Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.050090

ARTICLE

High-Secured Image LSB Steganography Using AVL-Tree with Random RGB Channel Substitution

Murad Njoum[1,2,*], Rossilawati Sulaiman[1], Zarina Shukur[1] and Faizan Qamar[1]

[1]Center of Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor, 43600, Malaysia

[2]Computer Science and Cyber Security, Faculty of Engineering and Information Technology, BirZeit University, BirZeit, Ramallah, P.O. Box 14, Palestine

*Corresponding Author: Murad Njoum. Email: mnjoum@staff.birzeit.edu

Received: 27 January 2024 Accepted: 09 August 2024 Published: 15 October 2024

ABSTRACT

Random pixel selection is one of the image steganography methods that has achieved significant success in enhancing the robustness of hidden data. This property makes it difficult for steganalysts' powerful data extraction tools to detect the hidden data and ensures high-quality stego image generation. However, using a seed key to generate non-repeated sequential numbers takes a long time because it requires specific mathematical equations. In addition, these numbers may cluster in certain ranges. The hidden data in these clustered pixels will reduce the image quality, which steganalysis tools can detect. Therefore, this paper proposes a data structure that safeguards the steganographic model data and maintains the quality of the stego image. This paper employs the Adelson-Velsky and Landis (AVL) tree data structure algorithm to implement the randomization pixel selection technique for data concealment. The AVL tree algorithm provides several advantages for image steganography. Firstly, it ensures balanced tree structures, which leads to efficient data retrieval and insertion operations. Secondly, the self-balancing nature of AVL trees minimizes clustering by maintaining an even distribution of pixels, thereby preserving the stego image quality. The data structure employs the pixel indicator technique for Red, Green, and Blue (RGB) channel extraction. The green channel serves as the foundation for building a balanced binary tree. First, the sender identifies the colored cover image and secret data. The sender will use the two least significant bits (2-LSB) of RGB channels to conceal the data's size and associated information. The next step is to create a balanced binary tree based on the green channel. Utilizing the channel pixel indicator on the LSB of the green channel, we can conceal bits in the 2-LSB of the red or blue channel. The first four levels of the data structure tree will mask the data size, while subsequent levels will conceal the remaining digits of secret data. After embedding the bits in the binary tree level by level, the model restores the AVL tree to create the stego image. Ultimately, the receiver receives this stego image through the public channel, enabling secret data recovery without stego or crypto keys. This method ensures that the stego image appears unsuspicious to potential attackers. Without an extraction algorithm, a third party cannot extract the original secret information from an intercepted stego image. Experimental results showed high levels of imperceptibility and security.

KEYWORDS

Image steganography; pixel random selection (PRS); AVL tree; peak signal-to-noise ratio (PSNR); imperceptibility; capacity

1 Introduction

As communication technology advances, public networks have seen an increase in digital media exchange, including images, movies, and audio. Digital media is susceptible to attacks such as eavesdropping, modification, privacy compromise, and theft of confidential information [1]. As data carriers, these images raise significant concerns about their security and confidentiality. Secure information is classified into cryptography, watermarking, and steganography. Cryptographers mainly focus on preventing intruders from breaking the encryption algorithm, which involves transforming plain text into something meaningless. Digital watermarking has been crucial in protecting copyright and intellectual property by embedding a unique watermark within digital content. Maintaining the invisibility of the watermark is essential for its effectiveness. Steganography protects the message from unauthorized access [2]. Cryptography provides one layer of security and confidentiality, while steganography adds an additional layer [3]. The steganographic message remains hidden from view. Even if code breakers suspect steganography in a transferred object, they cannot confirm it. Extracting the hidden message requires knowledge about the message, the file path, or the object's appearance. Without knowing the method used for hiding data within an object, code breakers will fail to pursue the purpose of the secret information. Together, the two techniques can complement each other [4].

Image steganography has two categories: spatial domain and frequency domain. In the spatial domain approach, the secret data to be embedded directly affects the pixel intensity values. Meanwhile, an appropriate transform is applied to the image in the frequency domain, and the resulting coefficients are altered to reflect the secret data [5,6]. Many techniques have been published to embed secret message bits in both spatial and frequency domain categories. The most familiar technique in the spatial domain is the least significant bit (LSB). Other methods such as pixel indicator techniques (PIT) [7], pixel locator sequence (PLS) techniques [8], pseudo-random number generator (PRNG) [9], and Rand-Stego pattern techniques are also used [10]. The main objective of steganalysis is to detect the transmission of steganographic communication between two relevant parties [11]. We can evaluate the performance of data hiding techniques using the peak signal-to-noise ratio (PSNR), mean square error (MSE), and Normalized Correlation Coefficient (NCC). We can also evaluate the data-hiding techniques with other parameters like hiding capacity and security [12,13].

According to the present research, hidden capacity is directly proportional to visual distortion and security. Therefore, there is scope for identifying a better solution. Current methods recommend using a PRNG to choose data-hiding locations. These methods require seeds for the PRNG to generate random sequence numbers to conceal bits of the secret messages. The main disadvantage of these methods is that they are expensive to generate non-repeating numbers, and the distribution of random numbers may not be typical. This issue aids in the detection of secret messages in digital media. A second disadvantage is that these methods need to share the seed key with the recipient. This research proposes a scheme to embed information bits in the LSB of non-consecutive pixels using a pixel indicator and an Adelson-Velsky and Landis (AVL) tree. This paper proposes a new random technique using an AVL tree data structure, pixel indicator, and 24-bit color RGB images as the cover medium instead of concealing the secret bits of confidential messages directly into the LSB of a sequence of pixels of data channels. The LSBs of random red and blue channels will conceal the secret message bits. The LSB of the green channel is used as an indicator because of human eye sensitivity [5].

The rest of the paper is organized as follows: Background studies are shown in Section 2. The literature review is presented in Section 3. The proposed method (AVL tree steganography, embedding, and extraction) is discussed in Section 4. Performance analysis and discussion are presented in Section 5. Finally, Section 6 presents the conclusion and future work.

2 Background Studies

In recent times, there has been a growing emphasis on cryptography and steganography. The need for secure communication and concerns about potential misuse, such as unauthorized distribution of digital information, have underscored the importance of integrating these techniques. As a result, it has become essential to have a comprehensive and well-defined understanding of both fields to implement them effectively. The primary focus of this paper is to address the security aspects associated with transferring text within images and to provide an overview of steganography techniques.

2.1 Cryptography

Cryptography is the art and science of sending messages so that only the intended recipient can read them [4]. In other words, cryptography aims to ensure that a message sent from a sender to a receiver remains confidential and secure. Even if someone intercepts the message, they should not be able to read it. Cryptography is a technique for protecting data using encryption formulas and functionalities so that the data held in a computer can only be interpreted by individuals who can decode it. Historically, cryptography was used mainly in the military and diplomacy until a few decades ago when the realization of digital data saw widespread application in many non-military fields such as e-commerce, e-banking, e-governance, telemedicine, e-shopping, and e-mailing.

There are three types of cryptographic schemes for securing data: public-key cryptography, private-key cryptography, and hash functions. The length and type of the keys used depend on the type of encryption algorithm.

2.2 Symmetric Key Cryptography

Symmetric key cryptography is a traditional cryptography implemented as a private key algorithm. They can be divided into conventional or fast algorithms and block ciphers. Fast algorithms like Data Encryption Standard (DES) focus on reducing the encryption and decryption time with a limited key length. Their nature does not meet the requirements of the current epoch; therefore, they are not secure. Modern-designed block ciphers like Advance Encryption Standard (AES) apply the substitution–permutation network structure. They are built using different transformations: substitution (S-boxes) and permutation (P-boxes) layers. They produce effective confusion, dispersion, and diffusion properties. The significant amounts of transformation rounds and complex S-boxes improve the security level of those ciphers. Generally, the main disadvantage of conventional symmetric key cryptography is the need for a secure key distribution channel to transmit key material over the network safely [14].

2.3 Asymmetric Key Cryptography

Asymmetric key cryptography is a primary component in public key cryptographic algorithms like Rivest–Shamir–Adleman (RSA), ElGamal, and Elliptic Curve. The main advantages of asymmetric key cryptography are facilitating the management of keys and the secure exchange of keys. The main problem, however, is that it is much slower than symmetric key cryptography. Therefore, the common practice is to use asymmetric key cryptography only to exchange a session key between the two sides and then use the session key with symmetric key cryptography to secure the rest of the communication between the two sides. During the asymmetric key cryptography exchange phase, the two sides must establish the authenticity of the other side and ensure that no man-in-the-middle attacker impersonates the other side to downgrade the communication security level [15].

2.4 Steganography: Concealing Information

Steganography is the art of hiding a secret data message within an ordinary, non-secret file or other medium to avoid detection and later extraction at its destination [16]. The security of concealing data in images has attracted researchers' attention to improving security aspects [17]. Researchers have proposed many steganography techniques to conceal secret messages in multimedia files such as images [18], audio files [19], video files [20], text [21], and internet protocols [22] to protect confidential data from attackers and transfer data without suspicion. Image steganography has two categories: spatial domain and frequency domain. In the spatial domain approach, the secret data to be embedded directly affects the pixel intensity values. Meanwhile, an appropriate transform is applied to the image in the frequency domain, and the resulting coefficients are altered to reflect the secret data [5,6].

3 Literature Review

Steganography focuses on capacity, imperceptibility, and security to achieve the invisibility of secret data. Imperceptibility is achieved by embedding data without suspicion, using techniques like localizing regions that can withstand pixel changes. Security involves making hidden information impossible to recover, impenetrable, or unpredictable. Proposed methods like LSB, randomization, pixel indication, and hybrid techniques have improved steganographic security.

3.1 Least Significant Bit (LSB)

LSB steganography is a basic technique that hides information within images, videos, or audio files by replacing the least significant bit of each pixel or sample with a bit from the secret message. However, it may not be secure against advanced detection methods or attacks. Information hiding in an image aims to alter less critical information in the carrier image. The LSB is the most used in steganography. It is among the most straightforward and widely used spatial image steganographic techniques [23–25]. The idea behind this technique is that the smallest components of an image only provide weak information, and human eyes cannot notice even minute changes in those components. The fundamental idea of the LSB embedding technique is shown in Fig. 1.

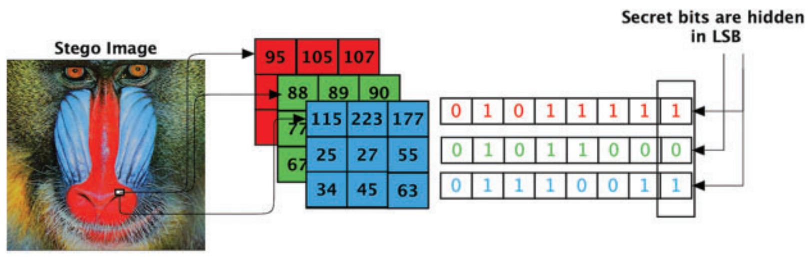

Figure 1: Basic LSB technique

In most proposed methods, the secret message bits are randomly or sequentially inserted in the LSB of the pixel positions. The advantages of LSB steganography are: (i) it is the simplest steganographic method to embed a message into digital media such as an image or sound; (ii) it does not require transformation of the cover medium; (iii) the LSB replacement ensures that the stego medium is indistinguishable from the original cover medium; (iv) it is the fastest technique for embedding speed; and (v) it is also possible to partially decode or detect the hidden message without completely decoding the stego object [26,27]. Conversely, the major drawbacks of LSB steganography are: (i) the stego medium is vulnerable to relatively simple statistical analysis; (ii) the hidden data is easily destroyed by simple manipulation or processing of the stego object; (iii) it is easy to insert data by comparing the magnitude of the pixel value or the absolute LSB value with the secret data; (iv) the steganographic algorithm is weak against the known-plaintext attack; (v) the depth or positional representation of the secret data is not considered; and (vi) it frequently provides unsatisfactory visual quality of the stego image [28,29].

