RELIABLE DELIVERY OF COVERT DATA THROUGH OPEN IMAGE SHARING PLATFORMS USING STEGANOGRAPHY

M. D. L FERNANDO

2025

Reliable Delivery of Covert Data Through Open Image Sharing Platforms Using Steganography

A Thesis Submitted for the Degree of Master of Computer Science

M. D. L. Fernando
University of Colombo School of Computing
2025

DECLARATION

Name of the student: M.D.L FERNANDO
Registration number: 19440278
Name of the Degree Programme: Master of Computer Science
Project/Thesis title: Reliable Delivery of Covert Data through Open Image Sharing Platforms using Steganography

1. The project/thesis is my original work and has not been submitted previously for a degree at this or any other University/Institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

2. I understand what plagiarism is, the various types of plagiarism, how to avoid it, what my resources are, who can help me if I am unsure about a research or plagiarism issue, as well as what the consequences are at University of Colombo School of Computing (UCSC) for plagiarism.

3. I understand that ignorance is not an excuse for plagiarism and that I am responsible for clarifying, asking questions and utilizing all available resources in order to educate myself and prevent myself from plagiarizing.

4. I am also aware of the dangers of using online plagiarism checkers and sites that offer essays for sale. I understand that if I use these resources, I am solely responsible for the consequences of my actions.

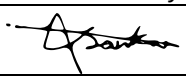
5. I assure that any work I submit with my name on it will reflect my own ideas and effort. I will properly cite all material that is not my own.

6. I understand that there is no acceptable excuse for committing plagiarism and that doing so is a violation of the Student Code of Conduct.

Signature of the Student	**Date**
	2025-06-22

Certified by Supervisor(s)

This is to certify that this project/thesis is based on the work of the above-mentioned student under my/our supervision. The thesis has been prepared according to the format stipulated and is of an acceptable standard.

	Supervisor 1	**Supervisor 2**	**Supervisor 3**
Name	Dr. Asanka P. Sayakkara		
Signature			
Date	2025-07-02		

I would like to dedicate this thesis to all who supported and encouraged me

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and special thanks to my research supervisor, Dr. Asanka P. Sayakkara, who provided me with the opportunity to conduct this study under his guidance and support. His insightful feedback and encouragement have been instrumental in overcoming challenges and enhancing the quality of this research.

Also, I would like to thank my parents, wife and my son for their encouragement and continuous support; it has been invaluable in helping me stay focused and committed to my academic journey.

ABSTRACT

In present days, sharing a secret message without drawing outsider attention is a challenge. However, image steganography facilitates this in a very easy way. Image steganography is a popular technique for embedding secret information within images to ensure confidentiality. There are various techniques available to manipulate steganographically processed images, called "stego-images". Transmitting stego-images via popular image-sharing platforms is more important due to several reasons. These platforms facilitate sharing images with a broader audience within a few seconds. Additionally, they are providing a less conspicuous means of communication compared to traditional methods. But these image-sharing platforms use various image processors like image compression, resizing, format conversion, and quality adjustment to optimize their performance and enhance the user experience. However, these image processors significantly degrade the integrity of secret information that is embedded using image steganography.

This research employs an experimental approach to explore the effects of platform-induced transformations on steganographic images, with a focus on the Least Significant Bit (LSB) image steganography technique. In this study, it involves embedding secret information into images, sharing it via the selected platforms (Facebook and WhatsApp), and assessing the data loss using the Payload Recovery Rate (PRR). Various image attributes are used to simulate the real-world scenario. This includes varying resolutions (low, mid, and high) and different formats (JPG, PNG). Two main hypotheses are tested: (H1) whether the application of error detection and correction methods significantly improves the reliability of information transmitted through a steganographic covert channel; and (H2) whether pre-processing techniques significantly enhance the reliability of information transmitted through a steganographic covert channel. Error correction methods (Reed Solomon, Hamming Code, and redundancy-based method) are used to investigate the hypothesis H1.

The experimental results of H1 reveal that the use of error correction methods can significantly improve the PRR, with high-resolution images on WhatsApp experiencing the most degradation due to resizing. To test H2, the high-resolution images that exhibited the most degradation on WhatsApp were resized to mid-resolution. The results demonstrate a significant improvement in PRR.

The outcome of this study highlights optimal image formats, resolutions, and error correction methods that help mitigate data loss, providing insights into enhancing the reliability of steganographic techniques in real-world scenarios.

Keywords: Image Steganography, Least Significant Bit, Error correction methods

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BMP	Bitmap
BPC	Binary Pattern Complexity
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
LSB	Least Significant Bit
PNG	Portable Network Graphics
PVD	Pixel Value Differencing
PRR	Payload Recovery Rate
TIFF	Tagged Image File Format

CHAPTER 1

INTRODUCTION

1.1 Chapter Overview

This chapter provides a detailed overview of the research work under these topics, including the domain, problem statement, motivation, specific computer science problem, research contribution, and its scope.

1.2 Problem Domain

Steganography is a method of hiding secret data inside a different digital medium, such as an image, audio, video, or text file. The ancient Greeks used this word "steganography" as a combination of two words. “Stego” means "cover,” and “Grayfia” means "writing." Therefore, people refer to it as "cover writing" or "hidden writing." Currently, the steganography method can be used for digital watermarking, secure data transmission without alerting, to save military secrets, etc. For the purpose of transmitting hidden messages, this can be achieved using both steganography and cryptography. Anyway, these two methods are serving in different ways to fulfill this requirement. Cryptography uses encryption to make the secret message unreadable. Therefore, even if outsiders intercept the secret message, they cannot understand it. However, the message still exists. Steganography serves the same function by embedding it within another medium, like an image, video, or audio file, without highlighting the concealed message [1].

The steganography domain categorizes image steganography because it involves embedding hidden information in an ordinary image. Figure 1 depicts the process of image steganography. The image steganography process consists of several components, as listed below:

- **Cover-Image:** Describes the original image that served as a carrier for secret information.
- **Message:** Secret information to be concealed within the cover image. This might be another image or text.
- **Stego-Image:** An image that contains embedded secret data.

- **Stego-Key:** This facilitates embedding hidden information inside the cover image or extracting the secret information from the Stego-Image.

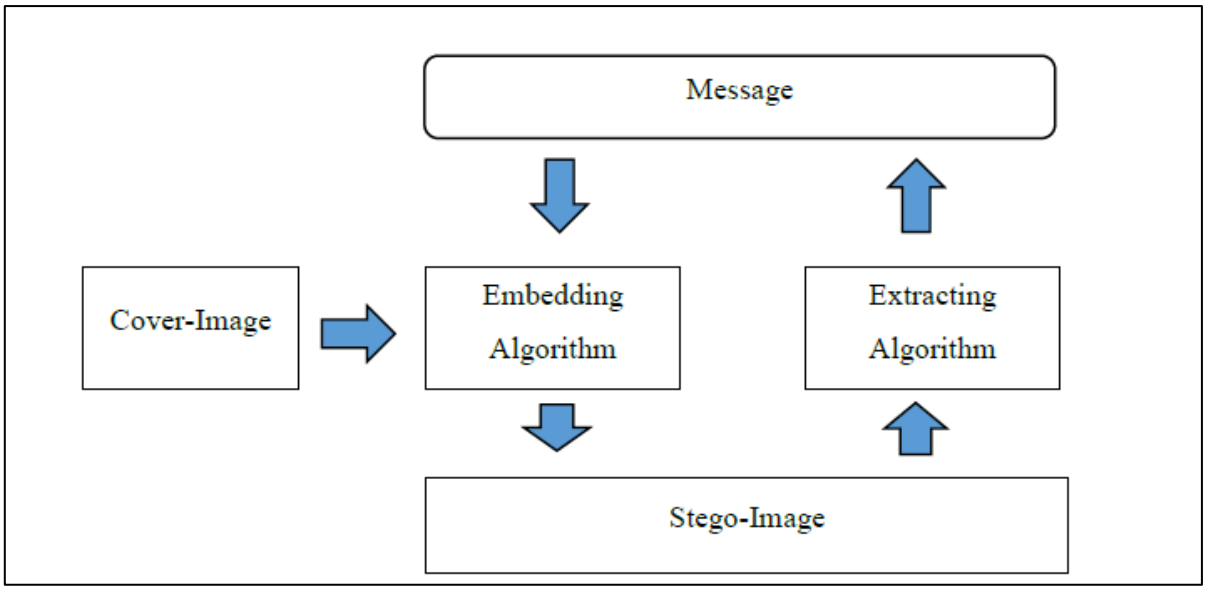


Figure 1. Image Steganography Process [2]

The transmission of stego-images through image sharing platforms holds significant importance for various reasons [2].

Stealthy Communication: With the advancement of the technologies and communication, image sharing via the common image sharing platforms has become a part of daily life since they provide a covert environment for hidden messages [1].

Wide Reach: People often register on these platforms because of their free availability, can be accessed using high end devices to low end devices, which leads to a larger audience. These platforms allow users to easily broadcast their secret messages to individuals and groups without drawing attention [1].

Ease to use: Usually image sharing platforms provide basic functionalities like simple and standard user interfaces for sharing images, support various file formats and sizes, offer faster image uploading and downloading facilities, include inbuilt image editing tools, and facilitate access via different devices in a responsive manner without reducing the user experience. Therefore, users

from various age groups, even those with minimal computer literacy, can easily use these functionalities [1].

However, common image sharing platforms like WhatsApp and Facebook Messenger use various image processing techniques to optimize their services. As a result, these processes can lead to partial or complete loss of embedded information.

1.3 Problem Statement

As previously mentioned, there are several well-known image processors available that lead to hidden information loss. Such as,

Image Compression: To reduce file sizes and facilitate faster uploading and downloading, images are often compressed. This compression can be lossy, such as in JPEG format, where some image data is irretrievably lost, or lossless, as in PNG format, where the image quality is preserved [3].

Resizing: Images may be resized to ensure they meet the platform's display requirements; optimize them for different devices and screen sizes. This resizing can alter the pixel structure of the image, potentially damaging the embedded data [4].

Format Conversion: To ensure consistency and compatibility, the platform may transform images into a standard format like JPEG, PNG, or WebP. During this process of image format conversion, there is a possibility to lose the hidden data embedded in the carrier image [3].

Quality Adjustment: Many image sharing platforms adjust their uploaded image qualities to optimize the overall performance. These platforms prioritize rapid image loading, display high-quality images, and minimal data consumption to deal with the users with different internet connection speeds. The main technique for altering image quality and file size is compression, and it leads to further degradation of the hidden information [3].

Watermarking: Some platforms may add watermarks to images for purposes of copyright protection or branding, which can overwrite or obscure the embedded steganographic data [3].

Considering those challenges, this study aims to investigate existing image steganography techniques, assess the impact of platform-specific image processing on stego-images, identify the most resilient image formats and techniques, and check whether it is possible to use pre-processing techniques and error detection and correction methods to improve the reliability of hidden data during transmission.

1.4 Motivation

The advancement of information and technology has increased the demand for ensuring the security and confidentiality of secret communications. Steganography has become more suitable for secret communication through the image-sharing platforms as it avoids the attention of outsiders. Therefore, it is crucial and timely to make it a more viable option for secret communication via the common image-sharing platforms. It is very interesting to investigate the reliability of existing image steganography techniques and image formats against modern image processing techniques and find ways to improve the reliability of information transmission.

1.5 Research Aim and Objectives

1.5.1 Research Aim

The research aims to assess how common image sharing platforms' image processors affect image steganography techniques and develop methods for reducing data loss by employing pre and post processing techniques, along with error correction and detection strategies.

1.5.2 Objectives

• **Identify Existing Image Steganography Techniques**: Start by conducting a literature survey to find existing steganography techniques that are used for embedding information into images.

• **Manipulate Steganographic Images**: Apply the identified steganography techniques to manipulate images, embedding hidden information. Then, share these stego-images through selected image-sharing platforms like WhatsApp and Facebook Messenger.

• **Analyze Information Loss**: Assess the information loss that occurs when the stego-images are transmitted and received through these platforms. This analysis will help understand the impact of platform-induced compression and manipulation on the hidden information.

• **Identify image formats and steganography techniques that are least affected**: Based on the previous analysis results, determine which image formats and steganography techniques exhibit the least data loss or distortion.

• **Algorithm/Methodology Development**: Develop algorithms or methodologies to minimize information loss in steganographic images during transmission through image sharing platforms. And also check the possibility of usage of error detection and correction methods as pre and post processors.

1.6 Scope

1.6.1 In Scope

- Identify existing steganography techniques.
- Analyze the information lost during transmission via image-sharing platforms.
- Check the possibility of using error detection and correction methods to minimize the loss.
- Check the possibility of using preprocessing techniques to minimize loss.

1.6.2 Out of Scope

- Create a new image steganography technique.

1.7 Structure of the Thesis

Introduction chapter, it provides a detailed introduction to the research, including the problem domain and problem statement, highlighting the key issues that it is going to address. Additionally, it explains the motivation for this research. Finally, it defines the research scope by specifying the boundaries and limitations with specific aspects of the problem that will be investigated.

The background and related work, second chapter will cover relevant information for this study, such as domain and technical reviews, existing steganography techniques, the impact of hidden information when sharing stego-images via image sharing platforms, and methods to minimize the data loss.

The methodology chapter is the third chapter and it details the methods that were chosen to accomplish the research objectives. It presents the proposed model, outlining the solution to address the existing gap, including key inputs and the stages of development.

The Evaluation and Results chapter discusses the outcomes of the proposed methodology with the statistics and details of the evaluation process. In other words, it describes the extent to which the proposed methodology meets the outlined objectives.

The Conclusion and Future Work chapter is the final chapter of this research document and it summarizes whether the research assumptions are achieved or not, challenges faced and future works that can be done to enhance the output.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Chapter Overview

The literature review chapter provides an overview of the steganography domain, including existing research works that are relevant to this study. It also provides a summary of current steganography techniques.

2.2 Image Representation

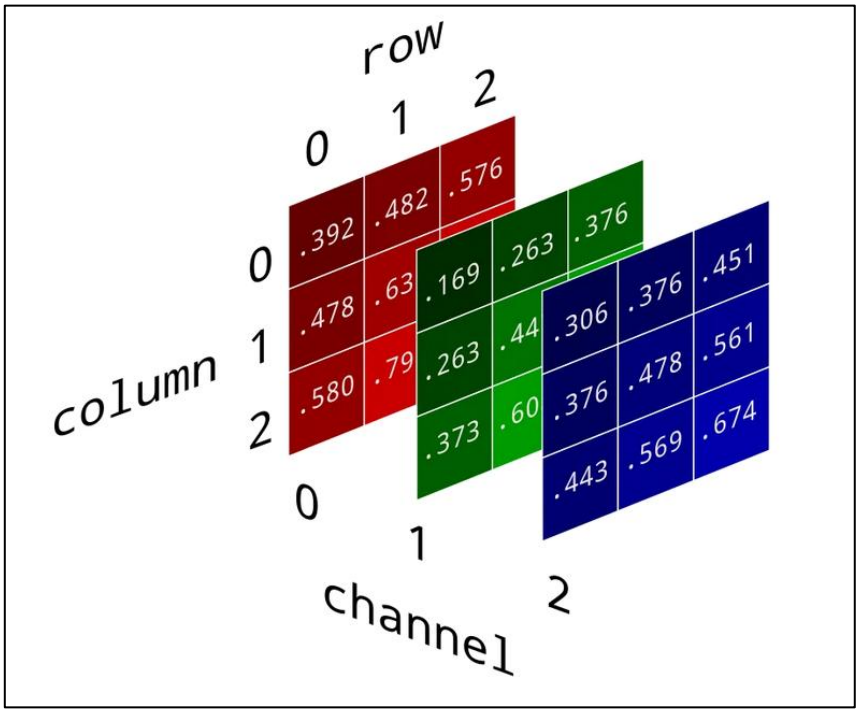


Figure 2. Pixel RGB Channel Array Representation [5]

An image is created with a collection of tiny pixels that are arranged in an array as in Figure 2. These pixels are displayed as very small squares filled with colors. The dimension of the pixel array is equal to the image's resolution. Basically, a color image's pixel contains 3 channels called (r.g.b), such as red, green, and blue, which are represented using a 3-dimensional array, and each cell value of the array represents the color intensity (i.e., brightness) of the corresponding channel [5].

