Steganography Report

IT 360: Information Assurance and Security

by

Chahine Jebabli

Achref Mouelhi

Ahmed Nour

April 2023

Information Technology Major

Tunis Business School

2022-2023

Contents

Chapter 1

Introduction

1.1 What is Steganography

Steganography is the practise of concealing information within a medium, originating from the Greek words "steganos" (covered) and "graphial" (writing). This entails concealing the data so that only the intended recipient is aware of its presence. While archaic methods entailed concealing data on the backs of wax, writing tables, or even rabbit bellies, modern methods often involve transferring data in the form of text, images, video, and audio across numerous mediums. Multimedia items are frequently employed as cover sources to conceal sensitive data during transmission. Steganography's goal is to permit invisible communication while guaranteeing confidentiality between communicative parties. We intend to discuss several security and data concealment techniques that can be utilised in steganography, such as LSB and PVD, in this study. We will investigate the strategies' strengths and drawbacks, as well as their applicability in various circumstances. Our goal is to provide a thorough understanding of steganography and its potential applications. Readers will have a better knowledge of steganography and its function in facilitating secure communication by the end of this paper.

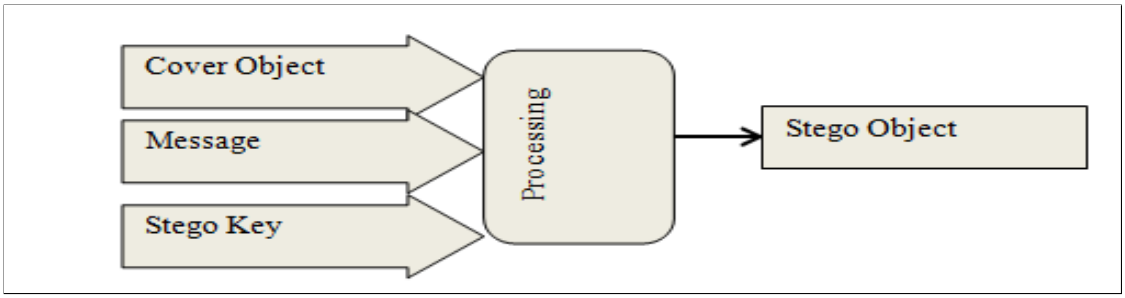

Figure 1.1: Steganography Action. Adapted from [1]

1.2 Process

The steganography process involves several steps:

1. Select and convert the secret data into a binary format.
2. Choose a cover object, such as an image or audio file.
3. Embed the secret data into the cover object using a steganographic algorithm.
4. Create a new file, known as the stego-object, which appears identical to the original cover object but contains the hidden secret data.
5. Transmit the stego-object over a chosen medium to the intended recipient.
6. Extract the secret data using a reverse steganographic algorithm.

By following these steps, steganography allows for the secure and discreet transmission of confidential information.

1.3 History

Steganography has an interesting history that dates back to ancient Greece. It has been used to disguise messages in a variety of methods, including ink and milk, as well as digital images, audio, and video files. Despite its growth, steganography's primary purpose remains the same: to keep information secret. This centuries-old strategy is still important in current communication.

1.4 Image steganography

Image steganography is a technique for concealing sensitive information within digital photographs. The secret information is encoded into the image pixels using a steganographic technique, resulting in a new image that appears identical to the original, known as the stego-image. The stego-image can be conveyed to the receiver via a specified medium, and the secret information can be extracted using a reverse steganographic technique. Despite the fact that image steganography is a common technique due to the popularity of digital photographs, it can be detected through statistical analysis or visual inspection.

Chapter 2

Basics

2.1 Cryptography Basics

Cryptography is a method of rendering information unintelligible to unauthorised individuals. This aids in the concealment of concealed information in steganography. Cryptography employs techniques to convert information into a secret code (ciphertext) that can be safely transferred. The recipient then decodes the information back into its original form (plaintext) using a key. Steganography employs several types of encryption, including symmetric key cryptography, public key cryptography, and hashing. Hidden information could be obtained by unauthorised parties if cryptography is not used.

2.2 Steganography Basics

Steganography is a method of concealing information within cover data in such a way that unauthorised users analysing the data are unable to find it. Unlike watermarking, steganography is designed to ensure that the hidden message is neither removed or altered by adversaries, but rather that it remains invisible. Steganography is very beneficial when encryption cannot be used to secure secret information during communication.

	Steganography	Cryptography
Definition	Depend on hiding the message existence	Depend on hiding the message meaning
Purpose	Keep communication secure.	Provide protection for data
Visibility	Never	Always
Failure	When discover the presence of a hidden message	When able to decrypt and read the message
Concern	Embedding capacity and detectability of cover object	Robustness against deciphering.
Carrier	Any type of digital media	Depend on text as a carrier
Key	Optional, but provide more security	Necessary

Figure 2.1: Steganography vs Cryptography. Adapted from [1]

Chapter 3

Main Concepts

Steganography is a method of hiding a hidden message within a cover image, audio file, or video file. The cover image acts as a vehicle for the hidden message, which is concealed within it via a steganography technique. To ensure safe communication, the algorithm defines how the secret message is encoded in the cover image, and a steganography key may also be employed. Understanding the fundamental fundamentals of steganography is essential for efficiently applying and employing this approach.

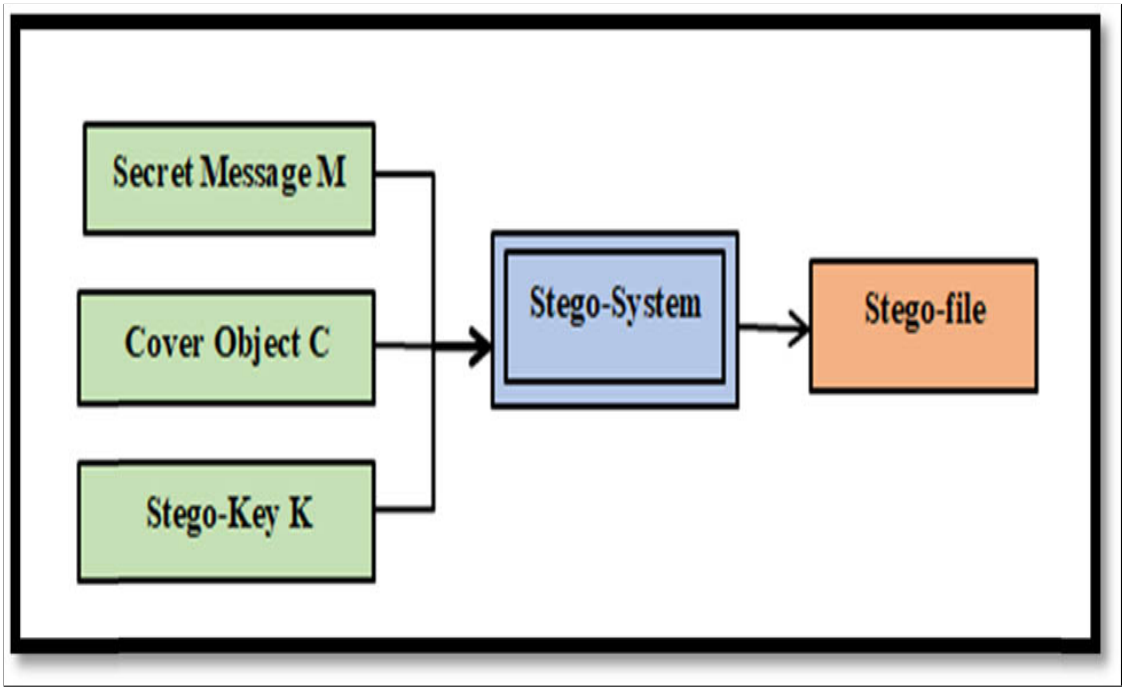

Figure 3.1: Steganography Main Concepts. Adapted from [2]

3.0.1 Cover Image

A cover image is the image in which the secret message is hidden. The cover image is chosen for its aesthetic complexity and resemblance to the original image. The cover image remains unmodified in image steganography, while the hidden message is embedded within it. The cover image can be in any format, including JPEG, PNG, and BMP.

3.0.2 Secret Message

The cover image's secret message is the message we want to conceal. The hidden message can take any form, including text, audio, video, or images. The secret message's size is determined by the capacity of the cover image to store it. To ensure confidentiality and integrity, the secret message is encrypted using a cryptographic technique.

3.0.3 Steganography Algorithm

The steganography algorithm is used to incorporate the hidden message into the cover image. Least Significant Bit (LSB), Pixel Value Differencing (PVD), and Spread Spectrum Steganography are several steganography algorithms. The LSB algorithm substitutes the least significant bit of each pixel in the cover image with the secret message's corresponding bit. The PVD algorithm embeds the secret message in the cover image by comparing the difference between neighbouring pixels. By spreading the hidden message across various frequency channels, the Spread Spectrum Steganography method embeds it.

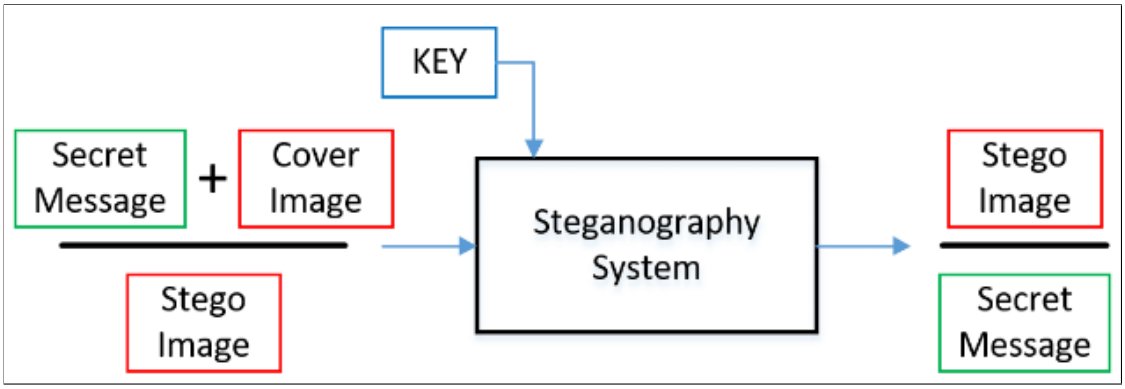

Figure 3.2: LSB-based image steganography system. Adapted from [3]

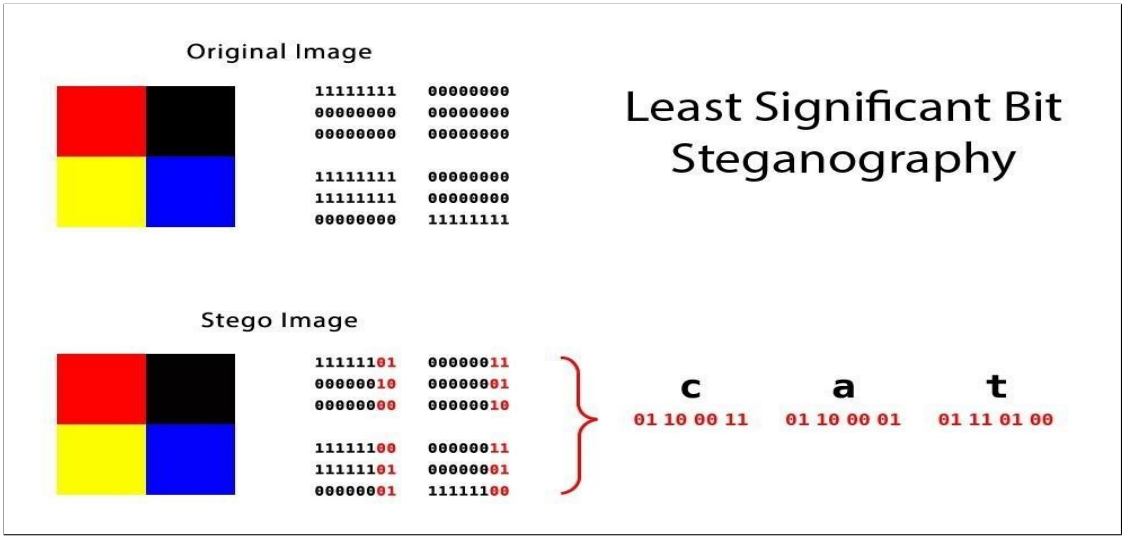

Figure 3.3: LSB Steganography [4]

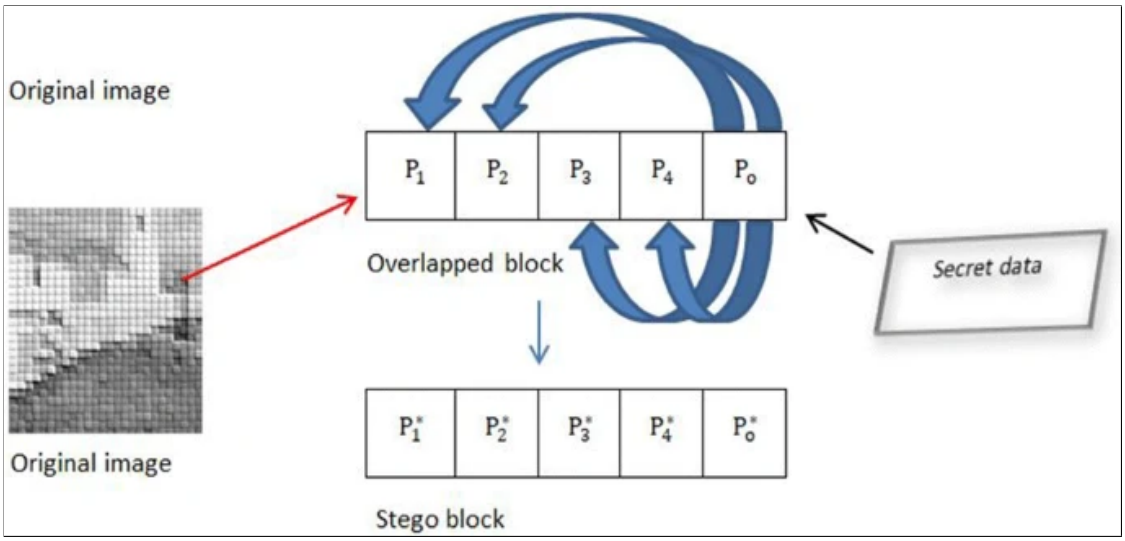

Figure 3.4: Pixel Value Differencing Image Steganography. Adapted from [5]

3.0.4 Steganography Key

A steganography key is a secret key that is used to encrypt the hidden message before it is included in the cover image. The steganography key is used to maintain the hidden message's confidentiality. Only the sender and intended recipient of the communication have access to the key. The hidden message is encrypted and decrypted using the steganography key. It is also used to choose which pixels in the cover image will contain the hidden message.