3.2 Pixel Indicator Techniques

The PIT is a method to conceal secret messages by adjusting each pixel's least significant bit, allowing subtle color changes undetectable to the human eye. Various factors, such as this pixel's location within the image or its color value, are considered when selecting it. The digital image steganographic method uses odd/even pixel allocation to hide encrypted messages in even pixels, ensuring the change is made on the first or second LSB without affecting the base of odd or even in the stego image [30]. The proposed method in [31] shows an image coding technique that conceals information along a chosen pixel and, on its subsequent value (pixel + 1). Two bits of the message can be concealed on each pixel based on a combination of these values. The authors of [25] use the indicator channel to select the data channel for hiding information. The technique depends on the length and parity bits of the secret message. If the length of the message is even, then data channel 1 is red, data channel 2 is green, and data channel 3 is blue; the three most significant bits (MSB) are considered indicators. Every bit of these three represents a channel (R, G, or B) that contains the hidden two bits of the secret message. For example, if three-MSB is 101, the red and blue channels will hide two secret bits; these will be hidden in the LSB of these channels. In [30], the method conceals secret information in red and blue channels using the LSB of the green channel as an indicator. The method uses 2, 2–4, or 4 LSBs of red and blue channels for up to six bits.

In [31], the proposed technique divides a cover image into four blocks of storage area and decides if each desired pixel can be saved based on the red channel's three MSBs. If the three MSBs have a value of 111, all red, green, and blue channels are candidates for data storage. No data can be stored in bits with a value of 000. If the bits have a value of 011, both green and blue channels are candidates. Four zeros in the LSBs determine the number of hidden bits. Pixel indicator techniques in steganography use one or more pixels on a cover image to signal whether the image contains hidden data. If these pixels are modified, they can lose their significance, and the steganalysis can fail. The significant advantage of pixel indicator techniques from the steganographic point of view is that, unless the indicators are disabled, no other changes are made to the image. This is very useful when the steganographic capacity is low or when the steganographic algorithm works in an adaptive mode, concealing data bit by bit and needing to access the stego image many times for different data to be embedded. The main disadvantage of pixel-indicator techniques is that the secret message is not directly embedded in the cover image. Data embedding, extraction, and possibly the adaptive hiding algorithm are more complex than desired [32–35].

3.3 Random Techniques

Randomization techniques enhance security in stego images by scattering secret message bits in a pattern known only to the method's owner, making this distribution undetectable by attackers. The least significant bit is used in a cover image to embed information in a random bit of a pixel using a PRNG. PRNG employs a 3-3-2 approach to hide a byte in a 24-bit color image, selecting random pixels and bit positions within the R, G, and B values. The authors in [36] proposed an efficient and adaptive data-hiding scheme based on a secure reference matrix. LSB steganography using a pixel locator sequence (PLS) with AES (Advance Encryption Standard) is suggested in [8], which uses a randomly generated secure reference matrix stored as a PLS file. PLS increases the security of LSB steganography by randomly distributing the secret data hidden in the image, making it difficult for unauthorized users to detect the hidden data. The PLS file is encrypted using AES. The random-bit selection algorithm in [37] enables users to conceal information within a cover image using randomly generated integers and user-specified parameters for the number of bits to be replaced in each pixel. The methods in [38–40] employ a chaotic pseudorandom generator to randomly determine the position and hierarchy of image pixels for embedding information while encrypting the secret message. In [10], a random varied pattern key is generated automatically or manually saved in a file and shared with the recipient. Both methods in [30,41] use the Henon map function and stego keys to generate a random sequence of pixels to hide the bits of the secret messages. Random selections in embedding locations or values generate an equal probability of steganographic messages, enhancing security and resistance to steganalysis. The distribution of these techniques is independent of the cover image position. Advanced steganographic programs extract secret messages from known steganographic implementations. Multi-dimensional random algorithms, like those used in deterministic techniques, prevent the detection of structures, ensuring system resistance to attack. The main advantages of this technique are: 1) Increased safety. By introducing an element of unpredictability and randomness, random procedures in steganography make it more difficult for unauthorized users to detect hidden information. 2) Resistance to detection: Randomly dispersing the hidden data throughout the cover file makes it less evident and difficult to find. 3) Robustness: Spreading the secret information throughout the carrier file in several places increases its resistance to corruption or data loss. The main disadvantages of the random technique are: 1) Complexity: Random techniques typically require more complex algorithms and procedures than straightforward steganography approaches. 2) Reduced hiding capacity: Randomly distributing the hidden information may result in fewer bits accessible in each location, leading to a lower overall hiding capacity. 3) Increased computational overhead: Random approaches frequently require more computational resources to produce and control the randomization process [32,33].

3.4 Hide Bits in Minimum and Maximum Color Intensity

In steganography, one approach is to hide bits of information by minimizing the color intensity of pixels in an image. By decreasing the color intensity of specific pixels, hidden data can be embedded without significantly altering the image's appearance. This method makes it difficult for third parties to detect the presence of hidden information, thus ensuring the security and confidentiality of the communication. Using the minimum color intensity of pixels, the hidden data can blend seamlessly with the cover media, making it virtually impossible for unauthorized users to detect or extract the hidden information [42–44].

The concept of hiding bits at maximum color intensity refers to concealing information within an image by manipulating the RGB values of pixels. This technique takes advantage of the imperceptibility of human vision to slight variations in color within a specific range. Secret information can be embedded without significantly altering the image's appearance by adjusting each pixel's most significant and second significant bits. Utilizing the maximum color intensity of each pixel allows more data to be hidden within the image while maintaining high image quality and minimizing visual distortions.

This information-hiding method provides transparency, security, and robustness in data transmission within images. The primary objective of hiding bits in maximum color intensity is to ensure the security and integrity of information during transmission through images [45,46].

Ultimately, the choice depends on the trade-off between hiding capacity and imperceptibility. If the priority is to maximize the amount of information that can be hidden while accepting a slightly higher risk of detection, using the maximum color intensity may be preferred. On the other hand, if the goal is to minimize the chance of detection at the expense of hiding capacity, the minimum color intensity approach could be more suitable. The minimum color intensity technique also provides several advantages: 1) It can hide many significant or more visually noticeable bits. 2) The degradation from hiding these bits is more evenly distributed. This helps prevent localized noticeable degradation, as the human visual system is more sensitive to alterations in specific image regions. 3) It is more protective of the secret, as it must first modify the encoded image to access and reveal the original message. The maximum color intensity technique offers several advantages: 1) The message colors never lose intensity. 2) It can conceal less significant bits outside the range of easily noticeable degradation. 3) It works with all types of images. 4) It's easier and more efficient to implement.

3.5 Summary and Comparison

The main features, pros, and cons of the literature methods discussed in this paper can be summarized in the following table (Table 1).

Table 1: Features, prose and, cos of literature techniques in steganography

Technique	Features	Pros	Cons
LSB	• LSB steganography is user-friendly and straightforward, making it accessible to those with limited technical expertise.	• Simplicity • Large hiding capacity • Imperceptibility. • They were used for grayscale images and RGB images.	• Vulnerability to detection. • Fragile due to image manipulations or compression. • Potential for accidental disclosure. • Limited security:

(Continued)

Table 1 (continued)

Technique	Features	Pros	Cons
PIT	• The PIT uses one channel to indicate the hidden data in the other two channels.	• Increased capacity for data hiding. • Enhanced security.	• The complexity of method. • Increased detection risk. • Fragility and potential data loss. • Limited availability of tools and resources. • Reduced image quality.
Random techniques	• Random techniques are methods for selecting embeddings or covering changes at specific locations.	• Enhanced security. • Resistance to the statistical analysis. • Increased robustness: by distributing the hidden information over file.	• Increased complexity. • Reduced hiding capacity. • It increased computational overhead.
Hide Bits in Min/ MaxColor intensity		• Increased security. • Improved imperceptibility. • Resistance to simple detection techniques.	• They limited hiding capacity. • Susceptibility to more advanced detection techniques. • Vulnerability to image modifications.

4 Proposed Method

This section emphasizes the modifications to conventional LSB steganography for high security. Our basic idea is to use an AVL tree for embedding the information bits and to use a random sequence of pixels for embedding the data bits to improve security while maintaining the capacity of the cover image. Firstly, the cover image must have more than enough pixels to conceal portions of the secret message, as it must be compatible with the text message length. Since each pixel will contain two bits of a secret message, the cover image must have more pixels than the message's length divided by two. Then, construct an AVL tree to embed the secret bits to minimize the distortion between the transformed and original cover images. The measure used to calculate the distortion is MSE. We modify the typical method of embedding the bits in random order. Instead of embedding two bits in sequential order at a time, this paper explores the possibilities of embedding two bits in non-sequential order, selecting the one that results in the slightest increase in MSE. The specific details of the method of concealing the secret bits are far too extensive to mention here, so we offer an outline in our paper.

4.1 Overview of the Proposed Method

This proposed steganography method increases the data payload and secures the secret data. The AVL tree is used to determine the exact location for the replacement, while the random RGB channel substitution is used to secure the planted secret data. Overall, this method eases the data detection process and can provide a high payload with minimum distortion to the cover image [47].

This paper uses an AVL tree, queue data structures, and pixel indicator channels of 24-bit RGB images in the LSB-based image steganography technique. The proposed algorithm embeds secret bits randomly in the LSBs of nonconsecutive pixels using the AVL tree without sharing a stego key or seed number used to generate a random number. The proposed randomization technique will depend on the selected pixels for hiding the bits that will be constructed based on the existing pixels in each level of the AVL tree. Furthermore, the green channel is selected to maintain the AVL tree construction without any change between the client and server. The construction of the AVL tree will be based on the green channel because the green color is more sensitive to human eyes than red and blue. Changing bits in the green channel will be detected by human eyes [48]. In addition, the green channel's LSB was selected to indicate the secret message bits' position in the other channels.

4.2 Implementation Details

The first step is to read the cover image pixels' RGB channels as decimal numbers. Then, a node is created to store information about the pixel containing the red, green, and blue decimal values. Additionally, the node will store the order of cover image pixels read from left to right and from top to bottom. The decimal value of the green channel will be used to construct the AVL tree because this value will remain unchanged and will not contain any hidden bits. The order of pixels will indicate the current position of the pixels after the extraction process.

The second step is constructing the AVL tree based on the green channel. For example, Fig. 2 shows the steps for inserting a secret bit into the green channel integer values of pixels from the first pixel to the fifth pixel.

Fig. 2a shows the green channel's first inserting value (164). Fig. 2b–d show the insertion of 202 to the right side since its value is greater than the parent's (164), then the insertion of 55 to the left side since its value is less than the parent's. When inserting 102, as shown in Fig. 2e, the tree needs a single rotation to the left. The result will be shown when all table pixels are read into the AVL tree. As seen in Fig. 2f, the pixel location will change according to the rebalancing of the tree.