Pixel’s array index can be used to navigate the image as coordinates. The parameters of the index are structured as [Channel Number, Row Number, Column Number]. This index will guide us through the image's channels, rows, and columns in that specific order. The intensity level range is between 0 and 255: 0 represents no light, and 255 represents the brightest. In other words, each color of these (r, g, b) are represented by 8-bit values. Therefore, a pixel has the ability to display more than 16 million color combinations [5].

Several popular image formats can be found on the Internet. Such as Joint Photographic Experts Group (JPEG), Graphics Interchange Format (GIF), Portable Network Graphics (PNG), etc. These image formats can be categorized into two groups considering the compression technique that they are used: lossy and lossless. JPEG uses a lossy compression algorithm to reduce the file size for storage purposes without impacting perceived image quality. But the PNG image format supports high-quality images, and it uses lossless compression. Bitmap (BMP) and Tagged Image File Format (TIFF) are not popular in image-sharing platforms due to their large file size [6].

2.3 Existing Image Steganography Techniques

There are various ways to classify the current steganography techniques, and these techniques use different types of algorithms in order to achieve their purposes. The algorithms that are used for cover types (i.e., file types) and techniques for embedding secret information serve as the primary foundation for these categories [6].

However, currently available steganography techniques primarily fall into two categories: spatial domain and frequency domain. The spatial domain techniques directly embed secret message bits in the pixel values of the cover image. Unlike the spatial domain, frequency domain techniques transform the cover image into a different domain for the purpose of embedding the secret message. Figure 3 and Figure 4 display the steganography process of both spatial and frequency domains [7].

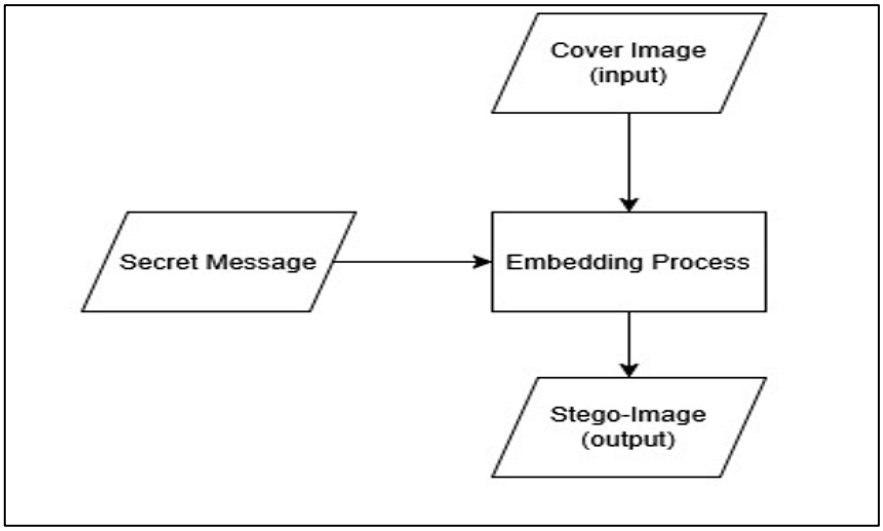


Figure 3. Steganography Process in Spatial Domain (Adapted from [7])

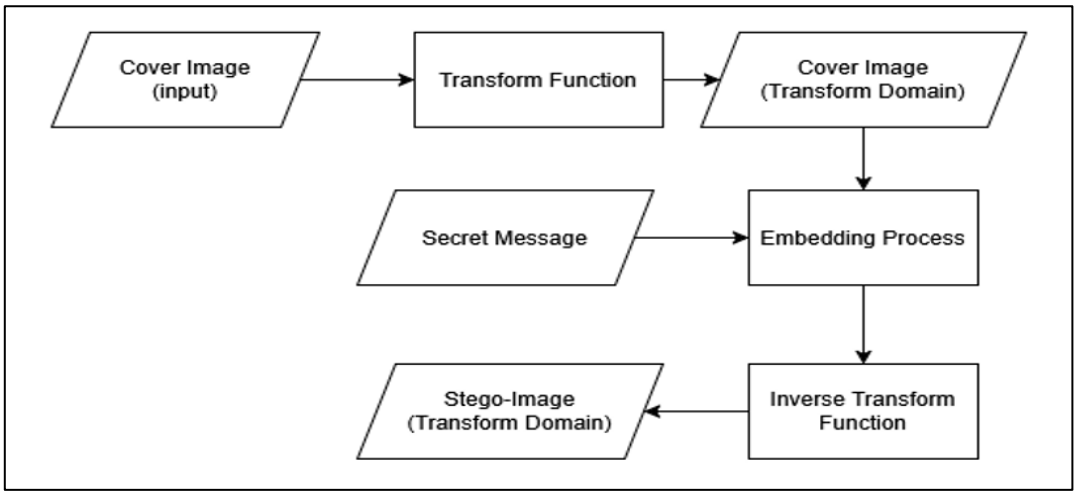


Figure 4. Steganography Process in Frequency Domain (Adapted from [7])

Recently, lossy image compression has become more important to assessing the robustness of the steganography techniques. The researchers have decided to look into the second strategy further and categorized it into five categories [6].

- Spatial domain technique
- Transform domain
- Spread spectrum
- Statistical methods
- Distortion techniques

2.3.1 Spatial Domain Techniques

- **Least Significant Bit (LSB) Insertion**

This is the simplest and most popular image steganography technique, which replaces the least significant bits of the carrier image with the bits of the hidden data. It requires low processing power and is easy to implement. It uses two types of data embedding schemes. Such as sequential and scattered. The scattered scheme embeds data in a random sequence. It is also very easy to use. However, this method is vulnerable to detection through statistical analysis, and image processors such as compression, filtering, and resizing can easily distort the hidden data in it [8].

As previously mentioned, a pixel color contains three channels. There are red, green, and blue. These three colors can generate other colors, as shown in Figure 5.

	R	G	B
black	0	0	0
red	255	0	0
green	0	255	0
blue	0	0	255
white	255	255	255

Figure 5. Colors Representation Using RGB Channels inside a Pixel [8]

Each channel within a pixel occupies 1 byte, or 8 bits. For instance, Table 1 represents the color blue as the integer 255, which is equivalent to 11111111 in binary. If we modify the last bit to 11111110, the color change in the output will not be noticeable to the human eye due to its minor nature. Therefore, we can store up to 3 bits per pixel using each channel [8].

	RED	GREEN	BLUE
Integer	0	0	255
Binary	00000000	00000000	11111111

Table 1. RGB Integer and Binary Representation [8]

- **Pixel Value Differencing (PVD)**

This method was initially proposed to hide secret data in grayscale images, and it provides a higher capacity for embedding secret data. However, it doesn't show any noticeable differences to the human eye. It determines the number of bits that can be embedded to the cover image based on the differences between adjacent pixel values. Therefore, it offers better resistance to visual attacks compared to LSB Insertion. But embedding capacity is slightly lower than LSB since it depends on the difference between neighboring pixels. It reads an image in a zigzag manner from the top left corner to the bottom. It is complex to implement and may lead to significant image quality degradation. *Figure 6* depicts the process of embedding the secret data using PVD [9].

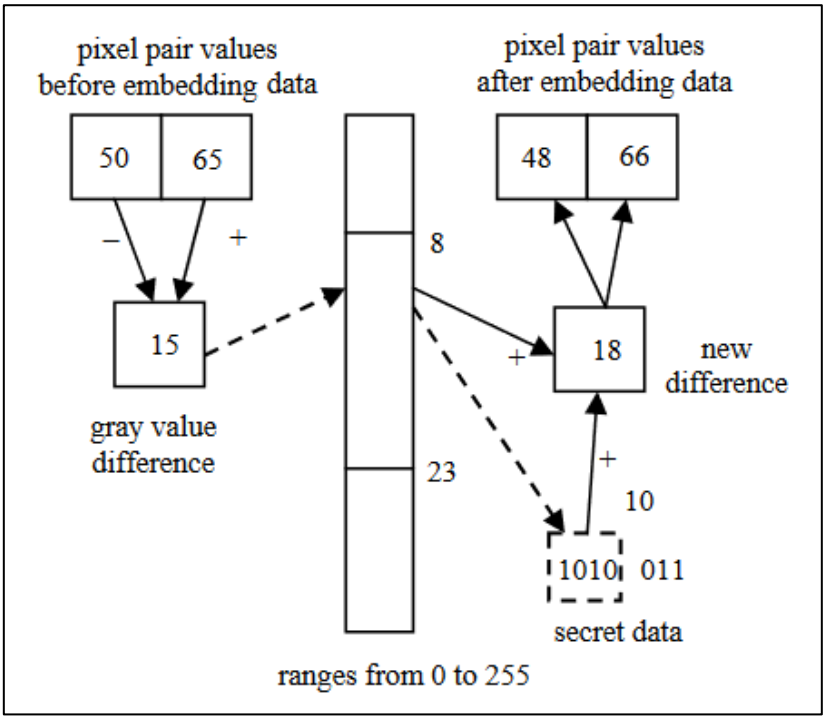


Figure 6. Embedding Secret Data Using Pixel Value Differencing [9]

- **Binary Pattern Complexity (BPC)**

The Binary Pattern Complexity technique assesses the level of noise in the image complexity and replaces the noisy area with a binary pattern that is mapped with hidden data. The image will revert to its original form once the reverse noise factor is applied. The process of analyzing bit plane complexity and selecting appropriate planes can be computationally expensive [10].

2.3.2 Transformation Domain Techniques

- **Discrete Fourier Transform (DFT)**

This technique is most widely used in the analysis of time-based sequences, and it was developed by Joseph Baptiste Fourier and primarily used to convert spatial visual images into a frequency-based representation. It provides higher embedding capacity, but it requires more processing power and also higher implementation complexity. This technique utilizes discrete time and transforms it into a continuous frequency. The primary disadvantage of this technique is that it eliminates all time-related information when converting from a time sequence to a frequency series [10].

- **Discrete Cosine Transform (DCT)**

This method converts images from spatial domain to frequency domain, like the discrete fourier transform (DFT) method. In steganography, DCT divides an image into 8x8 pixel blocks and converts them into 64 DCT coefficients. The process of converting each pixel block to DCT involves navigating the image from left to right and from top to bottom. These blocks are compressed by using a quantization table to scale the DCT coefficient for embedding the secret message. When retrieving the image, the inverse discrete cosine transformation (IDCT) method is used to reconstruct the image by decompressing the blocks. This method is more resilient against the compression, but computationally it requires more processing [10].

- **Discrete Wavelet Transform (DWT)**

The discrete wavelet transform technique is primarily used for analyzing non-stationary signals, and it relies on small waves that have finite duration, known as wavelets. DWT converts an image from the spatial domain to the frequency domain, like DFT and DCT. It provides a dual representation of the image by capturing both frequency and spatial description. DWT is a more precise model than DFT and DCT. But it requires more processing power, and it is complex to implement. The popular JPEG 2000 image compression standard is also based on the DWT. It requires more processing power, and it is complex to implement [10].

2.3.3 Spread Spectrum

This technique spreads hidden data across the broad frequency range, and it provides strong robustness, as even if parts of the data are lost from some bands, the secret message can still be recovered if enough data remains in other bands. When the signal-to-noise ratio is low in every frequency band, it is hard to detect the existence of the hidden message [11].

2.3.4 Statistical Methods

In this method, it embeds a hidden message by modifying some properties in the cover. It divides the cover image into blocks and embeds one message bit in each of those blocks. The cover block modification is required if the message bit equals one [10].

2.3.5 Distortion Techniques

This method uses distortion signals to embed secret data into the cover and its process can be divided into two phases. Such as encoder and decoder. In the encoder phase, it applies a sequence of changes to the cover image and in the decoder phase, it decodes the encrypted data to the originals using a secret key [10].

2.4 Challenges Introduced by Image Sharing Platforms

In this study, researchers investigated the feasibility of hiding data on four popular photo-sharing sites. There are Google+, Facebook, Twitter, and Flickr. They discovered that various image processing functions cause a significant loss of hidden data in stego-images. They are as follows:

- **Metadata Modification**: In Twitter, Flickr, and Facebook certain metadata fields' contents are modified or not available in the downloaded images. Twitter applies image processors when the image size exceeds 1024x768 pixels. Otherwise, it does not affect the image integrity [12].

- **Pixel Alteration:** When an image exceeds 2048 pixels in either the length or width dimensions, Facebook resizes it. Resizing causes the pixels to shift from their original locations or even lose them. Modification of pixel values in uploaded images can impact the hidden information of stego images. Some pixels' RGB (red, green, and blue) values undergo deviations up to 30 units, while 60% of the pixels maintain their original values. Furthermore, they discovered that these alterations conform to a Gaussian distribution, implying random behavioral changes [12].

- **Changes in Compression Ratio**: When comparing the original with the downloaded images on Facebook, they found different compression ratios. And also, they found that in 70% of all cases, there is a change in the quality factor for both the luminance and chrominance in color JPEG images, which ranges from 70 to 95 [12].

- **Changes in DCT Coefficients:** According to the results, DCT coefficients in Facebook are almost identical to the DCT coefficients in Flickr for certain images, and researchers discovered 15% of a change in downloaded images when compared with originals [12].

The researcher chose widely used and publicly available tools like GhostHost, StegHide, OutGuess, F5, and YASS to embed hidden data into images. Then they evaluated the retrieval of the hidden messages after sharing these images via the selected platforms. Results are listed in Figure 7.

× = Failure ✓ = Success ✓✓ = Conditional Success

Tool	Details on approach	Facebook	Twitter	Flickr	Google+
GhostHost	Embedding after the EOI marker	×	×	×	✓
Steghide	Changing Pixel Values	×	✓	×	✓
Outguess	Changing DCT coefficients (pseudo random)	×	✓	×	✓
F5	Changing DCT coefficients (non-zero)	×	✓	×	✓
YASS	Changing DCT coefficients (error correcting)	✓✓	✓	✓✓	✓

Figure 7. Steganography Tool Evaluation with Image Sharing-Platforms (Adapted from [12])

Using Facebook for Image Steganography

The authors of this paper selected Facebook as the image-sharing platform and used the JPEG image file type as a carrier for embedding secret messages. In this experiment, they uploaded images at different resolutions to observe the output image parameters. They used the tools OpenPuff, OutGuess Rebirth, F5, JP Hide & Seek, Steghide, Steg, OurScret, and Incognito to embed hidden messages in images. His findings were as follows [13].

The resolution of downloaded images is adjusted based on their original size. When the image size is larger than or equal to 960 pixels in width, it is typically resized to either 2048 × yyyy or 960 × yyyy pixels. When uploading images with high-quality settings, the ratio of the uploaded file size to the downloaded file almost reaches the range of 0.9 to 1.06 [13].

The tools OpenPuff, OutGuess Rebirth, OurSecret, and F5 did not support retrieving the hidden message from the downloaded carrier images. The Incognito was not compatible with JPEG format carriers. They defined the complete recovery of hidden text as a measure of success and found that

success rates were 15% for JPHide & Seek and 50% for StegHide. They also discovered that the hidden payload size ranged from 1 to 400 bytes, with the success rate decreasing as the size increased [13].

Currently, WhatsApp and Facebook have gained significant popularity; however, there are only a limited number of studies focused on the use of steganography for hidden message sharing through these platforms, which continue to evolve with frequent updates and new features. This represents a notable gap in the existing literature.

CHAPTER 3

METHODOLOGY

3.1 Chapter Overview

The methodology chapter describes the research design, data collection and implementing process, experimental procedures, and evaluation methods employed to address the research objectives and hypotheses.