3.0.5 Key Takeaways

In summary, the cover picture, secret message, steganography algorithm, and steganography key are the essential ideas of image steganography. These ideas are crucial for understanding how image steganography works and how to use it safely.

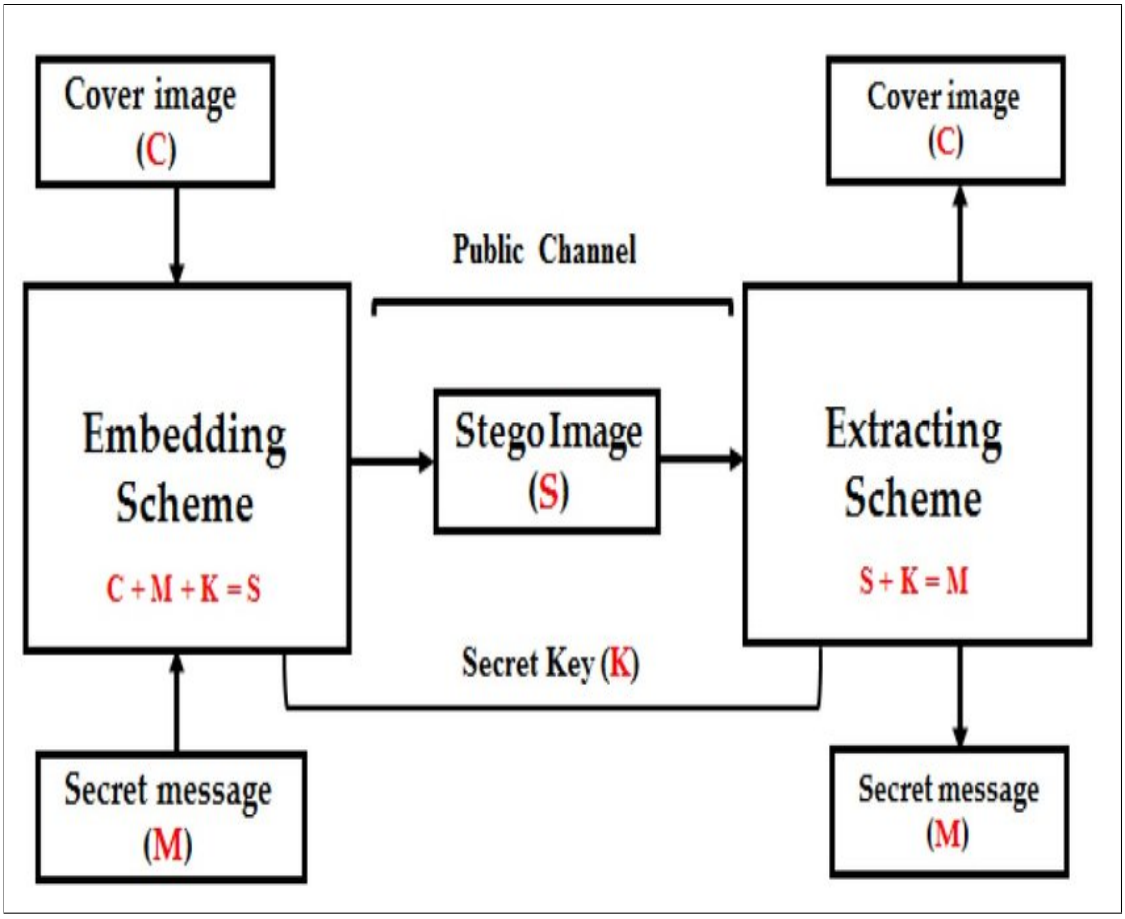

Figure 3.5: The basic concept of the steganography arrangement. Adapted from [6].

Chapter 4

Main Components

The four major components of steganography are embedding, extraction, cryptography, and user interface. The secret message is embedded, extracted, and encrypted using cryptography, and the user interface allows for system interaction. It is critical to grasp these crucial components in order to use steganography successfully and safely.

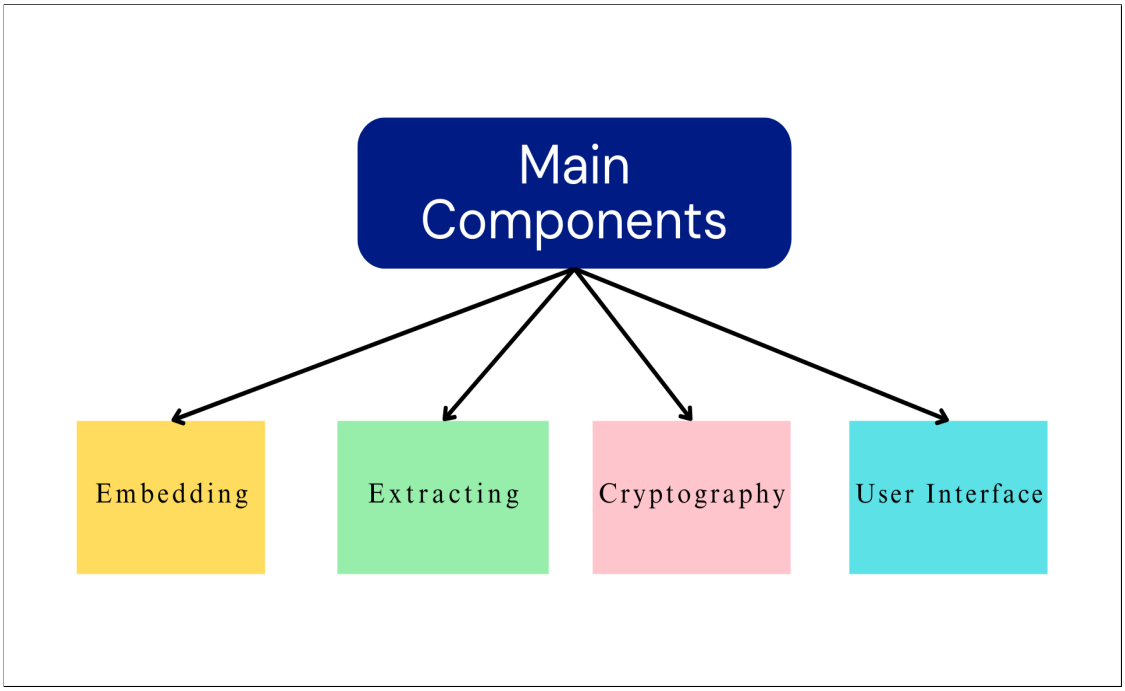

Figure 4.1: Steganography Main Components

4.0.1 Module for Embedding

Using techniques like as LSB and PVD, the embedding module conceals the hidden message in the image file. The least significant bit of each pixel is replaced with a bit from the secret message using LSB, whereas PVD alters the difference between pixel values in consecutive pixels. PVD can embed the message more effectively but may distort the image, whereas LSB has little influence on image quality but is open to assaults.

The embedding module usually consists of the following steps:

- Choose a carrier image and a hidden message.
- Convert the encrypted message to binary format.
- Disassemble the carrier image into individual pixels.
- Using the embedding procedure, replace the bits of the carrier picture with the bits of the hidden message.
- Save the changed image that contains the hidden message.

4.0.2 Extraction Module

The extraction module is in charge of obtaining the hidden message from the image file. The extraction procedure entails analyzing the image file and recovering the hidden message bits that were encoded in it. To retrieve the secret message, the extraction module employs the same procedures as the embedding module. To ensure that the extracted message is accurate and complete, the extraction module includes error correction methods.

The extraction module usually consists of the following steps:

- Load the carrier picture with the hidden message.
- Using the extraction algorithm, extract the bits of the secret message.
- Convert the extracted bits back to the secret message format.
- Using error correction methods, check the secret message's integrity.
- Show the user the extracted secret message.

A.LSB Embedding Algorithm

Input: photo cover (p) $\in P^L$, secret message (x) $\in \{0,1\}^m$, key (k)

Output: stego photo (g) with x implanted note bits

PRNG Seed with k

route = comb (l);

//comb (l) is a pseudo-random combination of {1,2,..l}

g = c;

z = min(x,n);

for j = 1 to z {

g[route[j]] = c[route[j]] + m[i] - c[route[j]] mod 2;

}

B.LSB Extracting Algorithm

Input: stego image (g) $\in P^L$, key (k)

Output: secret message (msg)

PRNG Seed with k

route = comb (l);

//comb (l) is a pseudo-random combination of {1,2,..l}

for i = 1 to m {

msg[i] = s[route[i]] mod 2;

}

Figure 4.2: LSB Embedding and Extracting Algorithms. Adapted from [2]

4.0.3 Module of Cryptography

The cryptography module is in charge of encrypting the secret message before embedding it in the picture file. The cryptography module ensures that the secret message is secure and that only the intended recipient has access to it. To encrypt the secret message, the cryptography module employs several encryption methods such as Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Rivest-Shamir-Adleman (RSA).

Typically, the cryptography module includes the following steps:

- Choose a secret message to encrypt.
- Decide on a cryptography algorithm and create a key
- Using the key and the cryptography technique, encrypt the secret message.
- Using the embedding module, insert the encrypted message into the carrier image.
- Distribute the key to the appropriate recipient

4.0.4 User Interaction

The user interface gives the steganography tool a graphical user interface (GUI). The user interface allows the user to select the carrier image and the secret message, as well as to configure the encryption technique and key and to start the embedding and extraction operations. The user interface is user-friendly and intuitive, allowing even non-technical individuals to utilize the steganography programme with ease.

User interface features:

- An interface for picking the carrier image and the hidden message from files.
- An encryption configuration interface that allows you to configure the encryption algorithm and key.
- An embedding interface used to start the embedding process.
- Using the embedding module, insert the encrypted message into the carrier image.
- An extraction interface used to start the extraction process.
- The retrieved secret message is shown using a message display interface.

4.0.5 Key Takeaways

In summary, embedding, extraction, cryptography, and user interface are the four essential components of steganography. The embedding module hides the secret message within the carrier image, and the extraction module retrieves it. Before embedding the message, cryptography encrypts it, and the user interface allows users to engage with the steganography tool. Both the embedding and extraction modules require a number of processes, including error correction to ensure proper message retrieval. Overall, understanding these fundamental components is critical for using steganography efficiently and safely.

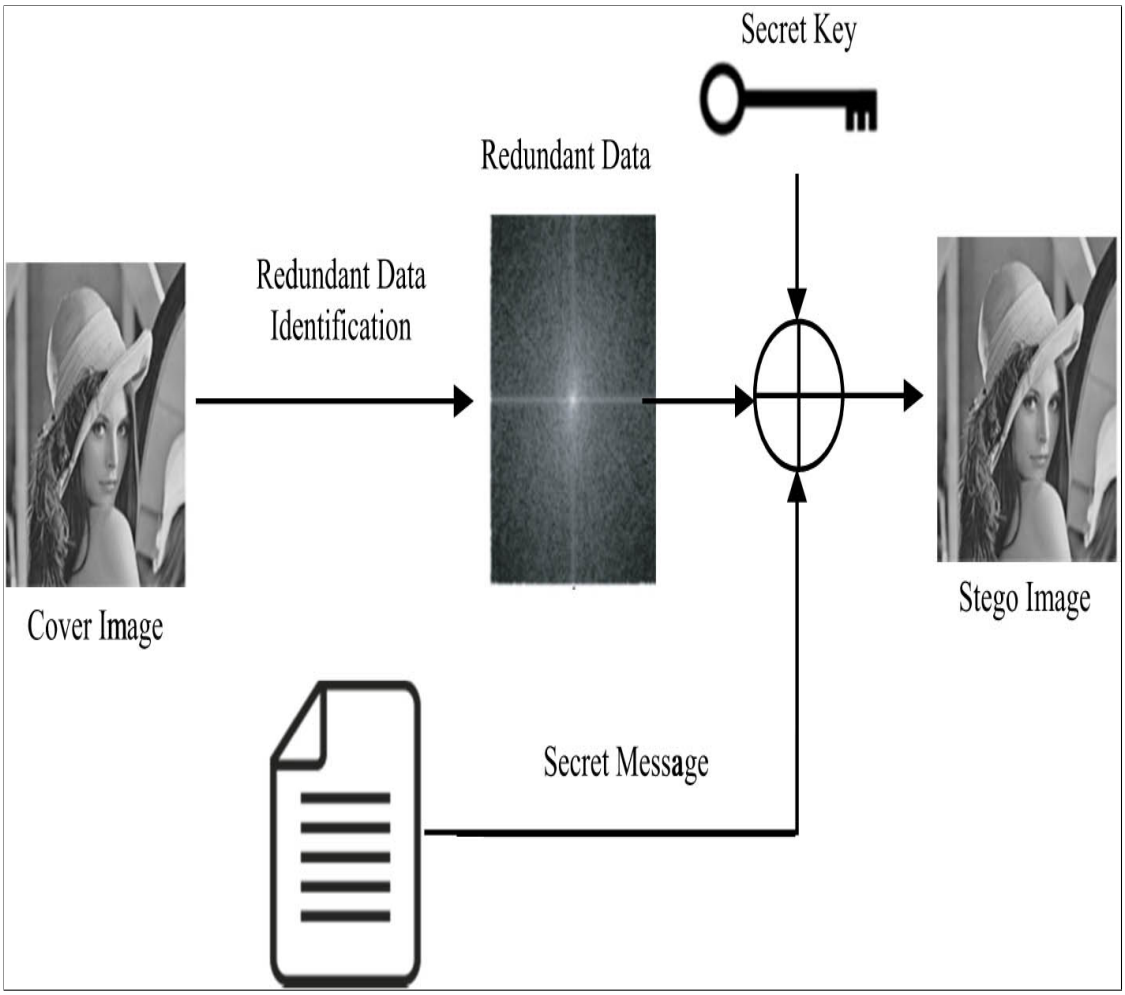

Figure 4.3: LSB System. Adapted from [2]

Chapter 5

Functional Flow

The step-by-step process of hiding a secret message behind a cover image to create a stego-image is referred to as steganography functional flow. Inputing the cover image and secret message, encrypting the secret message, embedding the encrypted message into the cover image, generating the stego-image, securely transmitting it, extracting the encrypted secret message, decrypting it, and displaying the hidden message are all common steps in this process. Understanding this functional flow is essential for using steganography for clandestine communication.