The formation of the AVL tree is used to classify the cover image's pixel location and balance the tree. At the same time, the sorting is based on comparing the pixel values of the green channel belonging to the cover image. The next step involves converting the secret message from ASCII codes into a sequence of bits, simplifying the concealing process within the cover image pixels. The red or blue channel will be used to save two bits. The last step of this method is hiding the message in the pixels of the cover image by combining the AVL trees and the concept of random RGB selection with bits of a secret message. The position of the pixels is essential to determining the location of pixel modification. So, the position of the pixels classified at the top will use the pixels of the cover image with a small RGB value. This is important due to the probability of certain RGB combinations in the bright pixels, as noise is smaller than the RGB combinations in color images. This process is done incrementally, starting from the tree's top node with the proposed AVL trees and level by level, and is stored in a blue or red channel based on the LSB two bits of the green channel. The following section will show the algorithm of this proposed method.

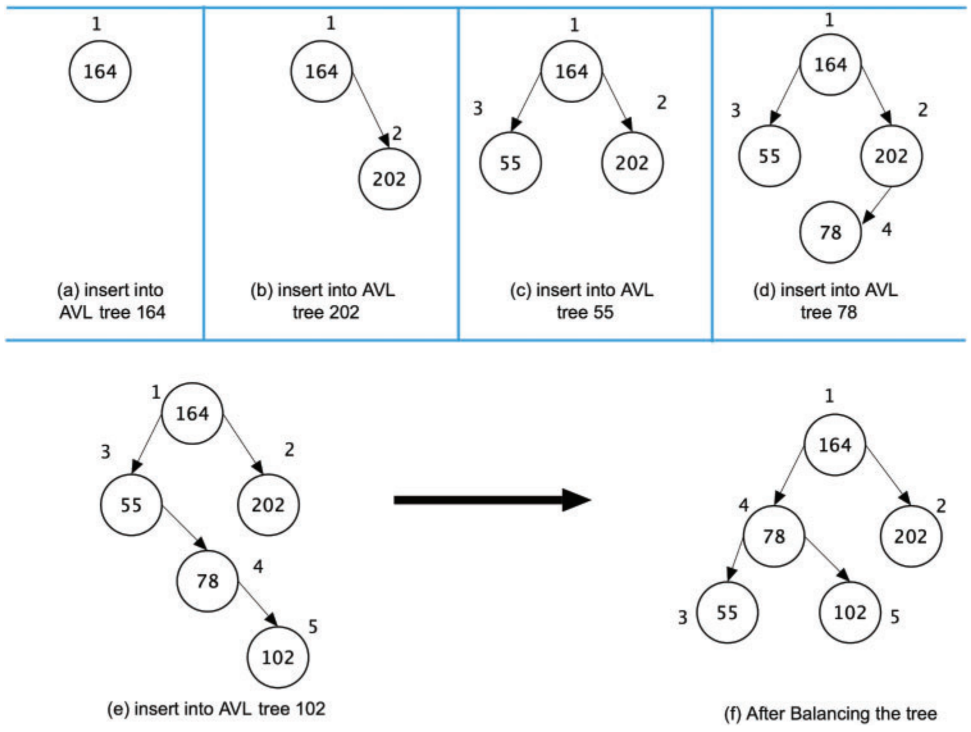
(a) insert into AVL tree 164
(b) insert into AVL tree 202
(c) insert into AVL tree 55
(d) insert into AVL tree 78
(e) insert into AVL tree 102
(f) After Balancing the tree
164
202
55
78
102

Figure 2: A sample steps for constructing an AVL tree (a) insert 164; (b) insert 202; (c) insert 55; (d) insert 78; (e) insert 102; (f) the balanced AVL tree

4.3 Embedding a Secret Message Algorithm

The following Table 2 shows the algorithm of the proposed method steps to embed a secret message into the cover image.

Table 2: Embedding a secret messaging algorithm

Algorithm: Embedding a secret messaging algorithm
Input: Cover image and text message.
Output: Stego image

```
1.    Start
2.    Read a Secret Message
      2.1.  Read the secret text message from a text file.
      2.2.  Define a "strlen" variable and store the length of the secret message inside this variable.
```

(Continued)

Table 2 (continued)

```
        2.3.  Convert the "strlen" into a binary number and store it in a new "strlenbin" variable with
              30 digits.
        2.4.  Read the secret text message and store it in a "secretmessage" variable.
        2.5.  Convert the "secretmessage" characters into binary.
        2.6.  Concatenate the string from Steps 3 and 5, then store them in a "strmesgbin" variable.
        2.7.  Define an integer variable " strlen2".
        2.8.  Set strlen2=string length of (strmesgbin)
3.      Reading the pixels of a cover image.
        3.1.  Load the cover image from the selected image directory.
        3.2.  Read RGB image pixels as integers.
        3.3.  Build an AVL tree data structure balanced based on the green channel value.
4.      Concealing the bits of the secret message:
        4.1.  Define the needed node variable (neededNode) that indicates the maximum length of text
              that can be stored in the cover image.
        4.2.  Set neededNode = strlen2 divided by 2.
        4.3.  If neededNode >image size, then throw an exception ("Message can't be hidden in this
              image")
        4.4.  Else, go to the next step.
        4.5.  Convert integer pixels of RGB from part C) into a binary system (red, green, and blue)
        4.6.  While (strmesgbin = 0), and (AVL tree).
                1. Read the LSB of the green-bit pixel channel of a node in the AVL tree
                2. Read the 2 most significant bits from part B) Step 8.
                3. If the LSB 2 bits of green pixel == "00" or "11"
                4. Then, store the 2 MSB encrypted bits in the LSB 2 bits of the red pixel
                5. Else, store these 2 bits in LSB bits of the blue pixel.
                6. update, strlen2 = strlen2-2
5.      Store the AVL-tree back to the buffered image file:
        5.1.  Read the AVL-tree pixels from part C) and store them in a new AVL-tree that is
              balanced according to the pixel number (pixel_no)
        5.2.  Read the pixels from the last step into the image file.
        5.3.  Stego image is the output.
```

The comparison with other hiding schemes shows that the proposed scheme has higher imperceptibility while maintaining high visual quality. Additionally, the proposed method shows a high level of security since steganalysis tools can't detect it. It's better than other techniques since the secret message isn't encrypted like other methods and is resistant to steganalysis tools. An example is given for this algorithm, as shown in Table 2. This example illustrates hiding the secret "hello" message inside the cover image.

The example starts with creating the AVL tree and then embedding the secret message "hello" inside the tree. This example will be illustrated in the following steps:

1. Read the secret text message from a file.
2. Set the length to 5, as "hello" has five characters.
3. Convert the length into binary with 30 digits: **01000 00000000 00000101**.
4. Convert "hello" into binary, totaling up to 40 bits (shown in Table 3).
5. The algorithm builds the AVL tree, as shown in Fig. 3. The construction of the AVL tree steps is illustrated in Fig. 2.

Table 3: "hello" ASCII code

h	e	l	l	o
01101000	01100101	01101100	01101100	01101111

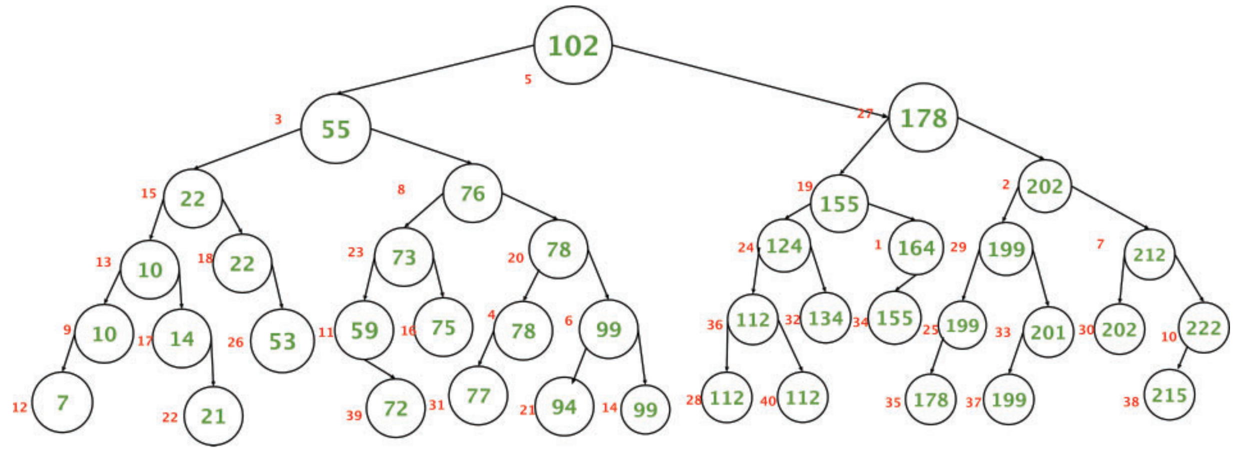

Figure 3: Part of AVL tree for image bits of Table 4

Table 4 shows a sample of pixels' cover image. These decimal values represent the decimal values for the RGB channel from pixels 1 to 40.

Table 4: Before embedding bits' message

Pixel	1	2	3	4	5	6	7	8
Red	163	204	173	157	255	212	233	7
Green	164	202	55	78	102	99	212	76
Blue	158	202	46	76	77	92	211	75
Pixel	9	10	11	12	13	14	15	16
Red	112	214	171	25	25	21	23	77
Green	10	222	59	7	10	99	22	75
Blue	15	212	47	9	7	92	21	76
Pixel	17	18	19	20	21	22	23	24
Red	13	214	171	157	208	23	72	123
Green	14	22	155	78	94	21	73	124

(Continued)

Table 4 (continued)

Blue	15	21	47	71	93	210	72	148
Pixel	25	26	27	28	29	30	31	32
Red	201	179	159	215	212	233	78	143
Green	199	53	178	112	199	202	77	134
Blue	199	48	176	79	191	201	76	148
Pixel	33	34	35	36	37	38	39	40
Red	202	173	151	253	223	231	75	220
Green	201	155	178	112	199	215	72	112
Blue	200	146	176	78	192	223	73	68

Explanation of the example:

1. Preparation:

The secret message "hello" has a length of 5 = $(101)_2$ = $(0000000000\ 0000000000\ 0000000101)_2$, and the message "hello" in binary = $(0110100001100101011011000110110001101111)_2$.

After the secret message is concatenated with the binary length, the binary result will be: (000000000000000000000000010101101000011001010110110001101100011011111).

2. Embedding process:

The secret bits representing the length of the message: **000000000000000000000000000101** will be embedded starting from the first level and up to the fourth level, which will then be followed by the secret message **0110100001100101011011000110110001101111**.

The embedding process starts by traversing the tree level by level, starting from the first level, the root node (green pixel is **102**). This pixel node contains **(*R:255, G:102, B:77, Pixel_no:5*)**. Converting the node into binary will give: $(102)_{10}$ = $(\boldsymbol{1100110})_2$, $(255)_{10}$ = $(\boldsymbol{1111111})_2$, and $(77)_{10}$ = $(\boldsymbol{1001101})_2$. Then, according to Table 4, check the LSB two bits of the green channel, pixel 5. If it is equal to ***00*** or ***11*,** then hide the two bits of the secret message in the ***red*** channel. Otherwise, the algorithm will hide the two secret bits in the ***blue*** channel.

According to the least significant two bits of the green value of pixel 5, which is (10), the two bits of the secret message from left (00) will be hidden in the blue channel of 77 value as highlighted in cyan in Table 4. After modification of the blue channel value, the pixel will be changed from 77 = $(1001101)_2$ to 76 = (1001100) since the last two significant bits of the blue channel (**01**) were replaced by (**00**).