3.2 Research Philosophy

This study employs a **pragmatic** research philosophy for several reasons. For instance, the study is outcome-oriented and focuses on practical solutions to address the real-world problems, such as improving the resilience of image steganography against image-sharing platforms that employ image processors. Furthermore, this study requires the flexibility of integrating both quantitative and qualitative methods to achieve comprehensive outcomes.

3.3 Research Design

The research design describes the systematic approach to fulfilling the research objectives and hypothesis. The study adopts **mixed-method** research that combines both **qualitative** insight with **quantitative experimental** approaches. Literature review involves identifying the existing steganography techniques, platforms' image processors, related works, and shortlisting methods for the experiment. The experiment measures the impact of platform-induced image processors by performing several identified phases. The experiment evaluates the following two hypotheses.

H1: The application of error detection and correction methods significantly improves the reliability of information transmitted through a steganographic covert channel.

H2: Pre-processing techniques significantly enhance the reliability of information transmitted through a steganographic covert channel.

The experiment mainly involves three stages. Figure 8 depicts the process flow of the research design.

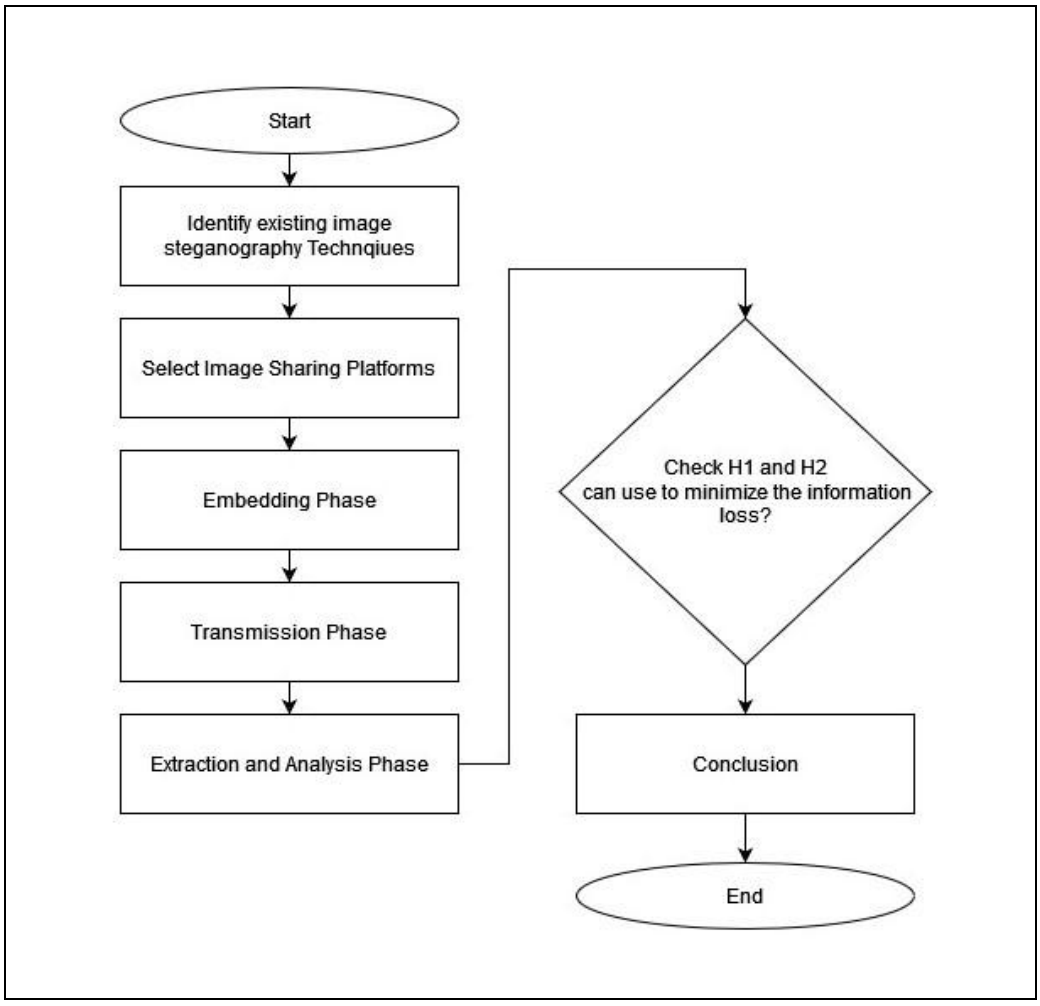


Figure 8. Process Flow of Research Design

Embedding Phase: Hide the data within the images using selected steganography techniques. The process embeds payloads into various types of images. Such as different formats, color spaces and resolutions.

Transmission Phase: Stego images are uploaded to and downloaded from selected image-sharing platforms.

Extraction and Analysis Phase: Extract the hidden data from downloaded images and assess the integrity of the payload. H1 and H2 hypotheses are applied to measure the improvement of the extracted data.

3.4 Literature Review Framework

This section outlines the theoretical foundation for implementing the research design and methodology, which is based on the insight of existing studies that are identified in Chapter 2.

Steganography Techniques

The review investigated the most popular steganography techniques in order to find their capabilities, and it discovered two categories: spatial domain and frequency domain.

- The spatial domain techniques directly embed secret message bits into the pixel values in the cover image. (e.g., Least Significant Bit, Pixel Value Differencing, Binary Pattern Complexity)

- The frequency domain techniques transform the cover image into the different domain for the purpose of embedding the secret message. (e.g., Discrete Fourier Transform , Discrete Cosine Transform , Discrete Wavelet Transform)

Based on the literature review chapter, Table 2 presents a comparison of popular steganography techniques considering several key attributes.

Technique	Complexity of Implementation	Processing Power	Embedding Capacity	Compression Resiliency
Least Significant Bit (LSB)	Low	Low	High	Low
Pixel Value Differencing (PVD)	Moderate	Moderate	Moderate	Moderate
Binary Pattern Complexity (BPC)	High	High	High	High
Discrete Fourier Transform (DFT)	Moderate	Moderate	Moderate	High
Discrete Cosine Transform (DCT),	Moderate	Moderate	Moderate	High
Discrete Wavelet Transform (DWT)	High	High	High	High

Table 2. Comparison of Steganography Techniques

According to the literature review, the LSB technique was chosen for analyzing data loss in image-sharing platforms due to its simplicity, high embedding capacity, and low computational complexity. And also it has low resiliency against compression and ease of implementation.

Image Formats

The review revealed that the majority of image-sharing platforms commonly utilize various image formats and their respective compression techniques. JPEG uses lossy compression to reduce the file size, and PNG uses lossless compression to store higher-quality images. Additionally, it emphasizes the importance of using both lossy and lossless formats in the experiment to analyze the integrity of the data against the compression.

Image Sharing Platforms and Image Processors

The review discovered the impact of image-sharing platform processors, including compression, resizing, format conversion, and other functions. Some platforms like Google+ and Twitter have less impact on images, while Facebook and Flickr platforms perform heavy compressions.

Research Gaps Identified

The review revealed some key areas that need further investigations.

- Fewer studies have integrated error detection and correction methods and utilized pre-processing techniques to mitigate the impact of image processors.

- Lack of recent studies on current updated image-sharing platforms. Platforms such as Facebook and WhatsApp frequently update their processors due to heavy usage.

3.5 Images for Steganography

The images are playing a vital role in this study because they enable the evaluation of steganography techniques with different conditions. This section outlines the characteristics, sources and preparation process of the image collection for steganography. This image collection consists of a varied set of images to ensure the experiment covers the real-world scenarios for this study. Table 3 lists the characteristics and sources of the images.

Image Characteristics	• Image formats: JPEG and PNG • Resolution Ranges: 640x480 (Low), 1280x720 (Mid), 1920x1080 (High) • Color Spaces: RGB

Data Sources	• Web pages: Pexels, vecteezy • Generated Images using photo editing tools • Captured Images from smartphone

Table 3. Images Characteristics and Sources

3.6 Implementation of the Experiment

3.6.1 Requirements

Hardware Requirements

- Processor: Intel core i5/i7
- Ram 8GB: or higher
- Storage: SSD 256GB
- Other: Smart Phone for capturing images

Software Requirements

- Programming
    - Language: Python
    - Libraries
        - Image Processing: Open CV, PIL
        - Steganography Implementation: PySteg, Steganography Libraries, or custom algorithms.
        - Error Correction: NumPy, Reed-Solomon libraries for advanced methods.
        - Data Analysis: Numpy, Pandas, Matplotlib
- Image Editing Tools: Photoshop, GIMP
- Web Browsers: Chrome, Mozilla
- Platforms: Facebook, WhatsApp

3.6.2 Implementation Steps

Image Preparation

- **Generate images for missing categories**: Image editing tools like Photoshop and GIMP are used to generate images when the required combinations of categories fail to be found on the internet.

- **Preprocessing**: Images are resized or padded to standard dimensions for consistency during testing. Metadata is stripped to prevent interference with steganographic embedding.

- **Image Categories**: Images are categorized based on their format, resolution, and encoded algorithms.

Embedding data using steganography

- **Generate stego images:** Embed payloads using previously selected image steganography technique. Payload contains text or binary data, and its size remains fixed to maintain consistency.

- **Embed Error Correction Code:** Payloads are encoded using error correction algorithms before embeds them into the image.

Data transmission via platforms

- **Upload and Download:** Share the payloads embedded images (stego-images) via selected platforms like Facebook, WhatsApp, etc., and download them. Downloaded images are properly stored along with the originally uploaded stego-images for further investigations.

Extraction and Analysis

- **Analysis:** Mainly use Payload Recovery Rate (PRR) to measure the data loss in the payload. It is calculated as follows.

  PRR = (correct bits in extracted payload / bits in embedded payload) x 100

Optimization

- **Check the hypotheses:** Apply image preprocessing and error detection and correction techniques to mitigate the information loss induced by image sharing platforms. Repeat the experiment and check the effect on PRR values.

CHAPTER 4

EVALUATION AND RESULTS

4.1 Chapter Overview

This chapter provides a detailed analysis of the evaluation results in order to validate the effectiveness of the proposed methods for improving the reliability of image steganography under various image sharing platform transformations. Summary of evaluations results and findings are detailed below.

4.2 Baseline Results

Table 4 shows the average PRR percentage values, calculated from the mean values of 10 images for each combination. Baseline results are generated using the LSB steganography method without the support of either pre-processing or error correction encoding methods. Figure 9 displays the Table 4 results graphically for further understanding.

Platform	**Resolution**	**Format**	**Average PRR (%)**
FACEBOOK	LOW	JPG	35.00
FACEBOOK	LOW	PNG	43.18
FACEBOOK	MID	JPG	40.97
FACEBOOK	MID	PNG	32.05
FACEBOOK	HIGH	JPG	39.55
FACEBOOK	HIGH	PNG	33.75
WHATSAPP	LOW	JPG	35.00
WHATSAPP	LOW	PNG	47.10
WHATSAPP	MID	JPG	40.97
WHATSAPP	MID	PNG	41.59
WHATSAPP	HIGH	JPG	34.89
WHATSAPP	HIGH	PNG	31.02

Table 4. Baseline Payload Recovery Rate (PRR)

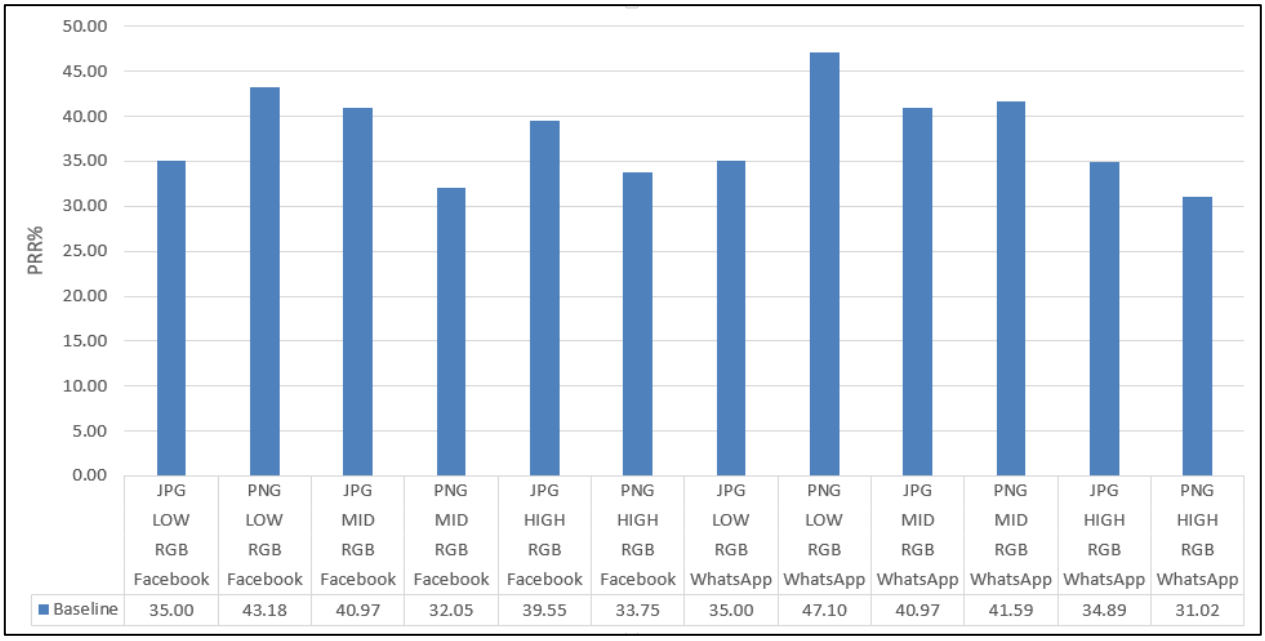


Figure 9. LSB Baseline PRR (%)

Table 5 presents a comparison of image attributes before uploading and after downloading, including format and resolution changes across Facebook and WhatsApp. The analysis highlights how selected image-sharing platforms are making changes to the format and the resolutions of the images. This comparison provides insights into the extent of image alterations that occur during transmission through these platforms.

Platform	**Uploaded Format**	**Uploaded Resolution**	**Downloaded Resolution**	**Downloaded Format**
FACEBOOK	LOW (640x480)	JPG	LOW (640x480)	JPG
FACEBOOK	LOW (640x480)	PNG	LOW (640x480)	JPG
FACEBOOK	MID (1280x720)	JPG	MID (1280x720)	JPG
FACEBOOK	MID (1280x720)	PNG	MID (1280x720)	JPG
FACEBOOK	HIGH (1920x1080)	JPG	HIGH (1920x1080)	JPG
FACEBOOK	HIGH (1920x1080)	PNG	HIGH (1920x1080)	JPG
WHATSAPP	LOW (640x480)	JPG	LOW (640x480)	JPG
WHATSAPP	LOW (640x480)	PNG	LOW (640x480)	JPG
WHATSAPP	MID (1280x720)	JPG	MID (1280x720)	JPG
WHATSAPP	MID (1280x720)	PNG	MID (1280x720)	JPG

WHATSAPP	HIGH (1920x1080)	JPG	MID (1280x720)	JPG
WHATSAPP	HIGH (1920x1080)	PNG	MID (1280x720)	JPG

Table 5. Image Attributes Comparison Uploaded vs. Downloaded

Figures 10, 11, and 12 display the original images compared to the baseline-encoded images in the order of low, mid, and high quality. Left side image represents the original image and right side image represents the baseline encoded image. According to these images it is not an easy to identify the difference between original and encoded image using naked eyes.

