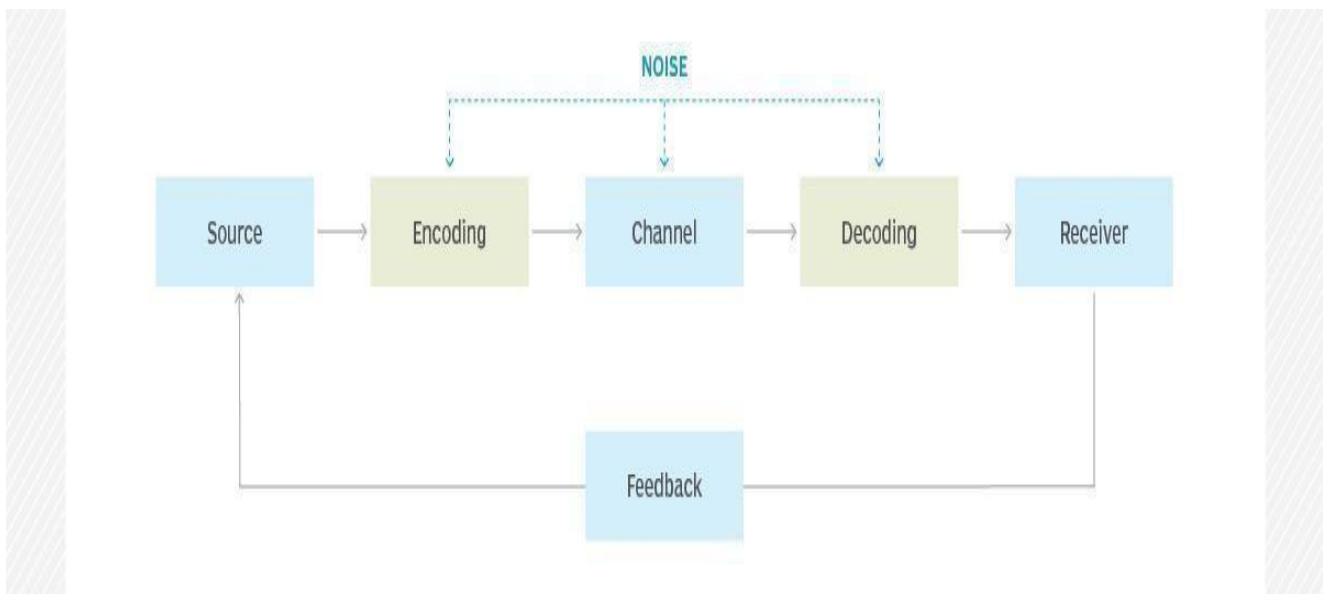


Figure 1.4: Encoding and Decoding Options [34]

Table No 1.2 : Functionality For Encoding and Decoding

Feature	Description
Image Selection	<ul style="list-style-type: none"> - Image Formats: Support for multiple image formats (e.g., JPEG, PNG, BMP) to accommodate user needs. - Selection Methods: Users can select images via drag-and-drop functionality or a file browser dialog. - User Convenience: Aims to make image selection easy and flexible, accommodating different user preferences.
Message Input	<ul style="list-style-type: none"> - Text Box: Provide a text box for users to input their message. - Message Length: Support reasonable message lengths, with an option to upload longer message files. - Message Formatting: Offer options for trimming white-space or converting text to a specific encoding format to ensure proper message preparation before encoding.
Encoding Options	<ul style="list-style-type: none"> - Techniques Offered: Users can choose from different encoding techniques like Least Significant Bit (LSB) modification, Discrete Cosine Transform (DCT), or Spread Spectrum methods. - Customization: Allow adjustments to encoding parameters, such as modification depth or encoding strength, to balance between data capacity and image quality.
Decoding Options	<ul style="list-style-type: none"> - Decoding Process: Provide a straightforward way to select an image and decode hidden messages. - Decryption Support: Include options for entering a decryption key or password if encryption is used.

The encoding process begins with the user selecting an image from their local system. The application should support multiple file formats (e.g., JPEG, PNG, BMP) and offer a simple drag- and-drop option or a file browser dialog for easy image selection. Users then input the message they want to encode into a provided text box. For longer messages, the application should allow uploading a text file. The message might be pre-processed, such as trimming white space or converting it to a specific encoding format. DCT or Frequency Domain Methods: In these advanced techniques, the message is embedded into the image's frequency components.

1.4 Problem Statement

In an era where digital communication is ubiquitous, the need for secure and private communication channels is paramount. Traditional encryption methods, while effective in protecting the contents of a message, often signal the presence of sensitive information, potentially attracting unwanted attention. Stenography offers an additional layer of security by obscuring the existence of the message itself. However, developing a stenographic method that is both secure and undetectable presents significant challenges. This thesis addresses these challenges by proposing a robust algorithm for image stenography, aimed at enhancing the security and unpredictability of hidden messages.

1.5 Objective

- Ensure the security of hidden messages through encryption technique.
- Create a user-friendly interface for easy message embedding and extraction.
- Evaluate system performance, security, and usability comprehensively.
- Improve the system's ability to handle changes to images without messing up the hidden message inside.

1.6 Structure of the Thesis

Chapter 1, Introduction, offers a foundational understanding of stenography, including its definition, history, and importance in modern communication. It outlines the problem statement, the need for undetectable communication methods, and the limitations of traditional encryption,

while setting the research objectives and discussing the significance of the study. This chapter concludes with an overview of the thesis structure.

Chapter 2, Literature Review, explores the historical background of stenography, its evolution, and existing techniques such as LSB, DCT, and Spread Spectrum. It reviews significant research work, highlighting innovations and limitations, and identifies gaps in the current research landscape.

Chapter 3, Methodology, details the theoretical framework underlying stenography, describes the proposed algorithm including the compression, encryption, and embedding processes, and justifies the choice of tools and technologies. It also outlines the step-by-step implementation plan with milestones and timelines.

Chapter 4, Design, Development, and Implementation, combines the design and development phases with implementation details. It covers the system architecture, user interface design, encoding and decoding algorithms, development environment, coding practices, integration processes, and testing procedures, and addresses any challenges encountered during implementation.

Chapter 5, Results, Analysis, and Discussion, presents the research findings, evaluating the encoding performance, decoding accuracy, and security analysis. It includes user feedback, discusses insights gained, explores the implications for real-world applications, identifies limitations, and provides recommendations for future research and practical use.

CHAPTER NO.02

LITERATURE REVIEW

2.1 Introduction

Image steganography, the practice of hiding information within digital images, has become a key focus in information security. Unlike cryptography, steganography conceals the existence of the message itself. This literature review explores the evolution of image steganography, examining various techniques, countermeasures, and applications. It also identifies gaps in existing research, setting the foundation for the contributions of this thesis.

2.2 Historical Background

Stenography, a term often intertwined with stenography, refers to the ancient and enduring practice of concealing information within another, more innocuous medium. This practice has a rich history that spans millennia, evolving from rudimentary methods in ancient civilizations to sophisticated techniques in the digital age. The need to transmit confidential information securely, whether for military, political, or personal purposes, has driven the development of stenography across different cultures and eras [6].

2.3 Early Methods of Stenography

The early methods of steganography are as diverse as the civilizations that utilized them, each developing distinctive techniques to obscure messages and protect communication. In ancient Greece, for example, historians record the use of hidden messages written on the wooden tablets that were then coated with wax. This method allowed the message to be concealed under a layer of wax, which could be scraped off and replaced with a new message if needed. The Greeks also employed the use of invisible ink, a technique where messages were written with substances.

2.3.1 Ancient Greece and Rome:

One of the earliest known instances of stenography was recorded by the Greek historian Herodotus. In ancient Greece, a particularly ingenious method involved shaving the head of a slave, tattooing a message on the scalp, and waiting for the hair to grow back before sending the slave off to deliver the message. This method ensured that the message remained hidden until the recipient shaved the slave's head again to reveal the text. Similarly, in Rome, methods of secret communication were crucial for military strategies and political maneuvers. The use of simple substitutions or physical obfuscation of text allowed messages to be hidden from enemy eyes [7].

2.3.2 Wooden Tablets and Wax:

Another early form of steganography involved using wooden tablets coated with wax, a method that was especially practical in times when paper was scarce. In this technique, a message was inscribed directly onto the wooden surface of the tablet. Once the message was written, the tablet was covered with a fresh layer of wax. This new wax layer concealed the original inscription, giving the appearance of an unused, blank tablet. This method was highly effective for the period, as it allowed messages to be hidden in plain sight, appearing to be ordinary objects. The concealment process was straightforward: the wax layer provided a clean surface that masked the underlying message, which would otherwise be visible. This method capitalized on the fact that wax was readily available and could be easily manipulated. When it was necessary to reveal the hidden message, the wax layer could be scraped away, exposing the original inscription beneath. This allowed for the message to be read without the need for complex tools or methods. The simplicity and effectiveness of this method made it an ideal solution in an era when alternative means of secure communication were limited. The wax-covered wooden tablet served as a practical medium for steganography, combining ease of use with effective concealment [8][9].

2.3.3 Invisible Ink:

The concept of invisible ink, which has played a significant role in covert communication throughout history, involves using substances that are not visible under normal conditions but can be revealed through specific processes. One of the earliest known forms of invisible ink involved lemon juice. When applied to paper, lemon juice dries clear, making it appear as if the paper is blank. However, when exposed to heat, the citric acid in the lemon juice reacts with the paper, causing the ink to turn brown and reveal the hidden message. This method was particularly useful in times when paper was not widely available or when messages needed to be concealed discreetly. During the Renaissance, invisible ink gained prominence among scholars, diplomats, and spies. The technique allowed for secret correspondence without arousing suspicion, which was crucial in an era marked by intense political and espionage activities. As chemistry advanced, so did the sophistication of invisible inks. Researchers discovered new substances that could be used to create inks visible only under certain conditions, such as ultraviolet light or chemical reagents. This advancement made invisible ink an even more effective tool for securing sensitive information. The strategic importance of invisible ink became particularly evident during the American Revolutionary War and World War II. Spies and military personnel used invisible ink to transmit critical intelligence securely [10].

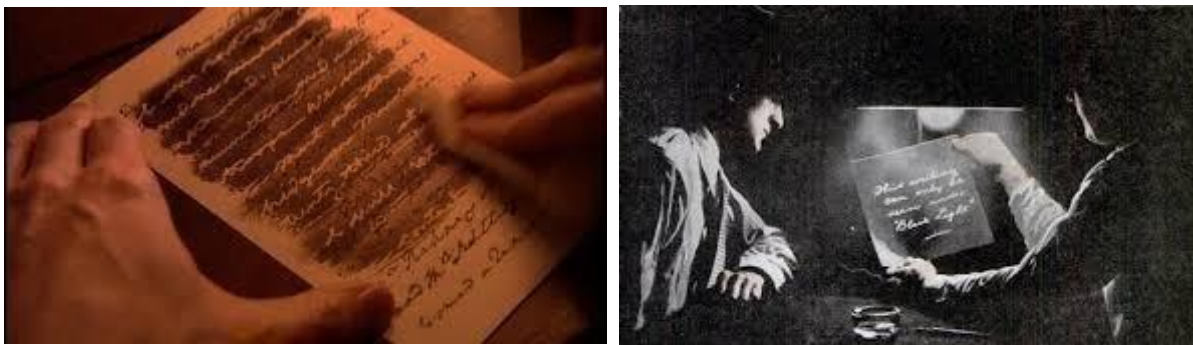


Figure 2.1: Invisible Ink [35]

2.3.4 Cipher Codes:

As Mention Table No 2.1 that While cryptography focuses more on the scrambling of messages, simple ciphers and codes have historically been a part of stenographic practices as well. Early users would embed a coded message within an otherwise normal text. Only the intended recipient, who knew the key or method to decode the message, could extract the hidden information as shown in figure 2.2. This blend of cryptography and stenography allowed for multiple layers of security, ensuring that even if the message were intercepted, it would be difficult to decipher without the correct key.