The following levels are considered for hiding the secret message bits:

(112,10,15), (13,14,15), (174,53,48), (171,59,47), (77,175,76), (157,78,76), (212,99,92), (215,112,79), (143,134,148), (173,155,146), (201,199,199), (202,201,200), (204,202,201), (214,222,212), (25,7,9), (23,21,210), (75,72,73), (78,77,76), (208,94,93), (21,99,92) contain the secret message, while the pairs (151,178,176), (253,112,78), (231,215,223), (233,199,192), and (220,112,68) do not contain concealed bits. Note that the pairs of pixels containing no secret information are not used and, therefore, are stopped.

4.4 Extracting the Secret Message

To extract the secret message from the stego image, the same algorithm will be rebuilt using the same techniques but in reverse order.

5 Performance Analysis

This paper has experimented with the proposed techniques using various authentic color images and 24-bit BMP images of different sizes to represent the flexibility of the technique on different types of images. For experimental purposes, this paper used 24-bit images as a cover image to compare the resulting stego image with the original cover image in terms of quality and hiding capacity. For each image size, the proposed method embedded data equal to a lesser 50% and 40% of the size of the cover image to represent the quality and hiding capacity of the stego image with minimal change in the actual image. The sample images used for experimentation are shown in Fig. 4a–c, and the corresponding resultant stego images are shown in Fig. 5a–c.

Figure 4: Cover sample images (a) Tiffany; (b) Papper; (c) Baboon

MSE is the mean square error between the ith pixel of the cover image C and stego image S, and M and N are the image's dimensions.

5.1 Experimental Environment

All examples of steganography discussed in this paper are limited to bitmap files. There are two reasons for this. First, working with bitmap files allows the manipulation of every bit of data in the file, and second, this provides a more accurate measurement of the capacity of the least significant bit available for secret message storage. However, the limitation of AVL tree steganography (hereafter will be called AVLTree-Steg) to one file type is arbitrary and was done to demonstrate the capabilities of steganography. Working with any file format is a simple exercise in code development. Several software programs were developed for this document to read and display the BMP image files produced on the Macintosh platform. Steganography is a technique used to hide files in digital media, such as images, videos, and audio. It can be used in image tools to hide files in digital images like BMP and JPEG, video tools in video files like AVI and MPEG, and audio tools in sound files like WAV and MP3. Some applications can hide files in executables, but this is more memory intensive. Most steganography tools are easy to handle and require no special knowledge. However, interface complexity can lead to confusion when introducing these tools.

Figure 5: Stego result images (a) Tiffany; (b) Pepper; (c) Baboon

Most steganography tools fall into the image, video, and audio categories. This paper will focus on images because many can be exchanged between clients simultaneously. Additionally, the size of images is preferable to other media, such as audio and video. The AVLTree-Steg program was written and compiled in Java using the Eclipse 2023 environment. Eclipse 2023 was chosen for the compiler platform due to its accessibility, cost-effectiveness, and cross-computer usability. No specific assembler command or function was used. The MacBook Pro (2021) and Enhanced IDE were utilized during the initial stages of the program. The development platform was an Apple M1 Pro with 16 megabytes of RAM and a 500-gigabyte hard disk.

5.2 *Evaluation Metrics*

This section presents a review of existing evaluation metrics. Since they departed from one another, the following different evaluation metrics are categorized into three different groups.

5.2.1 PSNR, MSE, and NCC Analysis

The metrics used to evaluate the approach are MSE and $PSNR$, as shown in Eqs. (1) and (2). High values of the $PSNR$ criterion are preferred for evaluating the stego image's quality. Here's the formula to compute MSE and $PSNR$:

$$PSNR = 10\ log_{10} \frac{255^2}{MSE} \tag{1}$$

$$MSE = \frac{\sum_{i=1}^{W} \sum_{j=1}^{H} \left(X_{ij} - Y_{ij}\right)}{WXH} \tag{2}$$

The image's dimensions are denoted by the values W and H, the image coordinates by i and j, and the cover and stego images by X and Y. If the $PSNR$ values are more than 50 dB, the images are considered high quality [12].

Eq. (3) shows the embedding capacity, also known as "embedding payload", refers to the proportion of the embedded secret bits in the cover image's pixels. The abbreviation ER stands for embedding rate.

$$ER = \frac{N}{WXH} bpp \tag{3}$$

where N is the number of hidden bits, W is the width, and H is the height of the cover image.

Normal Cross Correlation (NCC): The NCC value is calculated in Eq. (4). The Normalized Correlation Coefficient (NCC) can be used to assess the robustness of extraction results. If the NCC value is near 1, the likeness to the original message image is also close [13].

$$NCC = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} I\left(X_{ij}, Y_{ij}\right) * I'\left(X_{ij}, Y_{ij}\right)}{\sqrt{\sum_{i=1}^{W}\sum_{j=1}^{H} I\left(X_{ij}, Y_{ij}\right)}\sqrt{\sum_{i=1}^{W}\sum_{j=1}^{H} \left(X_{ij}, Y_{ij}\right)}} \tag{4}$$

where W and H are the width and height of the image, $I\left(X_{ij}, Y_{ij}\right)$ represents the coordination of the cover image and, $I'\left(X_{ij}, Y_{ij}\right)$ represent the coordination of stego image.

5.2.2 Histogram Analysis

The security level of the proposed methods is evaluated by comparing the cover image histogram with the stego image. The method is considered secure if there is no significant difference between them. When the payload is small (2 KB) the histogram of the stego image of the three techniques is relatively similar to the cover image histogram, implying low distortion. This will be explained in the next section.

The distortion remained low when the payload capacity was increased (32 KB). Because one channel was left unaltered and two channels were used for the embedding procedure, those two channels bore the brunt of the histogram's distortion. On the other hand, the original cover image's histogram and the other channel's histogram are the same. This will be explained in more detail in the following section.

5.2.3 The Probability of Detection Using Statistical Attack

The stego images were subjected to statistical analytical attacks such as RS Analysis and Sample Pairs Analysis to determine the probabilities of detecting the stego images [49]. It is based on the idea that the statistical characteristics of the cover media, the unaltered file, and the stego media that contains hidden data are different. The stego image steganography replacement technique involves hiding information by modifying the least significant bit of pixel values in a digital image. This technique reduces the difference in frequency between adjacent colors, making the changes less noticeable to the human eye. However, the number of pairings remains the same, maintaining the image's visual quality. The confidentiality of encrypted hidden messages using cryptography technology can make the hidden information harder to detect through statistical analysis. Even though the chi-square detection technique can successfully locate images with sequentially embedded data, it will only work if the data are integrated consecutively. The only images that chi-square detection technology can find are those that have been continuously implanted with secret messages [50]. For the stego image to resist statistical attacks, it must achieve a probability of detection near zero (0). Steganalysis tools save time and offer new attack angles for forensic analysts. StegExpose is a real-world solution that efficiently analyzes LSB steganography images using proven attacks. Its focus is intelligently combining steganalytic methods for more accurate results [51].

5.3 Details of Dataset

We use steganography to hide the secret message (binary text) in the cover images and generate the stego images. Each cover image will conceal 1000 random-length secret messages to ensure the safety of the steganography. Since a greater payload may lead to a larger payload, the secret message length is randomly selected from {2, 4, 8, 16, and 32} kilobytes by combining more than the file. This dataset is a collection of newsgroup documents. The ten newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and clustering. This paper will use the same dataset for the steganographic system. Each newsgroup file in the bundle represents a single newsgroup. Each message in a file is the text of some newsgroup document posted to that newsgroup. This is a list of the ten newsgroups: "business," The categories include "entertainment," "food," "graphics," "historical," "medical," "politics," "space," "sport," and "technology" [52].

This paper's dataset contains 3000 RGB-BMP images, dimensions 512 × 512, for steganography, steganalysis, and similar image processing applications. The dataset includes various image captures such as "paper," "flowers," "farmers," "beach," "birds," "river," "cow," "grapes," "strawberries," and "trees." All images are color images with 512 × 512 resolutions from a regular, high-definition camera. Moreover, we randomly select an additional seven cover images, "Tiffany," "Pepper," and "Baboon," from the internet as a benchmark. These images have pixel values whose size is 512 × 512, ensuring that all the resolutions of the cover images match those of the dataset. This paper uses the 3000 image matrices mentioned above as the cover images [53].

Finally, the number of cover image-containing stego images containing the secret message with a size of 16 KB, "steganography," accounts for 10% of all steganographic images, thus leading to a total of 300 stego images generated by steganography methods.

5.4 Results and Discussion

The simulation results using both the LSB image steganography, pixel indicator technique, and the proposed method applied to the Tiffany, Pepper, and Baboon benchmarks are shown in this section. The histogram, PSNR, MSE, NCC, and file size analyses for the dataset are also presented. The salient outcomes are discussed as follows:

5.4.1 Preparation

Select a set of cover images of varying sizes and content. These images represent a natural scene. Create a cover image that conceals a secret message. These messages are text files. We will load the AVL tree method. This technique uses the least significant bit (LSB) substitution method and random and pixel indicator techniques, where the least significant bits of pixel values are modified to encode the secret data.

5.4.2 Experimental Steps

Step 1: Load a cover image into the software.

Step 2: Apply the steganography algorithm to embed a secret message within the cover image.

Step 3: Save the modified image as a stego image.

Step 4: Repeat steps one to three for multiple cover images using the same secret message.

Step 5: Randomize the order of the stego images to avoid bias.

Step 6: Prepare a control group by selecting the original cover images without any embedded data.

Step 7: Conduct statistical analysis or evaluation metrics to measure the effectiveness of the steganography technique.

5.4.3 Image Steganography Using Benchmarks from the Proposed Method Dataset

The results regarding the imperceptibility compared with similar techniques are shown in Table 5. PSNR is another popular way of measuring the distortion of the cover image caused by embedding. It is the relationship between a signal's maximum value and distortion's noise power (MSE). A higher value of PSNR indicates better-quality embedding. As can be seen, the PSNR value (imperceptibility) of the proposed AVL tree method is highly satisfactory compared with other proposed methods. It fulfills the imperceptibility requirement at this stage since the MSE of this method converges to zero for all compared proposed methods. Fig. 6 shows the cover image "Tiffany" and the stego image after the proposed method is used.