Figure 10. Low Resolution Original vs. Baseline Encoded

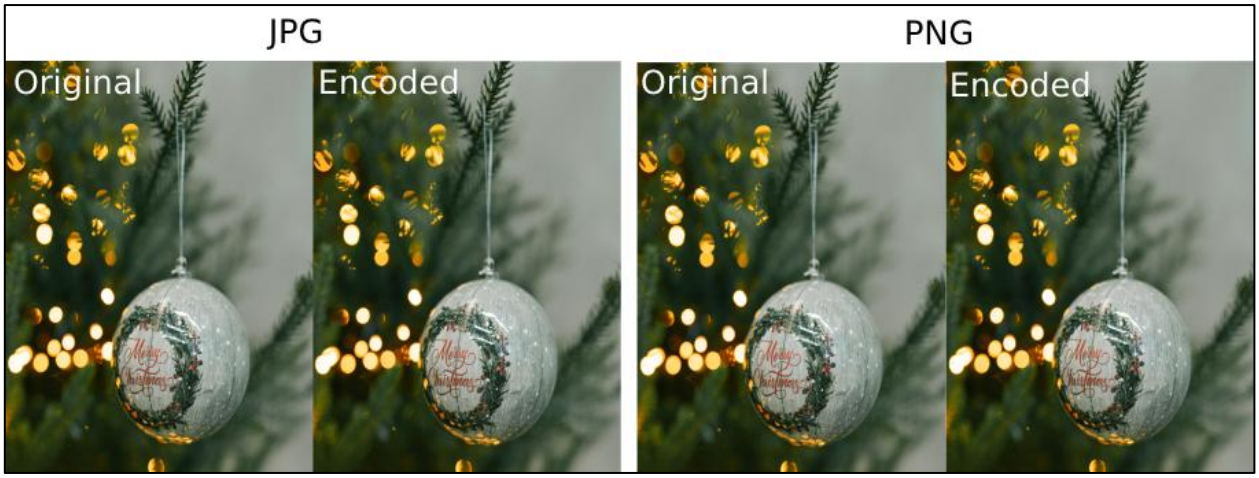


Figure 11. Mid resolution original vs. Baseline Encoded

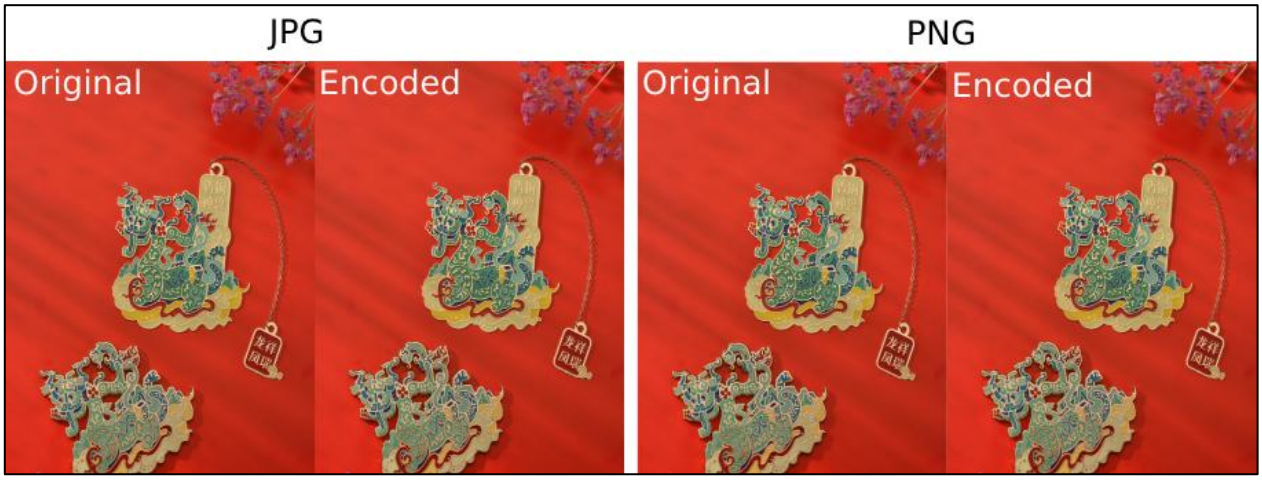


Figure 12. High Resolution Original vs. Baseline Encoded

Observations:

- According to the Figure 9, when the resolution increases, PRR value decreases for PNG format more than JPG format. It indicates that higher-resolution PNG files may experience more compressions and a higher rate of image quality loss. Neither JPG nor PNG has been capable of reaching 50% of PRR value due to the platform transformations.

- Without preprocessing or error correction, LSB has failed to reach 50% of PRR using either JPG or PNG formats due to the platform transformations.

- PNG format generally achieves a higher PRR than JPG in most cases and for the higher resolutions, JPG remains more stable than PNG.

- Low-resolution PNG images, especially on WhatsApp, offer the best baseline PRR.

- According to Table 5, all uploaded image formats are converted to JPG on both platforms, indicating that these platforms perform format conversion. Additionally, high-resolution images uploaded to WhatsApp are resized to mid-resolution. This resizing may lead to a lower PRR for high-resolution images on WhatsApp.

4.3 Error Correction Method - Hamming Code

Table 6 depicts the average percentage PRR values of Hamming Code with different redundancy bits. The secret message was encoded by using Hamming code before embedding it into the image. Figure 13 represents the graphical view of table 6.

Platform	**Resolution**	**Format**	**PRR(%) for number of bits attached**					
			2	**3**	**4**	**5**	**6**	**7**
FACEBOOK	LOW	JPG	46.70	43.81	41.99	43.75	37.33	42.05
FACEBOOK	LOW	PNG	37.78	48.35	43.64	36.65	37.22	40.34
FACEBOOK	MID	JPG	44.60	44.20	44.55	40.40	39.77	36.42
FACEBOOK	MID	PNG	40.00	43.13	39.20	45.68	39.26	41.42
FACEBOOK	HIGH	JPG	48.30	44.89	42.61	41.31	42.78	42.67
FACEBOOK	HIGH	PNG	43.41	50.91	42.61	40.68	42.44	47.10
WHATSAPP	LOW	JPG	46.70	43.81	41.99	43.75	37.33	42.05
WHATSAPP	LOW	PNG	42.16	47.73	41.93	38.24	37.33	41.99
WHATSAPP	MID	JPG	44.60	44.20	44.55	40.40	39.77	36.42
WHATSAPP	MID	PNG	35.00	37.39	43.07	45.97	41.31	34.89
WHATSAPP	HIGH	JPG	45.00	44.20	41.14	46.36	39.38	38.35
WHATSAPP	HIGH	PNG	45.17	44.38	34.72	34.66	41.65	38.01

Table 6. PRR Comparison Hamming Code with Different Redundancy Bits

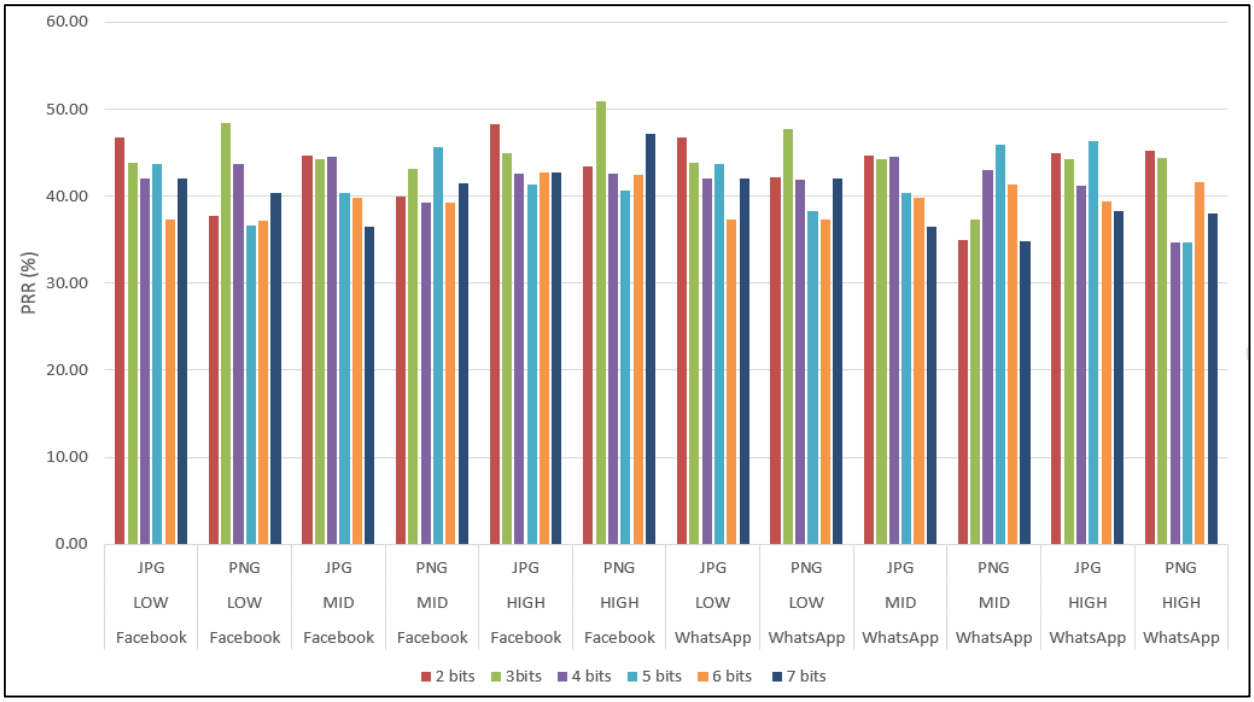


Figure 13. Hamming Code PRR (%) for Different Redundancy Bits

According to the results in Table 6, the Hamming code achieves higher PRR values when encoded with 3 redundancy bits. Figures 14, 15, and 16 illustrate the original images in the order of low, mid, and high quality alongside those encoded with the Hamming code using three redundancy bits.

Figure 14. Low Resolution Original vs. Hamming Encoded

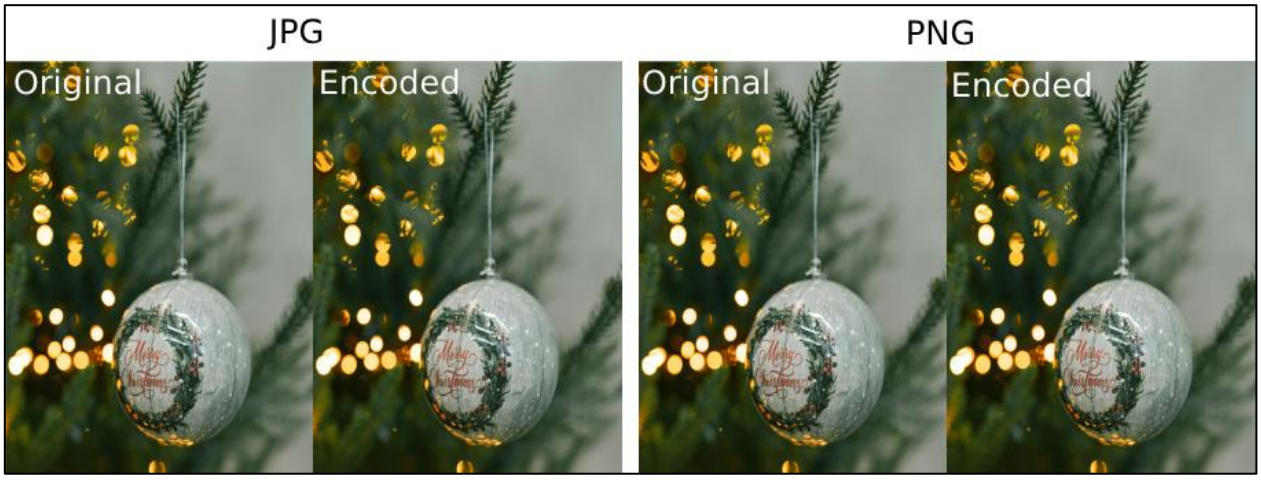


Figure 15. Mid Resolution Original vs. Hamming Encoded

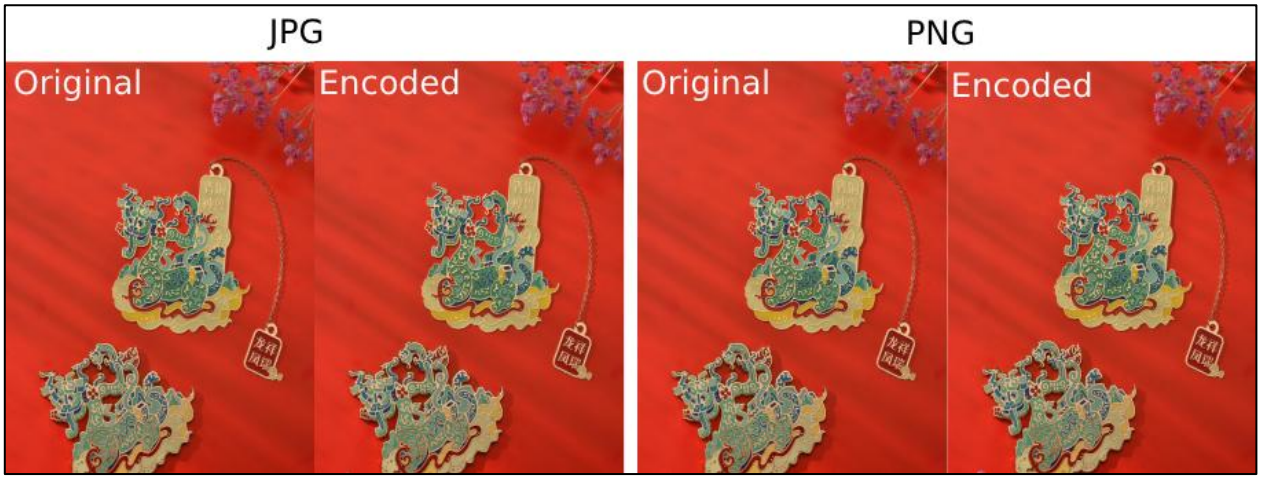


Figure 16. High Resolution Original vs. Hamming Encoded

Observations:

- In most cases, Hamming code PRR values are higher than baseline PRR values (without Hamming encoding). It clearly indicates that the use of the Hamming error-correction method helps improve data recovery.

- Three redundancy bits achieve the highest overall average PRR (44.75%).

- The PRR value does not always increase with the larger number of redundancy bits. It implies that higher redundancy may occur with a low PRR due to noise.

- The PRR tends to fluctuate as the number of redundancy bits increases, with lower values at 6 and 7 redundancy bits.

- PNG images show better PRR on Facebook (e.g., 50.91% for high-resolution PNG with 3 bits) and slightly lower PRR on WhatsApp at a higher redundancy level.

- High-resolution images show better PRR compared to low- and mid-resolution images, especially for PNG format.

- For JPG, mid-resolution images do not show significant improvement compared to low-resolution ones.

4.4 Baseline vs. Hamming (3-Bits Redundancy) PRR Comparison

The Hamming code with three redundancy bits achieves the highest overall PRR average, making it the preferred choice for further comparison. Table 7 presents a comparison of PRR values between the baseline results and Hamming code results, while Figure 17 provides a graphical representation of these findings.

