Research Proposal Title: Multiple Embedding Steganography for RGB Image

---

Submitted by: Dimas Anwar Aziz

NIM: 203022410012

Concentration: Cyber Security

E-mail address:

dimasanwaraziz@student.telkomuniversity.ac.id

Supervisor (I) : Prof. Ari Moesriami Barmawi, Ph.D.

Supervisor (II):

08102024

Topic Summary:

Image steganography is a crucial technique for securing confidential information by hiding it within digital images. This research proposes an improved multiple embedding steganography method for RGB images that aims to increase data hiding capacity while maintaining visual quality. Previous research on grayscale images demonstrated capacity improvements of up to 450%, but was limited by using only one channel. The proposed method extends this by utilizing all three color channels (Red, Green, Blue) and implementing sophisticated embedding techniques.

The research addresses key challenges in image steganography: optimizing

embedding capacity, preserving image quality, and ensuring robustness against various attacks. The method incorporates multiple embedding techniques with polar codes. The experimental evaluation focuses on imperceptibility (PSNR and MSE), robustness (against noise, cropping, scratching, and JPEG compression), and capacity (through image sizes and pixel variations).

Implementation uses Python with OpenCV and NumPy libraries. The research aims to achieve at least 50% capacity increase compared to conventional methods while maintaining PSNR above 40 dB. The evaluation will use standardized metrics and diverse test datasets.

The literature review analyzes previous work in image steganography, particularly focusing on code-based methods and RGB image applications. The proposed methodology includes comprehensive embedding and extraction processes, with special attention to key generation and security protocols. The experimental design incorporates various attack scenarios to test robustness.

This research contributes to information security by providing an enhanced steganography solution balancing capacity, quality, and security requirements. The results will benefit security professionals, researchers, and developers working on data hiding applications. The work is scheduled across four semesters, encompassing design, implementation, testing, and documentation phases.

1 INTRODUCTION (assessment 1)

In the increasingly evolving digital era, digital data security has become a top priority. Personal information, business data, and other critical data must be protected from threats of leakage or unauthorized access. In facing these challenges, cryptography and steganography emerge as important solutions to enhance digital data security.

Steganography is a technique of hiding messages within a message insertion media or cover image, so that the existence of the embedded secret message cannot be seen directly [**pujianto2021uji**]. Steganography enables the exchange of secret messages through hiding information in various digital media such as images, video, and audio, without raising suspicion [**siaulhak2023sistem**]. Steganography can be viewed as an extension of cryptography, and in practice, the secret message is encrypted first, then the ciphertext is hidden within another medium so that third parties are unaware of its existence. The hidden secret message can be extracted back exactly as the original [**soetarmono2012studi**].

Among various digital media that can be used for steganography, images become a popular choice for several reasons. First, digital images are widely available, and image exchange on the internet is common, thus not raising suspicion. Second, images have the capacity to hide information. Third, pixel manipulation in images can be done using various techniques. However, the main challenge in image steganography is finding the balance between data hiding capacity, image quality, and resistance to detection [**fikri2022optimasi**].

Previous research has shown significant progress in increasing data hiding capacity in grayscale images. Kingsley et al. [**kingsley2020improving**] successfully increased data hiding capacity up to 450% using code-based methods. However, the use of grayscale images limits practical applications as it only has one illumination channel. On the other hand, Fikri et al. [**fikri2022optimasi**] demonstrated that applying steganography to color images has good resistance, although images with black color dominance show less optimal compatibility. Based on these findings, this research aims to extend the embedding method to color images that have three color channels (Red, Green, and Blue). The main focus is to increase data hiding capacity while maintaining visual quality and resistance to detection. This research will also evaluate the compatibility of the proposed method with various types of color images, including those with large dark areas. By optimizing the use of all three color channels, it is expected to achieve a significant increase in data hiding capacity compared to methods that only use grayscale images, while maintaining the naturalness of the resulting images.

The knowledge aspect in this research focuses on steganography techniques for color images, code-based steganography methods, and digital image quality analysis. This includes a deep understanding of RGB color channel manipulation, effective data insertion techniques, and evaluation methods for the resulting stego image quality. Special emphasis is given to optimizing the use of all three color channels to maximize data storage capacity while maintaining visual quality.

From the user perspective, this research is aimed at several main groups. First, information security professionals who need reliable and secure data hiding solutions. Second, researchers in the field of steganography and data security who can utilize this method for further development. Third, developers of data security applications that require efficient and easily integrated steganography components.

The utility aspect of the research includes various practical applications in data hiding systems, copyright protection, and digital watermarking. The developed method is designed with consideration for ease of implementation, computational efficiency, and flexibility of use. The main focus is achieving an optimal balance between data hiding capacity, resulting image quality, and resistance to various steganography detection techniques.