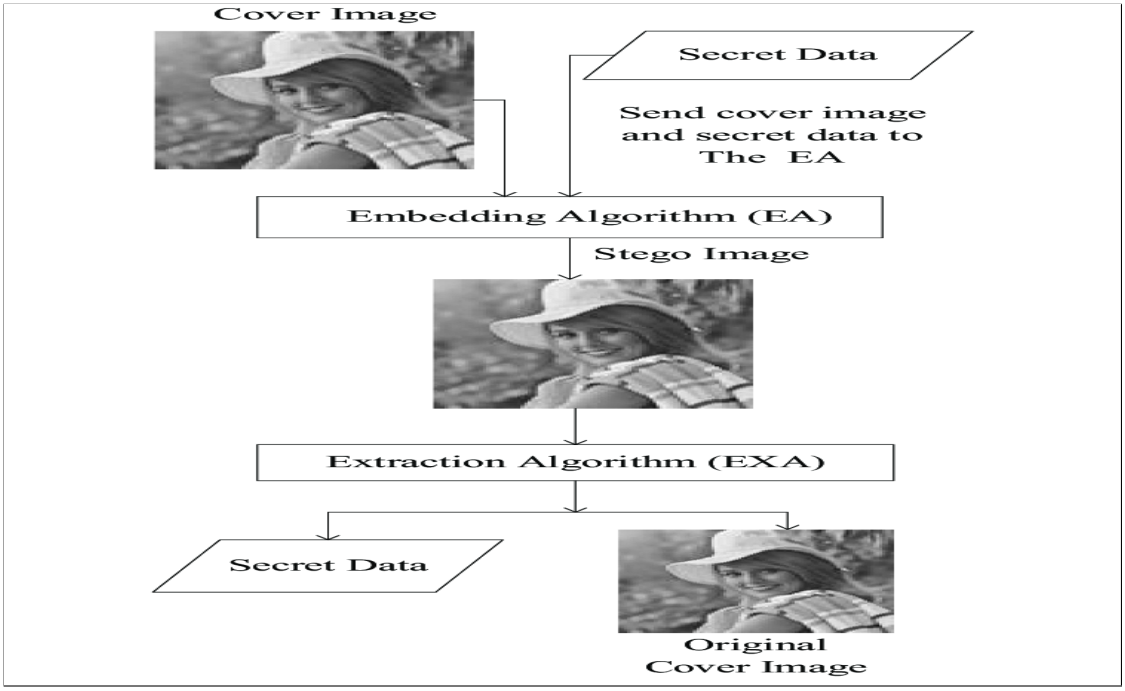

Figure 5.1: Steganography Functional Flow. Adapted from [7]

5.0.1 Input Cover Image and Secret Message

The cover image and secret message are entered into the system by the user: The user enters the input cover image and the secret message to be hidden in this phase. The cover picture can be any image that will be used as the carrier for the hidden message, such as a JPEG or PNG file. The secret message can be any data that needs to be kept private, such as text, audio, or video.

Here are some examples of cover images for an Image Steganography System:

- Image of natural scenery, such as a landscape or a beach view.
- Images of a creative nature, such as paintings or drawings.
- Photos of commonplace objects, such as a cup, a book, or a pen.
- Personal photographs, such as those of family members or pets.
- Images that are abstract, such as colourful patterns or textures.

The cover image for steganography should be carefully chosen to avoid suspicion. The chosen image should not raise any suspicions or appear suspicious. Following the submission of the cover image and secret message, the secret message is encrypted using a cryptographic technique to maintain confidentiality.

5.0.2 Secret Message Encryption

To guarantee confidentiality, the secret message provided by the user is encrypted in steganography. In the encryption process, a steganography key is utilised to generate a unique encryption pattern for each message. This key is a secret code known only to the sender and intended recipient, and it changes the arrangement of the message's bits, making deciphering difficult without it. Depending on the level of protection required and available processing power, many cryptographic methods can be used. Using a steganography algorithm, the encrypted message is subsequently inserted in the cover image.

5.0.3 Secret Message Embedding

A steganography algorithm is used to embed the encrypted secret message into the cover image. The algorithm is intended to obscure the message in an unnoticeable manner. Depending on the security level and type of cover picture, various methods such as LSB, PVD, and SS can be employed. The steganography algorithm alters particular parts of the cover image to include the encrypted message while retaining the image's aesthetic look. A stego-picture including both the original cover image and the concealed message is created. It can be viewed and sent to the intended recipient.

5.0.4 Generating the Stego-Image

A stego-picture is created by merging the cover image and changed pixels carrying the encrypted secret message. The user is then shown the stego-image to confirm that it appears natural and does not raise suspicion. To avoid unauthorised access, the stego-image should be transferred using a secure connection. The user has the option of keeping the stego-image for personal use or sending it to the intended recipient.

5.0.5 Transmitting the Stego-Image securely

The stego-image can be communicated to the receiver using secure methods such as email or messaging apps, and encryption techniques such as TLS or SSL can be used to protect the data while it is being transmitted. Password security can also be used to ensure that the stego-image is only accessible to the intended recipient. If an unauthorised entity gains access, the steganography technique and key can be utilised to extract the secret message. The recipient can utilise the extraction module to obtain the message after it has been transmitted.

5.0.6 Extracting the encrypted secret message

The recipient uses the extraction module to extract the secret message from the stego-image, which is then decrypted using the same technique and key as the cryptography module. To avoid security issues, the extracted message is displayed to the receiver, who should keep it secure and erase the stego-image.

5.0.7 Decrypting the encrypted secret message

The steganography key is used by the extraction module to decipher the embedded secret message that was encrypted by the cryptography module. The encryption and decryption modules share the same algorithm and key. The extracted message is then shown to the intended recipient, who must maintain the key secret in order for the message to be extracted from the stego-image. For the intended recipient to extract and read the buried secret message, the seventh step is critical.

5.0.8 Displaying the secret message

The original message is displayed to the receiver when the extraction module decrypts the secret message using the steganography key. The recipient can then take appropriate actions based on its contents. Because the decrypted message may contain critical information, it must be kept discreet and safe. The completion of the final step indicates that the secret message was securely transferred to the recipient and was not intercepted. The eighth and last phase of an Image Steganography System is critical for the recipient to access and interpret the secret message and take necessary actions as a result of it.

5.0.9 Key Takeaways

To summarise, the functional flow of steganography entails hiding a hidden message beneath a cover image to form a stego-image. To guarantee confidentiality, the input cover image and secret message are encrypted, and a steganography algorithm is employed to embed the encrypted message into the cover image in an imperceptible manner. Using encryption techniques and password security, the stego-image can be securely transferred to the intended recipient, and the receiver can extract and decrypt the concealed message using the steganography key. The original communication is shown to the receiver, who must keep it private and secure. Understanding the functional flow of steganography is critical for employing it for covert communication.

Chapter 6

Main existing solutions

6.1 SSIS (Spread Spectrum Image Steganography)

Spread Spectrum Image Steganography (SSIS) works by storing a message as Gaussian noise in an image (Marvel, Boncelet, and Retter 1998, Marvel et al. 1999). At low noise power levels, the image degradation is undetectable by the human eye, while at higher levels. the noise appears as speckles or "snow." The procedure is broken down into the following key steps, which are shown in figure :

- Create encoded message by adding redundancy via error-correcting code.
- Add padding to make the encoded message the same size as the image.
- Interleave the encoded message.
- Generate a pseudorandom noise sequence, n.
- Use encoded message, m, to modulate the the sequence, generating noise, s.
- Combine the noise with the original image, f.

6.1.1 Main Characteristics

- SSIS is a type of steganography that involves embedding secret data within an image without altering the perceptual quality of the image.
- It is based on the spread spectrum communication technique, which involves spreading the secret data signal over a wide bandwidth.
- Interleave the encoded message.
- The spread spectrum technique used in SSIS helps to make the embedded data more resistant to detection and removal by attackers.
- SSIS is a reversible process, meaning that the secret message can be extracted from the image without any loss of information.

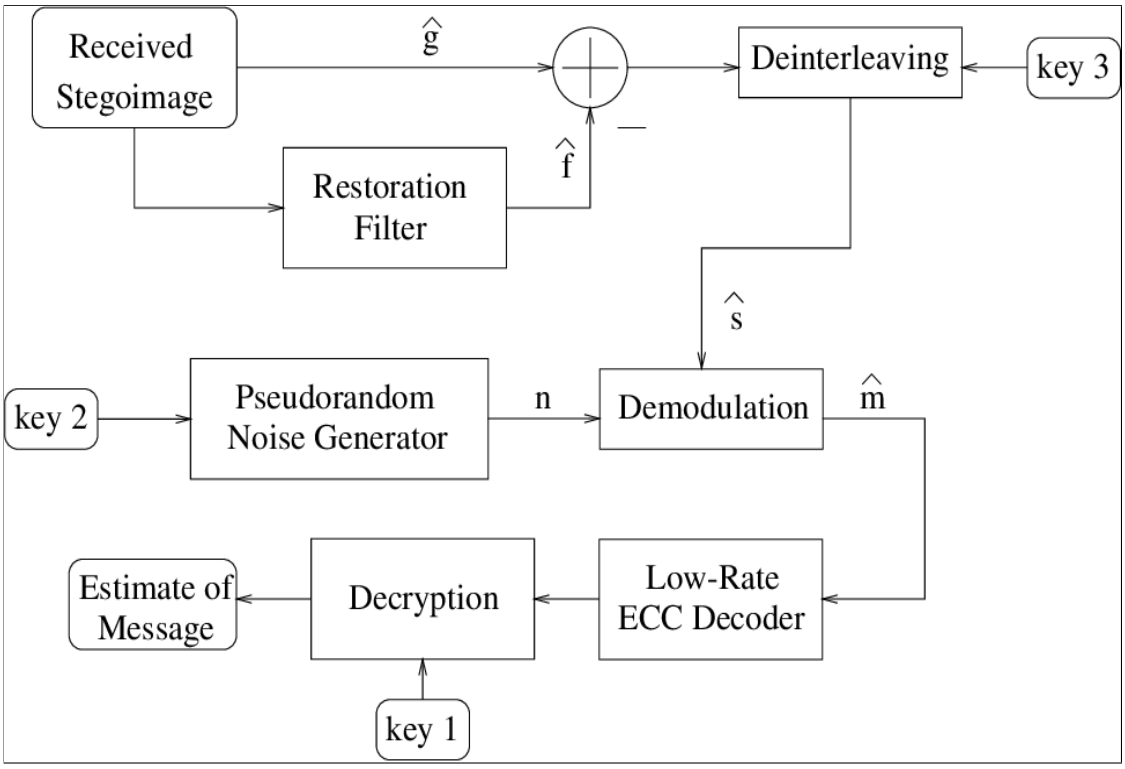

Figure 6.1: Spread spectrum image steganography. Adapted from [8]

6.1.2 Advantages

- SSIS is a robust technique for hiding secret data within an image, making it more difficult for attackers to detect and remove the embedded data.
- SSIS can be used for a wide range of applications, such as in digital watermarking, copyright protection, and covert communication.
- The use of spread spectrum communication helps to make the embedded data more resistant to attacks such as noise addition and cropping

6.1.3 Limitations

- SSIS requires a large bandwidth to spread the secret data signal, which can be a limitation in situations where bandwidth is limited or expensive.
- The use of SSIS can result in a slight degradation of the image quality, although this can be minimized through careful selection of the embedding parameters.
- SSIS can be vulnerable to attacks such as correlation and statistical analysis, which can be used to detect the presence of the embedded data.

6.2 Conclusion

In conclusion, SSIS is an effective and adaptable method for embedding hidden messages in images via spread spectrum communication. It has a large number of uses in digital watermarking, copyright protection, and clandestine communication, as well as high capacity data concealment, resistance to attacks such cropping and rotation. To prevent picture degradation, SSIS must be used with care as it is susceptible to statistical attacks. SSIS is a useful instrument for safe communication and data protection in the modern world.

6.3 BATCH steganography

Batch steganography is a technique for hiding a secret message within multiple files simultaneously, such as images, videos, or documents. The process involves dividing the secret message into small pieces and embedding each piece in a different file using a steganographic algorithm. The files containing the embedded message can be distributed publicly, and only the intended recipient who knows the extraction process can retrieve the original message.

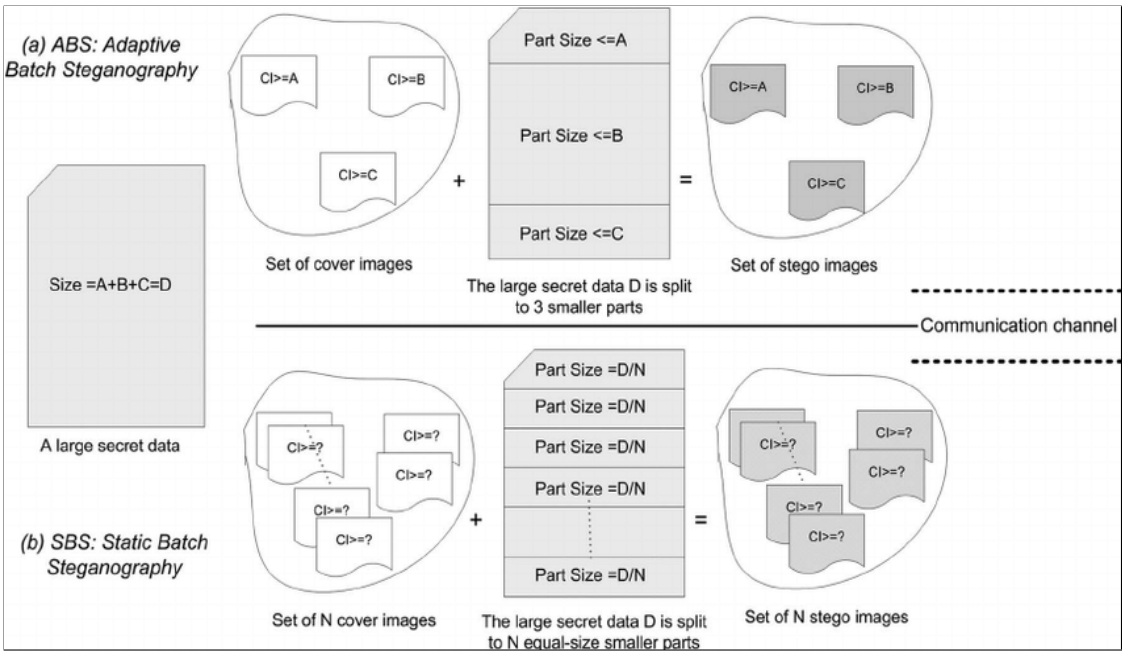

Figure 6.2: Batch image steganography. Adapted from [9]

6.3.1 Main Characteristics

- Batch steganography involves embedding a secret message within multiple files, such as images or documents, to increase the overall capacity of the message.
- Batch steganography can be used to hide a secret message in a variety of file formats, making it more difficult for attackers to detect the presence of the message.
- Batch steganography can be used to embed the secret message in a specific order, which can be used to convey additional information.