Platform	**Resolution**	**Format**	**Baseline**	**Hamming Code 3 bits**
FACEBOOK	LOW	JPG	35.00	43.81
FACEBOOK	LOW	PNG	43.18	48.35
FACEBOOK	MID	JPG	40.97	44.20
FACEBOOK	MID	PNG	32.05	43.13
FACEBOOK	HIGH	JPG	39.55	44.89
FACEBOOK	HIGH	PNG	33.75	50.91
WHATSAPP	LOW	JPG	35.00	43.81
WHATSAPP	LOW	PNG	47.10	47.73
WHATSAPP	MID	JPG	40.97	44.20
WHATSAPP	MID	PNG	41.59	37.39
WHATSAPP	HIGH	JPG	34.89	44.20
WHATSAPP	HIGH	PNG	31.02	44.38

Table 7. Comparison Between Baseline vs. Hamming 3 Bits Redundancy

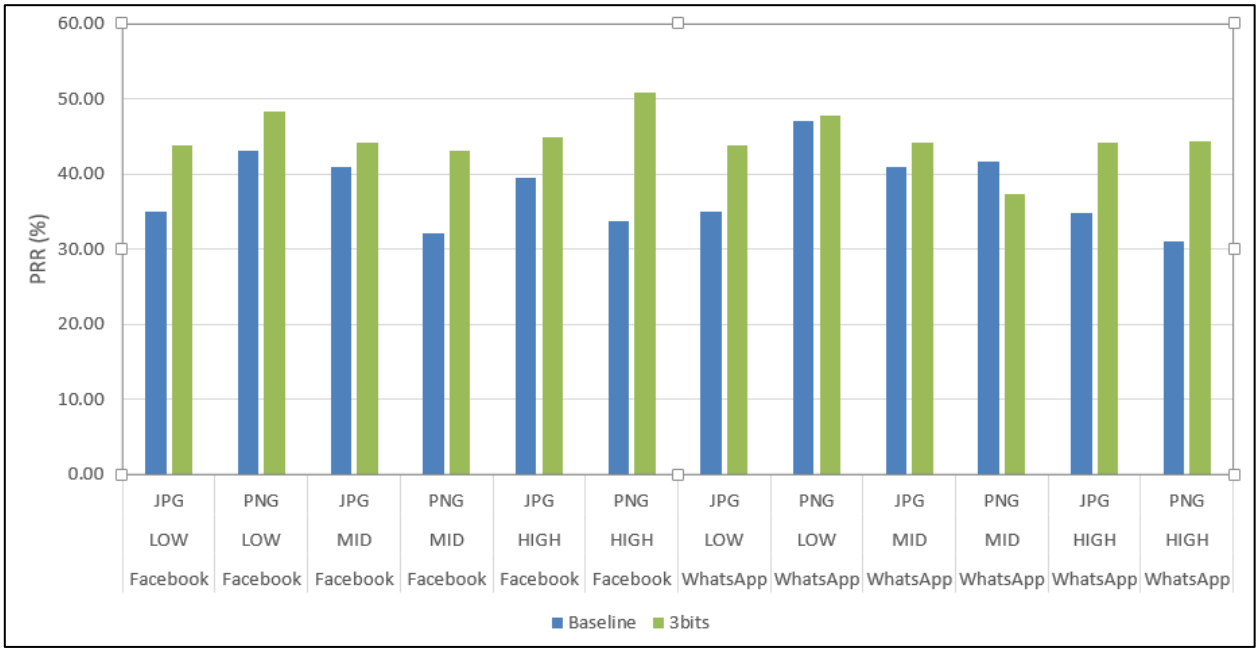


Figure 17. Comparison Between Baseline vs. Hamming Code with 3 Bits

Observations:

- Hamming code encoding with 3 redundancy bits improves PRR value across all cases when compared with the baseline.
- On both platforms, low-resolution images show an improvement of 8–10%.
- It shows the Hamming code performs better on Facebook than WhatsApp.
- PNG images show better PRR improvement with Hamming code compared to JPG.
- The best improvement is seen in Facebook HIGH PNG (+17.16%), while WhatsApp MID PNG has a slight PRR drop.

4.5 Error Correction Method - Reed Solomon

The Reed-Solomon error correction method was used to assess whether it can improve the reliability of hidden data that was embedded using the LSB method. However, it failed to recover the hidden data during the decoding phase. The decoding function of the Reed-Solomon returned an error indicating that the number of errors exceeded its error correction handling capability. This highlights that the image processors of Facebook and WhatsApp are making a significant impact on the degradation of the embedded data.

4.6 Error Correction Method - Redundancy Based method

This method was tested with different numbers of redundancy bits, and it gave different PRR values accordingly. Table 8 presents the average PRR values for each combination, and Figure 18 provides a graphical representation of these findings.

Platform	**Resolution**	**Format**	**No of Redundancy bits attached**						
			1	**2**	**3**	**4**	**5**	**6**	**7**
FACEBOOK	LOW	JPG	50.57	0.63	96.14	37.61	99.49	100.00	96.70
FACEBOOK	LOW	PNG	49.66	1.19	98.75	37.27	99.49	91.19	91.70
FACEBOOK	MID	JPG	50.80	0.63	99.94	40.06	99.20	91.65	91.76
FACEBOOK	MID	PNG	49.38	1.08	99.83	23.64	96.59	91.65	99.89
FACEBOOK	HIGH	JPG	52.05	0.00	99.94	30.97	96.70	92.44	97.05
FACEBOOK	HIGH	PNG	50.51	1.08	99.89	27.16	95.74	92.27	97.22
WHATSAPP	LOW	JPG	50.57	0.63	96.14	37.61	99.49	100.00	96.70
WHATSAPP	LOW	PNG	51.93	1.53	98.86	32.90	96.53	100.00	96.70
WHATSAPP	MID	JPG	50.80	0.63	99.94	40.06	99.20	91.65	91.76
WHATSAPP	MID	PNG	49.66	1.31	100.00	27.84	99.26	100.00	99.83
WHATSAPP	HIGH	JPG	36.65	1.08	43.30	0.00	40.00	3.64	37.22
WHATSAPP	HIGH	PNG	40.11	0.45	46.59	0.00	45.85	0.34	30.80

Table 8. PRR Comparison Redundancy Based Encoding with Different Redundancy Bits

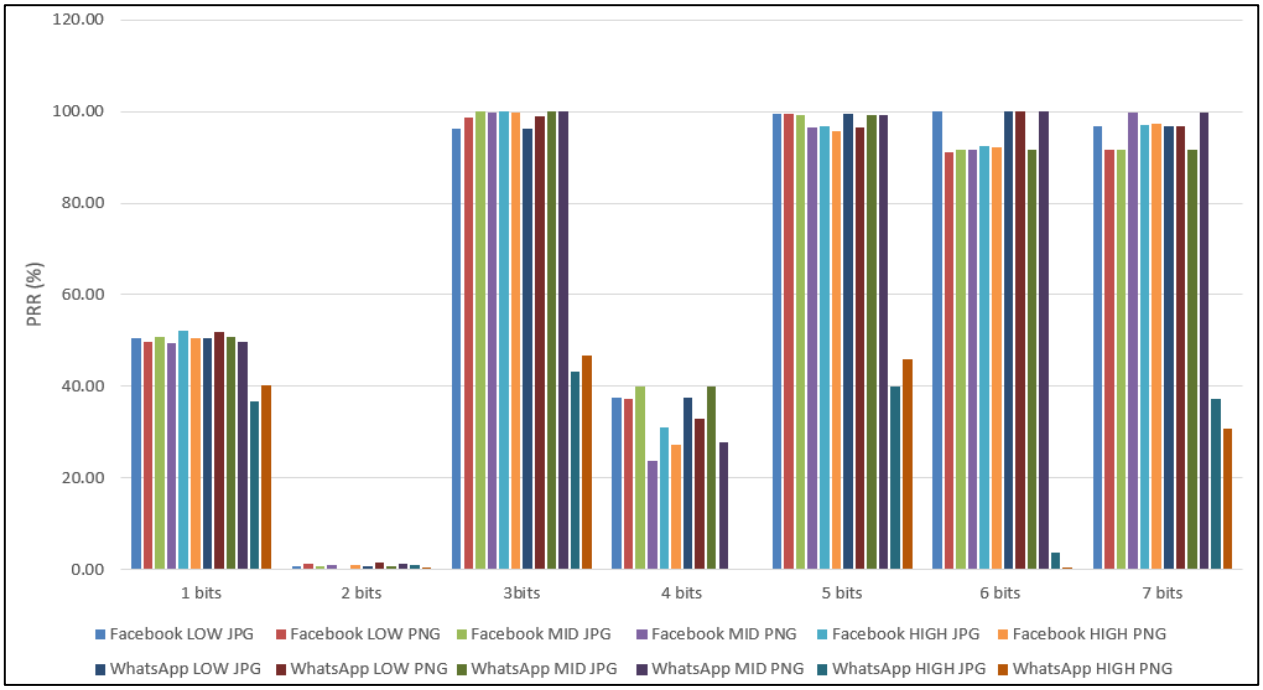


Figure 18. Redundancy Based PRR (%) for Different Redundancy Bits

Based on the results in Table 8, the Redundancy based method achieves higher PRR values when encoded with 3 redundancy bits. Figures 19, 20, and 21 illustrate the original images alongside those encoded with the Redundancy based method using three redundancy bits.

Figure 19. Low Resolution Original vs. Redundancy Based Encoded

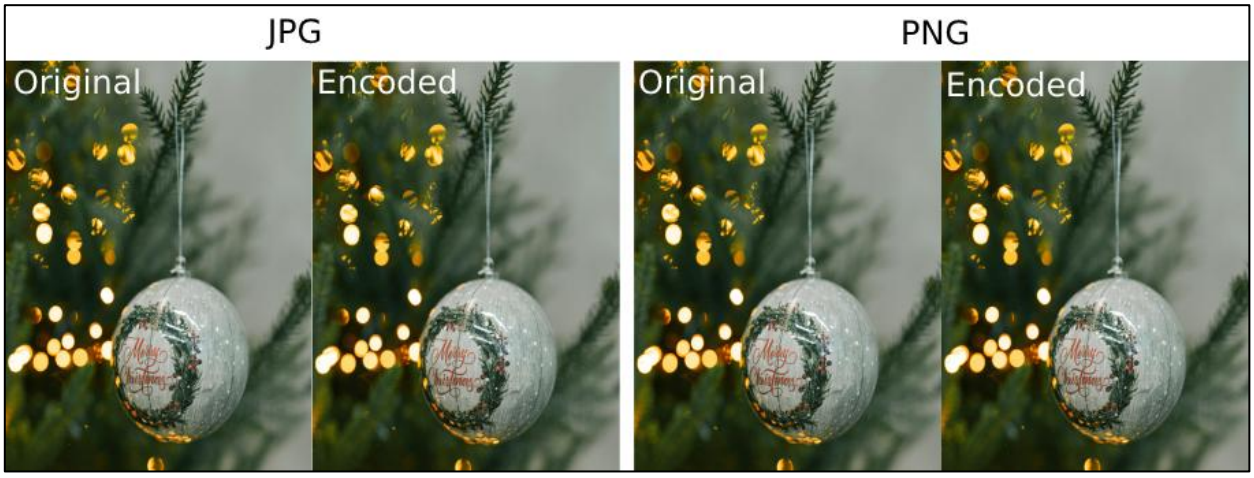


Figure 20. Mid Resolution Original vs. Redundancy Based Encoded

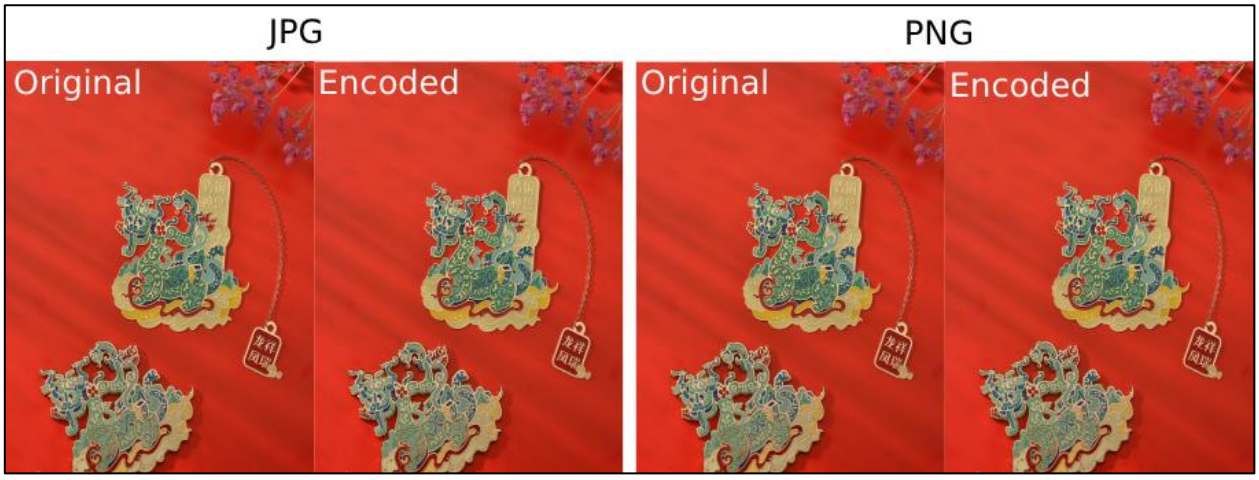


Figure 21. High Resolution Original vs. Redundancy Based Encoded

Observations:

- The PRR values increase with the number of redundancy bits increases, peaking around 3 to 5 bits, and decrease gradually after 5 redundancy bits.

- The highest overall average PRR (89.94%) is achieved with 3 redundancy bits, making it the most effective redundancy level for error recovery.

- For WhatsApp high-resolution images, PRR values drop drastically at 6 and 7 bits (e.g., JPG: 0.34% at 6 bits, PNG: 30.80% at 7 bits).

- High-resolution WhatsApp images exhibit the lowest PRR values, especially beyond three redundancy bits. Additionally, WhatsApp images experience noticeable resolution degradation. The original image resolution of 1080×1920 was reduced to 720×1280 upon download, indicating that WhatsApp applies image resizing when uploaded resolutions exceed the platform-defined limits. This resizing process leads to significant hidden information loss in high-resolution images with WhatsApp.

4.7 Hamming (3-Bits) vs. Redundancy Based (3-Bits) PRR Comparison

The Redundancy based method with three bits achieves the highest overall PRR average, making it the preferred choice for further comparison. Table 9 presents a comparison of PRR values between the Hamming code with 3 bits results and redundancy based 3 bits results, while Figure 22 provides a graphical representation of these findings.

Platform	**Resolution**	**Format**	**Hamming Code 3 bits**	**Redundancy based 3 bits**
FACEBOOK	LOW	JPG	43.81	96.14
FACEBOOK	LOW	PNG	48.35	98.75
FACEBOOK	MID	JPG	44.20	99.94
FACEBOOK	MID	PNG	43.13	99.83
FACEBOOK	HIGH	JPG	44.89	99.94
FACEBOOK	HIGH	PNG	50.91	99.89
WHATSAPP	LOW	JPG	43.81	96.14
WHATSAPP	LOW	PNG	47.73	98.86
WHATSAPP	MID	JPG	44.20	99.94
WHATSAPP	MID	PNG	37.39	100.00
WHATSAPP	HIGH	JPG	44.20	43.30
WHATSAPP	HIGH	PNG	44.38	46.59

Table 9. PRR Comparison Hamming Code 3 Bits vs. Redundancy Based 3 Bits

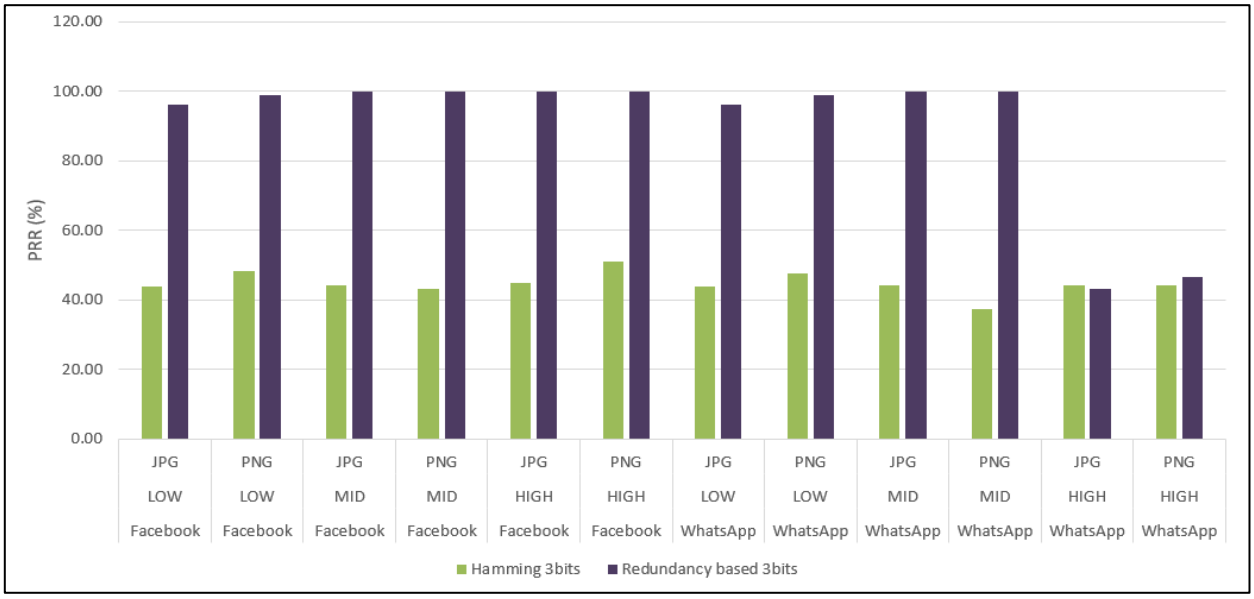


Figure 22. Comparison Between Hamming Code 3 Bits vs. Redundancy Based 3 Bits

Observations:

- The redundancy-based method with 3 bits achieves significantly higher PRR values compared to the Hamming code with 3 bits across all cases.