Table 5: Results of the AVL tree proposed method

Cover capacity	Tiffany			Baboon			Pepper		
Text file size (KB)	**MSE**	**PSNR**	**NCC**	**MSE**	**PSNR**	**NCC**	**MSE**	**PSNR**	**NCC**
1	0.0209	64.92	1.00	0.0218	64.75	1.00	0.0195	65.23	1.00
2	0.0392	62.20	1.00	0.0416	61.94	1.00	0.0373	62.42	1.00
4	0.0787	59.17	1.00	0.0820	58.99	1.00	0.0727	59.51	1.00
8	0.1573	56.16	1.00	0.1647	55.96	1.00	0.1466	56.49	1.00
16	0.3211	53.06	1.00	0.3249	53.01	1.00	0.2931	53.46	1.00
32	0.5720	50.56	1.00	0.5310	50.88	1.00	0.5326	50.87	1.00

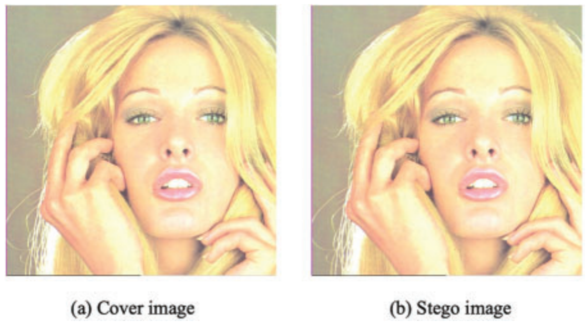

Figure 6: Tiffany's sample for Cover and Stego images (a) Cover image; (b) Stego image

Table 5 shows the metric results (MSE, PSNR, and NCC) of the proposed AVL tree method, tested on benchmarks (Tiffany, Baboon, and Pepper) with different plaintext file sizes (1, 2, 4, 8, 16, and 32 kilobytes). It is evident from Fig. 7 that the PSNR values of the cover object are approximately the same as those of the stego image for all three RGB layers. This indicates that the proposed method does not damage the cover object, and the stego image quality is sufficient to resist attacks from any harmful intruder. Additionally, comparing the results across all RGB layers shows that the PSNR values do not vary significantly. This consistency is due to the AVL tree technique, which hides secret message bits randomly in the cover image without sharing a stego key, as discussed in this proposed method. This technique embeds the secret message bits into the two significant bits of the red and blue channels, resulting in similar statistics for both the cover and stego objects. We achieve this by alternately hiding the two bits, once in the red channel and the next time in the blue channel.

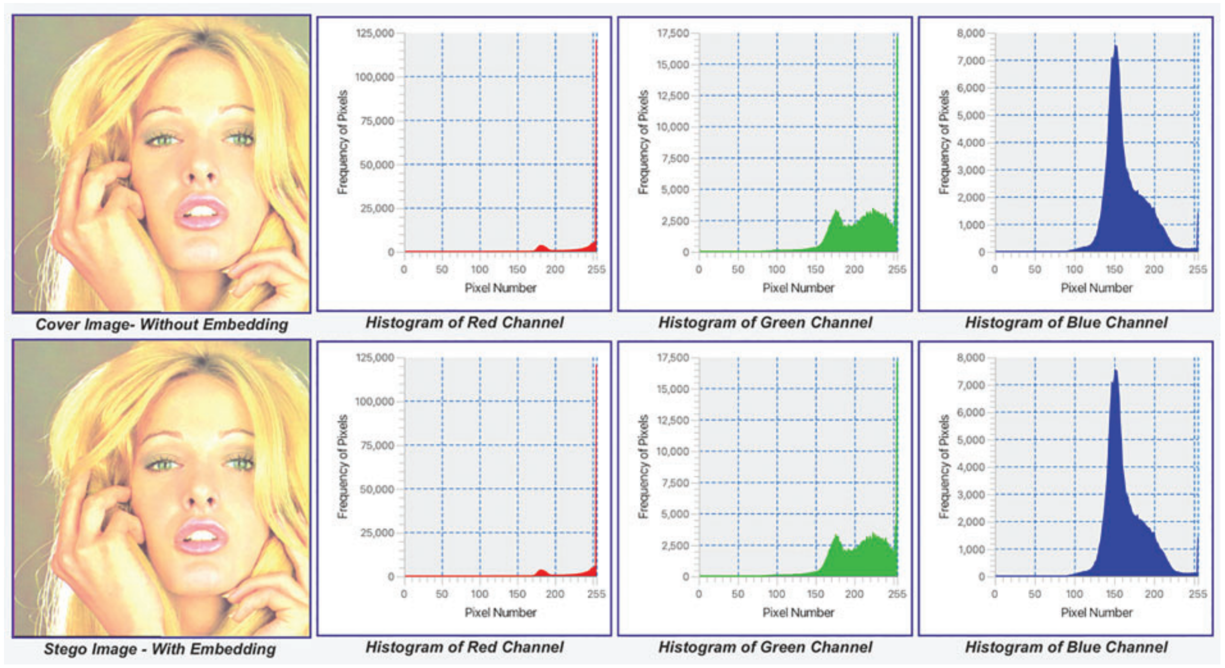

Figure 7: Tiffany.bmp (512 × 512) Histogram with a secret message size of 2 KB

The results of the proposed method are presented in line diagrams and tables, which authenticate the success of the proposed approach over existing methods. Figs. 7 through 12 represent the histograms of cover images *vs*. stego images for the benchmarks (Tiffany, Baboon, and Pepper). These figures were captured (via screenshots) from the developed AVLTreeStego App. As shown in Table 5, the benchmark results have a PSNR greater than 50 dB, indicating the high imperceptibility of the proposed method. Additionally, the mean standard error for this experiment is low, reflecting undetectable distortion and high image quality, with a Nice Cross-Correlation (NCC) of 1.0. This algorithm shows high imperceptibility because, when compared with recent research, the PSNR is consistently better than others except in studies [10] and [38]. In [38], the authors hide one character per pixel, whereas the AVL tree method hides only four bits. In [10], the image distortion is high due to a high MSE. The MSE of the AVL tree method is also better than that of other proposed methods, indicating high stego image quality. Furthermore, all compared methods use a stego key to conceal secret messages in the cover image, which must be shared with the recipient, but the AVL tree method does not. The capacity of this proposed method is also commendable.

5.4.4 Image Steganography Using Comparative Methods Benchmarks

In Table 6, the results of the comparative methods [8–10,30], and [36–38] are listed and compared with the proposed AVL tree method. These methods were tested on different 24-bit RGB cover images of various sizes (256 × 256, 481 × 321, or 512 × 512) and with different secret message lengths (114 bytes, 2 kilobytes, 5.3 kilobytes, 8 kilobytes, 10 kilobytes, and 18 kilobytes).

Table 6: Comparison of MSE and PSNR of the proposed method (AVL tree) with other techniques

Criteria	[8]	AVL tree	[9]	AVL tree	[10]	AVL tree	[30]	AVL tree	[36]	AVL tree	[37]	AVL tree	[38]	AVL tree
File size	114 bytes		18 KBytes		18 KBytes		10 KBytes		8 KBytes		5.3 KBytes		2 KBytes	
Image size	512 × 512		512 × 512		512 × 512		512 × 512		481 × 32		256 × 256		512 × 512	
PNSR	48.3	76.5	40.5	53.0	67.3	54.13	48.40	56.98	49.9	52.16	59.69	66.81	69.3	63.9
MSE	0.95	0.001	7.67	0.33	2.55	0.251	0.939	0.130	–	0.396	0.007	0.014	0.01	0.026
Colored	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Share keys	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No

The methods were tested on the same samples (cover images and secret messages). The PSNR results of [8,9,30] and [36] are above 50 dB, while those of [10], and [37,38] are below 50 dB. All proposed methods were shared with the recipient using stego keys. Testing the same dataset revealed that the suggested method had a PSNR of over 50 dB. The proposed method does not share any stego keys. Figs. 7 to 12 show some of the images' histograms. The small payload (2 KB) (Figs. 7, 9, and 11) resulted in a similar histogram for the stego image, where the payload capacity increased to 32 KB, resulting in low distortion as shown in Figs. 8, 10, and 12.