6.3.2 Advantages

- Batch steganography allows for high-capacity data hiding, as the secret message can be divided among multiple files.
- Batch steganography can be used to increase the robustness of the hidden message, as attackers would need to detect and extract the message from multiple files rather than a single file.
- Batch steganography can be used to hide the existence of the secret message, as it can be difficult for attackers to identify the specific files that contain the message.

6.3.3 Limitations

- Batch steganography can be time-consuming and resource-intensive, as it requires the embedding of the secret message in multiple files.
- The use of batch steganography can result in a loss of data fidelity or perceptual quality in the files containing the hidden message.
- The use of batch steganography may be detectable through statistical analysis, as the distribution of the data in the files may be altered by the embedding process.

6.4 Conclusion

In summary, batch steganography is a powerful technique for hiding secret messages in multiple files simultaneously, providing high-capacity data hiding and increased robustness. However, the use of batch steganography can be time-consuming and resource-intensive, and it may result in a loss of data fidelity or detectability through statistical analysis.

6.5 PIT (Pixel Indicator Technique)

The Pixel Indicator Technique is a steganography technique used to conceal secret information in the least significant bit of a pixel indicator, which is a flag that indicates whether a pixel has been updated. The hidden message is encoded in the least significant bit (LSB) of specific image pixels. These pixels are chosen based on their RGB values, and the message is embedded by changing the value of the least significant bit.

How it works:

- The cover image is separated into non-overlapping pixel blocks.
- The most significant bit (MSB) of each pixel in each block is altered to encode a secret message.
- Each pixel in the block's least significant bit (LSB) is updated to indicate whether or not it includes concealed information.

Diagram:

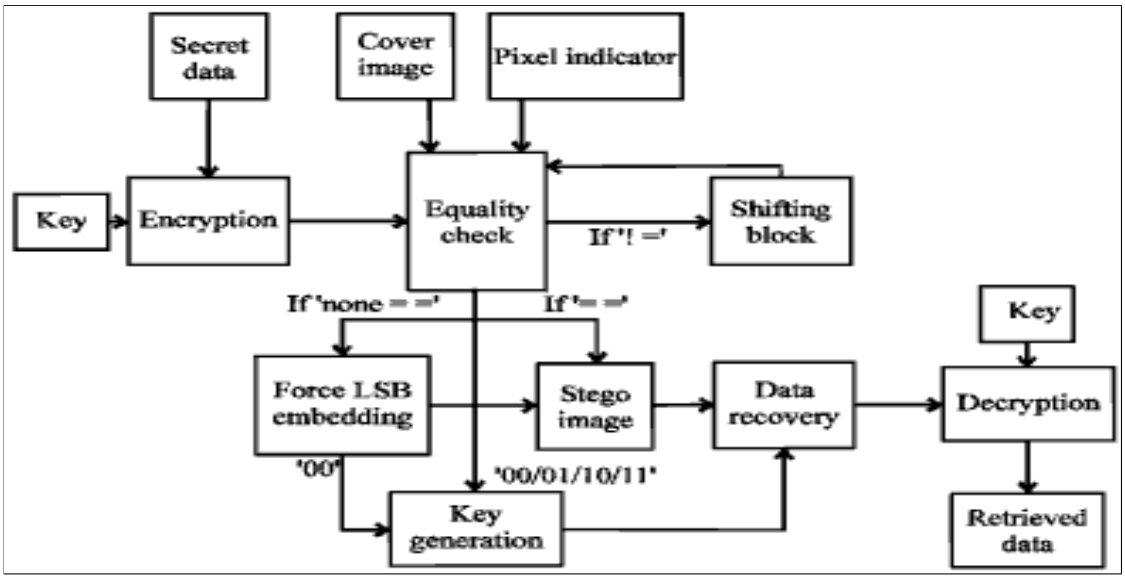

Figure 6.3: Block Diagram for PIT. Adapted from [10]

.

Hiding Process:

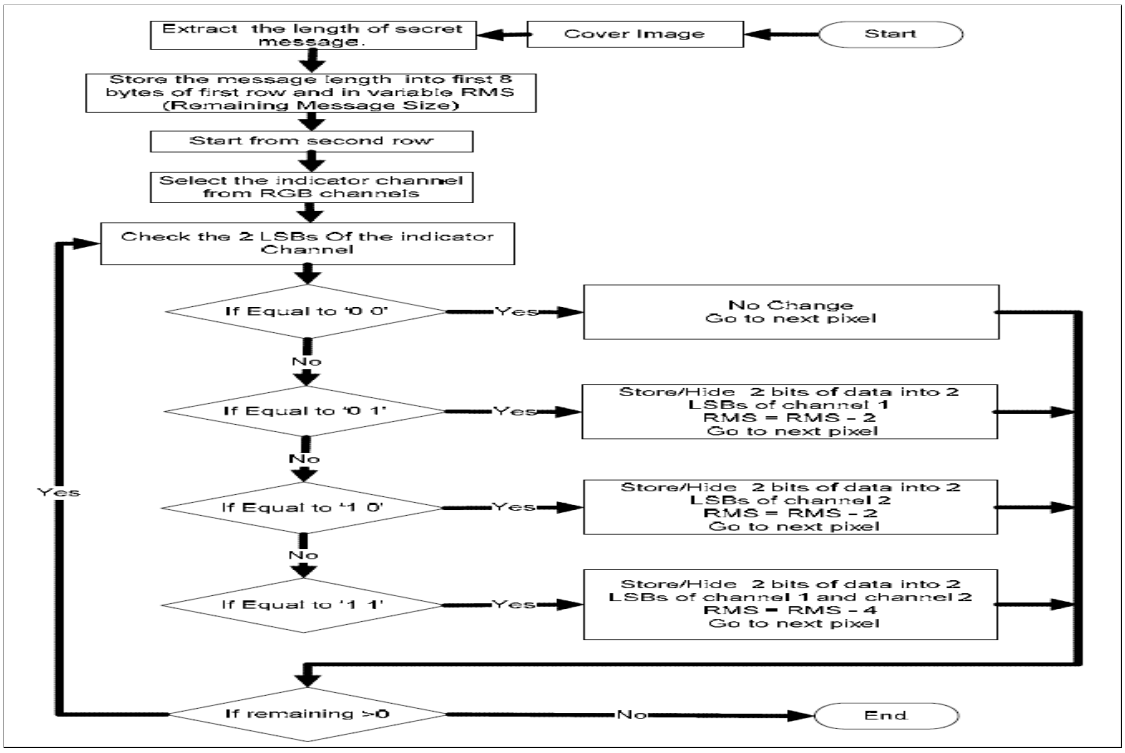

Figure 6.4: PIT Hiding Process. Adapted from [11]

Extraction Process:

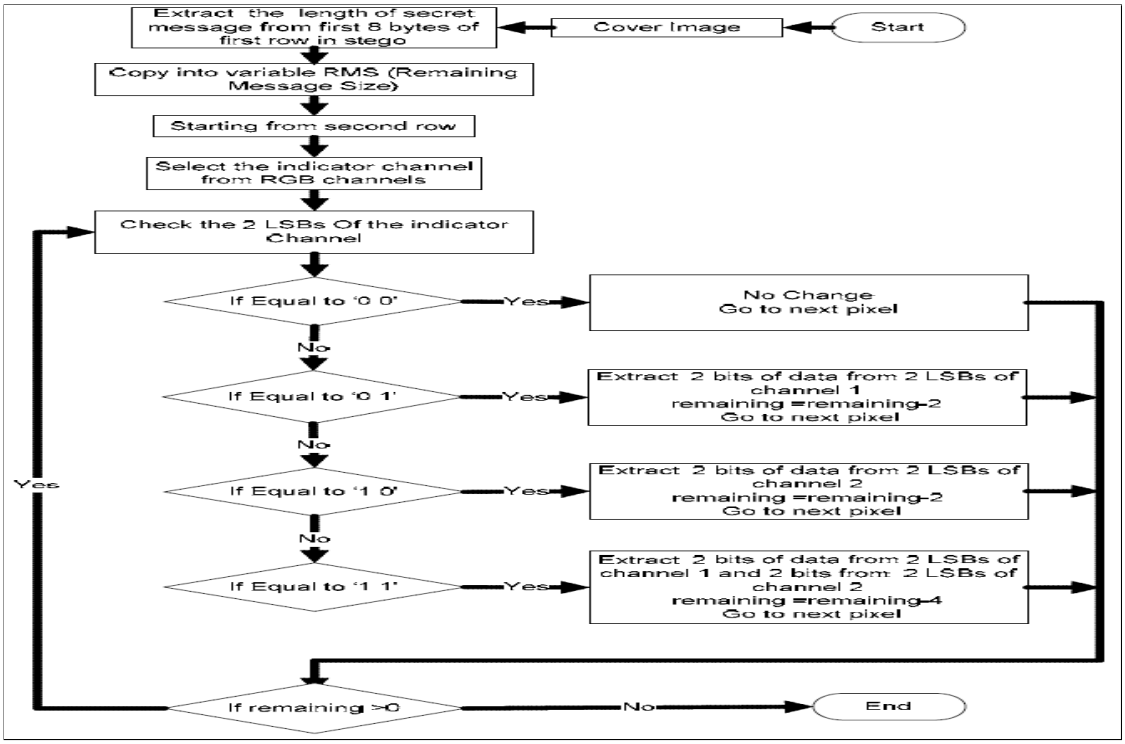

Figure 6.5: PIT Extractions Process. Adapted from [11]

6.5.1 Main Characteristics

- The embedding procedure is simple and quick.
- The resulting stego-image resembles the original image in appearance.
- The method is resistant to attacks that change the image on a global scale.
- Without the original cover image, the message can be extracted.

6.5.2 Advantages

- High capacity for message embedding
- The cover image has minimal distortion. from multiple files rather than a single file.
- Simple to implement and use
- Effective against global image attacks

6.5.3 Limitations

- Vulnerable to attacks that specifically target the pixels used for embedding
- Not appropriate for highly compressed images
- If too many pixels are modified, image quality may suffer.

6.6 Conclusion

Despite its limitations, the Pixel Indicator Technique remains a popular and effective method of steganography due to its ease of use, high capacity, and resistance to attacks. It is important to note, however, that the success of this technique is heavily dependent on the indicator matrix chosen, and it may not be appropriate for all types of cover images.

6.7 BPCS (Bit-Plane Complexity Segmentation)

The Bit-Plane Complexity Segmentation is a type of steganography in which a secret message is embedded in an image's complex bit planes. The complex bit planes are chosen for their randomness and complexity, making detection of the embedded message difficult.

How it works:

- Select the cover image and secret message
- Analyze the image and select the complex bit planes
- Embed the secret message into the complex bit planes using a specific algorithm.
- Send the stego-image to the receiver
- Extract the secret message by reversing the embedding algorithm

Hiding and Extracting Data: The BPCS approach conceals data by partitioning the image into non-overlapping blocks and embedding bits in each block's most complicated bit-planes. The receiver must first identify the complex bit-planes and block locations in order to retrieve the secret data.

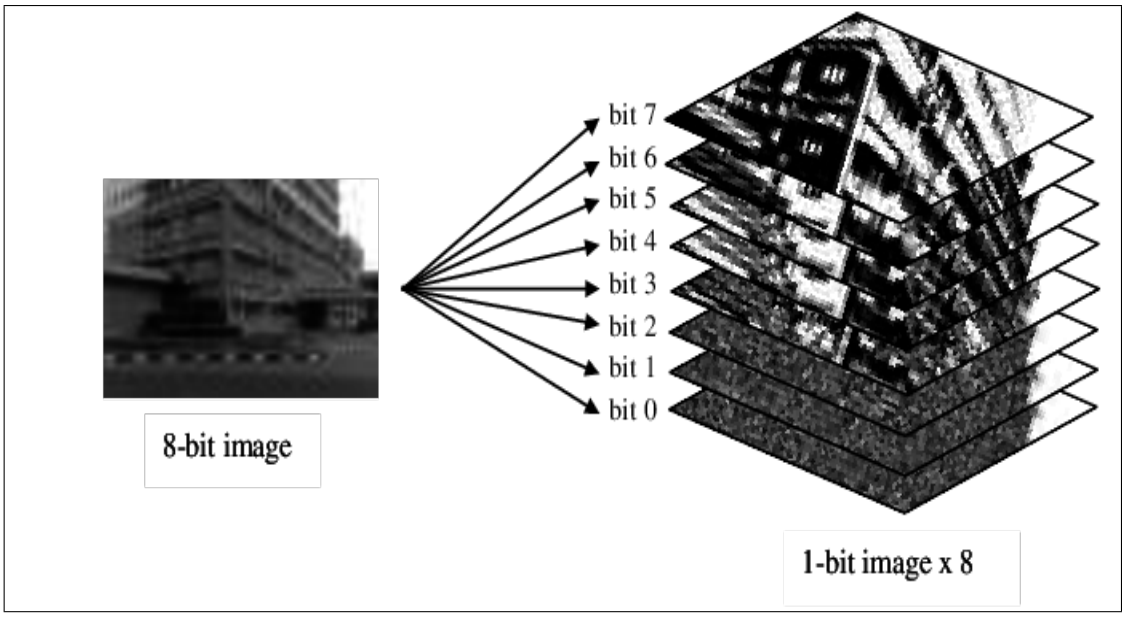

Figure 6.6: 8-bit image is decomposed into 8 binary images prior to the embedding process. [12]

6.7.1 Main Characteristics

- The embedding procedure necessitates a significant amount of processing power.
- The approach is immune to attacks that change the image on a global scale.
- Without the original cover art, the message can be extracted.
- Without understanding of the embedding algorithm, detecting the approach is challenging.

6.7.2 Advantages

- High capacity for message embedding.
- The approach is immune to attacks that change the image on a global scale.
- Without understanding of the algorithm, detecting the message is challenging.
- Suitable for photos that have been heavily compressed.

6.7.3 Limitations

- The embedding procedure is time-consuming and computationally demanding.
- The procedure may cause distortion to the cover image.
- Attacks that target the precise bit planes utilised for embedding are possible.
- The process is more difficult to implement and employ than other steganography techniques.