The infrastructure used in this research is based on Mac mini M2 with technical specifications that support efficient image processing. The system is equipped with an Apple M2 chip that has an 8-core CPU and 10-core GPU, supported by 8GB of memory and 256GB SSD storage. The use of macOS operating system provides a stable platform for the development and testing of the proposed steganography method.

2 Preliminary Literature Review (assessment 2)

Kingsley et al (2020) in their paper discussed the low embedding capacity and PSNR (Peak Signal-to-Noise Ratio) values in existing code-based steganography schemes, which only achieved 150% capacity and 48 dB visual quality of stegano

images. To address this issue, the proposed method is a multiple embedding technique, which aims to embed secret bits more than once in the LSB (Least Significant Bit) of pixels selected based on a secret key. The parameters used to measure the success of this method include embedding capacity and PSNR value of the resulting stego image. The advantage of this method is the ability to achieve higher embedding capacity up to 450% and increased PSNR value to 51 dB, as well as improved resistance to attacks such as scratching and JPEG compression. However, this research is still applied to black and white images [**kingsley2020improving**].

In research conducted by Dwi Andika et al (2020), they discussed the limited storage capacity in steganography techniques using the GifShuffle algorithm on GIF images. The main challenge faced was the difficulty in inserting large text messages. To overcome this problem, they modified the bit values in message storage, which aims to increase capacity without sacrificing image visual quality. Tests using parameters such as insertable data size and image quality showed that this method successfully increased storage capacity to over 256 KB without significantly reducing image quality. However, this method requires additional modifications to the GifShuffle algorithm which can increase implementation and processing complexity [**andika2020modifikasi**].

Susanto et al (2020) in their research combined the Least Significant Bit (LSB) steganography method with the RSA encryption algorithm to improve security in inserting encrypted messages into images. The parameter used to measure image quality after insertion is Peak Signal-to-Noise Ratio (PSNR),

with results showing the highest PSNR of 78 dB for 1024-bit messages. The advantage of this method is the almost invisible change in image quality, but the disadvantage is increased complexity with message size and RSA key size, which affects performance [**susanto2020kombinasi**].

Basri et al (2021) researched the use of steganography method with Least Significant Bit (LSB) technique to hide images within other images, particularly in the context of social interaction through digital media. This research measured the ratio of hidden image size to cover image and the ability of cover image to maintain its visual quality. The results showed that this method effectively hides information without significant changes to the cover image. However, this method has limitations in handling images with transparency, as it only considers red, green, and blue (RGB) components, without considering the alpha component [**basri2021penerapan**].

Research by Wiranata et al (2021) focused on embedding secret messages in images and audio using the Least Significant Bit (LSB) method combined with Caesar Chipper encryption and Rivest Code 4 to maintain data confidentiality. This research tested the application's ability to hide and retrieve messages intact, as well as changes in image and audio file size after the insertion process. The advantage of this method is its ability to maintain confidentiality without significant changes to image or sound quality. However, there are changes in file size caused by the encryption and message insertion process [**wiranata2021aplikasi**].

Research conducted by Abdillah et al (2023) discusses data protection and

access using steganography through the Least Significant Bit (LSB) technique, where text is inserted into images through changes in the smallest pixel values. Measured parameters include text insertion accuracy and maintained image quality after encoding and decoding processes. The main advantage of this method is its ability to hide information without significantly altering the visual quality of the image. However, this method has limitations in text insertion capacity and is vulnerable to image manipulation attacks that can damage hidden data [**abdillah2023implementasi**].

Table **??** shows the comparison of previous research results.

Table 1: Comparison of Previous Research

Author	**Method**	**Advantages**	**Disadvantages**
Kingsley et al. (2020)	Multiple embedding technique with secret key-based pixel selection	Increased capacity up to 450%, PSNR improved to 51dB	Limited to grayscale images only
Andika et al. (2020)	Modified GifShuffle algorithm for GIF images	Increased storage capacity over 256KB	Complex implementation, limited to GIF format
Susanto et al. (2020)	Combined LSB with RSA encryption	High PSNR (78dB) for 1024-bit messages	Increased complexity with larger messages and RSA keys
Basri et al. (2021)	LSB technique for image-in-image hiding	Effective information hiding with minimal visual changes	Limited handling of transparency (alpha channel)
Wiranata et al. (2021)	LSB combined with Caesar Cipher and RC4	Good security with dual encryption	File size increases after embedding
Abdillah et al. (2023)	LSB technique for text embedding	Maintains visual quality effectively	Limited text capacity, vulnerable to image manipulation