Letter	Code	Letter	Code	Letter	Code
A	AAAAA	I/J	ABAAA	R	BAAAA
B	AAAAB	K	ABAAB	S	BAAAB
C	AAABA	L	ABABA	T	BAABA
D	AAABB	M	ABABB	U/V	BAABB
E	AABAA	N	ABAAB	W	BABAA
F	AABAB	O	ABABA	X	BABAB
G	AABBA	P	ABBSA	Y	BABBA
H	AABBB	Q	ABBBB	Z	BABBB

Letter	Code	Letter	Code	Letter	Code
A	00000	I/J	01000	R	10000
B	00001	K	01001	S	10001
C	00010	L	01010	T	10010
D	00011	M	01011	U/V	10011
E	00100	N	01100	W	10100
F	00101	O	01101	X	10101
G	00110	P	01110	Y	10110
H	00111	Q	01111	Z	10111

Figure 2.2: Cipher Codes [36]

Cipher codes are systematic methods used to transform readable information, or plaintext, into an unreadable format, known as ciphertext, to ensure the confidentiality of the message. Historically, simple substitution ciphers like the Caesar cipher involved shifting each letter in the plaintext by a fixed number of places in the alphabet, making the message difficult to decipher without the shift key. More complex methods, such as the Vigenère cipher, use a keyword to vary the shift for each letter, adding a layer of security. Transposition ciphers, such as the Rail Fence and Columnar Transposition ciphers, rearrange the letters of the plaintext in different patterns to create ciphertext. In modern cryptography, symmetric key ciphers like AES [11].

Table No 2.1 :Methods of Stenography

Section	Explanation	Example
Ancient Greece and Rome	Hidden messages through physical methods, like tattooing on a slave's scalp or simple text substitution.	Tattoo on a slave's scalp hidden by hair growth; Caesar cipher shifting letters in a message.
Wooden Tablets and Wax	Messages written on wooden tablets and concealed under a layer of wax.	A message inscribed on a tablet and covered with wax, appearing blank until the wax is scraped off.
Invisible Ink	Writing messages with substances like lemon juice that become visible only when heated.	A spy writes a message with lemon juice; the message appears when the paper is held near a flame.
Cipher Codes	Embedding coded messages within normal text that only the recipient can decode.	A letter where every third letter spells out a hidden word, like "The eagle flies at midnight" hides "team."

In ancient Greece and Rome, steganography employed various physical methods to conceal messages. In Greece, one method involved tattooing a message onto the scalp of a slave, whose hair would then grow over the tattoo, hiding the message from casual observers. Only those who knew to look for the hidden message and had access to the slave could reveal it. Similarly, the Caesar cipher was a prevalent technique during this period, where each letter in the plaintext was shifted by a fixed number of places in the alphabet, making the message appear scrambled to anyone unaware of the shift. In addition to these methods, the use of wooden tablets and wax was a common practice. Messages were inscribed on wooden tablets, and a layer of wax was applied over the writing. To an observer, the tablet appeared blank, but the underlying message could be uncovered by scraping away the wax, revealing the inscription beneath. Invisible ink, another significant advancement, involved writing messages with substances.

2.4 Existing Stenography Techniques

Stenography has evolved significantly with the advent of digital technology, resulting in a variety of sophisticated methods that enable the concealment of information within different types of digital media. These techniques vary in complexity, robustness, and application, catering to different needs ranging from simple concealment to highly secure, resilient forms of hidden communication. This section provides a comprehensive overview of some of the most commonly used stenography techniques in the digital age [12].

2.4.1 Least Significant Bit (LSB) Embedding:

Least Significant Bit (LSB) embedding is a fundamental technique in digital stenography, known for its simplicity and widespread use. This method works by modifying the least significant bits of the pixels in an image to embed a secret message. In an 8-bit color image, each pixel consists of three bytes, corresponding to the red, green, and blue color channels as shown in figure 2.3. By altering the LSBs of these bytes, the embedded data does not significantly alter the visual quality of the image, making the changes nearly imperceptible to the human eye. The process of embedding a message using LSB involves several steps. Initially, the message to be hidden is converted into binary form. For example, the text "HELLO" becomes a series of binary digits. Pixels in the image are then selected for embedding, and each pixel's color channels (R, G, B) are used to embed bits of the message. The LSB of each color channel in these selected pixels is modified to include the bits of the hidden message. Since the LSB affects the pixel's color value only minimally, these changes are generally imperceptible to the human eye. After embedding the entire message, the image is saved and appears almost identical to the original, with only minor, unnoticeable alterations. To extract the hidden message, the image is analyzed to extract the LSBs from each color channel of the selected pixels.

The primary advantage of LSB embedding lies in its efficiency; it allows for the embedding of a relatively large amount of data with minimal computational resources and without the need for complex algorithms. However, the technique has notable limitations, particularly its vulnerability to various attacks such as image compression, noise addition, and other forms of image manipulation, which can easily corrupt or destroy the hidden message as shown in figure 2.3. Due to these weaknesses, LSB embedding is considered less secure and less robust compared to more advanced steganographic techniques. LSB embedding has notable advantages, including its subtlety, as the changes made to the LSBs are minimal and do not typically alter the image or audio quality significantly. It also allows for a significant amount of data to be hidden within a digital file, as multiple bits can be embedded across many pixels or audio samples. Moreover, LSB embedding is versatile and can be applied to various types of digital files, including images, audio, and video [13].

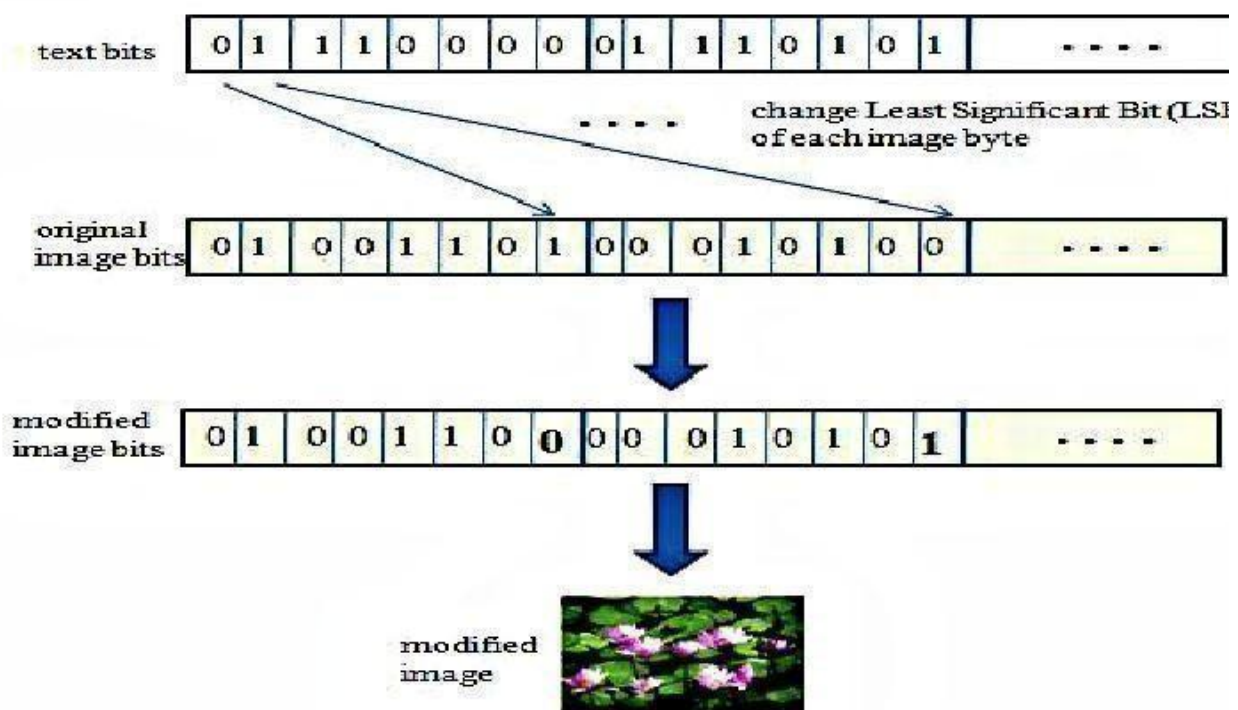


Figure 2.3: Least Significant Bit [37]

2.4.2 Discrete Cosine Transform (DCT):

The Discrete Cosine Transform (DCT) is a powerful mathematical technique widely used in digital signal processing and image compression. It transforms a signal or image from the spatial domain into the frequency domain, effectively breaking it down into a sum of cosine functions oscillating at various frequencies. This transformation is particularly valuable in the context of image and audio compression due to its ability to concentrate energy into a few significant coefficients, which simplifies data representation and reduces file size. In detail, the DCT operates by converting an image or signal into a set of coefficients that represent different frequency components. The process begins with dividing the image into small blocks, typically 8x8 pixels for images. Each block is then subjected to the DCT, which calculates the contribution of each frequency component within that block. The resulting coefficients indicate how much of each frequency is present in the block. These coefficients are organized such that lower-frequency components, which often contain the most critical information about the image or signal, are placed in the top-left corner of the coefficient matrix, while higher-frequency components are placed in the bottom-right. The key advantage of the DCT is its ability to separate an image into its constituent frequencies, with the majority of the image's visually significant information often concentrated in the lower frequencies. This characteristic is leveraged in image compression techniques such as JPEG, where the DCT coefficients are quantized and encoded to reduce redundancy and compress the image. Quantization involves reducing the precision of the DCT coefficients, particularly those corresponding to higher frequencies that are less perceptible to the human eye, thereby further decreasing file size. Once the DCT coefficients are quantized, they are typically encoded using techniques such as Run-Length Encoding (RLE) or Huffman coding, which further compresses the data by exploiting patterns and redundancies as shown in figure 2.4.

However, The compressed data is then stored or transmitted, and can be decompressed later using the inverse Discrete Cosine Transform (IDCT) to reconstruct the image or signal. The IDCT reverses the process, converting the frequency domain representation back into the spatial domain. In steganography, the DCT can be employed to embed secret messages within images by modifying the DCT coefficients of certain image blocks. This method leverages the fact that slight alterations to the DCT coefficients, particularly in the higher frequencies, are less perceptible to the human eye and can therefore be used to encode hidden data without significantly affecting the visual quality of the image. The encoded data can be extracted by performing the DCT on the modified image, retrieving the altered coefficients, and decoding the embedded information. Overall, the Discrete Cosine Transform is a fundamental tool in digital image processing and compression, offering a robust means of analyzing and modifying images and signals. Its efficiency in concentrating important information into a few significant coefficients makes it ideal for both data compression and steganographic applications, where maintaining high visual quality while embedding hidden information is crucial.

DCT-based steganography is more complex to implement compared to simpler methods like Least Significant Bit (LSB) embedding [14]. It requires a thorough understanding of the compression process and the effects of modifying DCT coefficients as shown in figure 2.4.