6.8 Conclusion

In conclusion, the Bit-Plane Complexity Segmentation technique has been demonstrated to be a reliable and secure steganography method with high capacity and resistance to attacks. However, the requirement for large cover images and longer embedding times may limit its use. It is also worth noting that the technique's success is dependent on the right selection of threshold values, which can affect the visual quality of the stego image.

6.9 LSB (Least Significant Bit)

In LSB (Least Significant Bit) steganography, the secret message bit is substituted for the least significant bit of each pixel value in the cover image. It is a preferred option for hiding hidden messages since the changes are frequently modest and hard to notice visually.

How it works:

- Retrieve the binary representation of the secret message.
- Begin with the first pixel in the upper-left corner of the cover image.
- For each bit of the secret message, replace the least significant bit of the current pixel with the bit of the secret message.
- Move to the next pixel in the cover image and repeat step 3 until all the bits of the secret message have been embedded.
- If the end of the cover image is reached before all the bits of the secret message have been embedded, wrap around to the beginning of the cover image and continue embedding the remaining bits.
- The resulting image can be transmitted or stored with the hidden message embedded within it. To extract the message, the same process is used to retrieve the least significant bits of the pixels and convert them back to the original message bits.

Diagram:

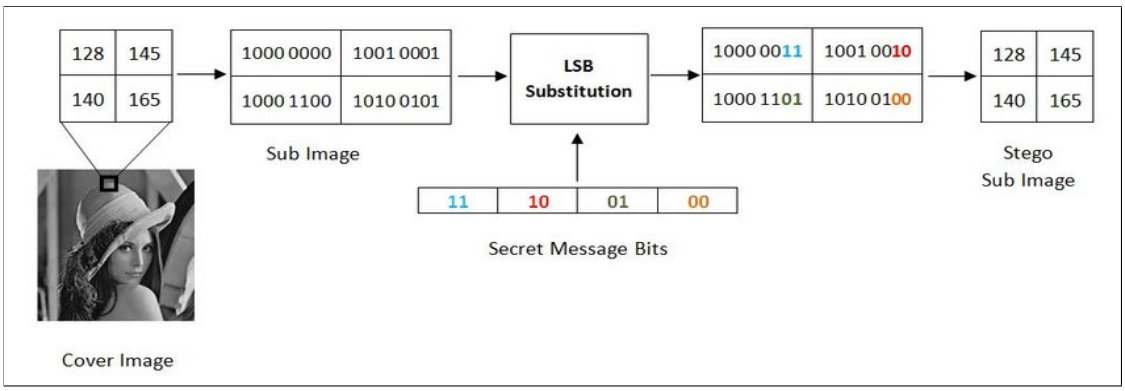

Figure 6.7: Digram of LSB substitution techniques. [13]

Hiding Process:

- Put the cover photo and the hidden message in binary format.
- Determine the number of LSBs that can be changed per RGB color channel without affecting the cover picture, which is usually 1-2 bits.
- Divide the secret message into equal-sized pieces corresponding to the number of LSBs in the cover picture.
- Change the LSBs of the cover picture for each message piece to match the binary values of that piece.
- Save the updated picture as the stego-image.

Extraction Process:

- Put the stego-image on.
- Find the quantity of least significant bits (LSBs) needed to encrypt the secret message.
- The binary values of the concealed message chunks may be obtained by separating the LSBs from each pixel in the stego-image.
- The secret message may be obtained by combining the binary parts.

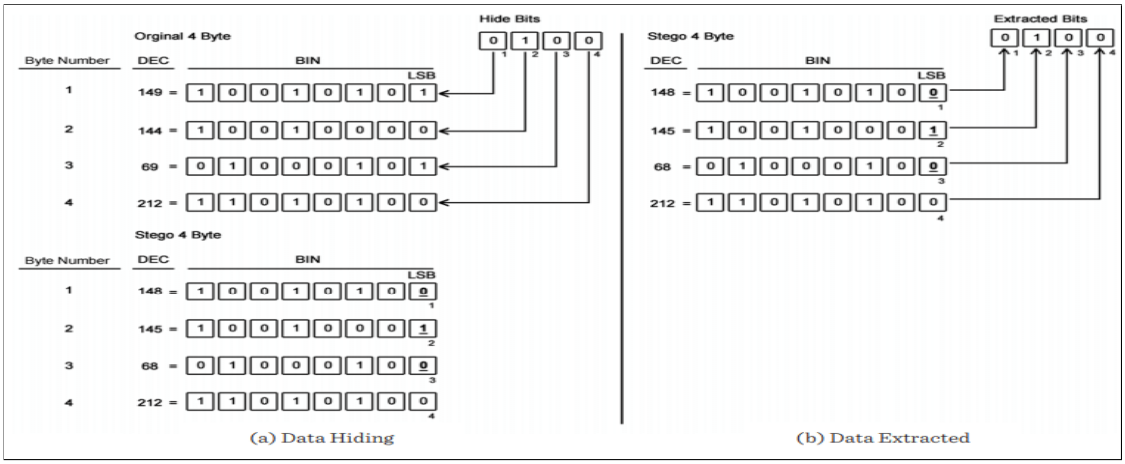

Figure 6.8: Hiding and extracting data with LSB. Adapted From [14]

6.9.1 Main Characteristics

- A straightforward and effective steganography method is LSB.
- It may be used with a variety of cover material, including as pictures, audio, and video.
- The changes to the cover picture are often subtle and undetectable to the naked eye.

6.9.2 Advantages

- Implementing LSB steganography is simple and doesn't require a lot of computer power.
- Without noticeably lowering the image's quality, it may conceal a lot of info in cover photos.
- Many software tools and libraries support and use LSB steganography.

6.9.3 Limitations

- Attacks that look for concealed messages via statistical analysis can find them using LSB steganography.
- It is also susceptible to attacks that change the cover image's LSBs, which might lead to the concealed message being lost or being shown to unauthorized people.
- Because LSB steganography is not very reliable, even minor changes to the cover image can result in the hidden message being lost or altered.

6.10 Conclusion

Finally, LSB steganography is a frequently used and simple technique for concealing data within digital photographs. It has various advantages, including high embedding capacity and message imperceptibility. However, it has several limitations, including vulnerability to attacks, image quality degradation, and the inability to embed large messages.

6.11 PVD (Pixel-Value Differencing)

The hidden message is concealed using the PVD (Pixel-Value Differencing) steganography method by using the difference between adjacent pixel values in the cover picture. Compared to LSB steganography, PVD steganography is more durable and more resistant to statistical analysis-based assaults.

How it works:

1. Calculate the difference between adjacent pixel values in the cover picture.
2. Alter the difference value between two successive pixels to insert the hidden message.
3. Ensure that the modifications are minor, and the difference values stay within a predetermined range.

Diagram:

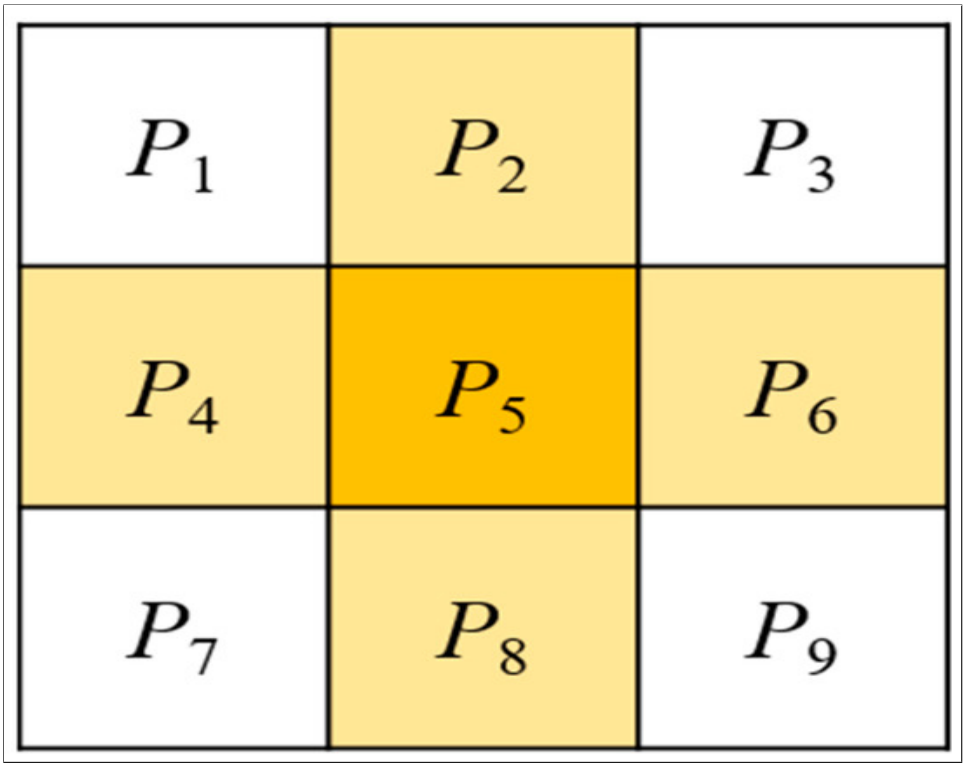

Figure 6.9: PVD diagram of a 3 × 3-pixel block. Adapted From [15]

Hiding Process:

- Create a binary bit stream from the secret message.
- Create non-overlapping blocks of size m x n.
- Compute the edge maps for each block using an edge detection operator.
- Change the LSB of a pixel with a strong edge to the corresponding bit of the binary stream for each block.
- Repeat step 4 for each bit in the binary stream.
- Combine all the changed blocks to create the stego-image.

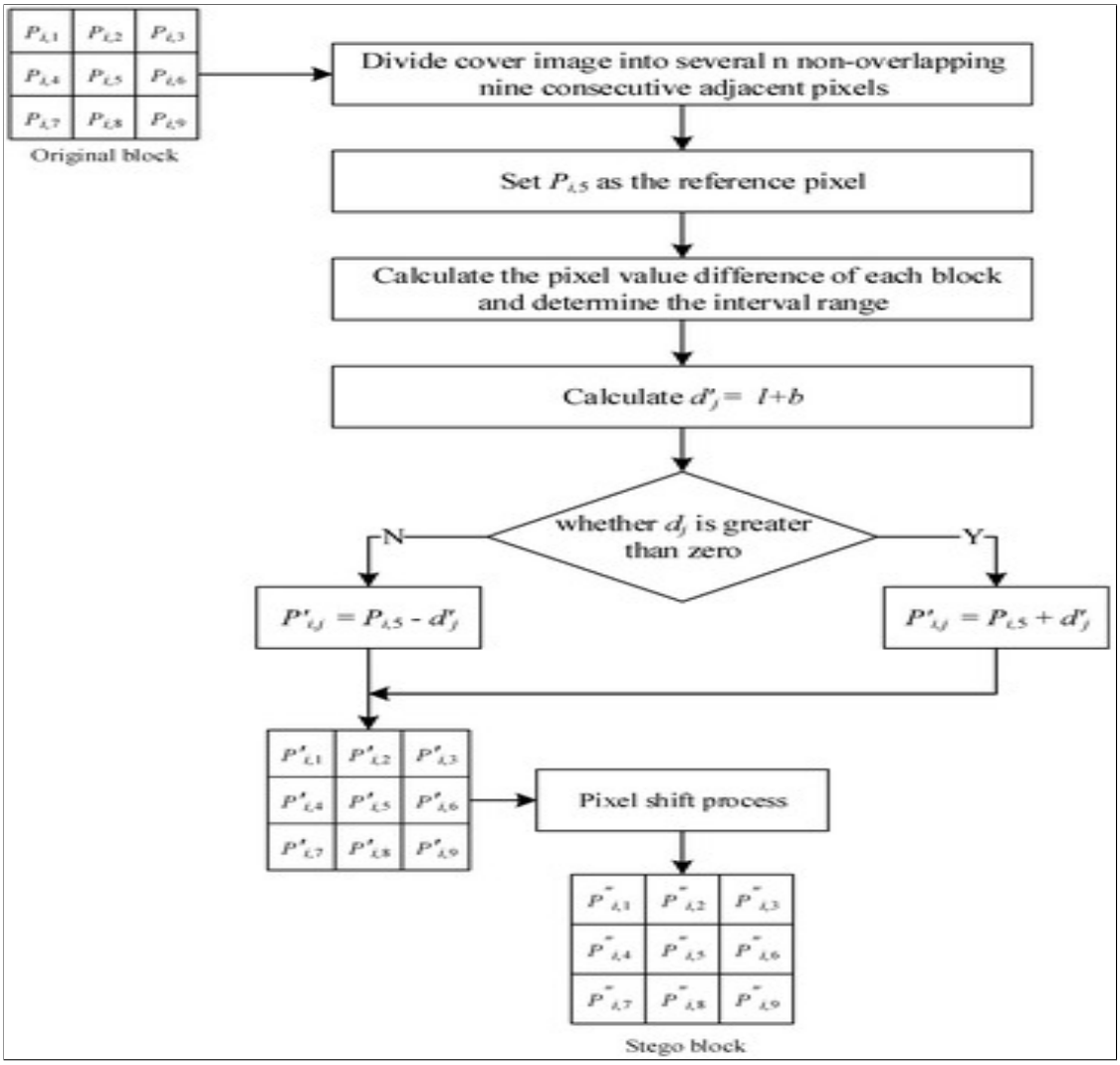

Figure 6.10: PVD embedding procedure. Adapted From [15]

Extraction Process:

- Create non-overlapping blocks of size m x n from the stego-image.
- Compute the edge maps for each block using an appropriate edge detection operator.
- Calculate the difference between the original and stego versions of each block.
- Concatenate the LSBs of the difference values from step 3 to reveal the hidden message.