Based on Table **??**, it can be concluded that various steganography methods have shown their respective advantages and disadvantages. The multiple embedding method by Kingsley et al. (2020) successfully increased data hiding capacity significantly up to 450% with good visual quality (PSNR 51 dB), but it is limited to grayscale images. The modified GifShuffle method by Andika et al. (2020) increased storage capacity in GIF images, but its implementation complexity is high. The combination of LSB and RSA encryption by Susanto et al. (2020) resulted in high PSNR (78 dB) but with increased complexity. The LSB method by Basri et al. (2021) is effective for hiding images within other images with minimal visual changes, but it is less optimal for images with transparency. The combination of LSB with Caesar Cipher and RC4 by Wiranata et al. (2021) provides good security but increases file size. Finally, the LSB method by Abdillah et al. (2023) is effective in maintaining visual quality but has limitations in text capacity and is vulnerable to image manipulation.

3 Problem Statement (Assessment 2)

Previous research in code-based steganography has shown significant progress. Kingsley et al. [**kingsley2020improving**] successfully increased data hiding capacity from around 150% to 450%, which is a substantial improvement for cases requiring high capacity. However, research on grayscale images [**soetarmono2012studi**] still limits its application due to lack of flexibility in using color channels.

4 Objective and Hypothesis (assessment 3)

This research has two main objectives along with related hypotheses. First, to develop code-based steganography techniques with multiple embedding for RGB images and increase data hiding capacity by at least 50% compared to previous methods, with the expectation that this method will significantly increase data hiding capacity and its application to RGB images will result in greater capacity improvement compared to grayscale images. Second, to maintain PSNR above 40 dB for stego images, with the hypothesis that capacity increase through multiple embedding will not significantly reduce visual quality [**nasution2018image**].

5 Research Method (assessment 4,5)

This research will use an experimental approach to test the proposed hypotheses. The research method includes several main stages, integrating basic steganography concepts and addressing specific challenges of code-based steganography with multiple embedding techniques.

5.1 Requirement Identification

Based on literature analysis and research objectives, the main requirements for developing code-based steganography method with multiple embedding are:

1. Increase data hiding capacity by at least 50% compared to the method proposed by Kingsley et al. (2020)

2. Techniques to maintain visual quality (PSNR $>$ 40 dB)

3. Embedding algorithm with multiple embedding on RGB images

This requirement identification will be the foundation for the design and implementation process of the proposed steganography system.

5.2 Design Process

Based on the identified requirements, this research will design algorithms for code-based steganography with multiple embedding.

The general system architecture will include

1. **Secret Image**

   Secret Image is confidential information to be hidden in the steganography system, which can include images in various formats (JPG, PNG) that will be protected for security and confidentiality. The selection of this type of secret data will affect the required storage capacity and embedding method used in the steganography process.

   This image is a dynamic user input to be embedded in a color cover image, and will be checked regarding the capacity and noise (quality) resulting from embedding this secret image.

2. **Encoding Process**

   The encoding process aims to insert messages (text) into images using the **Least Significant Bit (LSB)** of each pixel's color component (R, G, B).

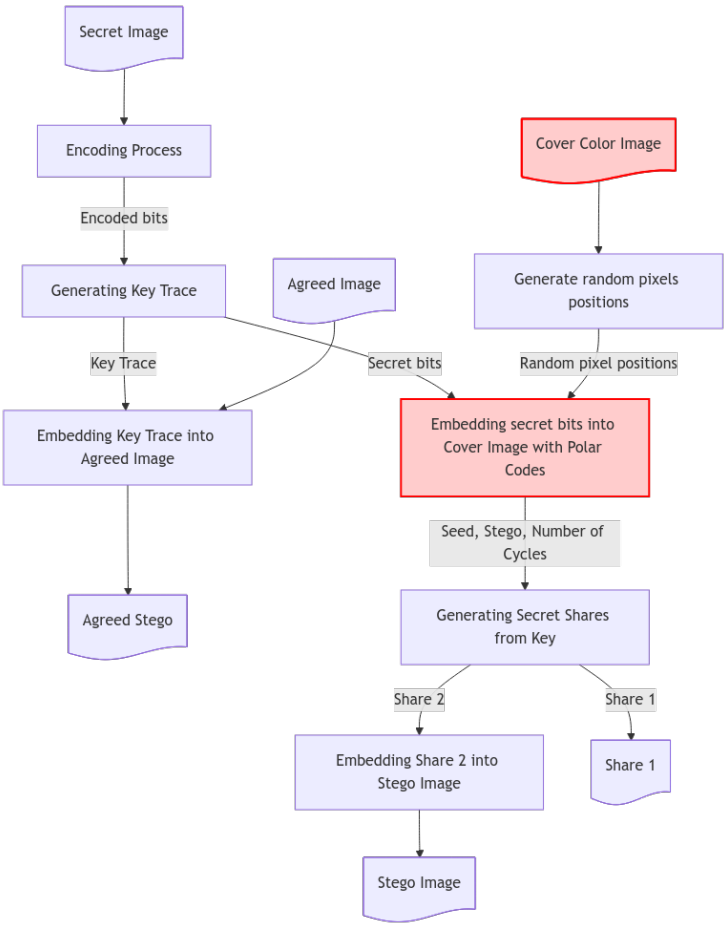

Figure 1: Design method

Here are the steps of the encoding process:

(a) **Data Preparation**

- **Input:**
  - Image that will be used as insertion media.
  - Text message to be hidden.
- **Terminator Addition:** Message is given a terminator (\000)

at the end as a marker that the message has ended.

  - Example:

    Initial message: "Hello, World!"

    Message to be processed: "Hello, World!\000"

- **Conversion to Bits:** Each character is converted to 8 bits (byte format).

  - Example letter "H" (ASCII 72): 01001000.

(b) **Image Iteration Pixel-by-Pixel**

Image is accessed pixel-by-pixel, and for each pixel:

- Color components R (Red), G (Green), and B (Blue) are extracted.
- Each color component has 8-bit value (0-255).

(c) **Inserting Bits to LSB**

From each message byte (8 bits), message bits are inserted one by one into the LSB (Least Significant Bit) of R, G, and B components. LSB is the lowest bit of the color component value. Changing LSB will not significantly alter the color.

- **Example of Bit Insertion:** Let's say:

  - Initial color component values:

$$R = 10110011, \quad G = 11001001, \quad B = 11111110$$

  - Message bits to be inserted: 0, 1, 1.

- **Change Steps:**
  - R (Red):
    - LSB before: 1
    - Replace with message bit 0.
    - R value becomes: 10110010.
  - G (Green):
    - LSB before: 1
    - Replace with message bit 1.
    - G value remains: 11001001.
  - B (Blue):
    - LSB before: 0
    - Replace with message bit 1.
    - B value becomes: 11111111.
- **Pixel Result:**

$$R = 10110010, \quad G = 11001001, \quad B = 11111111.$$

Process continues for each pixel, moving from left to right, top to bottom. If all message bits have been inserted, remaining pixels are not changed.

Encoding stops after all bits from the message (including terminator `\000`) have been successfully inserted. The new image (with hidden message) is saved in PNG format to avoid lossy compression.

(d) **Illustration**

Let's say the image has pixels with:

```
Pixel (R, G, B) = (10110011, 11001001, 11111110)
```

And the message to be inserted is the letter "A".

```
"A" in ASCII = 65 → 01000001 (8 bits).
```

Iteration Process:

Message Bit	Color Component	Old LSB	New LSB	Final Result
0	R (Red)	1	0	10110010
1	G (Green)	1	1	11001001
0	B (Blue)	0	0	11111110
0	R (Red)	0	0	10110010
0	G (Green)	1	0	11001000
0	B (Blue)	0	0	11111110
1	R (Red)	0	1	10110011
0	G (Green)	0	0	11001000

Table 2: Process of Inserting Message Bits into Color Components

Image Output:

The new image has modified LSB color component values, but the color changes are not visible to human eyes. This image contains the message that has been hidden.

3. Generating Key Trace

Process of generating keys that will determine where and how data will be hidden in the image. The system uses a special method to create random patterns that ensure data is evenly distributed throughout the image. This method makes the system more secure by adding a security layer, while facilitating data retrieval without damaging the image. If problems occur or someone tries to illegally access the data, this system also helps better manage and recover the data.

4. Agreed Image

   An image that has been agreed upon between sender and receiver as storage media for steganography key traces. The selection of this image is done by considering various technical characteristics such as texture complexity, color distribution, and optimal noise level to support the information hiding process without raising suspicion. The selected image must also have adequate capacity to store key traces while maintaining good visual quality after the insertion process.

5. Embedding Key Trace into Agreed Image

   A critical stage where the key trace is inserted into previously agreed images using steganography techniques. This process requires precision to ensure data integrity and optimal imperceptibility. Insertion is done by considering image characteristics and pixel value distribution to optimize the balance between storage capacity and visual quality. The steganography technique used also considers security aspects to prevent unau-

thorized detection and extraction.

6. Agreed Stego

   The final result of the key trace insertion process into mutually agreed images. This image has visual characteristics identical to the original image to avoid suspicion, but embedded within it is secret information in the form of key traces that will be used for the data extraction process. Image quality is maintained despite going through the insertion process, with minimal pixel value changes undetectable to the human eye. This image plays an important role as a carrier of control information needed for subsequent steganography processes.