- In many cases of mid-resolution images, the redundancy-based method achieves higher recovery rates (~99-100%).

- The redundancy-based method performs exceptionally well for Facebook mid-resolution and high-resolution images, with PRR values exceeding 99%.

- The Hamming code consistently achieves PRR values in the range of 43% to 51%, which is significantly lower than the redundancy-based method.

- This indicates that while Hamming code provides some error correction, it is less effective in recovering the payload compared to the redundancy-based approach.

4.8 Pre-Processing - Resizing

Table 10 presents the impact of resizing as a pre-processing technique to improve the reliability of hidden data in stego-images shared through WhatsApp. High-resolution images that exhibited the most degradation on WhatsApp were resized to mid-resolution, and a redundancy-based approach with 3 bits was applied.

Redundancy based 3 bits					
Platform	**Format**	**Original Resolution**	**PRR Before Resizing**	**Resized Resolution**	**PRR After Resizing**
WHATSAPP	JPG	HIGH (1920x1080)	43.30	MID (1280x720)	100.00
WHATSAPP	PNG	HIGH (1920x1080)	46.59	MID 1280x720)	100.00

Table 10. Redundancy Based 3 Bits PRR Comparison High resolution Original vs. Resized

Observations:

- Before resizing, high-resolution (1920×1080) images experienced significant data loss. After resizing to mid-resolution (1280×720), the PRR improved dramatically to 100.00% for both JPG and PNG images.

- The poor PRR before resizing indicates that WhatsApp's image processing significantly disrupts embedded hidden data, particularly in high-resolution images.

- The fact that PRR reaches 100% after resizing suggests that mid-resolution images are less affected by WhatsApp's compression mechanisms, preserving the hidden data more effectively.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Chapter Overview

This chapter elaborates a summary of this research, highlighting major findings, contributions, and lessons learned. Additionally, it assesses the achieved objectives, and the direction for future research.

5.2 Conclusion

This study investigated how image-sharing platforms affect stego-images with the goal of minimizing the amount of data lost by platform-performed image processors. A comprehensive literature review was conducted to identify existing image steganography techniques, with the Least Significant Bit (LSB) method selected for further experimentation. A variety of images were used to embed the hidden information and share them through Facebook and WhatsApp. Payload Recovery Rate (PRR) was used to analyze the information loss.

The outcome of the experiment discovers that image format, resolution, and platform-induced processors significantly affect PRR. PNG format images retained more embedded data compared to JPEG images. Error correction techniques with different redundancy bits were evaluated to determine the optimal configuration for retrieving the higher PRR. Redundancy-based encoding discovers a significant improvement in the reliability of embedded data However, the lowest PRR rates were observed in high-resolution images uploaded to WhatsApp due to platform-induced resizing in the redundancy-based approach. To address this, the high-resolution images that exhibited the most degradation on WhatsApp were resized to mid-resolution. The results then demonstrate a significant improvement in PRR.

The results confirm both hypotheses: (H1) error detection and correction methods enhance the reliability of covert communication, and (H2) pre-processing techniques contribute to reducing data loss. This study discovers the insights to enhance the reliability of hidden data using image steganography on image-sharing platforms.

5.3 Future Work

Future research can extend on this study by investigating the following areas;

- Analyzing other steganography techniques, such as Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT), to evaluate their resilience against platform-induced processors like compression, format conversion, resizing, etc.

- Analyzing other popular image-sharing platforms, such as Telegram, Instagram, and Twitter, to understand their impact on steganographic data transmission.

- Exploring machine learning-based error correction methods or hybrid approaches to enhance hidden data recovery.

- Exploring methods to optimize the hidden data capacity while maintaining the quality of the stego-image.

By addressing these areas, future studies can contribute to developing more resilient steganographic approaches that maintain data integrity across various image-sharing platforms.

APPENDICES

APPENDIX A: Test Results

Table A1, A2 and A3 depict the detailed representation of the PRR results for each image. Table A1 displays the baseline PRR results, while Table A2 provides the results after applying Hamming encoding with different bits. Table A3 presents the PRR results incorporating redundancy-based techniques.

Baseline Results

Platform	Resolution	Format	Img. No	PRR (%)
FACEBOOK	LOW	JPG	1	45.45
FACEBOOK	LOW	JPG	2	10.80
FACEBOOK	LOW	JPG	3	47.73
FACEBOOK	LOW	JPG	4	17.05
FACEBOOK	LOW	JPG	5	0.00
FACEBOOK	LOW	JPG	6	53.98
FACEBOOK	LOW	JPG	7	37.50
FACEBOOK	LOW	JPG	8	40.91
FACEBOOK	LOW	JPG	9	47.16
FACEBOOK	LOW	JPG	10	49.43
FACEBOOK	LOW	PNG	1	50.57
FACEBOOK	LOW	PNG	2	38.07
FACEBOOK	LOW	PNG	3	44.32
FACEBOOK	LOW	PNG	4	23.30
FACEBOOK	LOW	PNG	5	49.43
FACEBOOK	LOW	PNG	6	47.16
FACEBOOK	LOW	PNG	7	50.00
FACEBOOK	LOW	PNG	8	39.20
FACEBOOK	LOW	PNG	9	49.43
FACEBOOK	LOW	PNG	10	40.34

FACEBOOK	MID	JPG	1	46.59
FACEBOOK	MID	JPG	2	24.43
FACEBOOK	MID	JPG	3	49.43
FACEBOOK	MID	JPG	4	27.84
FACEBOOK	MID	JPG	5	54.55
FACEBOOK	MID	JPG	6	44.89
FACEBOOK	MID	JPG	7	17.05
FACEBOOK	MID	JPG	8	47.16
FACEBOOK	MID	JPG	9	48.30
FACEBOOK	MID	JPG	10	49.43
FACEBOOK	MID	PNG	1	46.59
FACEBOOK	MID	PNG	2	27.84
FACEBOOK	MID	PNG	3	42.05
FACEBOOK	MID	PNG	4	25.57
FACEBOOK	MID	PNG	5	43.75
FACEBOOK	MID	PNG	6	0.00
FACEBOOK	MID	PNG	7	27.27
FACEBOOK	MID	PNG	8	6.82
FACEBOOK	MID	PNG	9	50.00
FACEBOOK	MID	PNG	10	50.57
FACEBOOK	HIGH	JPG	1	46.59
FACEBOOK	HIGH	JPG	2	44.89
FACEBOOK	HIGH	JPG	3	48.30
FACEBOOK	HIGH	JPG	4	6.25
FACEBOOK	HIGH	JPG	5	45.45
FACEBOOK	HIGH	JPG	6	39.77
FACEBOOK	HIGH	JPG	7	22.16
FACEBOOK	HIGH	JPG	8	46.02
FACEBOOK	HIGH	JPG	9	46.59
FACEBOOK	HIGH	JPG	10	49.43
FACEBOOK	HIGH	PNG	1	46.59

FACEBOOK	HIGH	PNG	2	46.59
FACEBOOK	HIGH	PNG	3	48.30
FACEBOOK	HIGH	PNG	4	2.84
FACEBOOK	HIGH	PNG	5	55.68
FACEBOOK	HIGH	PNG	6	50.57
FACEBOOK	HIGH	PNG	7	6.82
FACEBOOK	HIGH	PNG	8	30.68
FACEBOOK	HIGH	PNG	9	46.02
FACEBOOK	HIGH	PNG	10	3.41
WHATSAPP	LOW	JPG	1	45.45
WHATSAPP	LOW	JPG	2	10.80
WHATSAPP	LOW	JPG	3	47.73
WHATSAPP	LOW	JPG	4	17.05
WHATSAPP	LOW	JPG	5	0.00
WHATSAPP	LOW	JPG	6	53.98
WHATSAPP	LOW	JPG	7	37.50
WHATSAPP	LOW	JPG	8	40.91
WHATSAPP	LOW	JPG	9	47.16
WHATSAPP	LOW	JPG	10	49.43
WHATSAPP	LOW	PNG	1	50.00
WHATSAPP	LOW	PNG	2	55.11
WHATSAPP	LOW	PNG	3	51.70
WHATSAPP	LOW	PNG	4	23.30
WHATSAPP	LOW	PNG	5	46.59
WHATSAPP	LOW	PNG	6	48.86
WHATSAPP	LOW	PNG	7	51.70
WHATSAPP	LOW	PNG	8	39.20
WHATSAPP	LOW	PNG	9	56.25
WHATSAPP	LOW	PNG	10	48.30
WHATSAPP	MID	JPG	1	46.59
WHATSAPP	MID	JPG	2	24.43

WHATSAPP	MID	JPG	3	49.43
WHATSAPP	MID	JPG	4	27.84
WHATSAPP	MID	JPG	5	54.55
WHATSAPP	MID	JPG	6	44.89
WHATSAPP	MID	JPG	7	17.05
WHATSAPP	MID	JPG	8	47.16
WHATSAPP	MID	JPG	9	48.30
WHATSAPP	MID	JPG	10	49.43
WHATSAPP	MID	PNG	1	46.59
WHATSAPP	MID	PNG	2	44.89
WHATSAPP	MID	PNG	3	44.89
WHATSAPP	MID	PNG	4	15.34
WHATSAPP	MID	PNG	5	52.84
WHATSAPP	MID	PNG	6	43.18
WHATSAPP	MID	PNG	7	42.05
WHATSAPP	MID	PNG	8	27.27
WHATSAPP	MID	PNG	9	53.98
WHATSAPP	MID	PNG	10	44.89
WHATSAPP	HIGH	JPG	1	46.59
WHATSAPP	HIGH	JPG	2	26.14
WHATSAPP	HIGH	JPG	3	14.20
WHATSAPP	HIGH	JPG	4	13.07
WHATSAPP	HIGH	JPG	5	53.98
WHATSAPP	HIGH	JPG	6	45.45
WHATSAPP	HIGH	JPG	7	12.50
WHATSAPP	HIGH	JPG	8	42.05
WHATSAPP	HIGH	JPG	9	43.75
WHATSAPP	HIGH	JPG	10	51.14
WHATSAPP	HIGH	PNG	1	46.59
WHATSAPP	HIGH	PNG	2	50.00
WHATSAPP	HIGH	PNG	3	10.23

WHATSAPP	HIGH	PNG	4	24.43
WHATSAPP	HIGH	PNG	5	49.43
WHATSAPP	HIGH	PNG	6	46.59
WHATSAPP	HIGH	PNG	7	22.16
WHATSAPP	HIGH	PNG	8	6.25
WHATSAPP	HIGH	PNG	9	6.82
WHATSAPP	HIGH	PNG	10	47.73

Table A 1. Baseline Results

Hamming Encoded Results

Platform	**Resolu.**	**Format**	**Img. No**	**No of Redundancy bits attached**					
				2	**3**	**4**	**5**	**6**	**7**
FACEBOOK	LOW	JPG	1	40.34	49.43	51.70	47.73	46.02	40.34
FACEBOOK	LOW	JPG	2	46.02	47.16	55.11	51.70	34.66	46.02
FACEBOOK	LOW	JPG	3	46.02	21.02	48.30	33.52	49.43	46.02
FACEBOOK	LOW	JPG	4	42.61	51.70	51.70	54.55	45.45	42.61
FACEBOOK	LOW	JPG	5	52.84	40.91	41.48	50.00	50.57	52.84
FACEBOOK	LOW	JPG	6	47.73	49.43	50.00	42.61	48.86	47.73
FACEBOOK	LOW	JPG	7	48.86	44.89	35.23	46.02	3.98	48.86
FACEBOOK	LOW	JPG	8	46.59	35.80	13.07	54.55	14.77	46.59
FACEBOOK	LOW	JPG	9	46.59	47.73	47.73	6.25	28.98	46.59
FACEBOOK	LOW	JPG	10	49.43	50.00	25.57	50.57	50.57	49.43
FACEBOOK	LOW	PNG	1	5.68	47.16	47.16	42.61	50.00	5.68
FACEBOOK	LOW	PNG	2	40.34	50.57	50.57	57.39	54.55	40.34
FACEBOOK	LOW	PNG	3	50.00	50.57	50.57	45.45	46.02	50.00
FACEBOOK	LOW	PNG	4	26.14	50.00	55.68	53.98	13.64	26.14
FACEBOOK	LOW	PNG	5	45.45	47.73	46.02	19.32	50.00	45.45
FACEBOOK	LOW	PNG	6	52.84	43.75	51.70	50.57	55.11	52.84

FACEBOOK	LOW	PNG	7	43.75	53.98	3.98	2.84	2.84	43.75
FACEBOOK	LOW	PNG	8	14.20	50.00	27.27	40.34	4.55	14.20
FACEBOOK	LOW	PNG	9	54.55	43.75	53.41	2.27	48.30	54.55
FACEBOOK	LOW	PNG	10	44.89	46.02	50.00	51.70	47.16	44.89
FACEBOOK	MID	JPG	1	14.77	47.16	49.43	43.75	46.02	14.77
FACEBOOK	MID	JPG	2	51.70	47.73	42.61	11.93	53.98	51.70
FACEBOOK	MID	JPG	3	55.68	13.64	48.30	46.59	49.43	55.68
FACEBOOK	MID	JPG	4	26.70	35.80	21.02	50.57	23.86	26.70
FACEBOOK	MID	JPG	5	52.84	46.59	48.30	46.59	45.45	52.84
FACEBOOK	MID	JPG	6	50.57	50.57	47.73	44.32	40.34	50.57
FACEBOOK	MID	JPG	7	43.18	47.16	36.36	50.57	6.82	43.18
FACEBOOK	MID	JPG	8	53.41	46.02	48.86	22.16	23.86	53.41
FACEBOOK	MID	JPG	9	53.41	52.84	51.14	43.18	54.55	53.41
FACEBOOK	MID	JPG	10	43.75	54.55	51.70	44.32	53.41	43.75
FACEBOOK	MID	PNG	1	3.41	48.30	48.30	44.32	48.86	3.41
FACEBOOK	MID	PNG	2	47.73	48.86	21.59	55.68	17.61	47.73
FACEBOOK	MID	PNG	3	44.32	48.30	47.16	52.27	42.05	44.32
FACEBOOK	MID	PNG	4	22.16	30.68	19.89	35.23	25.00	22.16
FACEBOOK	MID	PNG	5	47.73	49.43	46.59	53.41	10.80	47.73
FACEBOOK	MID	PNG	6	55.11	52.27	48.86	47.16	47.73	55.11
FACEBOOK	MID	PNG	7	22.73	44.32	48.86	25.00	52.84	22.73
FACEBOOK	MID	PNG	8	52.27	15.34	16.48	50.57	50.57	52.27
FACEBOOK	MID	PNG	9	51.14	50.00	46.59	46.02	44.89	51.14
FACEBOOK	MID	PNG	10	53.41	43.75	47.73	47.16	52.27	53.41
FACEBOOK	HIGH	JPG	1	46.02	47.16	48.86	44.32	48.86	46.02
FACEBOOK	HIGH	JPG	2	52.27	39.77	50.57	51.14	50.57	52.27
FACEBOOK	HIGH	JPG	3	46.02	46.02	40.34	22.16	52.27	46.02
FACEBOOK	HIGH	JPG	4	50.00	46.59	5.68	7.39	41.48	50.00