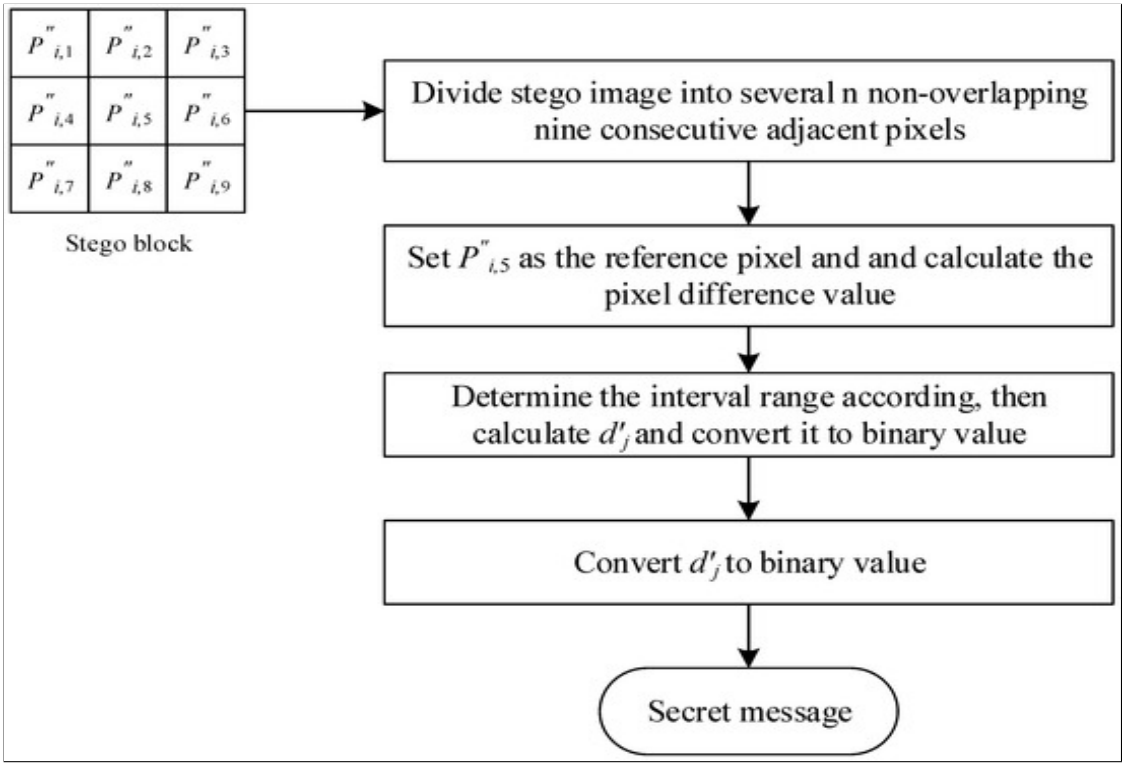

Figure 6.11: PVD extracting procedure. Adapted From [15]

6.11.1 Main Characteristics

- A straightforward and effective steganography method is LSB.
- It may be used with a variety of cover material, including as pictures, audio, and video.
- The changes to the cover picture are often subtle and undetectable to the naked eye.

6.11.2 Advantages

- Implementing LSB steganography is simple and doesn't require a lot of computer power.
- Without noticeably lowering the image's quality, it may conceal a lot of info in cover photos.
- Many software tools and libraries support and use LSB steganography.

6.11.3 Limitations

- Attacks that look for concealed messages via statistical analysis can find them using LSB steganography.
- It is also susceptible to attacks that change the cover image's LSBs, which might lead to the concealed message being lost or being shown to unauthorized people.
- Because LSB steganography is not very reliable, even minor changes to the cover image can result in the hidden message being lost or altered.

6.11.4 Conclusion

Finally, PVD steganography is a more advanced technique that seeks to overcome the limitations of LSB. It improves security, imperceptibility, and resistance to attacks. It does, however, have limits, such as a reduced embedding capacity and the complexity of the encoding and decoding operations. .

Chapter 7

Critique of existing solutions

7.1 SSIS Criticism

Attacks that take advantage of well-known weaknesses in the encryption technique may leave SSIS open to risk. For instance, to decrypt the concealed message, an attacker might be able to estimate the encryption key or take advantage of algorithmic weaknesses. To guarantee that SSIS is utilized properly and that the intended level of security is attained, careful evaluation of these restrictions is required. Overall, SSIS can be a useful steganography technique, but it's important to weigh its advantages and disadvantages when deciding which steganography method is best for a given application.

7.2 BATCH steganography Criticism

With the ability to embed and extract messages from many files, batch steganography can be more difficult to build and utilize than single-file steganography systems. To use batch steganography effectively and achieve the needed level of security, several restrictions must be carefully taken into account. Overall, batch steganography can be a useful steganography technique, but it's crucial to weigh its advantages and disadvantages when deciding which steganography method is best for a given application.

7.3 PIT Criticism

While PIT has some benefits, such as being easy to implement and not requiring the original image for extraction, it also has some significant drawbacks. PIT is vulnerable to statistical analysis attacks, in which an attacker can detect hidden information by analysing statistical features of the cover image. Furthermore, PIT has a limited embedding capacity, which means it can only hide a limited amount of information in a single image.

7.4 BPCS Steganography Criticism

While BPCS Steganography has some advantages, such as high security and embedding capacity, it also has some significant drawbacks. One of the main drawbacks of BPCS is that it requires a larger cover image than other steganography techniques, making it unsuitable for some applications. Furthermore, the embedding process can be computationally demanding, which can be difficult when working with large images or datasets. Finally, BPCS is vulnerable to some attacks, such as the RS analysis attack, which can reveal the presence of hidden data in the cover image by analysing the relationship between bit planes.

7.5 LSB Criticism

Because of its simplicity, the LSB approach is commonly employed, although it has a limited capability for hiding data in high-resolution photographs and might create visual distortion to the cover image, making concealed information simpler to discover. It's also susceptible to compression, which can lead to the loss or corruption of buried data.

7.6 PVD Criticism

PVD has advantages such as increased data hiding capacity and less aesthetic distortion to the cover image, as well as being compression resistant. It does, however, have drawbacks such as high computer resource requirements and sensitivity to noise and signal changes. Furthermore, due to specific requirements for colour space and image size, the PVD method may not be appropriate for some types of cover images or applications.

Chapter 8

SSIS UML Design

8.1 Use case diagram

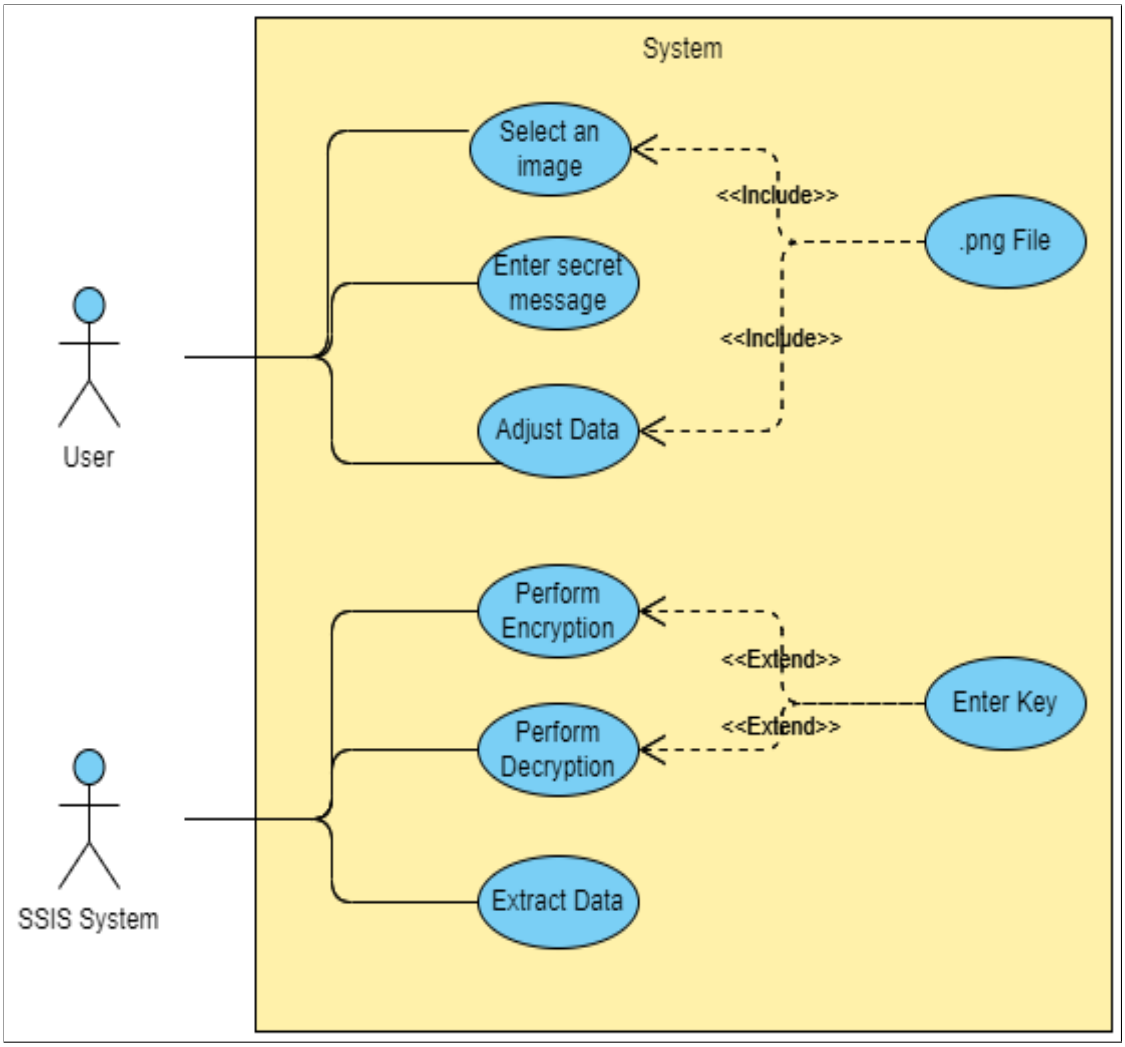

Figure 8.1: SSIS Use Case Diagram.

8.2 Sequence diagram

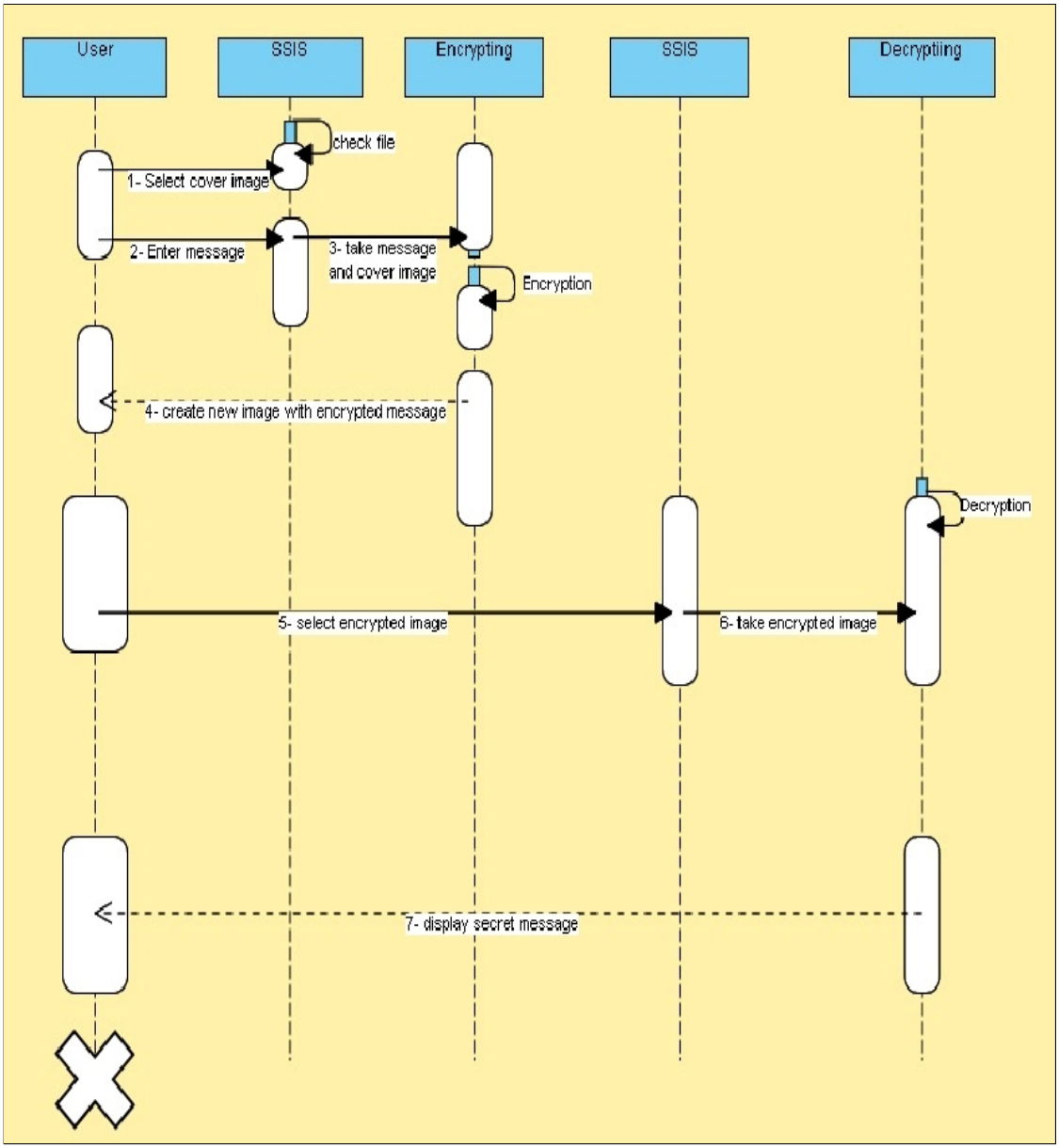

Figure 8.2: SSIS Sequence Diagram.