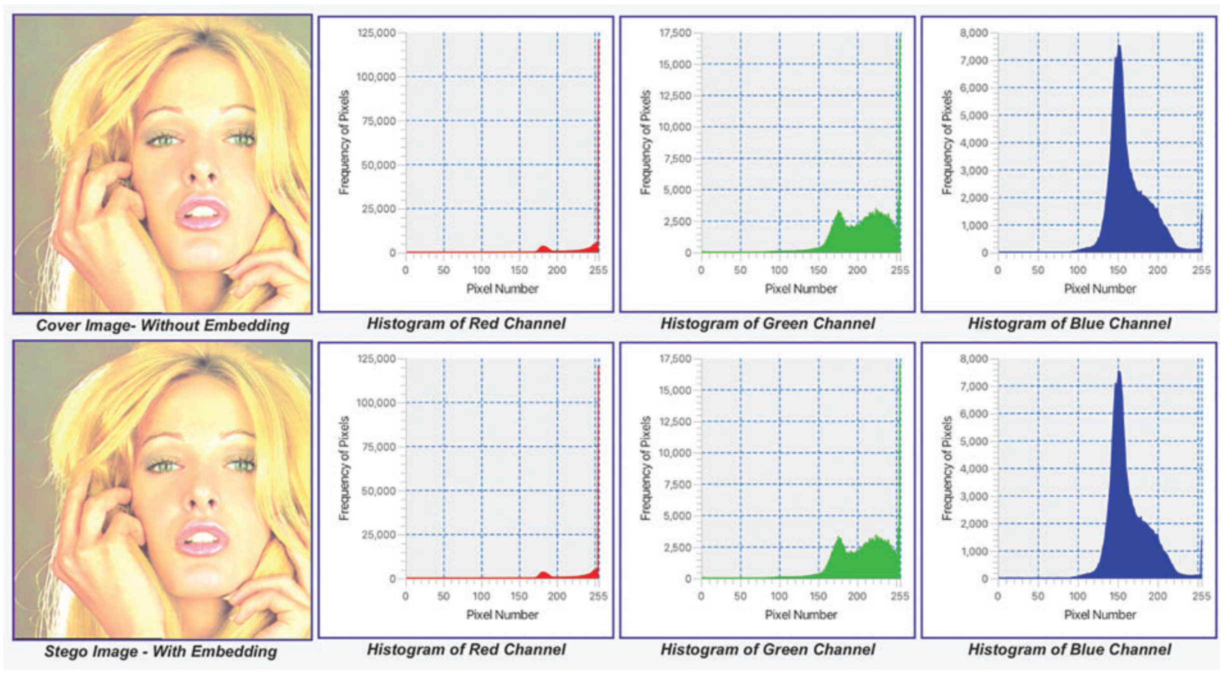

Figure 8: Tiffany.bmp (512 × 512) Histogram with a secret message size of 32 KB

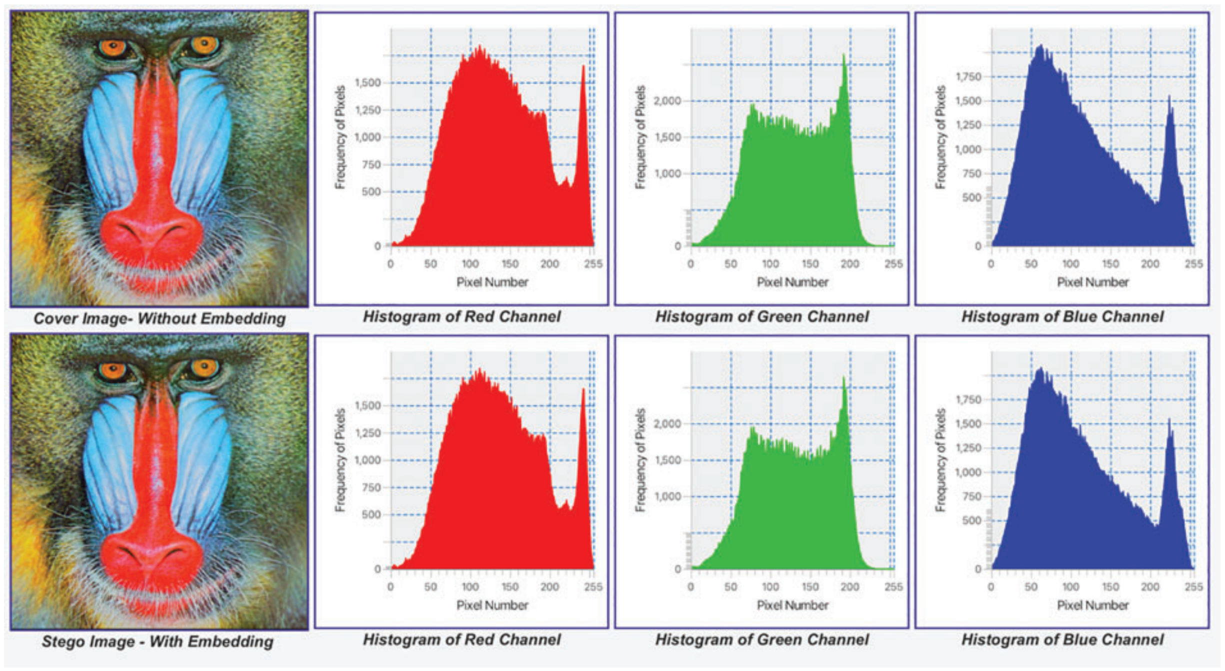

Figure 9: Baboon.bmp (512 × 512) Histogram with a secret message size of 2 KB

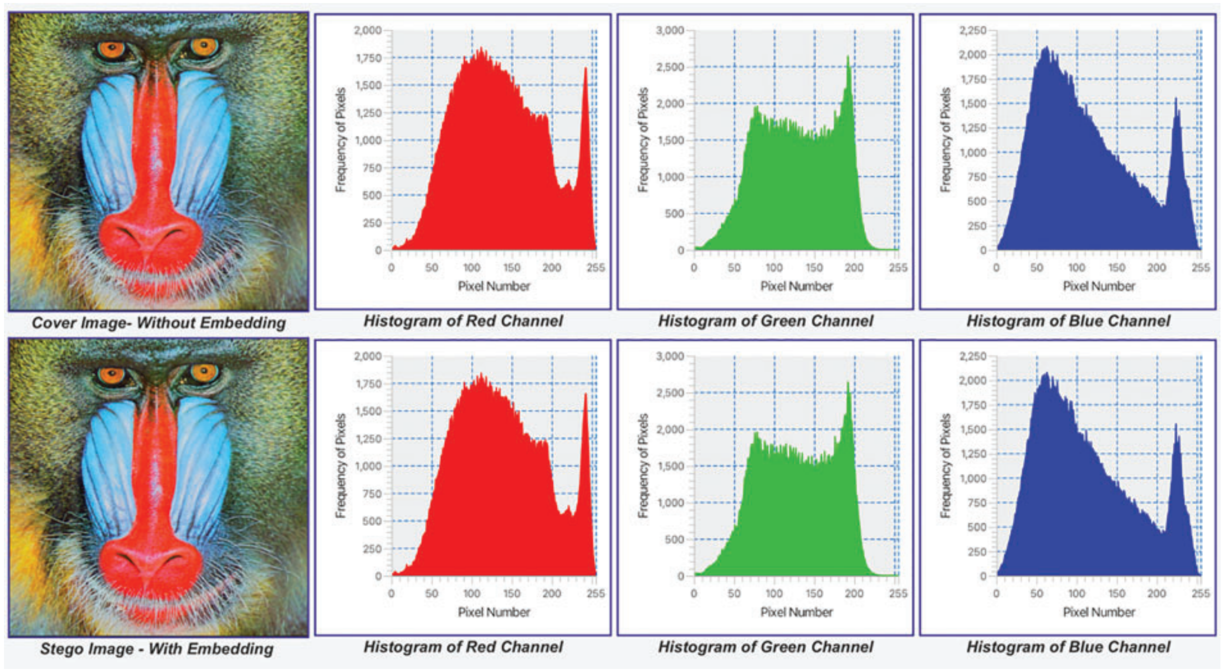

Figure 10: Baboon.bmp (512 × 512) Histogram with a secret message size 32 KB

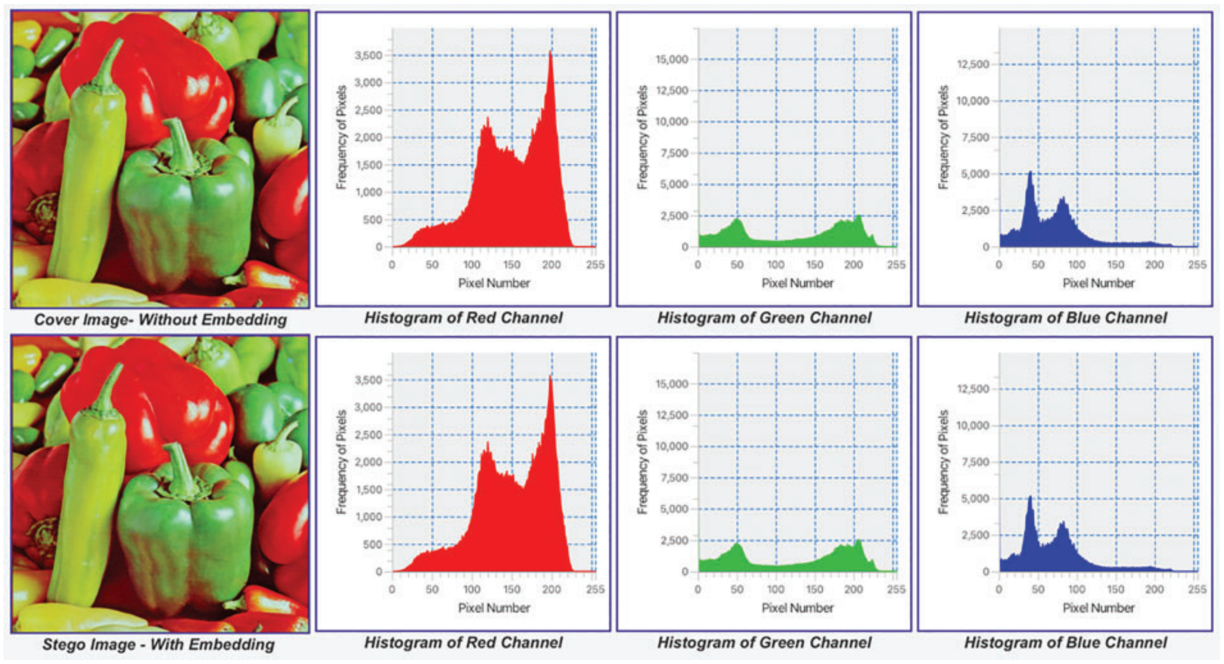

Figure 11: Pepper.bmp (512 × 512) Histogram with a secret message size of 2 KB

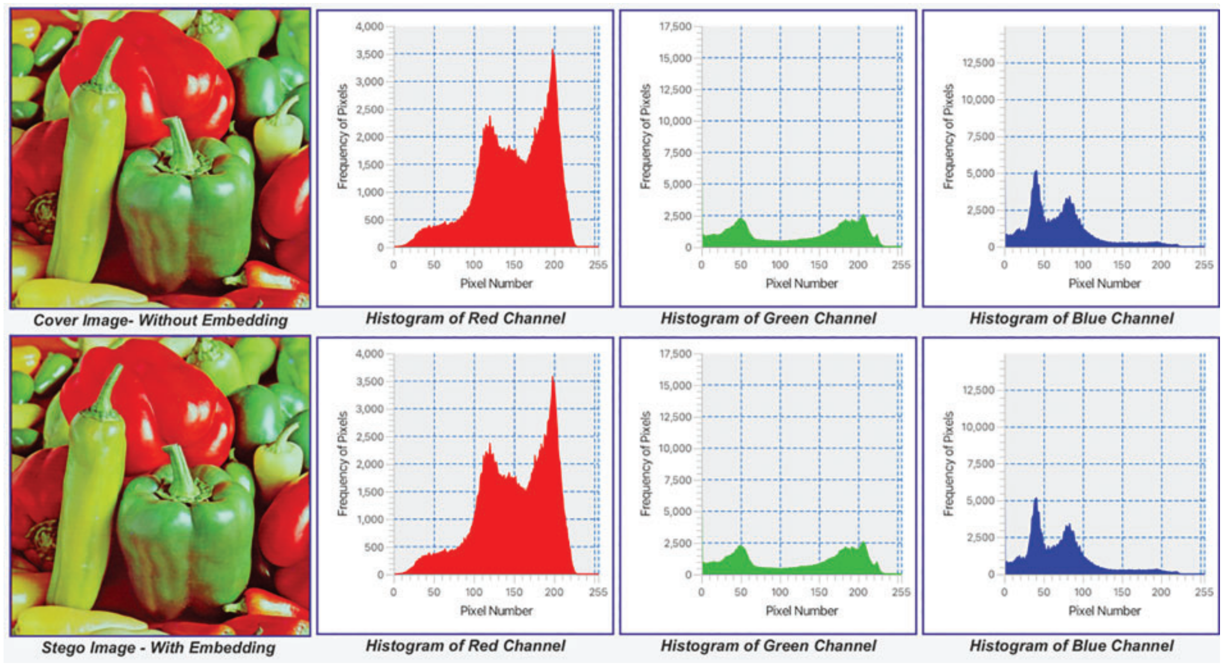

Figure 12: Pepper.bmp (512 × 512) Histogram with a secret message size of 32 KB

5.4.5 Image Steganography Using Benchmark of Proposed Methods

The probabilities of detecting the stego images with RS and chi-square using the StegoExpose tool [51] have averages of 0.05408144 for the RS Analysis and 0.13303362 for the chi-square analysis. When the same samples of the [8] method are tested with the proposed AVL tree data structure proposed method, the results show averages of 0.13303362 for the RS analysis and 0.05408144 for the chi-square analysis.

In Table 7, the results of the comparative method [8] are provided for security analysis obtained for the sample of stego images in Fig. 13.

Table 7: Security analysis comparison between [8] and AVL tree data structure proposed method

Method	[8]		AVL tree data structure method	
Images	Chi-Square	RS Analysis	Chi-Square	RS Analysis
Butterfly	0.02026724	0.02654193	0.25961672	0.16178021
Apple	0.29121700	0.01922108	0.24458866	0.01447541
Building	0.00509353	0.00480165	0.00315208	0.01529719
Fruit	0.02162384	0.02754041	0.02477702	0.02477294
Average	0.08455040	0.01952627	0.13303362	0.05408144

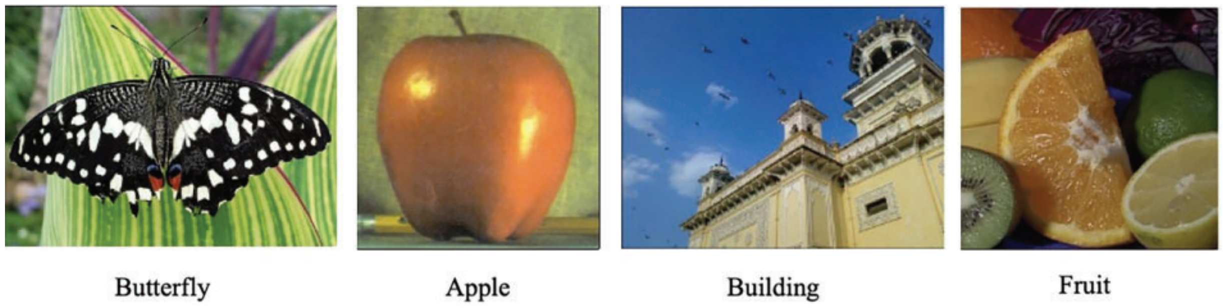

Figure 13: Cover samples for proposed method reprinted with permission from Tiwari et al. [8]

6 Discussion

As mentioned in Section 2, randomization techniques were used in many proposed methods to add more security to the hidden bits by selecting the pixels randomly. These techniques achieved random pixel selection differently. Many techniques use PRNG with seed keys like [9] and [37], while other proposed methods use mathematical equations to generate a random sequence number like [8,30], and [38]. The proposed methods [10,36] use patterns and matrices stored in files. All mentioned random techniques require sharing keys with the recipients. However, the method suggested in this paper does not share the stego key with the recipient to extract the secret data from the stego image, as it uses an AVL tree and queue data structure. The AVL tree data structure has the properties of sorted data and balanced node access, which is more efficient since accessing the tree nodes is O(log n). New integer nodes are added to the tree on the left side of the parent if their value is less than the parent node's value. Greater or equal values are placed on the right side after constructing the AVL tree, as shown in Fig. 3, based on the integer values of the green channel. This channel is sensitive to human eyes, so changes may be detected quickly. The traversal of the tree level by level can be done using a queue data structure. The root node is added to the queue. While the queue is not empty, access the root node and pop it from the queue. If the root has left or right nodes, push them to the queue. In this technique, the proposed method can access all nodes from top to bottom and from left to right, level by level, as shown in Fig. 3. These nodes [5, 3, 27, 15, 4, 19, ..., 38] are accessed randomly as tagged with the red numbers representing the pixels from the cover image. While traversing nodes of the tree from top to bottom, the 2-bit LSB of each red and blue channel will be altered with bits of length and secret message based on the two least significant bits of the green channel. The fluctuation between the red and blue channels to hide the message bits of a secret message depending on the two least significant bits of the green channel will enhance security because the sequence of hidden bits will not be in consecutive channels.