7. Embedding secret bits into Cover Image

   Embedding secret bits into cover image is the process of inserting secret data bits into the cover image using sophisticated and optimized steganography algorithms, combined with polar codes to improve data transmission reliability. This process involves analyzing cover image characteristics to determine optimal data insertion locations, then using precise transformation techniques to insert secret information bits.

   Polar codes, introduced by Arikan, are used to encode secret data before insertion. This technique utilizes channel polarization phenomena to achieve Shannon capacity with low encoding and decoding complexity ($O(N \log N)$).

   The algorithm used is specifically designed to maximize data storage capa-

city while maintaining image visual quality, with special focus on minimizing distortion that can be detected both visually and statistically. The use of polar codes provides additional advantages in terms of error correction and resistance to channel noise. This process also considers the balance between insertion efficiency and resistance to various steganalysis techniques.

8. Cover Image

   Cover image is the main medium for hiding secret data, selected with special consideration based on texture complexity, color distribution, and optimal statistical characteristics to support the information embedding process. Proper cover image selection is critical as it will affect data storage capacity, resistance to detection, and visual quality of the steganography result. Images with complex texture areas and high pixel value variations are generally more ideal because they can hide changes caused by the data embedding process more effectively.

9. Generate random pixels positions

   Algorithm to generate random but deterministic pixel positions based on a specified seed value. This process uses a verified pseudo-random number generator to produce pixel position sequences that are evenly distributed throughout the image. This deterministic approach is important to ensure consistency in data embedding and extraction processes, while enhancing security by creating data distribution patterns that are difficult for unau-

thorized parties to predict.

10. Generating Secret Shares from Key

    Process of dividing secret keys into multiple shares using Shamir's Secret Sharing scheme. This technique allows splitting the key into n shares with threshold k, where the original key can only be reconstructed if at least k shares are available. This process enhances security by distributing risk and preventing single point of failure in key management.

11. Embedding Share 2 into Stego Image

    Process of embedding the second share (share 2) of the divided key into the stego image using optimized steganography techniques. Embedding is done by considering the balance between data security and image visual quality. This process is designed to ensure shares can be accurately extracted without damaging other hidden information in the stego image.

12. Share 1

    The first part (share 1) of the divided key is stored separately by applying layered security protocols. Share 1 plays an important role as it is needed along with share 2 to reconstruct the original key. Storage and management of share 1 follows strict security standards to prevent unauthorized access.

13. Stego Image

    The final image after the secret data embedding process, designed to maintain visual characteristics identical to the original cover image. Although

containing hidden information, the stego image shows no differences detectable by visual or simple statistical analysis. Image quality is maintained through optimization of embedding algorithms that consider human visual perception characteristics.

14. Final Stego Image

    The final version of the stego image that has been equipped with public keys and all metadata needed for secure extraction and verification processes. This image provides comprehensive data recovery mechanisms while maintaining security and confidentiality aspects of hidden information. Additional data structures are integrated in a way that does not interfere with overall visual quality.

15. **Decoding Process**

    The decoding process aims to extract hidden messages that have been embedded in images using the **Least Significant Bit (LSB)** method. Here are the decoding steps:

    (a) **Data Preparation**

    - **Input:** Image that has been encoded with hidden message.
    - **Purpose:** Extract bits from color components of each pixel and reassemble into message characters.

    (b) **Image Iteration Pixel-by-Pixel**

    Image is accessed pixel-by-pixel (from left to right, top to bottom), then bit reading is performed from color components:

- Color components **R (Red)**, **G (Green)**, and **B (Blue)** are extracted one by one.
- From each color component, **Least Significant Bit (LSB)** is read because LSB is used to embed message bits.

Example:

$$R = 10110010$$
$$G = 11001001$$
$$B = 11111111$$

LSB from each color component:

- LSB $R = 0$
- LSB $G = 1$
- LSB $B = 1$

Bits obtained from that pixel are: $0, 1, 1$.

(c) **Assembling Bits into Bytes**

Every 8 bits obtained from LSB are reassembled into 1 byte.

- First LSB goes to leftmost bit.
- Next LSBs follow until 8 bits (1 byte) are formed.

Example:

Read LSBs: $0, 1, 0, 0, 1, 0, 0, 0$

Result: 01001000, which in ASCII is letter **H**.

The process of reading bits from LSB continues until finding **null**

terminator (\000), which is byte with value `0x00` (00000000 in binary).

Example:

`H → e → l → l → o → , → W → o → r → l → d → !`

Followed by null byte \000, decoding process stops.

Bytes that have been read are converted back to ASCII characters and assembled into the hidden message string.

Decoding Process Summary:

- Read LSB from color components R, G, and B sequentially.
- Assemble LSBs into 8 bits to form 1 byte.
- Repeat process until finding **null terminator** (\000).
- Convert collected bytes into message string.
- (Optional) Validate message by generating **key trace** using **SHA-256**.