8.3 Activity diagram

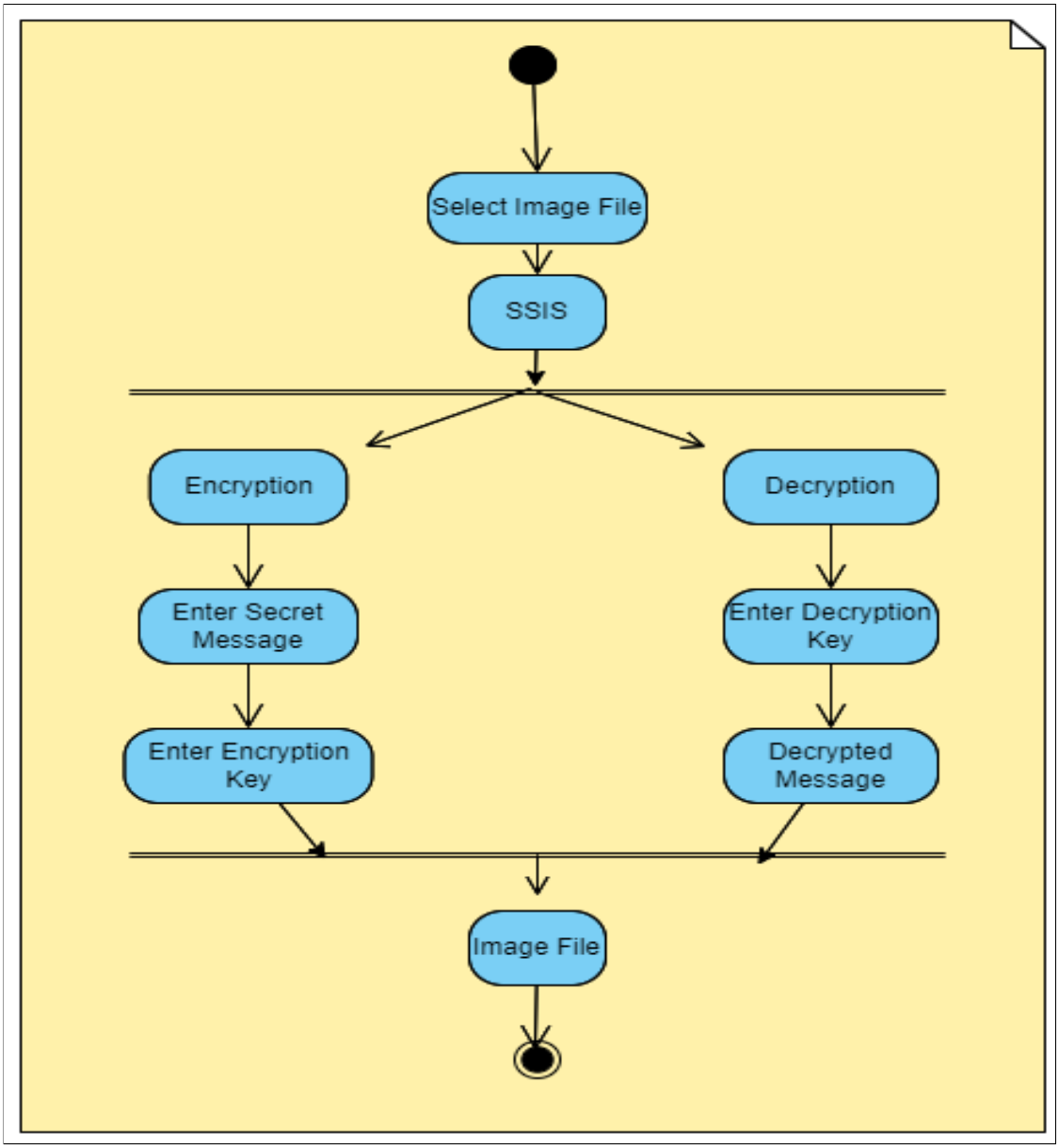

Figure 8.3: SSIS Activity Diagram.

8.4 Class diagram

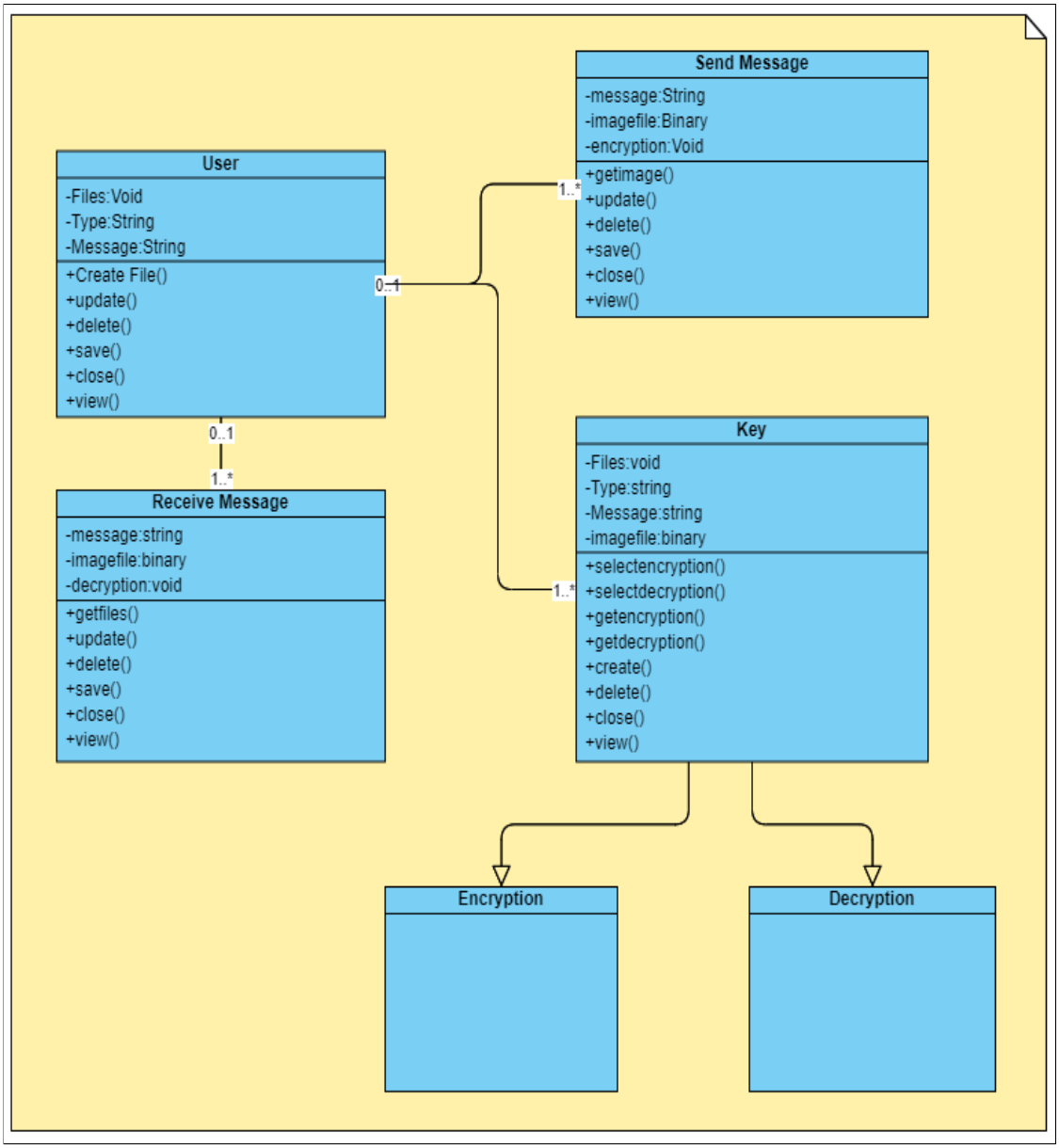

Figure 8.4: SSIS Class Diagram.

Chapter 9

SSIS Requirements Analysis

9.1 Functional requirements

9.1.1 Cover image selection and processing

The SSIS system should allow the user to choose a suitable cover image, process it to ensure that it can be used for steganography, and ensure that it has enough area to embed the hidden message.

9.1.2 Secret message encoding and encryption

To ensure confidentiality, the system should encode the secret message in a way that it can be readily inserted into the cover image and then encrypt it using a safe algorithm.

9.1.3 Embedding algorithm and SSIS application

The SSIS system should have a strong embedding algorithm capable of inserting the encrypted message within the cover image without significantly modifying it.

9.1.4 Extraction algorithm and SSIS application

The system should have a trustworthy extraction algorithm capable of extracting the secret message from the stego-image.

9.1.5 Decryption and decoding of the secret message

The system should include a safe decryption algorithm capable of decrypting the encrypted message and revealing the original secret message.

9.2 Non-functional requirements

9.2.1 Performance

The system should efficiently complete the embedding and extraction processes without generating substantial delays or compromising the quality of the cover image.

9.2.2 Security

The system should ensure that unauthorised users cannot discover the embedded message and that the decrypted message stays confidential.

9.2.3 Scalability

The system should be scalable and capable of handling a high volume of cover photos and texts.

9.2.4 User-friendliness

The system should be simple to operate and come with detailed instructions.

9.3 Key Takeaways

In conclusion, The SSIS system should provide a dependable and secure means of inserting hidden messages into cover images while maintaining confidentiality and authenticity.

Chapter 10

SSIS High-Level Design

10.1 Architecture overview

The SSIS system consists of five main components: Cover image selection and processing, Secret message encoding and encryption, Embedding algorithm and SSIS application, Extraction algorithm and SSIS application, and Decryption and decoding of the secret message.

10.2 Cover image selection and processing

The Cover Image Selection and Processing component of the SSIS system is responsible for selecting an appropriate cover image and processing it to ensure that it is suitable for embedding the secret message. The component may involve selecting an image with specific characteristics such as high resolution, minimal texture, or specific color combinations that can be used for steganography.

For example, in the SSIS system, the cover image may be selected based on the resolution and aspect ratio of the image. The image may be processed to ensure that it has enough space to embed the secret message and is in a suitable format for steganography.

10.3 Secret message encoding and encryption

The SSIS system's Secret Message Encoding and Encryption component is in charge of encoding the secret message so that it may be easily incorporated into the cover image and encrypting it with a safe algorithm to assure confidentiality.

In the SSIS system, for example, the secret message might be encoded using a binary code that can easily be integrated into the cover image without materially affecting the image. To maintain confidentiality, the message can then be encrypted using a secure technique such as AES.

10.4 Embedding algorithm

The SSIS system's Embedding technique and SSIS Application component is in charge of embedding the encrypted message into the cover image using a robust technique. The component must ensure that the message is integrated in such a way that unauthorised users cannot discover it and that it does not significantly affect the cover image.

In the SSIS system, for example, the message may be embedded using a Least Significant Bit (LSB) technique, which substitutes the least significant bits of the cover picture pixels with the encrypted message bits. The SSIS application may also have extra features such as the ability to choose the embedding rate, which controls how much data can be buried in the image.

10.5 Extraction algorithm

The fourth SSIS system component is in charge of extracting the secret message from the stego-image. It employs a dependable extraction technique to find and extract the encoded message from the cover image while preserving the quality of the cover image.

To allow users to quickly extract messages from stego-images, the SSIS programme must provide a user-friendly interface. The user can choose a stego-image and begin the extraction process, and the application will provide feedback on the extraction's progress.

10.6 Decryption and decoding of the secret message

The SSIS system's last component is in charge of decrypting and decoding the secret message derived from the stego-image. It decrypts the encrypted message using a secure decryption technique to get the original secret message. The SSIS programme must have an easy-to-use interface that allows users to simply decrypt and decode the secret message. The user can commence the decryption and decoding process by selecting an encrypted message, and the programme will provide feedback on the operation's progress.

10.7 Key takeaways

The SSIS system architecture is flexible and scalable, ensuring quick secret message embedding and extraction. Its five components ensure confidentiality and authenticity by embedding hidden messages into cover images in a safe and secure manner. The system's components work in tandem to provide users with an easy-to-use and efficient solution for secret message communication. The system satisfies both functional and non-functional requirements, including performance, security, scalability, and usability.

Chapter 11

Exchanged Messages and Data

11.1 Input data

The input data in the SSIS system consists of a cover image and a secret message that the user wishes to place into the cover image. The cover image should be in a steganography-compatible format, and the secret message should be in text or any other supported format that may be encoded and encrypted.

Assume that the user wants to send a confidential message to a buddy via SSIS. They select a cover image, such as a sunset photograph, then type their message in a text file. The picture of the sunset and the text file containing the secret message would be the input data for SSIS.

11.2 Output data

The SSIS system's output data comprises of a stego-image and the decrypted message. The stego-image is the cover image that contains the secret message, and the decrypted message is the original secret message that the user intended to convey.

In our example, after the user enters the cover image and secret message into SSIS, the system will generate a stego-image, which is the same picture of the sunset but with the secret message contained inside it. When the friend receives the stego-image, they can utilise SSIS to extract and decode the secret message, resulting in the original message the user intended to send.

11.3 Message/data flow between components

To enable the secure embedding, extraction, and decryption of the secret message, the SSIS system follows a precise message/data flow between its components.

First, the cover image is chosen and processed to guarantee it is steganographically suitable. The secret message is then encrypted and encoded using a safe technique. The encrypted message is subsequently embedded within the cover image by the SSIS system using a robust embedding method, resulting in the stego-image.

The stego-image is then put through the extraction algorithm, which extracts the secret message without modifying the cover image. Finally, the encrypted message is decoded using a secure decryption technique to yield the decrypted message.

11.4 Key takeaways

The SSIS system protects the security and confidentiality of data and messages exchanged between its components. The system can effectively embed, extract, and decode the secret message while retaining the integrity of the cover image by following the message/data flow between components.

Chapter 12

Implementation details

12.1 Programming language and libraries used

Any programming language with image processing and encryption libraries can be used to construct the SSIS system. Python, Java, and C++ are some of the most often used languages for building steganography systems.

Python is an appropriate language for SSIS since it offers a significant variety of libraries for image processing and encryption, such as OpenCV and cryptography. OpenCV is a popular image-processing toolkit that can be used for cover image selection, processing, and stego-image synthesis. Cryptography is a library that includes symmetric and asymmetric encryption methods that can be used for secret message encoding and encryption.

Pillow, a fork of the Python Imaging Library (PIL) that may be used for image processing and manipulation, and Stegano, a Python library for steganography that includes multiple embedding and extraction techniques, are two other libraries that can be used to build SSIS.

12.2 Sample code snippets

1- Importing necessairy libraries

In [ ]:

```
import cv2 #(OpenCV): A library used for image processing, including loading, manipulating, and saving images.

import base64 #A binary-to-text encoding scheme used to encode and decode binary data.

import hashlib #A library used for generating secure hash functions to encrypt the secret message.
import numpy as np #Used for image processing and manipulation.
```

2- Cover image selection and processing:

In [ ]:

```
# Read the cover image
cover_image = cv2.imread("sunset.png")

# Convert the image to grayscale
gray_image = cv2.cvtColor(cover_image, cv2.COLOR_BGR2GRAY)

# Verify that the image can be used for steganography
if gray_image.shape[0] * gray_image.shape[1] < len(secret_message):
    print("Cover image too small for the secret message.")
    exit()
```

3- Secret message encoding and encryption:

In [ ]:

```
# Convert the secret message to bytes
message_bytes = secret_message.encode('utf-8')

# Generate a hash of the secret message
hash_object = hashlib.sha256(message_bytes)
hash = hash_object.digest()

# Encrypt the secret message
encrypted_message = base64.b64encode(hash + message_bytes)
```

4- Embedding algorithm and SSIS application:

In [ ]:

```
# Convert the encrypted message to binary
binary_message = "".join([format(byte, "08b") for byte in encrypted_message])

# Embed the message into the cover image
stego_image = np.zeros(cover_image.shape, dtype=np.uint8)
for i in range(gray_image.shape[0]):
    for j in range(gray_image.shape[1]):
        binary_pixel = format(gray_image[i][j], "08b")
        if i * gray_image.shape[1] + j < len(binary_message):
            binary_pixel = binary_pixel[:-1] + binary_message[i * gray_image.shape[1] + j]
        stego_image[i][j] = int(binary_pixel, 2)
```

5- Extraction algorithm and SSIS application:

In [ ]:

```
# Extract the binary message from the stego-image
binary_message = ""
for i in range(gray_image.shape[0]):
    for j in range(gray_image.shape[1]):
        binary_pixel = format(stego_image[i][j], "08b")
        binary_message += binary_pixel[-1]

# Convert the binary message to bytes
message_bytes = bytearray(int(binary_message[i:i+8], 2) for i in range(0, len(binary_message), 8))

# Decrypt the secret message
decrypted_message = base64.b64decode(message_bytes[32:])
```

Figure 12.1: SSID Python code snippet.