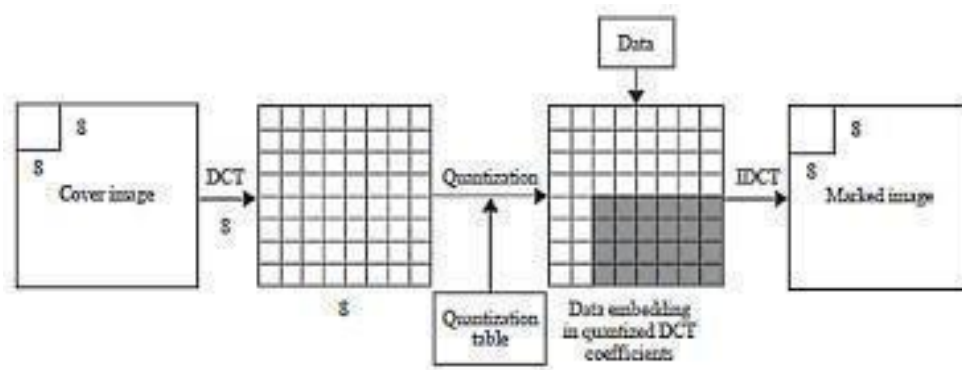


Figure 2.4: Discrete Cosine Transform [38]

2.4.3 Spread Spectrum:

Spread Spectrum is an advanced communication technique originally developed for enhancing the robustness of wireless signals, but it has also found applications in digital steganography. The core idea behind Spread Spectrum is to spread the signal's energy over a wider bandwidth than the minimum required, which helps to reduce interference, avoid detection, and improve signal robustness against eavesdropping and jamming. In more detail, the Spread Spectrum technique involves modulating the original signal with a spreading code that varies rapidly, thereby expanding its frequency range. This spreading process transforms the signal into a form that occupies a broader spectrum of frequencies than the original signal. The key benefits of this technique include increased resistance to signal interference and a higher level of security, as the signal appears as noise over the spectrum and is harder for unauthorized parties to detect and intercept. There are two primary types of Spread Spectrum techniques: Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). In FHSS, the signal rapidly hops between different frequency channels according to a pseudorandom sequence, effectively spreading the signal across various frequencies. This hopping sequence is synchronized between the transmitter and receiver, allowing for reliable communication while minimizing the impact of interference or jamming on any single frequency. In DSSS, the signal is spread by multiplying it with a high-rate pseudorandom noise (PN) sequence, which increases its bandwidth. This spreading sequence, known as a chipping code, modulates the data signal, expanding its frequency range. The resulting spread signal appears as noise to anyone not knowing the PN sequence, thus enhancing security and making it challenging for unintended recipients to decode the message without the correct sequence. In digital steganography, Spread Spectrum techniques are employed to embed hidden information within a digital file, such as an image or audio, by modulating the data in a way that disperses the hidden message [15].

2.4.4 Phase Coding:

Phase Coding is a sophisticated technique used in digital steganography and communication systems to embed hidden information by manipulating the phase of a signal. This method takes advantage of the phase component of a signal's waveform, which represents the position of the wave relative to a reference point in time. By altering the phase of a signal, hidden data can be encoded in a way that is often imperceptible to the human eye or ear, while still being retrievable by someone with the appropriate decoding method. The Phase Coding process involves several detailed steps. Initially, the original signal or image is divided into small segments or blocks, each of which will be individually processed. For images, this might involve dividing the image into smaller pixel blocks, while for audio, it could mean breaking the audio into short segments. Within each segment, the phase of the signal is adjusted according to the data to be hidden. This adjustment is typically done using a phase shift, which is a change in the phase angle of the waveform. To embed the hidden message, the data is first encoded into phase shifts. For instance, binary data can be represented as discrete phase shifts: a bit of "0" might correspond to a phase shift of 0 degrees, while a bit of "1" might correspond to a phase shift of 180 degrees. These phase shifts are then applied to the selected segments of the signal or image. The changes in phase are made small enough so that they do not significantly alter the overall appearance of the image or the quality of the audio, maintaining the integrity of the original content. After embedding, the modified signal or image is reconstructed and can be stored or transmitted. The key to successful Phase Coding is the precision with which the phase adjustments are made, ensuring that the hidden data is not easily detectable and does not introduce noticeable distortions. The hidden data is retrieved by performing the inverse operation: analyzing the phase shifts in the signal or image to decode the embedded message. This involves comparing the phase of the processed segments against a reference to determine the original data values [16].

Table No 2.2 : Existing Stenography Techniques:

Technique	Explanation	Example/Key Points
Least Significant Bit (LSB) Embedding	Embeds a secret message by modifying the least significant bits of an image's pixels. Simple and widely used, but vulnerable to image manipulations.	<ul style="list-style-type: none">- Changes are nearly imperceptible to the human eye.- Vulnerable to compression and noise attacks.
Discrete Cosine Transform (DCT)	Modifies DCT coefficients during JPEG compression to hide information. Robust against image processing but more complex than LSB.	<ul style="list-style-type: none">- Works in the frequency domain.- Resistant to compression, cropping, and resizing.
Spread Spectrum	Distributes hidden information across a wide range of frequencies or bandwidths, enhancing concealment and resilience to interference.	<ul style="list-style-type: none">- Robust against noise and image processing.- Lower embedding capacity due to data spread.
Phase Coding	Used in audio stenography; modifies the phase of an audio signal to embed data without affecting audio quality.	<ul style="list-style-type: none">- Effective in concealing information in audio.- Complex implementation, limited by audio signal characteristics

Table No 2.2 provides an overview of various steganography techniques and their characteristics. Least Significant Bit (LSB) Embedding is a straight forward method where secret messages are hidden by adjusting the least significant bits of an image's pixels. This technique is favored for its simplicity and subtlety, but it is vulnerable to alterations from image compression and noise. The Discrete Cosine Transform (DCT) approach embeds data by altering the DCT coefficients in JPEG-compressed images, making it robust against common image processing operations such as resizing and cropping. Despite its robustness, DCT embedding is more complex than LSB and requires careful implementation. Spread Spectrum techniques distribute the hidden data over a wide frequency range. This method typically results in a reduced capacity for data embedding due to the spreading of information. Phase Coding is specifically used in audio steganography, where the phase of audio signals is modified to conceal data without significantly affecting the audio quality. While effective in maintaining audio fidelity, Phase Coding presents challenges in terms of implementation complexity and is constrained by the properties of the audio.

2.5 Comparative Analysis- Strengths and Weaknesses

In the realm of digital stenography, various techniques offer different approaches to hiding information within digital media. Each method has its own set of strengths and weaknesses, making it suitable for specific applications depending on factors like security, data capacity, robustness, and complexity. This section provides a comparative analysis of the most common stenography techniques discussed earlier: Least Significant Bit (LSB) Embedding, Discrete Cosine Transform (DCT), Spread Spectrum, Phase Coding, and Transform Domain Techniques [17].

2.5.1 Least Significant Bit (LSB) Embedding:

Least Significant Bit (LSB) Embedding is a widely used technique in digital steganography that involves hiding secret information within the least significant bits of a digital file, such as an image, audio, or video. This method leverages the fact that the least significant bits of a digital file contribute minimally to the overall value of a pixel or sample, making changes to these bits less noticeable to the human eye or ear. In the context of images, an 8-bit color image uses three color channels for each pixel—Red, Green, and Blue (RGB)—each represented by 8 bits. The LSB of each color channel is the bit with the smallest value, and modifying it results in only a minor change to the pixel's color. The embedding process begins by converting the secret message into a binary format. Each bit of this message is then embedded into the LSB of the corresponding color channels in selected pixels of the image. Since these changes are minimal, the image retains its visual appearance while concealing the hidden data. The embedding process involves several steps: first, selecting the pixels in which the message will be embedded; next, modifying the LSBs of the chosen pixels to encode the message bits; and finally, reconstructing and saving the image with the embedded message.

2.5.2 Discrete Cosine Transform (DCT):

DCT-based steganography is robust against many image processing operations, especially those involving compression, due to its operation in the frequency domain. This method provides greater resilience compared to spatial domain techniques like LSB. However, its complexity requires a deep understanding of image processing, and the data embedding capacity is generally lower than LSB. There is also a risk of introducing visible artifacts if not applied carefully.

2.5.3 Spread Spectrum

This technique offers high robustness against noise and interference by distributing hidden data across a wide frequency range, making it difficult to detect or degrade. Despite its advantages, Spread Spectrum is complex to implement and results in lower data capacity as the data is spread thinly across the medium.

2.5.4 Phase Coding

Phase Coding is effective for audio steganography, as it hides data in phase components, as mentioned in Table No 2.3, which are less perceptible to the human ear compared to amplitude changes. This method is both discreet and efficient but requires careful phase management and has limited data capacity due to the need to avoid perceptible changes in the audio signal. Phase Coding is a sophisticated technique used in audio steganography that involves embedding data by manipulating the phase of an audio signal rather than its amplitude. This method takes advantage of the fact that phase changes are generally less noticeable to the human ear compared to amplitude changes, making it a discreet and effective approach for hiding information. In Phase Coding, the audio signal is divided into small segments or blocks, and each block's phase component is adjusted to encode the hidden data. These phase adjustments are made in such a way that they do not significantly alter the perceived quality of the audio. For instance, small phase shifts can be applied to encode binary data, where a bit of "0".

2.6 Comparative Analysis Table

The following table provides a summarized comparison of the techniques discussed, highlighting key attributes such as complexity, robustness, data capacity, and typical use cases.

Table No 2.3: Comparative Analysis Table

Technique	Complexity	Robustness	Data Capacity	Typical Use Cases	Vulnerabilities
Least Significant Bit (LSB)	Low	Low	High	Simple image steganography	Compression, noise, statistical analysis
Discrete Cosine Transform (DCT)	Medium	High	Medium	JPEG image steganography, watermarking	Implementation complexity, limited capacity
Spread Spectrum	High	High	Low	Secure communication, noise-resilient transmission	Complexity, lower data capacity
Phase Coding	Medium	Medium	Low	Audio steganography, covert communications	Implementation complexity, limited data capacity
Transform Domain (e.g., DWT)	High	High	Low to Medium	Advanced image/audio steganography, secure watermarking	Complexity, computational intensity
Least Significant Bit (LSB)	Low	Low	High	Simple image steganography	Compression, noise, statistical analysis

Steganography techniques vary significantly in terms of complexity, robustness, data capacity, typical use cases, and vulnerabilities. LSB Embedding is simple and has high data capacity but is vulnerable to compression, noise, and statistical analysis, making it less robust. Discrete Cosine Transform offers medium complexity and high robustness, ideal for JPEG image steganography and watermarking, but it faces challenges with implementation complexity and limited data capacity. Spread Spectrum provides high robustness and is used for secure communication and noise-resilient transmission, though it is complex and has a lower data capacity due to extensive information spreading. Phase Coding is effective for audio steganography with medium complexity and robustness, but it has a low data capacity and complex implementation.

2.7 Review of Significant Research Papers

The field of steganography has been significantly advanced through various research papers that have explored diverse techniques for embedding data within digital media. These studies have contributed to refining methods by addressing issues of security, detectability, and resilience against attacks. This section reviews several influential papers that have shaped the development of steganographic techniques.

2.7.1 Enhancing Security with Compression and Encryption

In their 2011 paper, Rosziati Brahmin and Teoh Suk Kuan introduced a method that enhances the security of Least Significant Bit (LSB) embedding through a combination of compression and encryption. Their approach involves compressing the secret message to reduce its size and obfuscate it, thereby making it more challenging for attackers to detect. The compressed data is then encrypted before being embedded into the host media using LSB. This dual-layer strategy mitigates some of LSB embedding's inherent vulnerabilities, such as susceptibility to statistical analysis and detection.

2.7.2 Watermarking in the Frequency Domain:

In their seminal work, "Digital Watermarking," Cox, Miller, and Bloom explored the application of frequency domain techniques like Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) for watermarking in digital images and videos. Their research underscored the robustness of these techniques against common image processing operations, such as compression and scaling, making them suitable for digital rights management (DRM). This foundational work highlighted the trade-offs between security, data capacity, and imperceptibility, paving the way for further research in transform domain steganography. As Mention Table No 2.4.

2.7.3 Detecting LSB Steganography:

The paper "Reliable Detection of LSB Steganography in Color and Grayscale Images," authored by Fridrich, Goljan, and Du, is a landmark in the field of steganographic detection. This research developed methods for identifying LSB steganography by analyzing the statistical properties of images to detect anomalies indicative of hidden data. Their work has been instrumental in understanding the vulnerabilities of LSB embedding and has spurred advancements in creating more secure and less detectable steganographic techniques.