5.3 Implementation Process

The designed algorithm will be implemented using Python, utilizing libraries such as OpenCV for image processing and NumPy for numerical operations. This implementation can support both RGB and grayscale images to enable comparative analysis.

5.3.1 Development Environment Setup

Start with installing Python as the main programming language. Install OpenCV library which will be used to read, write, and manipulate digital images. Add NumPy to support mathematical operations and arrays needed in image processing.

5.3.2 Main Module Development

1. Implementation of image reading functions:
    (a) Reading RGB format images using OpenCV
    (b) Reading grayscale format images using OpenCV
    (c) Validation of image input format
2. Implementation of pixel processing functions:
    (a) Pixel value modification for data insertion
    (b) Embedding capacity checking
    (c) Data integrity validation
3. Implementation of data extraction functions:
    (a) Reading modified pixel values
    (b) Hidden data extraction
    (c) Original message reconstruction
4. Implementation of quality evaluation functions:

(a) PSNR (Peak Signal-to-Noise Ratio) calculation

(b) MSE (Mean Square Error) calculation

(c) Steganography result imperceptibility analysis

5.3.3 Steganography Algorithm Implementation

In steganography algorithm implementation, system development begins with implementing a comprehensive multiple embedding mechanism. This process includes identifying potential areas for data insertion, developing layered embedding methods, and managing metadata for multiple embedding. Coordination between embedding layers becomes crucial to ensure the integrity of embedded data.

The developed system is designed to support various image formats, with main focus on handling RGB (24-bit) and grayscale (8-bit) images. Algorithm adaptation is done to accommodate both formats, along with performance optimization that enables the system to work efficiently on various types of carrier images.

Embedding capacity optimization becomes one of the main priorities, with a target of at least 50% of carrier size. This is achieved through efficient bit-plane slicing technique implementation, supported by payload data compression mechanism, and adaptive pixel selection to maximize storage capacity without sacrificing visual quality.

Visual quality preservation becomes a fundamental aspect in development, with PSNR target set above 40 dB. The system implements various strategies to

minimize visual distortion, including pixel change distribution techniques and adaptation to specific characteristics of carrier images.

Implementation is equipped with comprehensive validation mechanisms, including data integrity verification, effective capacity measurement, visual quality evaluation, and robustness testing. This series of tests ensures that the developed system meets all established requirements, both in terms of capacity, quality, and data security.

5.3.4 Testing System Development

Create testing modules for:

- Measuring maximum data embedding capacity - Evaluating visual quality using PSNR and SSIM - Testing resistance to steganalysis techniques - Comparing with conventional LSB methods

5.3.5 Analysis and Evaluation

Conduct comprehensive analysis including:

- Statistical testing using t-test and ANOVA - Subjective visual evaluation - Performance metric measurements (capacity, PSNR, SSIM) - Computational efficiency evaluation

5.3.6 Documentation

Prepare complete documentation including:

1. Source code with clear comments

2. System usage manual

3. Testing results and analysis

4. Recommendations for further development

5.4 Experiment Design and Data Collection

To evaluate the performance of the proposed method compared to other methods, an experiment is conducted based on four main quantitative measures. Figure **??** illustrates the suggested experimental design for the proposed method.