One of the main advantages of using the AVL tree data structure is sorting the object nodes in order. Traversing the AVL tree sequentially from left, parent, and right will result in an orderly printing of this tree. When the AVL tree in the figure is traversed, the result will be like 7, 10, 10, 14, 21, 22, ..., 215. This property is potent for sorting the color intensity in ascending order. So, when hiding two bits in LSB, the value of the red or blue channel will be incremented by +3, +2, +1, or decremented by –1, –2, –3, or unchanged. As a result, the changes in red and blue channels will be undetectable by attacker tools like StegoExpose. Furthermore, this proposed method uses traversal of the node of the AVL tree level by level to conceal the secret message bits in nonadjacent pixels to enhance the security level because when the bits are hidden in nonadjacent pixels, the attack tools, which depend on statistical analysis, cannot detect the existence of a bit or bits in these pixels.

When the AVL method is compared with the proposed method in [8], the tool detects one (Butterfly.bmp) image out of five from the sample in Fig. 13 that might have concealed data inside. However, it can detect the secret message, but the estimated file size was 37,008 bytes, which is not the correct size of the secret message because it was 114 bytes. Also, the extraction of secret messages cannot be done without an extraction algorithm. The AVL tree data structure showed strong security against the attack tool, although it doesn't use any encryption method to encrypt the secret message as in the method [8].

The chi-square statistical test is the foundation of the chi-square steganalysis algorithm, which compares the expected distribution of pixel values in a cover image to the distribution in an image under steganalysis. This method detects hidden information in steganographic methods, which often involve minor modifications to the image's statistical characteristics, such as shifting the pixel value distribution or adding correlations between nearby pixels [49]. Residual statistics (RS) refer to the statistical properties of differences between original cover media and stego media, obtained by subtracting the cover media from the stego media [13]. When the statistical results of both techniques are near zero, the attacker fails to detect the existence of the secret message in the stego images.

The proposed method was tested on a benchmark (Tiffany, Baboon, and Pepper) from the dataset, and the test showed high security after being checked by the StegoExpose tool. This tool 100% fails to detect secret messages in the cover images of traditional datasets (Baboon, Tiffany, Pepper) with different lengths of secret files (1, 2, 3, 4, 5, 8, 16, and 32 KB). When the chi-square test is applied to the sample images (Tiffany, Baboon, and Pepper) of the research using the StegExpose tool [51], the probability of detecting the stego images (Tiffany.bmp, Baboon.bmp, and Pepper.bmp) using Chi-square detection averages is 0.00683106. The results showed that chi-square failed to discover the existence of the hidden message in the stego image. Also, the suggested scheme shows that the system is resistant to RS attacks in various payloads. The probability of detecting the stego images (Tiffany.bmp, Baboon.bmp, and Pepper.bmp) using RS analysis averages is 0.09962817. The results showed that RS analysis failed to discover the hidden message in the stego image.

Authors in [8] use the Advance Encryption Standard (AES) algorithm to encrypt the message before hiding bits in the cover image to show that the proposed algorithm is highly secure. Still, the AVL tree data structure algorithm showed the same resistance to attackers with approximately the same results as Table 7 shows without encrypting the message.

It is feasible to spot possible alterations that point to the existence of steganographic content by comparing the histograms of the stego and cover images. Depending on the needs of the investigation, the entire image can be subjected to histogram-based steganalysis or just particular sections of interest. The presence of concealed information can be found by looking at the distribution of pixel values and finding disparities. It's crucial to remember that although histogram analysis might be helpful in steganalysis, it is not infallible.

7 Conclusion and Future Work

In conclusion, the current practice uses simple data structures that do not require a steganographic context or data modeling. This paper offers more sophisticated data structures, which can be considered robust against common steganalytic attacks. In steganography, a high PSNR indicates the hidden data is less detectable, making the stego image appear more like the original cover image. This ensures that the embedded data remains secure and undetected by casual observers or automated detection methods. Testing the AVLTree-Steg method on the benchmark demonstrates high imperceptibility, with the PSNR exceeding 50 dB for all tested benchmark step images. The MSE is critical in steganography as it quantifies the average squared difference between the cover and stego images. A lower MSE indicates that the stego image closely resembles the original cover image, reducing the likelihood of detecting hidden data. Minimal distortion is essential for maintaining the cover image's quality and ensuring the steganographic method's security and effectiveness.

The proposed method demonstrates a low MSE, indicating minimal distortion between the cover and stego images. The NCC measures the similarity between the original cover image and the stego image, with a value close to 1.0 indicating strong similarity. This NCC value is crucial as it guarantees the inconspicuousness of the embedded data and the preservation of the cover image integrity. All stego images in the benchmark exhibit an NCC of approximately 1.0. Moreover, when comparing the benchmark images to the stego images, the histograms of all channels closely match those of the cover images, indicating the method's excellent security.

Comparing the AVLTree-Steg method with other recent methods reveals high imperceptibility and large capacity. Furthermore, the proposed method has a high level of security, as steganalysis tools cannot detect it. This work uses an AVL tree in steganography to generate random pixel locations that conceal secret bits. This paper implements a randomization technique that eliminates the need for sharing stego keys, significantly reducing the risk of interception or unauthorized access. Steganography becomes inherently secure without the need for stego keys. One example of keyless steganography is using image-based techniques, where secret messages are hidden within the pixel values of an image. By slightly altering the least significant bits of the image's pixels, hidden data can be embedded without noticeable changes to image quality. This method allows the message to be extracted simply by reversing the pixel alterations. This paper demonstrates improved imperceptibility, by altering the red or blue pixels of the cover image to occur in a different order. This AVL tree contains information from top to bottom and left to right, enhancing the concealment method and significantly reducing the detectability of hidden data. The node-hiding process relies on the green channel, providing consistent results when analyzing the tree horizontally and level-by-level.

In the future, we can enhance capacity by concealing confidential data in all channels of the cover image and altering pixels based on various criteria. By distributing hidden data more evenly across the red, green, and blue channels, we can increase the overall capacity and improve the imperceptibility of the cover image. We can embed a higher volume of confidential information using this approach without compromising the visual quality of the cover image. Additionally, spreading the data across multiple channels reduces the likelihood of detection, as changes in pixel values are less concentrated and, therefore less noticeable. We can also encrypt sensitive bits before embedding them inside the cover image. We will also explore using alternative data structures, such as a hash, heap, stack, queue, or graph, to improve capacity, imperceptibility, efficiency, or security. These data structures can store a secret message that is extremely difficult to decipher. It is possible to further improve the security of a secret message by encrypting it with encryption algorithms. For instance, the Advanced Encryption Standard (AES) is widely used to secure sensitive data. AES is known for its robustness and efficiency, making it a popular choice for both government and commercial applications. By using AES, you can ensure that your message remains confidential and protected from unauthorized access. Finally, we can utilize compression algorithms, such as Huffman coding, to further reduce the encrypted messages, and improve the overall security of the method.

Acknowledgement: The authors would like to thank the editors and reviewers for their detailed review and insightful advice.

Funding Statement: This research was funded by the University Kebangsaan Malaysia.

Author Contributions: The authors confirm their contribution to the paper as follows: Study conception and design: Murad Njoum, Rossilawati Sulaiman, and Zarina Shukur; Analysis and interpretation of results: Murad Njoum; Draft manuscript preparation: Murad Njoum, Rossilawati Sulaiman, and Faizan Qamar. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and codes supporting this study's findings are available from the corresponding authors upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that there are no conflicts of interest to report regarding the present study.

References

[1] V. K. Trivedi, S. Stalin, S. Joshi, M. H. Alkinani, P. K. Shukla and B. Gaur, "User data privacy in multimedia domain using 3-3 LSB-based color image steganography with RC4 and Bernoulli map protection," *J. Electron. Imaging*, vol. 31, no. 3, 2022, Art. no. 033021. doi: 10.1117/1.JEI.31.3.033021.

[2] M. M. Iqbal, U. Khadam, K. J. Han, J. Han, and S. Jabbar, "A robust digital watermarking algorithm for text document copyright protection based on feature coding," in *2019 15th Int. Wirel. Commun. Mobile Comput. Conf. (IWCMC)*, Tangier, Morocco, IEEE, Jun. 2019, pp. 1940–1945.

[3] K. C. Nunna and R. Marapareddy, "Secure data transfer through internet using cryptography and image steganography," in *2020 SoutheastCon*, Raleigh, NC, USA, IEEE, vol. 2, 2020. doi: 10.1109/SoutheastCon44009.2020.9368301.

[4] V. B. Savant and R. D. Kasar, "A review on network security and cryptography," *Res. J. Eng. Technol.*, vol. 12, no. 4, pp. 110–114, 2021. doi: 10.52711/2321-581X.

[5] L. Akhila and V. J. Manoj, "Image steganography using pixel value differencing with modulus function and optimization," in *2022 4th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Tirunelveli, India, 2022, pp. 1369–1373.

[6] F. Al-Shaarani and A. Gutub, "Securing matrix counting-based secret-sharing involving crypto steganography," *J. King Saud Univ.–Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6909–6924, 2022. doi: 10.1016/j.jksuci.2021.09.009.

[7] C. K. Deo, A. Singh, D. K. Singh, and N. K. Soni, "Developing a highly secure and high Capacity LSB steganography technique using PRNG," in *2020 Int. Conf. Comput. Perform. Eval. (ComPE)*, Shillong, India, 2020, pp. 136–140.

[8] K. Tiwari and S. J. Gangurde, "LSB steganography using pixel locator sequence with AES," in *2021 2nd Int. Conf. Secure Cyber Comput. Commun. (ICSCCC)*, Jalandhar, India, 2021, pp. 302–307.

[9] U. M. E. Ali, E. Ali, M. Sohrawordi, and M. N. Sultan, "A LSB based image steganography using random pixel and bit selection for high payload," *Int. J. Math. Sci. Comput.*, vol. 3, pp. 24–31, 2021.