2.7.4 Robustness of Spread Spectrum Techniques:

As Mention Table No 2.4, M. Ramkumar and A. Akansu's 2001 paper, "A Robust Digital Watermarking Scheme Using Spread Spectrum and DCT," examined the application of Spread Spectrum techniques to steganography, particularly in audio and video files. Their research demonstrated that distributing hidden information across a broad bandwidth enhances the resilience of the data against noise and interference. This approach has been a significant contribution to secure communication systems and digital watermarking by offering robust protection against various attacks.

2.7.5 Use of Wavelet Transforms:

Barni, Bartolini, and Piva made substantial advances in steganography through their exploration of Discrete Wavelet Transform (DWT) for embedding information. In their study, "A DWT-Based Technique for Spatio-Frequency Localization of Video Watermarks," they focused on embedding data in less noticeable components of the signal, which enhances robustness while preserving the quality of the host media. Their work has been pivotal in developing advanced steganographic techniques that balance security and imperceptibility effectively.

Table No 2.4: Review of Significant Research Papers

Research Paper	Focus Area	Key Contributions
Enhancing Security with Compression and Encryption	LSB embedding combined with compression and encryption.	<ul style="list-style-type: none">- Improved security by compressing and encrypting the secret message before embedding.- Reduced detectability
Watermarking in the Frequency Domain	Application of DCT and DWT for digital watermarking.	<ul style="list-style-type: none">- Demonstrated robustness against image processing operations.- Highlighted trade-offs in transform domain steganography.
Detecting LSB Steganography	Detection of LSB steganography through statistical analysis.	<ul style="list-style-type: none">- Developed methods for identifying hidden data in images.
Detecting LSB Steganography	Detection of LSB steganography through statistical analysis.	<ul style="list-style-type: none">- Developed methods for identifying hidden data in images.- Influenced the creation of more secure steganographic techniques.
Robustness of Spread Spectrum Techniques	Use of Spread Spectrum and DCT in audio and video steganography.	<ul style="list-style-type: none">- Enhanced resilience of hidden data against noise and interference.- Contributed to secure communication systems.

Table No 2.4 provides an overview of significant research papers in the field of steganography. The paper on Enhancing Security with Compression and Encryption explores the combination of LSB embedding with compression and encryption, which significantly improves the security of hidden messages by reducing detectability through these additional layers of protection. The research on Watermarking in the Frequency Domain focuses on applying DCT and DWT for digital watermarking, demonstrating how these transform domain techniques offer robustness against various image processing operations while also highlighting the inherent trade-offs in these methods. The study on Detecting LSB addresses the challenge of uncovering hidden data in images through statistical analysis, contributing to the development of more secure.

CHAPTER NO.03

RESEARCH METHODOLOGY

3.1 Introduction

This chapter elaborates on the research methodology used to design and implement a novel steganography algorithm as depicted in the workflow shown in Figure 3.1. The methodology includes the steps for data embedding (hiding) and retrieving processes, which form the core of the proposed security system. This system integrates multiple layers of security to ensure the safe and confidential transmission of sensitive information. This chapter details the research methodology employed in designing and implementing a novel steganography algorithm. The methodology encompasses a comprehensive workflow that addresses both the data embedding (hiding) and retrieving (extraction) processes, which are central to the proposed security system. The embedding process involves several key steps: first, selecting appropriate carrier media (such as images or audio files), then encoding the secret data into the carrier through specialized algorithms that minimize perceptible changes to the media. This may involve techniques like Least Significant Bit (LSB) embedding or more advanced methods like Discrete Cosine Transform (DCT). Following the embedding, the system integrates multiple layers of security to enhance data confidentiality. These layers might include encryption of the hidden message, using robust algorithms to ensure that even if the embedding process is detected, the data remains protected. The retrieval process involves extracting the hidden data from the carrier medium, where the system applies decoding techniques to reverse the embedding process and recover the original message. The methodology emphasizes a multi-faceted approach to security, combining data hiding techniques with encryption and possibly other protective measures to ensure that the sensitive information remains secure throughout its transmission.

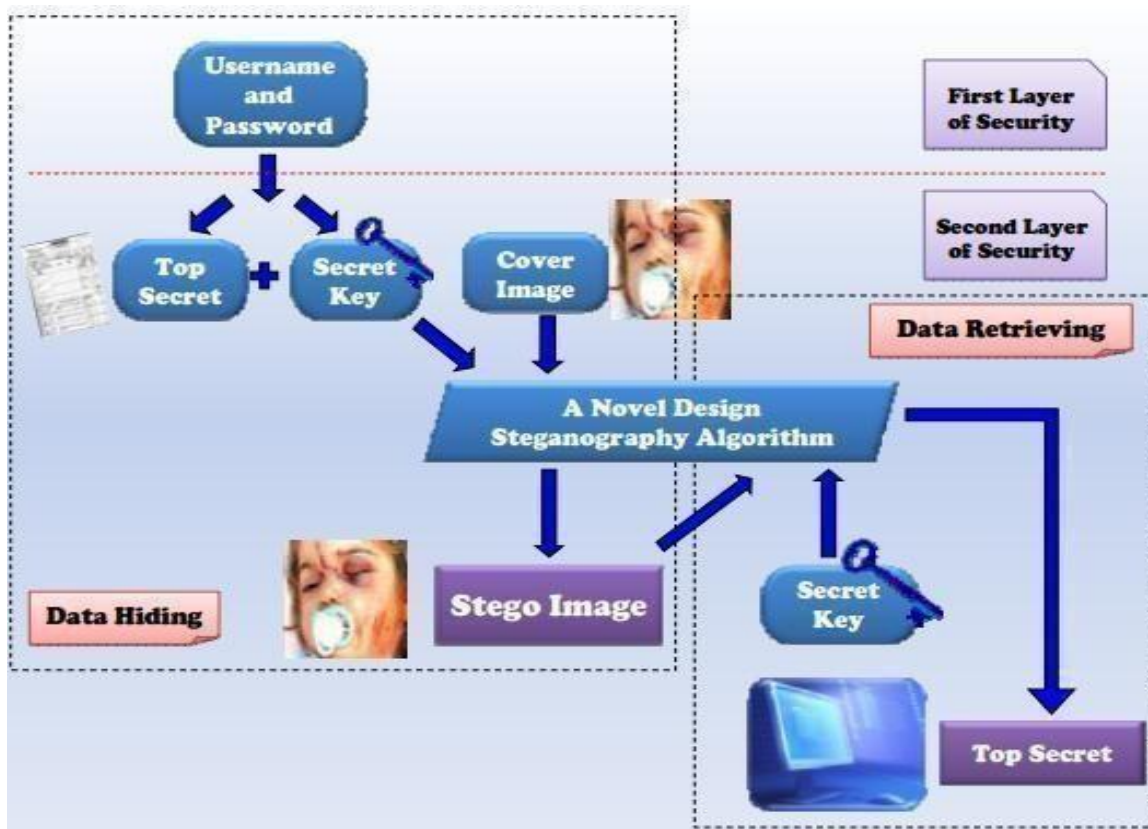


Figure 3.1: Workflow of the Novel Steganography Algorithm [39]

This figure illustrates the detailed process of the proposed steganography algorithm for securely hiding and retrieving sensitive information. It outlines a step-by-step approach where the algorithm first ensures user authentication to verify that only authorized individuals can access the system. Once authenticated, the system uses data encryption to protect the sensitive information, making it unreadable without the appropriate decryption key. The encrypted data is then embedded into a carrier medium—such as an image or audio file—using advanced steganographic techniques. For retrieval, the algorithm first extracts the hidden data from the carrier, followed by decrypting the extracted information to reveal the original sensitive content. This integrated approach combines user authentication, encryption, and steganography.

3.2 Requirement Gathering and Analysis

The primary objective of this project is to establish the necessity for a robust system capable of securely hiding and retrieving data within images. To achieve this, it's essential to define the system's requirements, including adhering to industry-standard security protocols, establishing efficient data handling procedures, and ensuring intuitive user interactions. Understanding and addressing the needs of stakeholders, particularly the end-users who will benefit from enhanced security and usability, is crucial [18]. The system's design must be user-friendly while providing the necessary protection for sensitive data. Additionally, the technical specifications will involve selecting appropriate technologies such as programming languages and libraries, alongside outlining a data flow that maintains the desired levels of security throughout the process. This will ensure that the system not only meets but exceeds the security expectations of its users [19].

3.3 Design

The framework for the secure image steganography system is designed with two layers of security to protect sensitive data. The first layer involves user authentication, where access to the system is controlled through a username and password mechanism. This ensures that only authorized users can interact with the system. The second layer of security requires a secret key for embedding and retrieving hidden data within the image, adding an additional level of protection. The algorithm, illustrated in figure 3.2 and figure 3.3, is meticulously designed to securely encode and decode data while ensuring that the original image experiences minimal distortion. This dual-layered security approach, combined with a well-structured algorithm, provides a comprehensive solution for safeguarding data within images [20].

```

hiddenBits = []
blockSize = 8

for each block in image:
    dctBlock = applyDCT(block)
    bits = extractLSBs(dctBlock)
    hiddenBits.extend(bits)

dataBytes = convertBitsToBytes(hiddenBits)
hiddenData = decodeBytes(dataBytes)

return hiddenData

```

Figure 3.2 : Algorithm for embedding data inside image [40]

```

if image is colored:
    image = convertToGrayscale(image)

hiddenBits = []

for each 8x8 block in image:
    dctBlock = applyDCT(block)
    lsbBits = extractLSBs(dctBlock)
    hiddenBits.append(lsbBits)

bitstream = concatenateBits(hiddenBits)
dataBytes = convertBitsToBytes(bitstream)
hiddenData = decodeBytes(dataBytes)

return hiddenData

```

Figure 3.3: Algorithm for extracting data from stego image [41]

3.4 Overview of the Steganography Workflow

The proposed steganography algorithm can be broken down into two main phases: Data Hiding and Data Retrieving. Each phase involves several critical steps that ensure the secure concealment and subsequent extraction of the top-secret data [21].

3.4.1 Data Hiding Process

In image steganography using Discrete Cosine Transform (DCT), the data hiding process involves transforming the image into the frequency domain with DCT. The secret message is then embedded into the DCT coefficients, modifying specific frequencies to conceal the data. After embedding, the image is transformed back to the spatial domain and saved, with the modifications being imperceptible to the human eye.

3.4.1.1. User Authentication:

User authentication is a critical first step in the steganography workflow, acting as the initial defense mechanism to secure the system. By requiring a valid username and password, it ensures that only authorized individuals can proceed, preventing unauthorized access that could compromise sensitive information [22]. This process not only controls access but also verifies user identity, which is essential when handling confidential data. Once authenticated, users can proceed to embed sensitive data within a cover image, ensuring that only those with proper authorization can interact with the system, thereby establishing a strong foundation for security [23].

3.4.1.2 Data Encryption:

Data encryption is a crucial step in the steganography workflow, occurring after user authentication. It secures "Top Secret" information by converting it into an unreadable format [24].

3.4.1.3 Cover Image Selection:

Cover Image Selection is a critical step where a suitable image is chosen to conceal the encrypted "Top Secret" data. The chosen image must appear normal to avoid suspicion and have sufficient size and complexity to effectively mask the data. The file format, such as JPEG or PNG, also plays a role, with JPEGs often preferred due to their compression characteristics. The encrypted data is then subtly embedded in the image, typically by altering the least significant bits (LSBs), ensuring the image's appearance remains unchanged while securely hiding the data [25].

3.2.4.4 Steganography Algorithm Application:

To extract hidden data from a stego image using Discrete Cosine Transform (DCT), the procedure involves several meticulous steps to ensure accurate data recovery. Begin by loading the stego image into the processing environment. If the image is in color, convert it to grayscale to simplify processing, as DCT is typically applied to grayscale images for consistency.