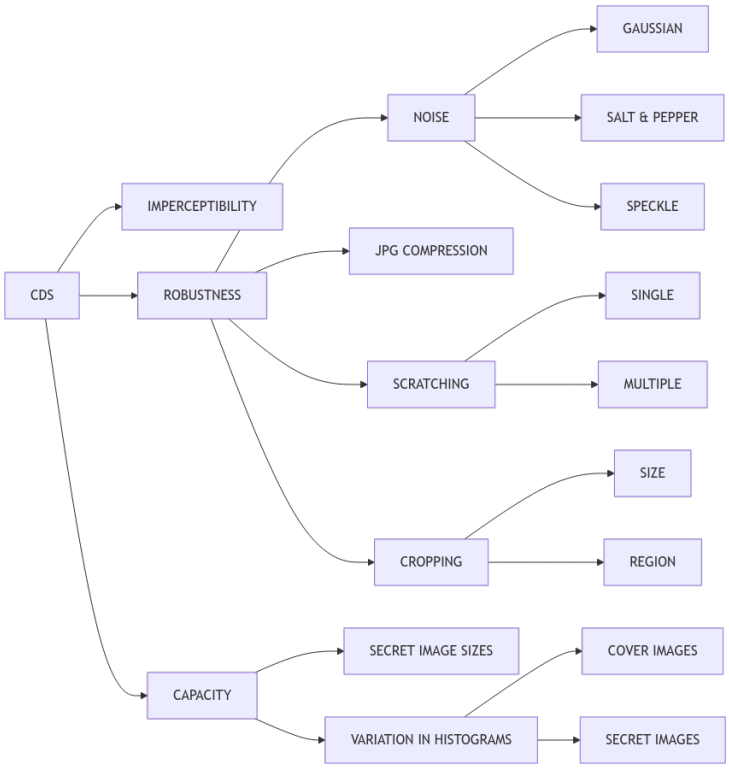

Figure 2: Design experiment

5.4.1 Imperceptibility

The steganography scheme will be imperceptible if the human eye cannot distinguish between the cover and stego images. The focus in this section is the visual quality of the resulting stego images. Groups of secret images with the same image size and diverse tonal distributions are formed. Each secret image in a particular group is embedded in the cover using both the proposed method and A. M. Molaei's method.

Imperceptibility is then measured using Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR). A higher PSNR indicates that the stego image is more similar to the original image, meaning better visual quality.

5.4.2 Robbustnes

This section tests the ability of the proposed method to handle various attacks. This allows the receiver to recover secret messages that have been destroyed if the stego image is intentionally or unintentionally destroyed. The quantitative measure to be determined is the PSNR of the recovered secret image.

After each embedding scenario is performed as described, the resulting stego images are exposed under the following attacks

1. Noise Attack: Stego images are attacked with several methods below with different intensities.

   (a) **Gaussian Noise**: This name is taken from Carl Friedrich Gauss [**selamistudy**]. Gaussian Noise is statistical noise that has a prob-

ability density function (PDF) equal to the normal distribution, also known as Gaussian distribution. In other words, the values that can be taken by the noise are Gaussian distributed. For Gaussian Noise, two variance variations are considered in the experiment: Gaussian noise with mean 0 variance 0.01 and Gaussian noise with mean 0 variance 0.1.

(b) **Salt and Pepper Noise**: This is a form of noise that is also visible in images. It is also known as impulse noise. This noise can be caused by sharp and sudden disturbances in the image signal. This noise appears as rarely occurring black and white pixels [**kaisar2008salt**]. Salt and Pepper noise comes in various forms with different densities. Two density variations are considered in the experiment: Salt and Pepper noise with density 0.05 and Salt and Pepper noise with density 0.5.

(c) **Speckle Noise**: Speckle noise is noise that appears due to environmental conditions affecting imaging sensors during image acquisition. Speckle noise is most commonly detected in medical images, active Radar images, and Synthetic Aperture Radar (SAR) images [**mansourpour2006effects**]. Two variance variations are considered in the experiment: Speckle noise with variance 0.04 and Speckle noise with variance 0.4.

After the stego is exposed to these Noise attacks, the extraction process retrieves the secret image and performs error correction. The PSNR of

the retrieved secret image is determined and tabulated. The PSNR of all retrieved secret images is determined and compared with A. M. Molaei's Method.

2. Cropping Attack: Stego images with the same PSNR range are grouped and exposed to cropping attacks. Cropping attacks are performed based on different cropping ratios and cropping regions. Each stego image in a particular group is exposed to the same cropping ratio and cropping region as other stego images in the same group. The stego image will undergo extraction process to retrieve the secret image by performing error correction. For each set or group, the PSNR value of the retrieved secret image is determined and tabulated and compared with A. M. Molaei's Method.

3. Scratching Attack: Scratch attacks come in two different forms: different length with same width, and different width with same length. After each form of scratch attack is performed on the stego image, error correction is used to retrieve the secret image. The PSNR of the retrieved secret image is determined, tabulated and compared with previous methods.

   Single scratching is performed by scratching one line on the stego image. Multiple scratching is performed by scratching several lines on the stego image.

4. JPEG Compression Attack: Stands for Joint Photographic Experts Group. JPEG is a lossy compression method commonly used for digital images,

especially for images produced by digital photography. JPEG typically achieves 10:1 compression with little visible loss in image quality [**ch2015medical**]. JPEG compression is used in a number of image file formats. JPEG is the most common image format used by digital cameras and the most common format for storing and transmitting photographic images on the Web.

To conduct the experiment, all stego images embedded with different sizes of the same secret message are exposed to JPG attacks. After exposing the stego to JPEG compression attacks, secret image extraction is performed by applying error correction to recover the secret image. The PSNR of the recovered secret image is determined, tabulated, and compared with A. M. Molaei's Method.

5.4.3 Capacity

In this section, the embedding capacity of the proposed method and the method used is examined. The ratio of the number of secret bits embedded in the cover image to the number of pixels in the cover image is evaluated.