FACEBOOK	HIGH	JPG	5	43.75	49.43	46.02	50.00	47.73	43.75
FACEBOOK	HIGH	JPG	6	51.14	47.73	45.45	47.16	45.45	51.14
FACEBOOK	HIGH	JPG	7	47.16	24.43	35.23	50.57	39.20	47.16
FACEBOOK	HIGH	JPG	8	50.00	51.14	47.73	51.14	43.75	50.00
FACEBOOK	HIGH	JPG	9	46.59	45.45	52.84	45.45	11.36	46.59
FACEBOOK	HIGH	JPG	10	50.00	51.14	53.41	43.75	47.16	50.00
FACEBOOK	HIGH	PNG	1	42.61	47.16	48.86	44.32	48.86	42.61
FACEBOOK	HIGH	PNG	2	48.30	51.14	50.57	47.73	36.36	48.30
FACEBOOK	HIGH	PNG	3	50.00	54.55	40.34	17.61	12.50	50.00
FACEBOOK	HIGH	PNG	4	5.68	56.25	5.68	40.34	43.75	5.68
FACEBOOK	HIGH	PNG	5	51.70	48.30	46.02	47.73	48.30	51.70
FACEBOOK	HIGH	PNG	6	44.32	48.86	45.45	49.43	43.75	44.32
FACEBOOK	HIGH	PNG	7	51.14	53.98	35.23	51.14	48.30	51.14
FACEBOOK	HIGH	PNG	8	52.84	47.16	47.73	2.84	46.59	52.84
FACEBOOK	HIGH	PNG	9	44.89	50.00	52.84	48.30	51.70	44.89
FACEBOOK	HIGH	PNG	10	42.61	51.70	53.41	57.39	44.32	42.61
WHATSAPP	LOW	JPG	1	40.34	49.43	51.70	47.73	46.02	40.34
WHATSAPP	LOW	JPG	2	46.02	47.16	55.11	51.70	34.66	46.02
WHATSAPP	LOW	JPG	3	46.02	21.02	48.30	33.52	49.43	46.02
WHATSAPP	LOW	JPG	4	42.61	51.70	51.70	54.55	45.45	42.61
WHATSAPP	LOW	JPG	5	52.84	40.91	41.48	50.00	50.57	52.84
WHATSAPP	LOW	JPG	6	47.73	49.43	50.00	42.61	48.86	47.73
WHATSAPP	LOW	JPG	7	48.86	44.89	35.23	46.02	3.98	48.86
WHATSAPP	LOW	JPG	8	46.59	35.80	13.07	54.55	14.77	46.59
WHATSAPP	LOW	JPG	9	46.59	47.73	47.73	6.25	28.98	46.59
WHATSAPP	LOW	JPG	10	49.43	50.00	25.57	50.57	50.57	49.43
WHATSAPP	LOW	PNG	1	11.36	50.00	49.43	45.45	45.45	11.36
WHATSAPP	LOW	PNG	2	50.57	51.70	54.55	43.18	52.84	50.57

WHATSAPP	LOW	PNG	3	44.89	51.70	45.45	47.16	44.89	44.89
WHATSAPP	LOW	PNG	4	32.39	48.86	25.00	13.64	19.32	32.39
WHATSAPP	LOW	PNG	5	51.14	48.86	14.77	53.98	50.57	51.14
WHATSAPP	LOW	PNG	6	47.73	51.14	46.02	52.84	50.57	47.73
WHATSAPP	LOW	PNG	7	39.77	42.05	42.61	22.16	2.27	39.77
WHATSAPP	LOW	PNG	8	49.43	35.80	46.59	8.52	9.66	49.43
WHATSAPP	LOW	PNG	9	46.02	45.45	48.30	47.73	47.16	46.02
WHATSAPP	LOW	PNG	10	48.30	51.70	46.59	47.73	50.57	48.30
WHATSAPP	MID	JPG	1	14.77	47.16	49.43	43.75	46.02	14.77
WHATSAPP	MID	JPG	2	51.70	47.73	42.61	11.93	53.98	51.70
WHATSAPP	MID	JPG	3	55.68	13.64	48.30	46.59	49.43	55.68
WHATSAPP	MID	JPG	4	26.70	35.80	21.02	50.57	23.86	26.70
WHATSAPP	MID	JPG	5	52.84	46.59	48.30	46.59	45.45	52.84
WHATSAPP	MID	JPG	6	50.57	50.57	47.73	44.32	40.34	50.57
WHATSAPP	MID	JPG	7	43.18	47.16	36.36	50.57	6.82	43.18
WHATSAPP	MID	JPG	8	53.41	46.02	48.86	22.16	23.86	53.41
WHATSAPP	MID	JPG	9	53.41	52.84	51.14	43.18	54.55	53.41
WHATSAPP	MID	JPG	10	43.75	54.55	51.70	44.32	53.41	43.75
WHATSAPP	MID	PNG	1	14.77	50.00	50.00	44.89	48.86	14.77
WHATSAPP	MID	PNG	2	44.89	51.70	53.41	50.00	56.82	44.89
WHATSAPP	MID	PNG	3	25.00	46.02	52.84	47.16	17.05	25.00
WHATSAPP	MID	PNG	4	25.00	5.11	19.89	22.73	25.57	25.00
WHATSAPP	MID	PNG	5	47.16	51.14	50.57	53.98	56.82	47.16
WHATSAPP	MID	PNG	6	49.43	46.59	49.43	47.73	48.86	49.43
WHATSAPP	MID	PNG	7	4.55	6.25	5.11	50.00	13.64	4.55
WHATSAPP	MID	PNG	8	42.05	16.48	44.89	47.16	49.43	42.05
WHATSAPP	MID	PNG	9	50.00	51.70	51.14	49.43	46.59	50.00
WHATSAPP	MID	PNG	10	47.16	48.86	53.41	46.59	49.43	47.16

WHATSAPP	HIGH	JPG	1	35.23	49.43	50.00	44.89	47.73	35.23
WHATSAPP	HIGH	JPG	2	47.73	50.57	51.14	46.02	46.02	47.73
WHATSAPP	HIGH	JPG	3	46.59	50.00	40.91	47.73	45.45	46.59
WHATSAPP	HIGH	JPG	4	28.98	46.59	21.02	47.73	36.36	28.98
WHATSAPP	HIGH	JPG	5	47.73	44.89	51.70	47.16	51.14	47.73
WHATSAPP	HIGH	JPG	6	52.84	44.89	52.84	51.14	48.30	52.84
WHATSAPP	HIGH	JPG	7	46.02	9.09	5.11	23.86	13.07	46.02
WHATSAPP	HIGH	JPG	8	46.59	52.27	40.91	53.98	49.43	46.59
WHATSAPP	HIGH	JPG	9	47.16	52.27	43.75	49.43	7.95	47.16
WHATSAPP	HIGH	JPG	10	51.14	42.05	53.98	51.70	48.30	51.14
WHATSAPP	HIGH	PNG	1	41.48	50.00	50.00	48.30	48.86	41.48
WHATSAPP	HIGH	PNG	2	41.48	46.59	32.39	50.57	39.77	41.48
WHATSAPP	HIGH	PNG	3	15.91	44.89	42.05	48.86	48.30	15.91
WHATSAPP	HIGH	PNG	4	44.89	5.68	20.45	25.57	14.20	44.89
WHATSAPP	HIGH	PNG	5	49.43	50.00	7.95	46.02	47.73	49.43
WHATSAPP	HIGH	PNG	6	51.70	52.27	47.73	50.57	51.70	51.70
WHATSAPP	HIGH	PNG	7	48.86	48.86	7.39	14.20	12.50	48.86
WHATSAPP	HIGH	PNG	8	51.14	49.43	44.89	3.98	51.70	51.14
WHATSAPP	HIGH	PNG	9	54.55	45.45	44.89	2.27	51.14	54.55
WHATSAPP	HIGH	PNG	10	52.27	50.57	49.43	56.25	50.57	52.27

Table A 2. Hamming Encode Results

Redundancy Based Method Results

Platform	Resolu.	Format	Img. No	No of Redundancy bits attached						
				1	2	3	4	5	6	7
FACEBOOK	LOW	JPG	1	58.52	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	LOW	JPG	2	52.27	0.00	100.00	100.00	99.43	100.00	100.00
FACEBOOK	LOW	JPG	3	55.11	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	LOW	JPG	4	56.25	0.00	100.00	7.39	97.16	100.00	100.00
FACEBOOK	LOW	JPG	5	56.82	0.00	72.16	11.93	99.43	100.00	67.61
FACEBOOK	LOW	JPG	6	7.95	6.25	89.20	2.84	100.00	100.00	100.00
FACEBOOK	LOW	JPG	7	53.41	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	LOW	JPG	8	56.25	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	LOW	JPG	9	56.82	0.00	100.00	51.70	100.00	100.00	100.00
FACEBOOK	LOW	JPG	10	52.27	0.00	100.00	0.00	98.86	100.00	99.43
FACEBOOK	LOW	PNG	1	55.11	2.27	100.00	100.00	100.00	100.00	100.00
FACEBOOK	LOW	PNG	2	52.84	0.00	100.00	0.00	100.00	100.00	100.00
FACEBOOK	LOW	PNG	3	55.68	0.00	100.00	2.84	99.43	100.00	99.43
FACEBOOK	LOW	PNG	4	56.25	0.00	100.00	2.84	96.02	100.00	100.00
FACEBOOK	LOW	PNG	5	45.45	0.00	98.86	16.48	99.43	11.93	99.43
FACEBOOK	LOW	PNG	6	8.52	6.25	89.20	2.84	100.00	100.00	100.00
FACEBOOK	LOW	PNG	7	51.70	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	LOW	PNG	8	59.09	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	LOW	PNG	9	61.36	2.27	100.00	100.00	100.00	100.00	100.00
FACEBOOK	LOW	PNG	10	50.57	1.14	99.43	46.59	100.00	100.00	18.18
FACEBOOK	MID	JPG	1	57.39	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	MID	JPG	2	51.70	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	MID	JPG	3	54.55	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	MID	JPG	4	56.25	0.00	100.00	7.39	96.59	100.00	100.00
FACEBOOK	MID	JPG	5	58.52	0.00	99.43	18.18	98.30	16.48	99.43
FACEBOOK	MID	JPG	6	7.95	6.25	100.00	2.84	99.43	100.00	100.00

FACEBOOK	MID	JPG	7	52.27	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	MID	JPG	8	59.66	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	MID	JPG	9	57.95	0.00	100.00	51.70	99.43	100.00	100.00
FACEBOOK	MID	JPG	10	51.70	0.00	100.00	18.18	98.30	100.00	18.18
FACEBOOK	MID	PNG	1	51.70	2.27	100.00	1.14	100.00	100.00	100.00
FACEBOOK	MID	PNG	2	50.57	0.00	100.00	0.00	100.00	100.00	100.00
FACEBOOK	MID	PNG	3	55.11	0.00	100.00	2.84	99.43	100.00	100.00
FACEBOOK	MID	PNG	4	56.82	0.00	100.00	2.84	97.16	100.00	100.00
FACEBOOK	MID	PNG	5	55.11	2.27	98.30	22.73	70.45	16.48	99.43
FACEBOOK	MID	PNG	6	7.95	6.25	100.00	2.84	99.43	100.00	100.00
FACEBOOK	MID	PNG	7	51.70	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	MID	PNG	8	57.95	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	MID	PNG	9	56.25	0.00	100.00	84.66	100.00	100.00	100.00
FACEBOOK	MID	PNG	10	50.57	0.00	100.00	18.18	99.43	100.00	99.43
FACEBOOK	HIGH	JPG	1	57.39	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	HIGH	JPG	2	55.11	0.00	100.00	7.39	100.00	100.00	100.00
FACEBOOK	HIGH	JPG	3	55.11	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	HIGH	JPG	4	57.39	0.00	100.00	7.39	96.59	100.00	100.00
FACEBOOK	HIGH	JPG	5	59.66	0.00	99.43	37.50	71.02	24.43	71.02
FACEBOOK	HIGH	JPG	6	7.95	0.00	100.00	24.43	100.00	100.00	99.43
FACEBOOK	HIGH	JPG	7	53.98	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	HIGH	JPG	8	59.09	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	HIGH	JPG	9	60.80	0.00	100.00	29.55	100.00	100.00	100.00
FACEBOOK	HIGH	JPG	10	53.98	0.00	100.00	1.14	99.43	100.00	100.00
FACEBOOK	HIGH	PNG	1	51.14	2.27	100.00	93.75	100.00	100.00	100.00
FACEBOOK	HIGH	PNG	2	53.41	0.00	100.00	2.84	100.00	100.00	100.00
FACEBOOK	HIGH	PNG	3	55.11	0.00	100.00	2.84	98.30	100.00	100.00
FACEBOOK	HIGH	PNG	4	55.68	0.00	100.00	2.84	86.36	100.00	100.00
FACEBOOK	HIGH	PNG	5	58.52	2.27	99.43	34.66	74.43	22.73	72.16
FACEBOOK	HIGH	PNG	6	7.95	6.25	100.00	2.84	99.43	100.00	100.00

FACEBOOK	HIGH	PNG	7	50.00	0.00	100.00	1.14	100.00	100.00	100.00
FACEBOOK	HIGH	PNG	8	59.66	0.00	100.00	100.00	100.00	100.00	100.00
FACEBOOK	HIGH	PNG	9	57.95	0.00	99.43	29.55	100.00	100.00	100.00
FACEBOOK	HIGH	PNG	10	55.68	0.00	100.00	1.14	98.86	100.00	100.00
WHATSAPP	LOW	JPG	1	58.52	0.00	100.00	1.14	100.00	100.00	100.00
WHATSAPP	LOW	JPG	2	52.27	0.00	100.00	100.00	99.43	100.00	100.00
WHATSAPP	LOW	JPG	3	55.11	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	LOW	JPG	4	56.25	0.00	100.00	7.39	97.16	100.00	100.00
WHATSAPP	LOW	JPG	5	56.82	0.00	72.16	11.93	99.43	100.00	67.61
WHATSAPP	LOW	JPG	6	7.95	6.25	89.20	2.84	100.00	100.00	100.00
WHATSAPP	LOW	JPG	7	53.41	0.00	100.00	1.14	100.00	100.00	100.00
WHATSAPP	LOW	JPG	8	56.25	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	LOW	JPG	9	56.82	0.00	100.00	51.70	100.00	100.00	100.00
WHATSAPP	LOW	JPG	10	52.27	0.00	100.00	0.00	98.86	100.00	99.43
WHATSAPP	LOW	PNG	1	55.11	2.27	100.00	100.00	100.00	100.00	100.00
WHATSAPP	LOW	PNG	2	55.11	0.00	100.00	2.84	100.00	100.00	100.00
WHATSAPP	LOW	PNG	3	55.68	0.00	100.00	2.84	99.43	100.00	99.43
WHATSAPP	LOW	PNG	4	56.25	0.00	100.00	2.84	96.02	100.00	100.00
WHATSAPP	LOW	PNG	5	59.66	2.27	99.43	16.48	70.45	100.00	67.61
WHATSAPP	LOW	PNG	6	8.52	6.25	89.20	2.84	100.00	100.00	100.00
WHATSAPP	LOW	PNG	7	51.70	0.00	100.00	1.14	100.00	100.00	100.00
WHATSAPP	LOW	PNG	8	59.09	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	LOW	PNG	9	61.36	2.27	100.00	100.00	100.00	100.00	100.00
WHATSAPP	LOW	PNG	10	56.82	2.27	100.00	0.00	99.43	100.00	100.00
WHATSAPP	MID	JPG	1	57.39	0.00	100.00	1.14	100.00	100.00	100.00
WHATSAPP	MID	JPG	2	51.70	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	MID	JPG	3	54.55	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	MID	JPG	4	56.25	0.00	100.00	7.39	96.59	100.00	100.00
WHATSAPP	MID	JPG	5	58.52	0.00	99.43	18.18	98.30	16.48	99.43
WHATSAPP	MID	JPG	6	7.95	6.25	100.00	2.84	99.43	100.00	100.00