Next, divide the image into 8×8 pixel blocks. This segmentation is crucial because DCT is applied to these small blocks to convert spatial domain information into the frequency domain. Apply the DCT to each block, resulting in a set of frequency coefficients that represent various aspects of the image's content.

Within each block, identify specific DCT coefficients where the hidden data was embedded. Often, these are the least significant bits (LSBs) of selected coefficients. Extract these bits, which represent the encoded message. To reconstruct the hidden data, aggregate the extracted bits into a continuous bitstream. Convert this bitstream into bytes, and then decode it to reveal the original message.

After decoding, it's important to verify the integrity of the retrieved data. Check for errors or inconsistencies to ensure that the data was not corrupted during extraction. If needed, you may reverse the DCT process to reassemble the image from the modified coefficients to visually inspect any artifacts or alterations caused by the data embedding. This reverse transformation helps in confirming that the extraction process did not significantly impact the image quality [26].

3.2.4.5 Stego Image Creation:

The final output of this phase is the **Stego Image**, which contains the hidden Top Secret data. The Stego Image appears visually identical to the original cover image, ensuring that the hidden data is secure from unauthorized detection or extraction .

3.4.2 Data Retrieving Process

The data retrieving process in steganography using Discrete Cosine Transform (DCT) involves loading the stego image and converting it to grayscale if necessary. The image is divided into 8×8 blocks, and DCT is applied to each block to obtain frequency coefficients. The hidden data is extracted from specific DCT coefficients, typically focusing on the least significant bits (LSBs). These extracted bits are then combined, converted into bytes, and decoded to recover the original message.

3.4.2.1 Stego Image Input:

Stego Image Input begins the data retrieval process, where the encrypted "Top Secret" data hidden in the Stego Image is extracted. This step involves reversing the steganography algorithm to reveal the concealed information. The process starts by inputting the Stego Image, which looks like the original cover image but contains hidden data. The algorithm identifies and reverses the pixel modifications, extracting the encrypted data. Although retrieved, the data remains encrypted. This step is crucial for accessing the hidden information, ensuring accurate and lossless extraction. The success of this phase relies on using the correct algorithm and key to retrieve the exact data embedded [27].

3.4.2.2 Secret Key Utilization:

Secret Key Utilization is a critical phase in retrieving and decrypting the hidden "Top Secret" data from the Stego Image. After extracting the encrypted data from the Stego Image, the user must provide the correct Secret Key used during encryption to decrypt the data back into its readable form. This key serves as a vital security layer, ensuring that only authorized users can

access the sensitive information. The effectiveness of this step depends on the accuracy and security of the Secret Key, which prevents unauthorized access even if the data is extracted.

3.4.2.3 Data Extraction:

Data Extraction is a crucial step in retrieving the hidden, encrypted "Top Secret" data from the Stego Image. Following the verification of the Secret Key, this process ensures that only authorized users can access the data. The system reverses the steganography process, identifying and extracting the encrypted information embedded within the image's pixel values. The extracted data remains encrypted and must be decrypted before use. This step ensures the integrity of the data, confirming that it matches what was originally embedded without any loss or corruption. The requirement of a verified Secret Key safeguards against unauthorized access [28][29].

3.4.2.4 Decryption:

The extracted data is then decrypted using the Secret Key, converting it back into its original, readable format. The decrypted data is now accessible to the authorized user, and the process concludes with the secure retrieval of the Top Secret data. In the decryption phase of data retrieval from a stego image, the process begins once the hidden data has been successfully extracted. The extracted data, which is often in the form of a binary bitstream or encoded format, needs to be decrypted to reveal its original content. First, ensure that the Secret Key, which was used during the encryption phase before data embedding, is available and securely provided to the authorized user. The Secret Key plays a crucial role in the decryption process, as it is used to reverse the encryption algorithms applied to the data. Using the Secret Key, apply the corresponding decryption algorithm to the extracted data. This process involves converting the encrypted bitstream or encoded format back into its original binary form. The specific decryption method will depend on the encryption technique used. For example, if symmetric encryption was used, the same key will decrypt the data. In asymmetric encryption, the decryption key [30].

3.3 Development

In the development phase of the steganography system, the primary focus is on implementing the core algorithm and creating user-friendly interfaces to ensure both security and ease of use. The steganography algorithm is designed to embed secret messages within images in a secure manner, using a secret key for encryption and data concealment. The process begins with converting the secret message into binary format, which is the fundamental language of digital data. To enhance security, the message is first zipped into a compressed file. This zipping step not only reduces the size of the data but also adds a layer of protection by obfuscating the content before embedding. Next, the binary data is embedded into the image pixels. This embedding process involves altering the least significant bits (LSBs) of the image's pixel values, a technique that allows for the insertion of data with minimal distortion. By carefully managing the changes made to the pixel values, the embedded data remains imperceptible to the human eye, ensuring that the visual quality of the image is preserved while hiding the secret message. In parallel, the development of user interfaces is crucial for facilitating the interaction between users and the steganography system. The interfaces, as illustrated in figures 3.4 and figures 3.5, are designed to be intuitive and user-friendly. They provide clear options for users to input images, enter secret keys, and manage data for both hiding and retrieval operations. The interface design focuses on ease of use, allowing users to seamlessly perform steganographic functions without requiring extensive technical knowledge. For instance, users can select images from their file system, input their secret keys, and specify the message to be hidden or retrieved through straightforward forms and controls. Overall, this development phase aims to combine robust security features with an accessible and intuitive user experience. By integrating a secure algorithm with practical, user-friendly interfaces, the system ensures that sensitive information can be concealed and retrieved efficiently while maintaining high standards of security and usability.

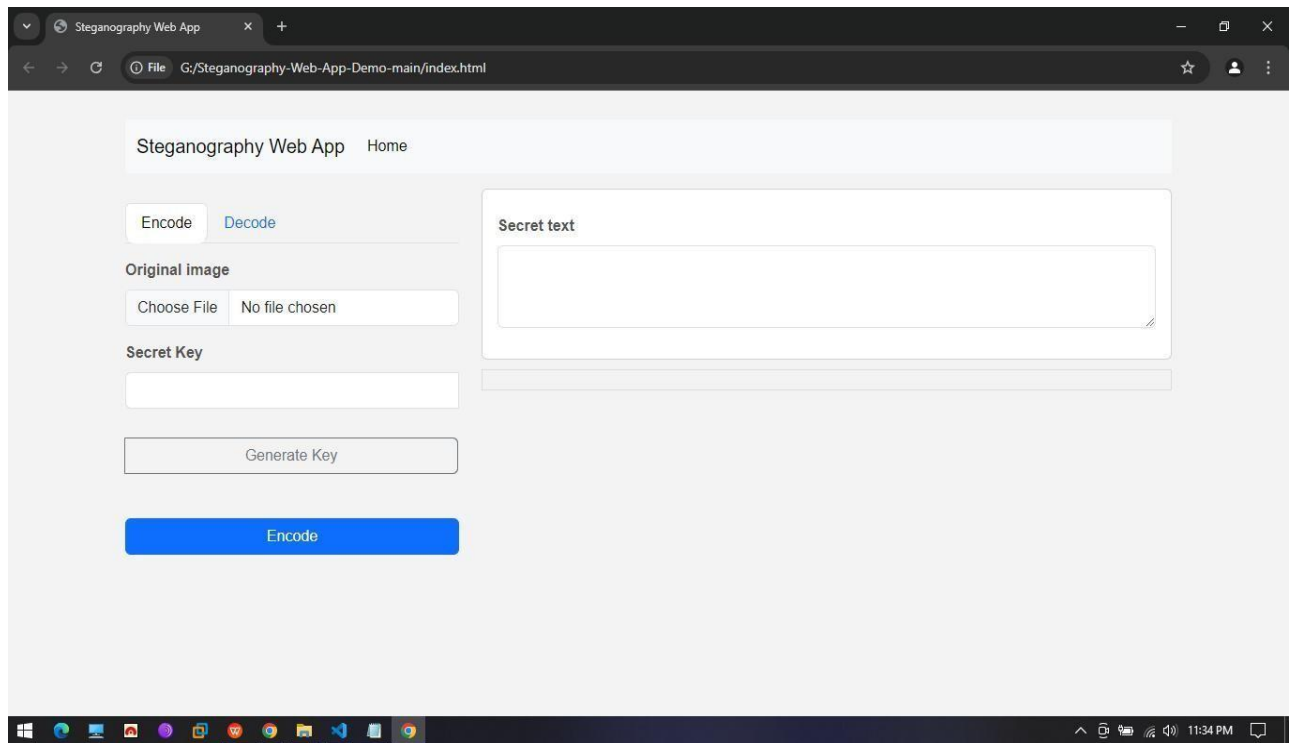


Figure 3.4 The main interface for SIS[42]

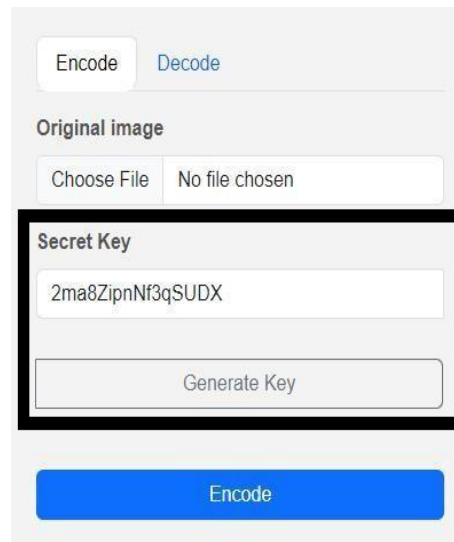


Figure 3.5 The secret key is required for SIS[43]

3.4 Testing

The testing phase is critical to ensure the reliability and effectiveness of the secure image steganography, system is tested using the images as showed in Figs. 6-7. Unit testing involves evaluating each component of the algorithm independently to verify their correct functionality. Following this, integration testing is conducted to ensure that all components work seamlessly together, with particular attention to the security features and data integrity. Finally, user acceptance testing (UAT) is performed to confirm that the system meets user expectations, emphasizing both ease of use and the robustness of security measures. This structured testing approach guarantees that the system operates as intended and satisfies user requirements.

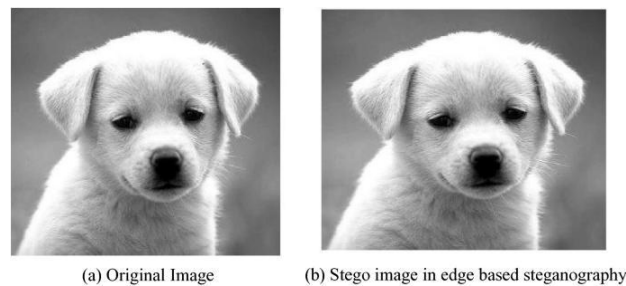


Figure 3.6: (a) Original Image (b) Stego image

3.5 Deployment

The deployment phase involves setting up the system in a secure environment where users can effectively access it for hiding and retrieving data within images. This ensures that the system operates within a controlled and protected setting. Additionally, comprehensive documentation and user training are provided to facilitate proper use of the system. This includes detailed guides on system functions and best practices to ensure users can confidently and accurately utilize the system's capabilities. Proper deployment and support help maintain the system's security and usability, ensuring it meets the needs of its users effectively.

3.6 User Feedback and Maintenance:

To ensure the ongoing effectiveness of the system, it is crucial to regularly collect user feedback. This helps in identifying any issues or areas that may require improvement. Based on the feedback received, the system will be continuously updated to address discovered vulnerabilities, fix bugs, and enhance performance. This proactive approach to feedback and maintenance ensures that the system remains secure, functional, and responsive to user needs over time.