12.3 Limitations and challenges

The size of the secret message that can be placed in the cover image is restricted by the size of the cover image, which is one constraint of SSIS. Furthermore, if the cover image is compressed or altered, the encoded message may become corrupted or lost. Another issue is keeping the encoded message hidden from unauthorised users, as even minor alterations to the stego-image can reveal it.

12.4 Key Takeaways

SSIS can be implemented using a variety of programming languages and libraries, and it consists of numerous processes, including cover picture selection and processing, secret message encoding and encryption, embedding and extraction methods, and secret message decryption and decoding. When using SSIS, there are some restrictions and problems to consider. SSIS offers a dependable and safe solution for inserting secret messages into cover images while maintaining secrecy and authenticity. The system is made up of multiple components that collaborate to pick and process the cover image, encode and encrypt the secret message, embed the message in the cover image, extract the message from the stego-image, and decrypt and decode the secret message.

Chapter 13

Tools And Development Phases

13.1 Programming Language

- Python was chosen for implementing SSIS due to its simplicity, readability, and availability of relevant libraries.
- Python's easy-to-understand syntax and emphasis on code readability simplify the implementation of complex algorithms like SSIS.

13.2 Development Environment

- **Visual Studio Code (VS Code):** Lightweight code editor with Python support through extensions. Provides syntax highlighting, debugging, and Git integration. Customizable and widely used.
- **Jupyter Notebook:** Interactive computing environment for creating and sharing code, visualizations, and text. Ideal for data analysis and documentation. Supports multiple languages, including Python.

13.3 Libraries and Frameworks

- **OpenCV: [16]** OpenCV is an open-source computer vision library that provides a variety of functions for image processing tasks. It can be used in SSIS to manipulate, filter, and analyze images.
- **NumPy: [17]** NumPy is a popular Python library for scientific computing that provides a wide range of tools for working with arrays and matrices. It can be used for a variety of image processing tasks in SSIS.
- **Cryptography Libraries:** Used for encryption and decryption operations in SSIS.
- **Pillow: [18]** Pillow is a user-friendly library for image processing. It can be used alongside OpenCV for image handling and transformation in SSIS.
- **Scikit-learn: [19]** Scikit-learn is a comprehensive machine learning library.It can provide advanced feature extraction and data analysis techniques to enhance the performance of the SSIS algorithm.
- **Other libraries:** Depending on specific requirements, additional libraries can be explored for tasks such as image manipulation, noise generation, and matrix operations in SSIS.

These libraries and frameworks offer a range of functionalities and tools to support different stages of the SSIS implementation.

13.4 Additional Software or Hardware Tools

1. **Hardware Tool:**
    - GPUs for accelerating image processing tasks or specialized image capture devices for higher-quality cover images.
2. **Software Tool:**
    - **Git:**As version control system for code management.
    - **GitHub:**As a collaboration platform for team coordination, and project management.

13.5 Development Phases

13.5.1 Functions

1. **Function 1: Key Validation**

    - This function will check the format and validity of a given key.
    - It will take a key as input.
    - The function will verify if the key meets the required format criteria.
    - If the key is invalid, the program will terminate.

2. **Function 2: Matrix Interleaving**

    - This function will perform the reorganization of a matrix based on a given key.
    - The function will iterate through each row of the matrix.
    - For each row, it will iterate through the elements of the key
    - The value from the source matrix corresponding to the key element will be assigned to the destination matrix.

3. **Function 3: Noise Generation**

    - This function will generate Additive White Gaussian Noise (AWGN) [20] for a given image length.
    - It will take the image length as input.
    - The function will use appropriate libraries to generate a sequence of random numbers following a Gaussian distribution.
    - The sequence will be reshaped into a matrix format based on the image length.
    - The normalized noise matrix will be returned.

4. **Function 4: Second Stage Noise Generation**

    - This function will generate noise for the second stage of encryption.
    - It will take the image length as input.
    - The noise generation function will be called to obtain the AWGN noise matrix.
    - An empty matrix will be initialized to store the interleaved noise values.
    - The user will be prompted to input the second key for AWGN.
    - The matrix interleaving function will be called to reorganize the noise matrix using the key.
    - Finally, the AWGN noise matrix will be returned.

5. **Function 5: Image to Binary Matrix Conversion**

    - This function will convert the red color channel of an image matrix into a binary matrix.
    - The function will iterate through the rows of the image matrix and extract the red color channel values.
    - The extracted values will be converted into binary representation and stored in the binary matrix.
    - The binary matrix will be returned.

13.5.2 Encoding Process

The encoding process involves these steps:

1. Reading in the cover image using the OpenCV library and converting the red color channel of each pixel into a binary representation.
2. Generating a noise signal matrix using a pseudorandom number generator.
3. Embedding the binary message into the cover image by multiplying the binary representation of each pixel with the corresponding value from the noise signal matrix.
4. Interleaving the pixels of the cover image using a secret key.
5. Embedding the interleaved pixels into the LSBs (Least Significant Bits) of the blue color channel of the cover image, utilizing another secret key.

13.5.3 Decoding Process

The decoding process involves these steps: (Reversing Encoding Steps)

1. Reading in the stego image using the OpenCV library and extracting the binary representation of the blue color channel of each pixel.
2. Reconstructing the interleaved pixels using the secret key.
3. Recovering the noise signal matrix using the pseudorandom number generator.
4. Dividing the binary representation of each pixel by the corresponding value from the noise signal matrix to restore the original binary representation.
5. Extracting the binary message from the LSBs of the blue color channel using the secret key.

13.6 Summary

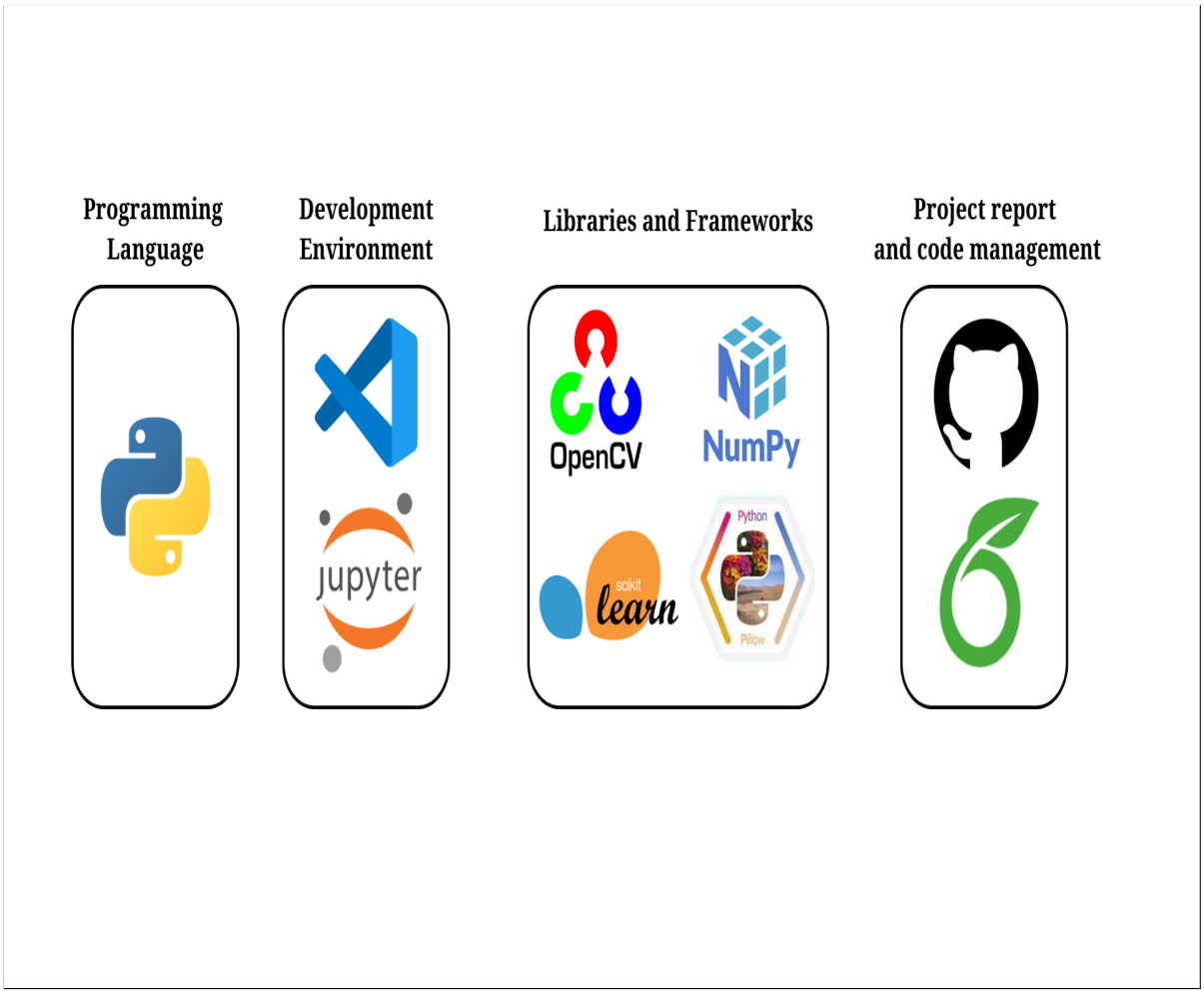

Figure 13.1: Tools Used

Chapter 14

Conclusion

In this study, we looked at the basics of steganography, such as the cover image, secret message, steganography algorithm, and key. We also talked about the main steganography components, such as the embedding and extraction modules, the cryptography module, and user interface. We also demonstrated the steganography functional flow, which includes entering the cover image and secret message, encrypting the secret message, embedding the message, generating the stego-image, securely transmitting it, extracting the encrypted message, decrypting it, and displaying the secret message.

Furthermore, we examined the importance of steganography in numerous domains, such as military and forensic investigations, as well as its limitations. To prevent unauthorised access to the secret message, we also highlighted the requirement for secure communication routes and encryption.

In conclusion, Finally,steganography is an effective approach for securely transferring information that is not limited to a certain field. However, it is critical to select appropriate steganography algorithms and keys to ensure the secret message's confidentiality. Future study could concentrate on the development of more robust steganography techniques and their use in other fields.

We would like to express our gratitude to our professor Mrs. Manel Abdelkader for providing us with the guidance and support that helped us reach this point. We extend a personal thank you to her for her invaluable contributions to our learning journey.

Bibliography

[1] M. Anbarasi and V. Karthikeyani, "A survey of steganography and steganalysis techniques in digital images," *International Journal of Computer Science Information Security*, vol. 1, pp. 1–6, 06 2009.

[2] F. A. Baothman and B. S. Edhah, "Toward agent-based lsb image steganography system," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 903–919, 2021.

[3] Q. Do and I. Koo, "Fpga implementation of lsb-based steganography," *Journal of Information and Communication Convergence Engineering*, vol. 15, pp. 151–159, 09 2017.

[4] WonderHowTo, "Steganography: Hide secret data inside image or audio file in seconds." `https://shorturl.at/EJX25`, accessed on 2023-04-22.

[5] S. Chauhan and G. Singh, "Pixel value differencing based image steganography: a review," *SN Computer Science*, vol. 9, pp. 321–326, 11 2018.

[6] M. Hashim, A. A.Mahmood, and M. Mohammed, "A pixel contrast based medical image steganography to ensure and secure patient data," *The International Journal of Nonlinear Analysis and Applications (IJNAA)*, vol. 12, pp. 2008–6822, 12 2021.

[7] P. Maniriho and T. Ahmad, "Information hiding scheme for digital images using difference expansion and modulus function," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, 07 2019.

[8] L. M. Marvel, C. G. Boncelet, and C. T. Retter, "Spread spectrum image steganography," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 8, no. 8, pp. 1075–1083, 1999.

[9] H. Sajedi and M. Jamzad, "Adaptive batch steganography considering image embedding capacity," *Optical Engineering - OPT ENG*, vol. 48, 08 2009.

[10] M. Padmaa and Y. Venkataramani, “Random image steganography using pixel indicator to enhance hiding capacity,” *Journal of Applied Sciences*, vol. Volume Number, no. Issue Number, p. Page Range, 2014.

[11] A. A.-A. Gutub, “Pixel indicator technique for rgb image steganography,” *Journal of Emerging Technologies in Web Intelligence*, vol. 2, pp. 56–64, 2010.

[12] S. KHAIRE and S. Nalbalwar, “Review: Steganography – bit plane complexity segmentation (bpcs) technique,” *International Journal of Engineering Science and Technology*, vol. 2, 09 2010.

[13] B. Mondal and T. Mandal, “A secret shearing algorithm based on lsb substitution,” *International Journal of Computer Applications*, vol. 92, 03 2014.

[14] A. Durdu, “Nested two-layer rgb based reversible image steganography method,” *Information Technology and Control*, vol. 50, pp. 264–283, 06 2021.

[15] C.-T. Huang, N. S. Shongwe, and C.-Y. Weng, “Enhanced embedding capacity for data hiding approach based on pixel value differencing and pixel shifting technology,” *Electronics*, vol. 12, no. 5, 2023.

[16] Wikipédia, “Opencv — wikipédia, l’encyclopédie libre,” 2022. [En ligne; Page disponible le 2-octobre-2022].

[17] Wikipédia, “Numpy — wikipédia, l’encyclopédie libre,” 2022. [En ligne; Page disponible le 3-juin-2022].

[18] “Pillow Documentation.” [Online; accessed 14-May-2023].

[19] “scikit-learn Documentation.” [Online; accessed 14-May-2023].

[20] Wikipedia contributors, “Additive white gaussian noise — Wikipedia, the free encyclopedia,” 2023. [Online; accessed 14-May-2023].