In this test, the experiments conducted above will compare 2 methods: polar codes with multiple embedding steganography and the previously used method (Reed-Muller code with multiple embedding). Each method will be tested using 30 different cover images and 30 different secret images.

Since the previous method used grayscale images while the proposed method uses color images, there will be 30 of those images that

5.5 Analysis and Evaluation Method

This research will design a series of experiments to evaluate the performance of the proposed method:

5.5.1 Imperceptibility

Mean Square Error (MSE) is a metric used to measure the difference between two images. MSE is calculated using the following formula:

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} [I(i,j) - K(i,j)]^2 \tag{1}$$

Where:

- M and N are the dimensions of the image.
- $I(i,j)$ is the pixel value at position (i,j) of the original image.
- $K(i,j)$ is the pixel value at position (i,j) of the stego image.

A lower MSE indicates that the stego image is more similar to the original image, meaning better visual quality.

Peak Signal-to-Noise Ratio (PSNR) is a metric used to measure the quality of the stego image compared to the original image. PSNR is calculated using the following formula:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{2}$$

Where:

- MAX_I is the maximum image intensity value (e.g., 255 for 8-bit images).
- MSE is the Mean Square Error between the original and stego images.

A higher PSNR indicates that the stego image is more similar to the original image, meaning better visual quality.

5.5.2 Robbustnes

The quantitative measure to be determined is the PSNR of the recovered secret image. This is determined using equations **??** and **??**:

Reed-Muller code RM(1,4) has been applied in this method due to its higher error correction capability. The number 1 indicates the order/level of the Reed-Muller code. The number 4 indicates the dimension or number of input variables.

Since r = 1 and m = 4, the message size k is calculated by:

$$k = \sum_{i=0}^{r} mi = 40 + 41 = 5bits \tag{3}$$

And the code size N is calculated by:

$$N = 2^m = 16bits \tag{4}$$

$$ECC = \frac{numberofcorrectablebits}{totalbits} = \frac{3}{16} = 18.75\% \tag{5}$$

After each embedding scenario is performed as described, the resulting stego

images are exposed under the following attacks:

1. Noise Attack
2. Cropping Attack
3. Scratching Attack
4. JPEG Compression Attack

5.5.3 Capacity

Embedding capacity, EC is then determined by the following equation:

$$EC = \frac{|S|}{H \cdot W}(bpp) \tag{6}$$

Where —S— is the number of embedded secret bits, H and W are the height and length of the image respectively. The resulting payload is expressed as a percentage. To evaluate the performance of the proposed method in terms of capacity, the following two main scenarios are considered:

1. Pixel Value Variation: Cover images and secret images are analyzed based on their image histograms.
    - 40 color scale cover images of size 512x512 pixels are used.
    - 40 color scale secret images of various sizes are used.
    - Each secret image with unique tonal distribution is embedded in each cover image with unique tonality as well.

2. Secret Image Size: For each secret image, 33 different size variations are performed.

    - Each size variation is embedded in a different cover image.
    - The capacity resulting from each embedding is recorded.

Results for each group are tabulated and PSNR values are determined and compared with the previous Method.

6 Work Plan and Time Schedule

The work plan is divided into several main phases spread across 4 semesters. The first semester focuses on foundational work including literature study, problem identification, contribution formulation and hypothesis development. Literature study begins in month 1, while problem identification through hypothesis formulation spans months 1-2, culminating in the research proposal in month 3.

The second semester concentrates on design and initial implementation. The encoding process design occupies months 4-7, while embedding schema implementation spans months 5-8. These overlapping periods allow for iterative refinement as implementation insights inform design improvements.

The third semester focuses on decoding process design (months 7-10) and sharing schema design (months 8-11). This phase integrates the encoding and embedding work from the previous semester into a complete system.

The final semester emphasizes testing and documentation. Imperceptibility, robustness and capacity testing are conducted in month 11. The thesis draft is

developed over months 10-12, incorporating all research findings and analysis.

The detailed schedule is shown in Table **??**, with blue cells indicating active work periods for each activity. This timeline allows for systematic development while maintaining flexibility to accommodate challenges that may arise during implementation.

Table 3: Activity Schedule

Activity		**SEMESTER**											
		1			**2**			**3**			**4**		
1	Literature study												
2	Problem identification												
3	Contribution formulation												
4	Hypothesis formulation												
5	Proposal												
6	Design Encoding Process												
7	Implement embedding schema												
8	Design decoding process												
9	Design sharing schema												
10	Imperceptibility testing												
11	Robbustnes testing												
12	Capacity testing												
13	Thesis draft												

Supervisor (I)'s Comments:

Comments about the title
Comments about the research method
Sign **Date:** (_____)

Supervisor (II)'s Comments:

Comments about the title
Comments about the research method
Sign **Date:** (_____)