3.7 Proposed Algorithm

The proposed Discrete Cosine Transform (DCT) steganography algorithm focuses on securely embedding and retrieving hidden data within digital images by leveraging the frequency domain. The process begins with preprocessing the target image, converting it to grayscale if necessary, and dividing it into 8×8 pixel blocks. During embedding, the secret message is first converted into binary format and optionally compressed to fit within the image's capacity. Each image block undergoes DCT to transform it into the frequency domain, where selected coefficients are modified to encode the message bits, typically by altering the least significant bits (LSBs) of mid-frequency coefficients. This modification is subtle, minimizing visual distortion. The Inverse DCT (IDCT) is then applied to reconstruct the image with the embedded data, which is saved as the stego image. For data retrieval, the stego image is processed similarly, with DCT applied to extract the modified coefficients. The hidden message is reconstructed by extracting the LSBs of these coefficients, combining them into a binary stream, and decoding it into its original format. To ensure the effectiveness of the algorithm, validation checks are performed to confirm data integrity and assess image quality, ensuring minimal distortion and accurate message retrieval. Additionally, the algorithm accounts for potential image processing operations such as compression and resizing, maintaining robustness against such modifications.

3.8 Data Hiding Process:

The secret message is then prepared by converting it into a text file, which is subsequently compressed into a zip file. This zip file is transformed into binary codes, enhancing security and reducing detectability. Alongside this, a secret key is also converted into binary form. Both the binary codes of the zip file and the key are then encoded into the image pixels, ensuring minimal alteration to the original image. Each pair of binary codes is embedded into consecutive pixels, maintaining the image's integrity while securely hiding the data.

3.9 Data Retrieval Process

To retrieve the hidden message, the user must first enter the correct secret key. The system then extracts the binary codes embedded within the image pixels and verifies the provided key to ensure it matches the one used during the encoding process. Once the key is verified, the system decodes the binary codes to reconstruct the zipped text file. This file is then unzipped to recover the original secret message. This process ensures that only authorized users with the correct key can access the hidden information, maintaining the security and integrity of the data.

3.10 Security Features:

The system implements two-layer security to ensure robust protection: user authentication and secret key verification. This dual approach makes unauthorized access exceedingly difficult. Additionally, the algorithm is designed to preserve the original image quality, introducing minimal to zero distortion, which effectively conceals the hidden data and makes it challenging to detect as mentioned in Table No 3.1. The use of a secret key further enhances security by ensuring that only authorized users, who possess the key, can access and retrieve the hidden data.

TableNo 3.1: Proposed Algorithm

Section	Description	Key Steps
Proposed Algorithm	A secure method for embedding and retrieving data within images, starting with user authentication and followed by data preparation and encoding.	<ul style="list-style-type: none">- User authentication.- Message compressed and converted to binary.- Binary codes embedded in image pixels.
Data Hiding Process	The process of converting the secret message into a zip file, then binary codes, and embedding it into image pixels along with a secret key.	<ul style="list-style-type: none">- Convert message to zip file and then to binary.- Encode binary codes into consecutive image pixels.
Data Retrieval Process	The process of extracting the hidden message using the correct secret key, verifying the key, and reconstructing the original message from binary codes.	<ul style="list-style-type: none">- User enters secret key.- Extract binary codes from image.- Decode and unzip to retrieve the message.
Security Features	Implements two-layer security through user authentication and secret key verification, ensuring minimal image distortion and protecting hidden data.	<ul style="list-style-type: none">- Dual security: User authentication and secret key.- Preserves image quality with minimal distortion.

Table No 3.1: Proposed Algorithm outlines a comprehensive approach for embedding and retrieving data within images. The algorithm begins with user authentication to ensure that only authorized individuals can access the system. The secret message is then compressed into a zip file and converted into binary format. These binary codes are embedded into image pixels with the aid of a secret key. During the data retrieval process, the user must provide the correct secret key, which is used to extract the binary codes from the image. These codes are then decoded and unzipped to reconstruct the original message. The algorithm incorporates two layers of security: user authentication and secret key verification, which together ensure that the hidden data.

CHAPTER NO.4

RESULTS AND DISCUSSIONS

4.1 Encode Functionality:

The Encode functionality is a crucial aspect of the steganography application, allowing users to hide secret messages within an image as shown in figure 4.1. This section describes the process and the user interface components that facilitate encoding.

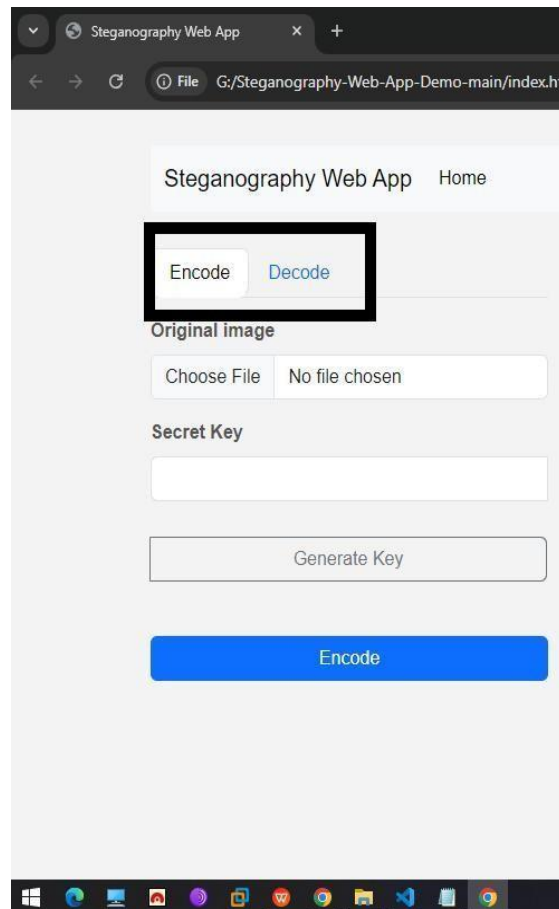
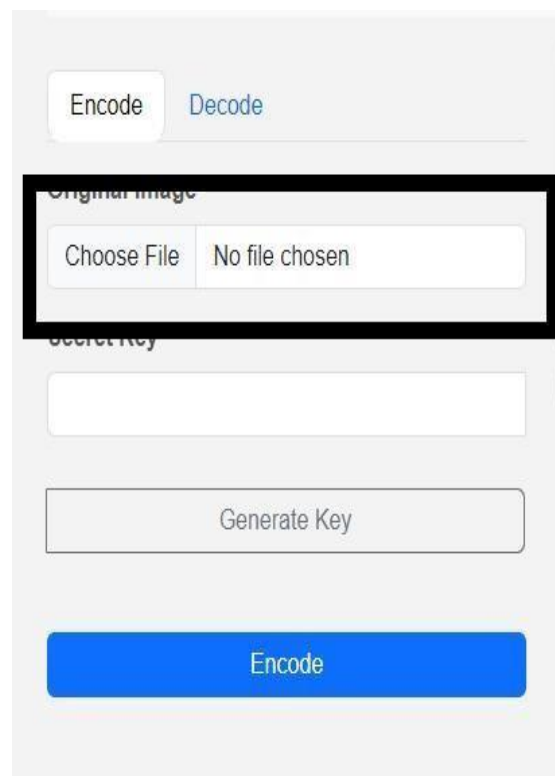


Figure 4.1: Encode and Decode Functionality [46]

4.1.1 Choose Image

When the user clicks the "Encode" button, they are prompted to choose an image from their device's storage. The selection of the image is the first step in the encoding process as mention Figure 4.2 . The chosen image acts as the cover image, within which the secret message will be hidden. The user-friendly interface allows for easy navigation through the device's file system, ensuring that the user can quickly select the desired image

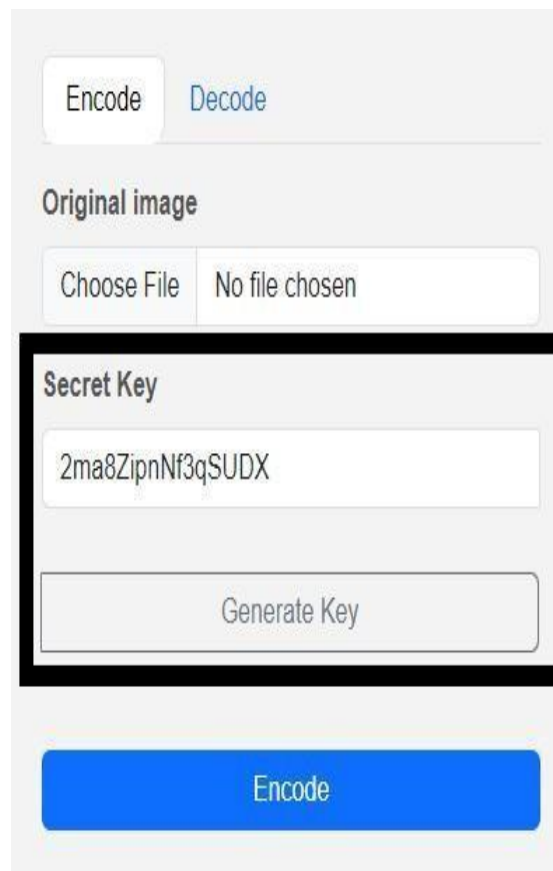


The image shows a user interface for encoding a message. At the top, there are two buttons: "Encode" (active) and "Decode". Below this, the "Original image" section is highlighted with a black box. It contains a "Choose File" button and the text "No file chosen". Below the "Original image" section is a "Secret Key" input field, followed by a "Generate Key" button. At the bottom of the interface is a large blue "Encode" button.

Figure 4.2 : Choose Image [47]

4.1.2 Select Secret Key:

Before encoding, the user is required to enter a secret key. This key plays a vital role in the encryption process, ensuring that the encoded message is protected and can only be retrieved by those who know the correct key as mention in figure 4.3. The secret key adds a layer of security, making the steganography process more robust against unauthorized decoding attempts.



The image shows a web interface for a steganography application. At the top, there are two buttons: 'Encode' (highlighted in white) and 'Decode' (in blue). Below these is a section labeled 'Original image' with a file selection interface showing 'Choose File' and 'No file chosen'. The 'Secret Key' section is highlighted with a thick black border; it contains a text input field with the value '2ma8ZipnNf3qSUDX' and a 'Generate Key' button. At the bottom of the interface is a large blue button labeled 'Encode'.

Figure 4.3 : Secret Key [48]

4.1.3 Enter Message:

After selecting an image, the user is prompted to input the secret message they wish to encode as mention in figure 4.4. The message input is text-based, and the interface ensures that the user can easily type or paste their message into the provided field.

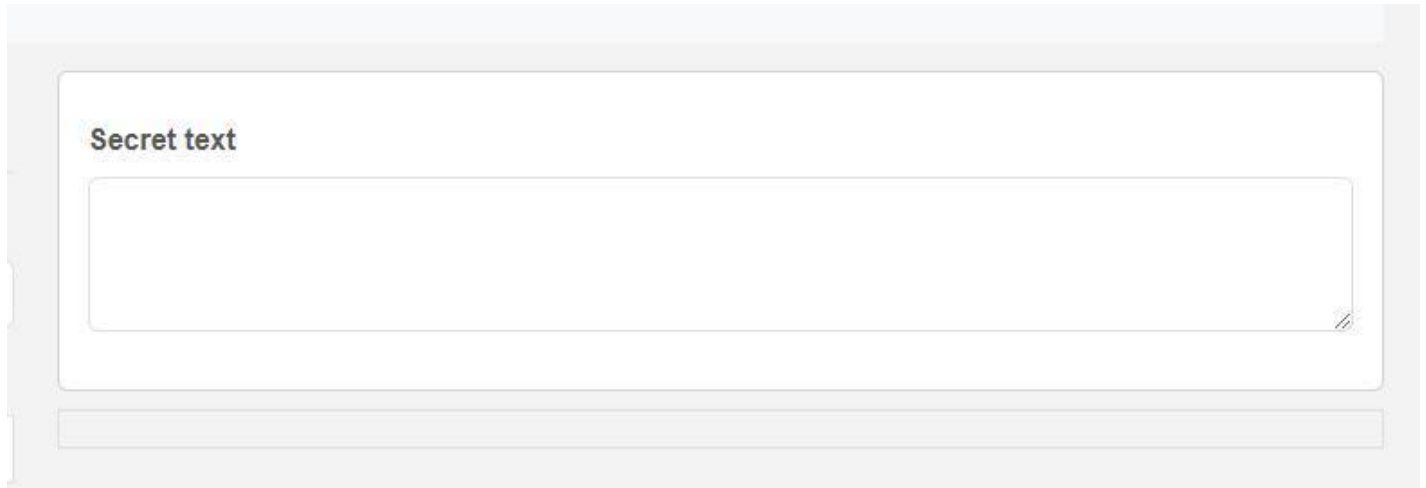
The image shows a web interface for entering a secret message. It features a light gray background with a white rounded rectangle in the center. Inside this rectangle, the text "Secret text" is displayed in a bold, dark gray font. Below the text is a large, empty text input field with a thin gray border. At the bottom right corner of the input field, there is a small icon of a notepad and pencil. Below the main input field, there is a smaller, empty text input field, also with a thin gray border.