[10] K. H. Abuhmaidan, A. K. Kayed, and M. Alrisia, "Steganography: A flexible embedded randomization technique," *KSII Trans. Internet Inf. Syst.*, vol. 17, no. 1, pp. 120–144, 2023. doi: 10.3837/tiis.2023.01.007.

[11] D. Megías, W. Mazurczyk, and M. Kuribayashi, "Data hiding and its applications: Digital watermarking and steganography," *Appl. Sci.*, vol. 11, no. 22, 2021, Art. no. 10928. doi: 10.3390/app112210928.

[12] S. Panchikkil, V. M. Manikandan, Y. Zhang, and S. Wang, "A multi-directional pixel-swapping approach (MPSA) for entropy-retained reversible data hiding in encrypted images," *Entropy*, vol. 25, no. 4, 2023, Art. no. 563. doi: 10.3390/e25040563.

[13] M. N. M. Najih, E. H. Rachmawanto, C. A. Sari, and S. Astuti, "An improved secure image hiding technique using PN-sequence based on DCT-OTP," in *2017 1st Int. Conf. Inform. Comput. Sci. (ICICoS)*, Semarang, Indonesia, 2017, pp. 47–52.

[14] M. Santhanalakshmi, K. Lakshana, and G. M. Shahitya, "Enhanced AES-256 cipher round algorithm for IoT applications," *Sci. Temper*, vol. 14, no. 1, pp. 184–190, 2023. doi: 10.58414/SCIENTIFICTEMPER.2023.14.1.22.

[15] P. Dijesh, S. Babu, and Y. Vijayalakshmi, "Enhancement of e-commerce security through asymmetric key algorithm," *Comput. Commun.*, vol. 153, no. 7, pp. 125–134, 2020. doi: 10.1016/j.comcom.2020.01.033.

[16] R. Wazirali, W. Alasmary, M. M. Mahmoud, and A. Alhindi, "An optimized steganography hiding capacity and imperceptibly using genetic algorithms," *IEEE Access*, vol. 7, pp. 133496–133508, 2019. doi: 10.1109/ACCESS.2019.2941440.

[17] K. Maheswari, C. Siva, and G. Nalinipriya, "An innovative model for secure environment using steganography," in *2022 8th Int. Conf. Smart Struct. Syst. (ICSSS)*, Chennai, India, 2022, pp. 1–5.

[18] A. A. Almayyahi, R. Sulaiman, F. Qamar, and A. E. Hamzah, "High-security image steganography technique using XNOR operation and fibonacci algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 10, pp. 511–522, 2020. doi: 10.14569/issn.2156-5570.

[19] A. H. Ali, L. E. George, A. A. Zaidan, and M. R. Mokhtar, "High capacity, transparent and secure audio steganography model based on fractal coding and chaotic map in temporal domain," *Multimed. Tool Appl.*, vol. 77, no. 23, pp. 31487–31516, 2018. doi: 10.1007/s11042-018-6213-0.

[20] H. Zhao, Y. Liu, Y. Wang, S. Liu, and C. Feng, "A video steganography method based on transform block decision for H. 265/HEVC," *IEEE Access*, vol. 9, pp. 55506–55521, 2021. doi: 10.1109/ACCESS.2021.3059654.

[21] M. A. Majeed, R. Sulaiman, and Z. Shukur, "New text steganography technique based on part-of-speech tagging and format-preserving encryption," *KSII Trans. Internet Inf. Syst.*, vol. 18, no. 1, pp. 170–191, 2024.

[22] O. H. Alhabeeb, F. Fauzi, and R. Sulaiman, "Developing a novel DNA-based steganography algorithm using random table generation with segmentation," *Multimed. Tools Appl.*, vol. 83, no. 14, pp. 40529–40567, Apr. 2024. doi: 10.1007/s11042-023-16699-7.

[23] Y. P. Astuti, E. H. Rachmawanto, and C. A. Sari, "Simple and secure image steganography using LSB and triple XOR operation on MSB," in *2018 Int. Conf. Inf. Commun. Technol. (ICOIACT)*, Yogyakarta, Indonesia, 2018, pp. 191–195.
[24] A. Ahmed and A. Ahmed, "A secure image steganography using LSB and double XOR operations," *Int. J. Comput. Sci. Net.*, vol. 20, no. 5, pp. 139–139, 2020.
[25] N. M. Al-Aidroos and H. A. Bahamish, "Image steganography based on LSB matching and image enlargement," in *2019 First Int. Conf. Intell. Comput. Eng. (ICOICE)*, Hadhramout, Yemen, 2019, pp. 1–6.
[26] F. Baso, "Performance analysis of the last significant bit (LSB) method in steganography for data hiding in image data," *J. Secur., Comput., Inf., Embedded, Netw., Intell. Syst.*, pp. 58–62, 2023. doi: 10.61220/scientist.v1i2.20234.
[27] S. Bilgaiyan, R. Ahmad, and S. Sagnika, "Adaptive image steganography using rotating color channels and inverted LSB substitution," *SN Comput. Sci.*, vol. 4, no. 5, 2023, Art. no. 565. doi: 10.1007/s42979-023-01949-0.
[28] Y. Y. Demircan and S. Ozekes, "A novel LSB steganography technique using image segmentation," *J. Univers. Comput. Sci. (JUCS)*, vol. 30, no. 3, pp. 308–332, 2024. doi: 10.3897/jucs.105702.
[29] A. K. Patel and D. Vekariya, "A literature review on image quality and embedding payload for image steganography," in *AIP Conf. Proc.*, AIP Publishing, Dec. 2023, vol. 2855, no. 1.
[30] N. A. F. Abbas, N. Abdulredha, R. K. Ibrahim, and A. H. Ali, "Security and imperceptibility improving of image steganography using pixel allocation and random function techniques," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 1, pp. 694–705, 2022.
[31] K. Joshi, S. Gill, and R. Yadav, "A new method of image steganography using 7th bit of a pixel as indicator by introducing the successive temporary pixel in the grayscale image," *J. Comput. Netw. Commun.*, vol. 2018, no. 8, pp. 1–10, 2018. doi: 10.1155/2018/9475142.
[32] S. Ghoul, R. Sulaiman, and Z. Shukur, "A review on security techniques in image steganography," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 6, 2023. doi: 10.14569/issn.2156-5570.
[33] O. Kuznetsov, E. Frontoni, and K. Chernov, "Beyond traditional steganography: Enhancing security and performance with spread spectrum image steganography," *Appl. Intell.*, vol. 54, no. 7, pp. 5253–5277, 2024. doi: 10.1007/s10489-024-05415-z.
[34] Y. G. Yang, B. P. Wang, Y. H. Zhou, W. M. Shi, and X. Liao, "Efficient color image encryption by color-grayscale conversion based on steganography," *Multimed. Tools Appl.*, vol. 82, no. 7, pp. 10835–10866, 2023. doi: 10.1007/s11042-022-13689-z.
[35] V. Rahmani and M. Mohammadpour, "High hiding capacity steganography method based on pixel indicator technique," in *2017 5th Iranian Joint Congress Fuzzy Intell. Syst. (CFIS)*, 2017, pp. 144–149.
[36] M. Zhang, S. Zhang, and L. Harn, "An efficient and adaptive data-hiding scheme based on secure random matrix," *PLoS One*, vol. 14, no. 10, 2019, Art. no. 222892. doi: 10.1371/journal.pone.0222892.
[37] A. Gahan and G. D. Devanagavi, "A secure steganography model using random-bit select algorithm," in *2020 Third Int. Conf. Adv. Electron., Comput. Commun. (ICAECC)*, Bengaluru, India, 2020, pp. 1–5.
[38] K. Kordov and S. Zhelezov, "Steganography in color images with random order of pixel selection and encrypted text message embedding," *PeerJ Comput. Sci.*, vol. 7, no. 11, 2021, Art. no. 380. doi: 10.7717/peerj-cs.380.
[39] W. Wu and H. Li, "A novel scheme for random sequential high-capacity data hiding based on PVD and LSB," *Signal, Image Video Process.*, pp. 1–11, 2023.
[40] M. K. Abdul-Hussein and H. T. S. ALRikabi, "Secured transfer and storage image data for cloud communications," *Int. J. Online Biomed. Eng.*, vol. 19, no. 6, pp. 4–17, 2023.
[41] L. Zhang, X. Song, A. A. A. El-Latif, Y. Zhao, and B. Abd-El-Atty, "Reversibly selective encryption for medical images based on coupled chaotic maps and steganography," *Complex Intell. Syst.*, vol. 10, no. 2, pp. 2187–2213, 2024.
[42] J. Horng, S. Xu, C. Chang, and C. Chang, "An efficient data-hiding scheme based on multidimensional Mini-SuDoKu," *Sensors*, vol. 20, no. 9, 2020, Art. no. 2739. doi: 10.3390/s20092739.

[43] G. Shao, J. Liu, and D. Shen, "A novel steganography scheme for color image based on HLS translation," *J. Phys.: Conf. Ser.*, vol. 2025, no. 1, 2021, Art. no. 12058. doi: 10.1088/1742-6596/2025/1/012058.

[44] N. Pan, J. Qin, Y. Tan, X. Xiang, and G. Hou, "A video coverless information hiding algorithm based on semantic segmentation," *EURASIP J. Image Video Process.*, vol. 2020, no. 1, 2020. doi: 10.1186/s13640-020-00512-8.

[45] Q. Jiang, "An image hiding algorithm based on bit plane and two-dimensional code," in *2021 Third Int. Conf. Intell. Commun. Technol. Virtual Mobile Netw. (ICICV)*, Tirunelveli, India, 2021.

[46] Z. Tang, H. Nie, C. M. Pun, H. Yao, C. Yu and X. Zhang, "Color image reversible data hiding with double-layer embedding," *IEEE Access*, vol. 8, pp. 6915–6926, 2020. doi: 10.1109/ACCESS.2020.2964264.

[47] H. T. Elshoush, M. M. Mahmoud, and A. Altigani, "A new high capacity and secure image realization steganography based on ASCII code matching," *Multimed. Tools Appl.*, vol. 81, no. 4, pp. 1–47, 2022. doi: 10.1007/s11042-021-11741-y.

[48] D. Deshmukh and D. G. Kurundkar, "Video steganography using edge detection techniques," in *Proc. Int. Conf. Commun. Inf. Process. (ICCIP)*, Chongqing, China, 2019.

[49] C. C. Chang, G. D. Su, C. C. Lin, and Y. H. Li, "Position-aware guided hiding data scheme with reversibility and adaptivity for dual images," *Symmetry*, vol. 14, no. 3, 2022, Art. no. 509. doi: 10.3390/sym14030509.

[50] I. H. Pan, K. C. Liu, and C. L. Liu, "Chi-square detection for PVD steganography," *2020 Int. Symp. Comput., Consumer Control*, no. 3C), pp. 30–33, 2020. doi: 10.1109/IS3C50286.2020.

[51] B. Boehm, "Stegexpose-A tool for detecting LSB steganography," 2014, *arXiv:1410.6656*.

[52] J. Baxter, "Dataset text document classification," 2020. Accessed: Jun. 8, 2020. [Online]. Available: https://www.kaggle.com/datasets/sunilthite/text-document-classification-dataset

[53] M. Al-Jarrah, "RGB-BMP steganalysis dataset," *Mendeley Data*, vol. V1, 2018. doi: 10.17632/sp4g8h7v8k.1.