WHATSAPP	MID	JPG	7	52.27	0.00	100.00	1.14	100.00	100.00	100.00
WHATSAPP	MID	JPG	8	59.66	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	MID	JPG	9	57.95	0.00	100.00	51.70	99.43	100.00	100.00
WHATSAPP	MID	JPG	10	51.70	0.00	100.00	18.18	98.30	100.00	18.18
WHATSAPP	MID	PNG	1	51.70	2.27	100.00	1.14	100.00	100.00	100.00
WHATSAPP	MID	PNG	2	51.70	0.00	100.00	2.84	100.00	100.00	100.00
WHATSAPP	MID	PNG	3	55.11	0.00	100.00	2.84	99.43	100.00	100.00
WHATSAPP	MID	PNG	4	56.82	0.00	100.00	2.84	97.16	100.00	100.00
WHATSAPP	MID	PNG	5	58.52	2.27	100.00	61.93	98.30	100.00	98.86
WHATSAPP	MID	PNG	6	7.95	6.25	100.00	2.84	99.43	100.00	100.00
WHATSAPP	MID	PNG	7	51.70	0.00	100.00	1.14	100.00	100.00	100.00
WHATSAPP	MID	PNG	8	57.95	0.00	100.00	100.00	100.00	100.00	100.00
WHATSAPP	MID	PNG	9	56.25	0.00	100.00	84.66	100.00	100.00	100.00
WHATSAPP	MID	PNG	10	48.86	2.27	100.00	18.18	98.30	100.00	99.43
WHATSAPP	HIGH	JPG	1	43.18	2.27	55.68	0.00	52.84	0.00	48.30
WHATSAPP	HIGH	JPG	2	41.48	0.00	52.27	0.00	25.00	2.27	49.43
WHATSAPP	HIGH	JPG	3	56.25	0.00	26.14	0.00	47.16	34.09	40.34
WHATSAPP	HIGH	JPG	4	56.82	0.00	26.14	0.00	53.41	0.00	53.41
WHATSAPP	HIGH	JPG	5	39.77	2.27	38.07	0.00	30.11	0.00	25.57
WHATSAPP	HIGH	JPG	6	0.00	6.25	26.14	0.00	3.41	0.00	3.41
WHATSAPP	HIGH	JPG	7	42.61	0.00	55.68	0.00	53.98	0.00	48.86
WHATSAPP	HIGH	JPG	8	44.32	0.00	55.68	0.00	51.70	0.00	48.30
WHATSAPP	HIGH	JPG	9	4.55	0.00	43.75	0.00	44.32	0.00	48.86
WHATSAPP	HIGH	JPG	10	37.50	0.00	53.41	0.00	38.07	0.00	5.68
WHATSAPP	HIGH	PNG	1	43.18	0.00	55.68	0.00	52.84	0.00	50.00
WHATSAPP	HIGH	PNG	2	42.05	0.00	52.27	0.00	27.84	0.00	28.41
WHATSAPP	HIGH	PNG	3	48.86	0.00	26.14	0.00	42.05	3.41	38.64
WHATSAPP	HIGH	PNG	4	53.98	0.00	55.68	0.00	53.41	0.00	55.11
WHATSAPP	HIGH	PNG	5	39.77	0.00	39.77	0.00	51.14	0.00	26.70
WHATSAPP	HIGH	PNG	6	3.41	0.00	26.14	0.00	53.98	0.00	3.41

WHATSAPP	HIGH	PNG	7	43.75	2.27	55.68	0.00	53.98	0.00	48.86
WHATSAPP	HIGH	PNG	8	40.34	0.00	55.68	0.00	51.14	0.00	47.73
WHATSAPP	HIGH	PNG	9	46.02	0.00	45.45	0.00	40.91	0.00	3.41
WHATSAPP	HIGH	PNG	10	39.77	2.27	53.41	0.00	31.25	0.00	5.68

Table A 3. Redundancy-Based Results

APPENDIX B - Source Codes

LSB without Error Correction and Pre-processing - Encode Function

```
from PIL import Image # type: ignore
import numpy as np # type: ignore

def encode_message(input_img_path, message, output_png, output_jpeg,
jpeg_quality=95 , img_format = "jpg"):

    # Binary conversion of the message
    message_binary = ''.join(format(ord(char), '08b') for char in message) +
'00000000'

    # Read the image from location
    image = Image.open(input_img_path)

    # Array representation of the image
    image_array = np.array(image, dtype=np.uint8)

    # Converting array into one dimension for processing
    flat_pixels = image_array.flatten()

    # Check if the message fits into the image
    if len(message_binary) > len(flat_pixels):
        raise ValueError("Message is not support for encoding.")

    # Encode the message into the least significant bits of the image
    for i, bit in enumerate(message_binary):
        flat_pixels[i] = (flat_pixels[i] & 0xFE) | int(bit)  # Ensure LSB is
set to the message bit

    # Reshape the array back to the original image shape
    encoded_array = flat_pixels.reshape(image_array.shape)

    # Save as PNG
    encoded_image = Image.fromarray(encoded_array)
    encoded_image.save(output_png)
```

```
# Convert PNG to given format
    if(img_format == 'jpg'):
        encoded_image.save(output_jpeg, "JPEG", quality=jpeg_quality)
    elif(img_format == 'bmp'):
        encoded_image.save(output_jpeg, "BMP", quality=jpeg_quality)
    else:
        encoded_image.save(output_jpeg)

    print(f"Message encoded and saved to {output_png} and {output_jpeg}")
```

LSB without Error Correction and Pre-processing - Decode Function

```
def decode_message(image_path):

    # Read the image from location
    image = Image.open(image_path)

    # Array representation of the image
    image_array = np.array(image, dtype=np.uint8)  # Ensure the array is uint8

    # Converting array into one dimension for processing
    flat_pixels = image_array.flatten()

    # Extract the least significant bits
    binary_message = ''.join(str(pixel & 1) for pixel in flat_pixels)

    # Convert binary to string, stopping at the null terminator
    chars = [chr(int(binary_message[i:i+8], 2)) for i in range(0,
len(binary_message), 8)]
    message = ''.join(chars).split('\x00', 1)[0]  # Stop at the first null
character

    return message
```

Calculating PRR Function

```
def calculate_prr(original_payload: str, extracted_payload: str) -> float:

    # Convert strings to bytes
    original_payload_bytes = original_payload.encode('utf-8')
    extracted_payload_bytes = extracted_payload.encode('utf-8')

    # Total bits in the original payload
    total_bits = len(original_payload_bytes) * 8

    # Edge case: If original payload is empty
    if total_bits == 0:
        return 100.0 if len(extracted_payload_bytes) == 0 else 0.0

    # Compare overlapping bytes bit by bit
    matching_bits = 0
    for orig_byte, ext_byte in zip(original_payload_bytes,
extracted_payload_bytes):
        # Count matching bits in each byte
        matching_bits += 8 - bin(orig_byte ^ ext_byte).count('1')

    # Add penalty for bits in missing bytes
    unmatched_bits = (len(original_payload_bytes) -
len(extracted_payload_bytes)) * 8

    # Total matching bits
    total_matching_bits = matching_bits

    # Calculate PRR
    prr = (total_matching_bits / total_bits) * 100
    return prr
```

Hamming Encode Function

```
from PIL import Image
import numpy as np


def hamming_encode(data):

    if len(data) % 4 != 0:
        # Pad data to make it a multiple of 4
        padding_length = 4 - (len(data) % 4)
        data += '0' * padding_length

    encoded = ""
    for i in range(0, len(data), 4):
        d1, d2, d3, d4 = map(int, data[i:i+4])
        # Calculate parity bits
        p1 = d1 ^ d2 ^ d4
        p2 = d1 ^ d3 ^ d4
        p3 = d2 ^ d3 ^ d4
        # Append encoded block
        encoded += f"{p1}{p2}{d1}{p3}{d2}{d3}{d4}"
    return encoded
```

Hamming Decode Function

```
def hamming_decode(encoded):

    if len(encoded) % 7 != 0:
        # Pad the encoded message to make it a multiple of 7
        padding_length = 7 - (len(encoded) % 7)
        encoded += '0' * padding_length

    decoded = ""
    for i in range(0, len(encoded), 7):
        p1, p2, d1, p3, d2, d3, d4 = map(int, encoded[i:i+7])
        # Check for errors
        c1 = p1 ^ d1 ^ d2 ^ d4
        c2 = p2 ^ d1 ^ d3 ^ d4
        c3 = p3 ^ d2 ^ d3 ^ d4
        error_pos = c1 * 1 + c2 * 2 + c3 * 4
```

```
block = [p1, p2, d1, p3, d2, d3, d4]
        if error_pos != 0:
            # Correct the error
            block[error_pos - 1] ^= 1

        # Extract data bits
        decoded += f"{block[2]}{block[4]}{block[5]}{block[6]}"
    return decoded
```

Redundancy Based Encode Function

```
def encode_message_with_redundancy(image_path, message, output_path,
bit_position=1, redundancy=8, img_format="jpg"):

    redundancy (int): Number of times to repeat each bit for robustness.

    # Validate bit position
    if not (0 <= bit_position < 8):
        raise ValueError("bit_position must be between 0 and 7.")

    # Convert the message to binary and add a null terminator
    message_binary = ''.join(format(ord(char), '08b') for char in message) +
'00000000'

    # Repeat each bit for redundancy
    redundant_message = ''.join(bit * redundancy for bit in message_binary)

    # Load the image
    image = Image.open(image_path)
    image_array = np.array(image, dtype=np.uint8)
    flat_pixels = image_array.flatten()

    if len(redundant_message) > len(flat_pixels):
        raise ValueError("Message is too large to encode in the given image.")

    # Create a mask to clear the target bit
    mask = 255 - (1 << bit_position)

    # Encode the message
    for i, bit in enumerate(redundant_message):
        flat_pixels[i] = np.uint8((flat_pixels[i] & mask) | (int(bit) <<
bit_position))
```

```
# Reshape and save the encoded image
    encoded_array = flat_pixels.reshape(image_array.shape)
    encoded_image = Image.fromarray(encoded_array)
    if img_format == 'jpg':
        encoded_image.save(output_path, format="JPEG", quality=90)  # Save as
JPEG to test robustness
    else:
        encoded_image.save(output_path)  # Save as JPEG to test robustness
    print(f"Message encoded and saved to {output_path}")
```

Redundancy Based Decode Function

```
def decode_message_with_redundancy(image_path, bit_position=1, redundancy=8):

    from collections import Counter

    # Load the image
    image = Image.open(image_path)
    image_array = np.array(image, dtype=np.uint8)
    flat_pixels = image_array.flatten()

    # Extract the embedded bits
    extracted_bits = [(flat_pixels[i] >> bit_position) & 1 for i in
range(len(flat_pixels))]

    # Group bits by redundancy
    grouped_bits = [extracted_bits[i:i + redundancy] for i in range(0,
len(extracted_bits), redundancy)]

# Decode each group using majority voting and validate consistency
    decoded_bits = []
    for group in grouped_bits:
        if len(group) == redundancy:
            most_common_bit, count = Counter(group).most_common(1)[0]
            # Include the bit only if it appears in the majority
            if count > redundancy // 2:
                decoded_bits.append(most_common_bit)
            else:
                # Inconsistent group - discard
                break
        else:
            # Incomplete group - stop decoding
            break
```

```
# Convert bits to characters
    decoded_message = ''.join(chr(int(''.join(map(str, decoded_bits[i:i +
8])), 2)) for i in range(0, len(decoded_bits), 8))

    # Stop at the null terminator
    decoded_message = decoded_message.split('\x00', 1)[0]
    #print(f"Decoded Message: {decoded_message}")
    return decoded_message
```

This repository contains the source code samples related to this thesis. The code includes implementations and examples discussed in the document.

Repository Name: mcs

Author: Danushka Fernando

Repository URL: https://github.com/dlf91/mcs

Access Type: Public

REFERENCES

[1]. www.tutorialspoint.com. (n.d.). A Comprehensive Guide To Understanding Image Steganography Techniques And Types. [online] Available at: https://www.tutorialspoint.com/a-comprehensive-guide-to-understanding-image-steganography-techniques-and-types [Accessed 5 Mar. 2024].

[2]. Hussain, M. (2013). A Survey of Image Steganography Techniques. [online] 54, pp.113–125. Available at: https://www.researchgate.net/publication/271863505_A_Survey_of_Image_Steganography_Techniques [Accessed 5 Mar. 2024].

[3]. Faiz, S. (2024). JPEG Files Explained – Everything You Need to Know. [online] File Format Blog. Available at: https://blog.fileformat.com/image/everything-you-need-to-understand-jpeg-images [Accessed 6 Mar. 2024].

[4]. Derek Goodwin Photography. (n.d.). Guide to Digital Image Terminology: file size, resolution, compression, image format, pixel dimensions. [online] Available at: https://derekgoodwinphotography.com/blog/understanding-digital-image-sizes-and-terminology [Accessed 6 Mar. 2024].

[5]. BenMauss (2021). Pixels, Arrays, and Images - Level Up Coding. [online] Medium. Available at: https://levelup.gitconnected.com/pixels-arrays-and-images-ef3f03638fe7 [Accessed 20 Oct. 2024].

[6]. Hamid, N., Yahya, A., Ahmad & Osamah, R. and Al-Qershi, M. (2012). Image Steganography Techniques: An Overview. International Journal of Computer Science and Security (IJCSS), [online] (6), p.168. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fe7b3da2b67c74ac2c6ec99f25507b00c5de684f [Accessed 25 Oct. 2024].

[7]. Singh, J., Kaur, G., Bfcet, Deon, Bathinda, P., Manveer, K. and Garcha (n.d.). Review of Spatial and Frequency Domain Steganographic Approaches. [online] Available at:

https://www.ijert.org/research/review-of-spatial-and-frequency-domain-steganographic-approaches-IJERTV4IS061082.pdf [Accessed 17 Nov. 2024].

[8]. RenanTKN (2020). LSB Steganography — Hiding a message in the pixels of an image. [online] Medium. Available at: https://medium.com/@renantkn/lsb-steganography-hiding-a-message-in-the-pixels-of-an-image-4722a8567046 [Accessed 28 Oct. 2024].

[9]. El-Alfy, E.-S. and Al-Sadi, A. (n.d.). Pixel-Value Differencing Steganography: Attacks and Improvements. [online] Available at: https://repository.root-me.org/St%C3%A9ganographie/EN%20-%20Pixel-Value%20Differencing%20Steganography%20-%20El-Alfy%20-%20Al-Sadi.pdf [Accessed 27 Oct. 2024].

[10]. Kaur, H. and Rani, J. (2016). A Survey on different techniques of steganography. MATEC Web of Conferences, [online] 57, p.02003. doi:https://doi.org/10.1051/matecconf/20165702003 [Accessed 27 Oct. 2024].

[11]. Powar, S., Dinde, H. and Patil, R. (n.d.). A Study and Literature Review on Various Image Steganography Techniques. [online] International Research Journal of Engineering and Technology. IRJET. Available at: https://www.vivekanandcollege.ac.in/uploads/dptbcs/research%20paper/radhika%20patil_all%20research%20papers-33-36.pdf [Accessed 28 Oct. 2024].

[12]. Ning, J., Singh, I., Madhyastha, H., Krishnamurthy, S., Cao, G. and Mohapatra, P. (n.d.). Secret Message Sharing Using Online Social Media. [online] Available at: http://www.cs.ucr.edu/~krish/cns14.pdf [Accessed 28 Oct. 2024].

[13]. Hiney, J., Tejas Dakve, Krzysztof Szczypiorski and Gaj, K. (2015). Using Facebook for Image Steganography. [online] Available at: https://www.researchgate.net/publication/277959373_Using_Facebook_for_Image_Steganography [Accessed 28 Oct. 2024].