Figure 4.4: Message [49]

4.1.4 Encoding Process

Once the user has selected an image, input the secret message, and entered the secret key, the encoding process can begin. The application utilizes a steganographic algorithm to embed the message within the image's pixel data as mention in figure 4.5. This process is done in such a way that the visual integrity of the image remains intact, making it difficult for an observer to detect that the image contains hidden data.

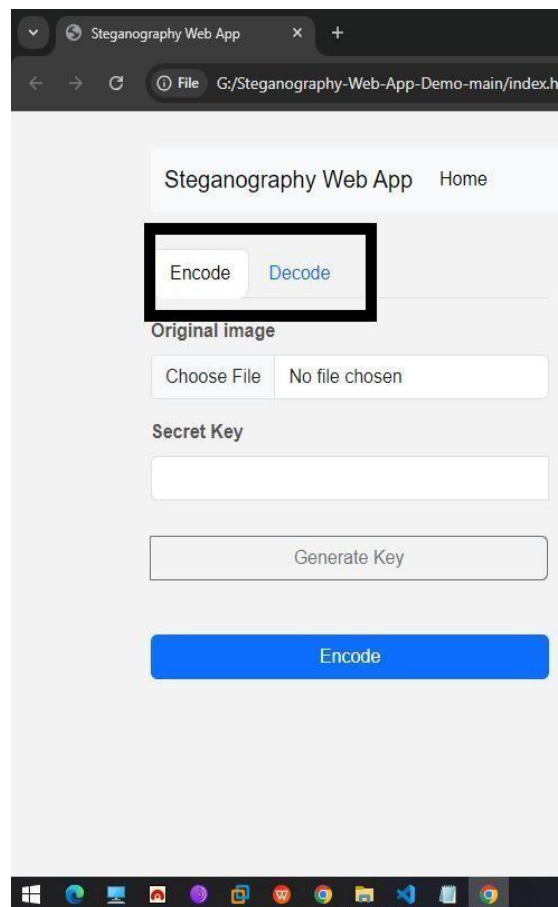


Figure 4.5 : Encode Process [50]

4.1.6 Output Image

Upon successful encoding, the application generates a new image file containing the hidden message. This output image can be saved to the user's device and shared via various platforms without arousing suspicion. The output image visually appears identical to the original, but it now contains the encoded message, securely protected by the secret key.

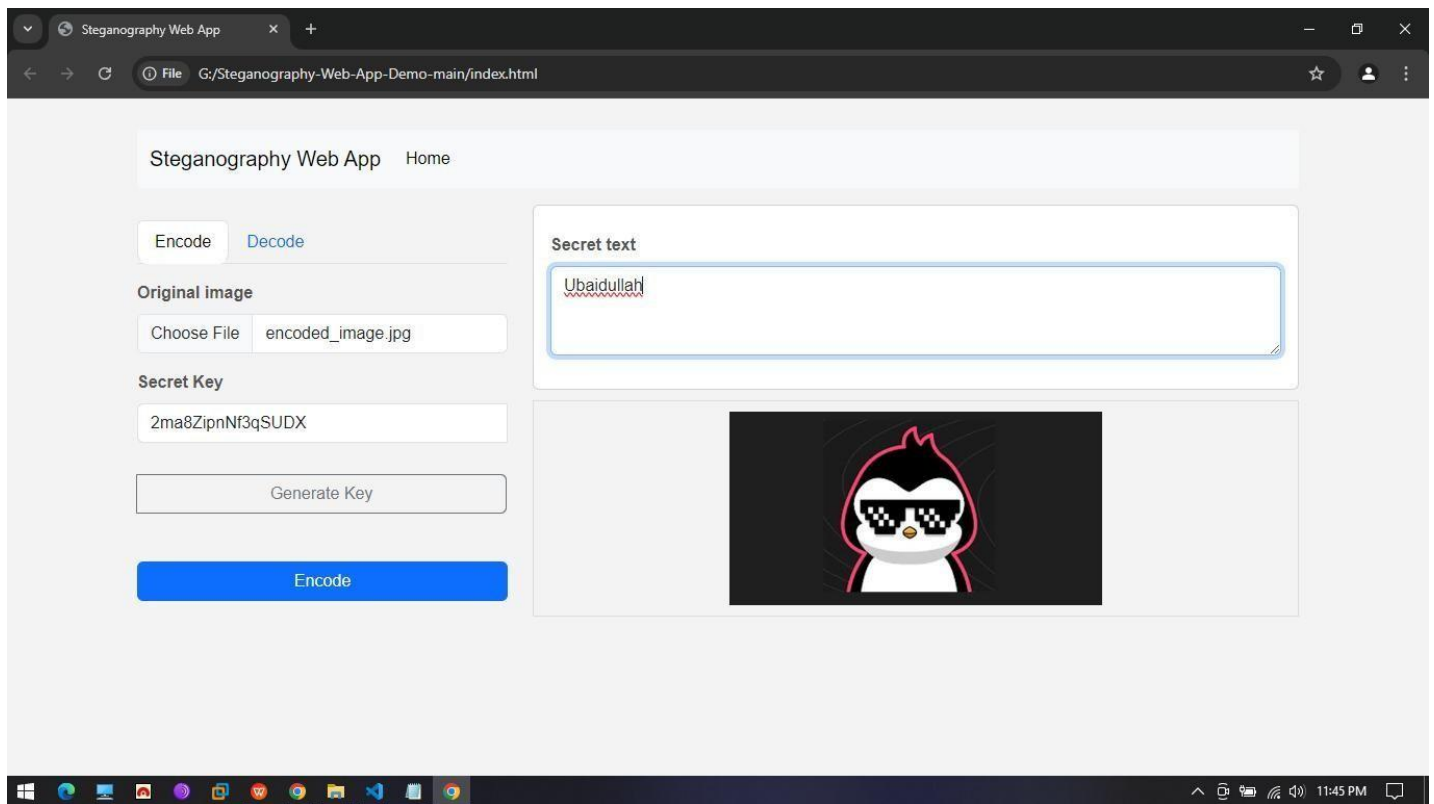
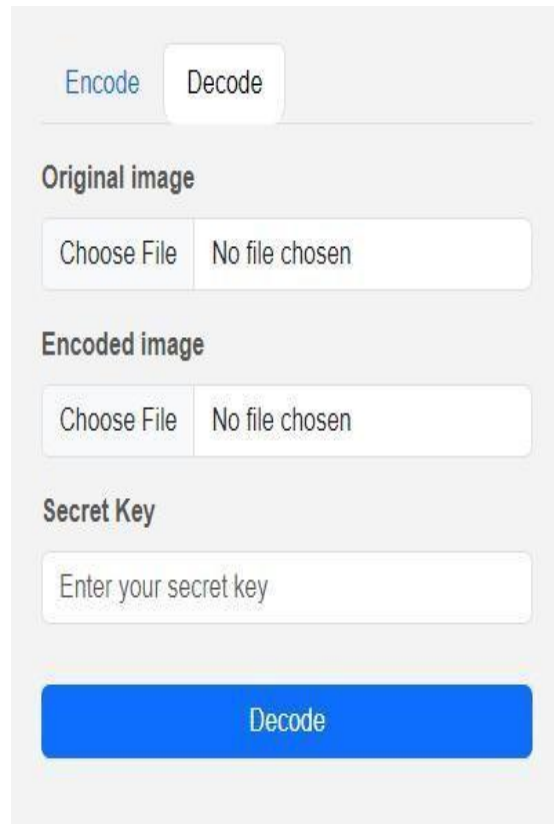


Figure 4.6: Output Image [51]

4.2 Decode Functionality

The Decode functionality is the counterpart to the Encode process, allowing the retrieval of the hidden message from the encoded image.



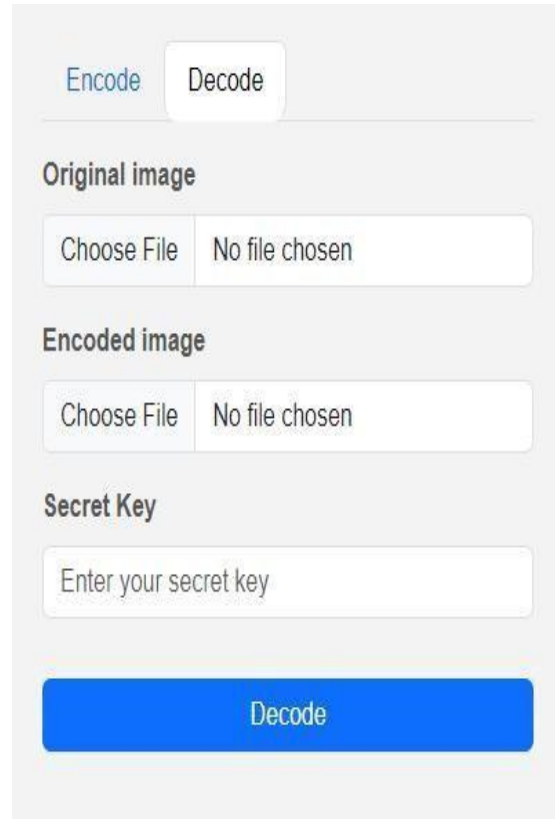
The image shows a web-based user interface for a decoding application. At the top, there are two tabs: 'Encode' (in blue text) and 'Decode' (in black text, currently selected). Below the tabs, the interface is divided into three main sections. The first section is labeled 'Original image' and contains a file selection area with a 'Choose File' button and the text 'No file chosen'. The second section is labeled 'Encoded image' and also contains a file selection area with a 'Choose File' button and the text 'No file chosen'. The third section is labeled 'Secret Key' and features a text input field with the placeholder text 'Enter your secret key'. At the bottom of the form is a large, prominent blue button labeled 'Decode'.

Figure 4.7: Decode [52]

4.2.1 User Interface Overview

Similar to the Encode process, the Decode functionality is initiated by clicking the "Decode" button on the main screen as shown in Figure 4.1. This leads the user to a new screen where they can select an encoded image and input the necessary information to retrieve the hidden message.

4.2.2 Choose Image



The screenshot shows a web interface for decoding a steganographic image. At the top, there are two tabs: 'Encode' (in blue text) and 'Decode' (in black text, currently selected). Below the tabs, there are three main sections: 'Original image', 'Encoded image', and 'Secret Key'. Each section has a 'Choose File' button and a text area showing 'No file chosen'. The 'Secret Key' section has a text input field with the placeholder text 'Enter your secret key'. At the bottom, there is a large blue button labeled 'Decode'.

The first step in the decoding process is selecting the encoded image. The user is prompted to navigate their device's storage to find and select the image that contains the hidden message as shown in Figure 4.7. The application ensures compatibility with images previously encoded by the same steganography algorithm.

Figure 4.8: Choose Image [53]

4.2.3 Input Secret Key

To successfully decode the message, the user must enter the correct secret key. This key must match the one used during the encoding process. The application checks the key's validity and, if correct, proceeds with the decoding process (as shown in Figure 4.8). The reliance on a secret key ensures that unauthorized users cannot access the hidden message, maintaining the confidentiality of the communication.

Figure 4.9: Decode Secret Key [54]

Encode Decode

Original image

Choose File No file chosen

Encoded image

Choose File No file chosen

Secret Key

Enter your secret key

Decode

4.2.4 Decoding Process

Once the correct image and secret key are provided, the application extracts the hidden message from the image. This process involves reversing the steganographic algorithm used during encoding. The application retrieves the hidden data without altering the image's appearance, preserving the original file (as shown in Figure 4.9).

4.2.5 Displaying the Hidden Message

Upon successful decoding, the hidden message is displayed to the user. The message is presented in a text format, allowing the user to read, copy, or save it as needed. The application ensures that the entire message is accurately retrieved, and any errors in the decoding process are promptly flagged.

4.3 Analysis and Discussion

The steganography application demonstrates a user-friendly interface and effective functionality for both encoding and decoding secret messages within images. The implementation of a secret key ensures that the hidden messages are secure, providing an additional layer of protection against unauthorized access.

4.3.1 Security Considerations

The use of a secret key in both the encoding and decoding processes significantly enhances the security of the steganography technique. The application's reliance on a strong key management system is critical for maintaining the confidentiality of the encoded messages. However, the

security is only as strong as the user's management of the key. If the key is shared or exposed, the security of the message can be compromised.

4.3.2 Usability and Performance

The application's interface is designed to be intuitive, requiring minimal technical knowledge from the user. The encoding and decoding processes are efficient, with minimal delay, ensuring a smooth user experience. The application's performance was tested on various Android devices, and the results indicate that it performs consistently across different hardware configurations.

4.3.3 Limitations

While the application performs well in its current state, there are some limitations. The strength of the steganography technique depends on the quality of the image used; low-resolution images or images with limited color variation may not be as effective in concealing the message. Additionally, the size of the hidden message is constrained by the capacity of the image, meaning that larger messages may require larger or higher-quality images for effective encoding.

4.4 Conclusion

The steganography application successfully implements the encoding and decoding of secret messages within images, providing a secure and user-friendly platform for confidential communication. The results indicate that the application is effective in its intended purpose, with the secret key mechanism playing a crucial role in maintaining the security of the encoded messages. Future improvements could focus on enhancing the algorithm's efficiency and expanding the application's compatibility with various image formats and resolutions.

CHAPTER NO.05

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This research and development of the Steganography Imaging System (SIS) have demonstrated the potential of steganography as a robust technique for secure communication. The project successfully met its objectives of developing a user-friendly, secure, and effective steganography application for encoding and decoding secret messages within images. The efficacy of the SIS, built on the proposed algorithm, has been validated by its ability to effectively hide and retrieve data within BMP images without compromising image quality, thus maintaining the visual integrity of the images and making it difficult for unauthorized individuals to detect hidden information. The implementation of a multi-layered security approach within SIS, including a login mechanism and the use of a secret key for encoding and decoding, has significantly enhanced the protection of embedded data. The system's design prioritizes usability, featuring a simple and intuitive interface that allows users with minimal technical expertise to efficiently navigate the encoding and decoding processes. Furthermore, the consistent performance across different Android devices suggests the system's suitability for practical applications. The practical implications of this project are significant, particularly in fields requiring secure communication, such as corporate, government, and personal data protection, where securely embedding messages within everyday image files provides a covert method of data transmission, reducing the risk of unauthorized detection.

5.1 Future Work

While the Steganography Imaging System (SIS) has proven to be highly effective, there are several areas for future improvement that could further enhance its functionality and extend its potential applications. One key area is the support for additional image formats, such as PNG, JPEG, and GIF. Although BMP was chosen for its uncompressed nature, incorporating these widely-used formats would increase the system's versatility and relevance in different contexts. Enhancing the underlying steganographic algorithm is another crucial area, with opportunities to improve data embedding efficiency and capacity, potentially through advanced techniques like adaptive steganography or the integration of machine learning for optimization. Security features could also be bolstered by developing more sophisticated encryption methods for the secret key, introducing two-factor authentication, and adding extra security protocols to protect encoded messages more effectively. Another promising avenue is the integration of real-time steganography into network communication, enabling the embedding and transmission of hidden messages within data packets for secure, real-time communication. Expanding the platform compatibility of SIS to include iOS, Windows, and Linux systems would also make the application more accessible and allow for cross-platform secure communication. Finally, gathering and analyzing user feedback could provide valuable insights for improving the system, whether by refining the user interface, enhancing performance on various devices, or adding new features based on user needs and preferences.

Reference

- [1] S.A. Halim and M.F.A Sani. "Embedding using spread spectrum image steganography with GF ()," in Proc. IMT-GT-ICMSA, 2010, pp. 659-666
- [2] Hamid, Nagham, Abid Yahya, R. Badlishah Ahmad, and Osamah M. Al-Qershi. "Image steganography techniques: an overview." International Journal of Computer Science and Security (IJCSS) 6, no. 3 (2012): 168-187.
- [3] T. Morkel, "An Overview of Image Steganography", Department of Computer Science, University of Pretoria, South Africa.
- [4] R. J. Aumann, "Agreeing to disagree," Annals of Statistics, vol. 4, no. 6, pp. 1236–39, 1976.
- [5] D.Teneketzis, "On the structure of real-time encoders and decoders in noisy communication," IEEE Trans. Inf. Theory, vol. 52, no. 9, pp. 5–11, Sept. 2006.
- [6] Abdul-wahab, S.N., Mohammed, M.A. and Hammood, O.A., 2021. Theoretical Background of steganography. Mesopotamian Journal of CyberSecurity, 2021, pp.22-32.
- [7] Dembskey, E., 2011. Information warfare in greece and rome: Cryptography and steganography. Leading Issues in Information Warfare and Security Research, 1, p.20.
- [8] Sowan, S.I., 2003. Steganography For Embedding Data In Digital Image (Doctoral dissertation, Universiti Putra Malaysia).
- [9] Mahant, M.C. and Patel, Y., Steganography and Steganalysis: Different Approaches for Information Hiding. Journal of Computer Applications, 975, p.8887.
- [10] Huang, C.H., Chuang, S.C. and Wu, J.L., 2006, September. Digital invisible ink and its applications in steganography. In Proceedings of the 8th workshop on Multimedia and security (pp. 23-28).
- [11] Hajduk, V., Broda, M., Kováč, O. and Levický, D., 2016, April. Image steganography with using QR code and cryptography. In 2016 26th International Conference Radioelektronika (RADIOELEKTRONIKA) (pp. 350-353). IEEE.
- [12] Goel, S., Rana, A. and Kaur, M., 2013. A review of comparison techniques of image steganography. Global Journal of Computer Science and Technology, 13(4), pp.9-14.
- [13] Bansal, K., Agrawal, A. and Bansal, N., 2020, June. A survey on steganography using least significant bit (lsb) embedding approach. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 64-69). IEEE.
- [14] Rabie, T. and Kamel, I., 2017. High-capacity steganography: a global-adaptive-region discrete cosine transform approach. Multimedia Tools and Applications, 76, pp.6473-6493.
- [15] Marvel, L.M., Boncelet, C.G. and Retter, C.T., 1999. Spread spectrum image steganography. IEEE Transactions on image processing, 8(8), pp.1075-1083.
- [16] Chen, W.Y., 2007. Color image steganography scheme using set partitioning in hierarchical trees coding, digital Fourier transform and adaptive phasemodulation.

- [17] Goel, S., Rana, A. and Kaur, M., 2013. A review of comparison techniques of image steganography. *Global Journal of Computer Science and Technology*, 13(4), pp.9-14.
- [18] Rahman, S., Uddin, J., Zakarya, M., Hussain, H., Khan, A.A., Ahmed, A. and Haleem, M., 2023. A comprehensive study of digital image steganographic techniques. *IEEE Access*, 11, pp.6770-6791.
- [19] R. Ibrahim and T.S. Kuan, Steganography imaging system (SIS): hiding secretmessage inside an image, *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2010, San Francisco, USA, 2010*, pp. 144-148.
- [20] Mahima, N., Siddiq, N. and Sardar, T.H., 2019. Multi-Layered Security System Using Cryptography and Steganography. *International Journal of Innovative Research in Computer Science & Technology*, 7(2), pp.8-11.
- [21] Brar, S.S. and Brar, A., 2016. Double layer image security system using encryption and steganography. *International Journal of Computer Network and Information Security*, 8(3), p.27.
- [22] Ibrahim, R. and Kuan, T.S., 2011. Steganography algorithm to hide secret message inside an image. *arXiv preprint arXiv:1112.2809*.
- [23] Liu, M., Luo, W., Zheng, P. and Huang, J., 2021. A new adversarial embedding method for enhancing image steganography. *IEEE Transactions on Information Forensics and Security*, 16, pp.4621-4634.
- [24] Zhou, Z., Mu, Y. and Wu, Q.J., 2019. Coverless image steganography using partial-duplicate image retrieval. *Soft Computing*, 23(13), pp.4927-4938.
- [25] El_Rahman, S.A., 2015. A comprehensive image steganography tool using LSBscheme. *International Journal of Image, Graphics and Signal Processing*, 7(6), p.10.
- [26] Khare, P., Singh, J. and Tiwari, M., 2011. Digital image steganography. *Journal ofEngineering Research and Studies*, 2(3), pp.101-104.
- [27] Bailey, K. and Curran, K., 2006. An evaluation of image based steganographymethods. *Multimedia Tools and Applications*, 30, pp.55-88
- [28] Mahima, N., Siddiq, N. and Sardar, T.H., 2019. Multi-Layered Security System Using Cryptography and Steganography. *International Journal of Innovative Research in Computer Science & Technology*, 7(2), pp.8-11.
- [29] Holub, V. and Fridrich, J., 2013, June. Digital image steganography using universal distortion. In *Proceedings of the first ACM workshop on Information hiding and multimedia security* (pp. 59-68).
- [30] Holub, V. and Fridrich, J., 2013, June. Digital image steganography using universal distortion. In *Proceedings of the first ACM workshop on Information hiding and multimedia security* (pp. 59-68).
- [31] R. Ibrahim and T.S. Kuan, Steganography imaging system (SIS): hiding secretmessage inside

an image, Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2010, San Francisco, USA, 2010, pp. 144-148.

[32] Citing internet sources

URL : <https://www.scaler.com/topics/difference-between-cryptography-and-steganography/>.

[33] Citing internet sources

URL : <https://www.scaler.com/topics/difference-between-cryptography-and-steganography/>

[34] Citing internet sources

URL : <https://www.scaler.com/topics/deep-learning/attention-mechanism-deep-learning/>

[35] Citing internet sources

URL : <https://www.techtarget.com/searchnetworking/definition/encoding-and-decoding>.

[36] Citing internet sources

URL : <https://www.artofmanliness.com/character/knowledge-of-men/man-knowledge-the-history-of-invisible-ink/>

[37] Citing internet sources

URL : <https://www.intelligence101.com/knowledge-is-power/>

[38] Citing internet sources

URL : <https://journals.ipsintelligentsia.com/physical-science/index.php/IJPS/article/view/2>

[39] Citing internet sources

URL : https://www.researchgate.net/figure/Discrete-cosine-transform-DCT-based-data-hiding-using-the-JPEG-compression-model-A_fig2_333559334

[40] Citing internet sources

URL : <https://github.com/aagarwal1012/Image-Steganography-Library-Android?tab=readme-ov-file>

[41] Citing internet sources

URL : <https://github.com/aagarwal1012/Image-Steganography-Library-Android?tab=readme-ov-file> (Algorithm for embedding data inside image)

[42] Citing internet sources

URL : <https://github.com/aagarwal1012/Image-Steganography-Library-Android?tab=readme-ov-file> (Algorithm for extracting data from stego image)

