High-Security Image Steganography Technique using XNOR Operation and Fibonacci Algorithm

Ali Abdulzahra Almayyahi[1], Rossilawati Sulaiman[2], Faizan Qamar[3], Abdulwahhab Essa Hamzah[4]
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia UKM Bangi 43600, Selangor, Malaysia[1, 2, 3]
Photonics Technology Laboratory, Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, UKM Bangi 43600, Selangor, Malaysia[4]

***Abstract*—Since the number of internet users is increasing and sensitive information is exchanging continuously, data security has become a problem. Image steganography is one of the ways to exchange secret data securely using images. However, several issues need to be mitigated, especially in the imperceptibility (security) aspect, which is the process of embedding secret data in the images that can be vulnerable to attacks. This paper focuses on developing a secure method for hiding secret messages in an image, based on the standard Least Significant Bit (LSB). Before proceeding with the embedding stage, the secret message's size is reduced by compression using the Huffman algorithm, followed by two operations, which are the Boolean operation Exclusive-NOR (XNOR) operation and the Fibonacci algorithm when selecting pixels to embed the secret message. As a result of these processes, a stego-image is created with two secret keys. We obtained promising results against standard images with higher Peak Signal-to-Noise Ratio (PSNR) values of 66.6170, 65.8928, and 65.9386 dB for Lena.bmp, Baboon.bmp, and Pepper.bmp, respectively, as compared to other state-of-the-art schemes. The evaluation stage proves the increasing level of security as well as imperceptibility.**

***Keywords*—Image steganography; Huffman algorithm; XNOR operation; Fibonacci algorithm; LSB; PSNR**

I. Introduction

Steganography is the technique used to hide secret data in a cover-media. It consists of two parts, "stego" which means "cover" in Greek and "grafia" which means "writing" and both can be defined as "covered writing" [1]. Various types of multimedia files can be used to conceal secret data such as text [2], image [3], video [4]-[5], audio [6][39] protocols, and Deoxyribonucleic acid (DNA), with the most commonly used, are images as a cover-media [7]–[9] The cover-media can also be referred to as stego-media. There are two types of embedding methods in a steganography system, which are spatial domain and frequency domain. In the spatial domain, pixels intensity is used to insert or embed the secret message directly into the Least Significant Bit (LSB) of a pixel. The LSB substitution is the most common technique of embedding in the spatial domain, whereas, in the frequency domain, the image is converted into various frequency classes, and the embedding process has been performed by using the coefficient factors [10]–[12].

Several issues and challenges are associated with steganography, which includes (1) lack of security of the secret message hidden in the image, (2) lack of payload capacity, which reflects the amount of data that can be embedded into the image, and (3) maintain high imperceptibility or quality of the stego-image as similar as the cover image (original image), which is the desired property that supports security [13]–[15]. Several literature attempts to find a balance between the quantity of data embedded and the protection of the secret message while maintaining the quality of the image [16], [17]. Some algorithms have succeeded in increasing the data hiding while keeping the image's quality, but the majority have not [18], [19]. These existing methods have made more efforts in hiding information to get high security so that the concealing method makes the hidden secret message not to be seen by a hacker (or not seen by human eyes). However, more research in steganography techniques need to do to achieve high security and high capacity [20].

This paper proposes a new technique for enhancing the security and capacity of the standard LSB method. A balance between high security and high capacity is important to maintain the cover-image quality similar to stego-image after embedding stage. We exploit the Most Significant Bit (MSB) and LSB in the cover-images to add a security level of the standard LSB steganography methods. Also, capacity can be increased by applying compression to the secret message.

The rest of the paper is organized as follows: Section II defined related works on secret data hiding techniques. Then, Section III illustrates the proposed method in details, followed by Section IV, which presents the experimental results. Section V states the conclusion of the work, and lastly, Section VI shows recommendations for future research work.

II. Related Work

This section discusses the existing solutions to counter the issues in image steganography. Various versions of bit inversion techniques are proposed to enhance the LSB embedding process. In [3], the authors proposed an improved LSB technique where the LSB's of some pixels of the cover-image is inverted when inputs of specific bit patterns are found. Their result shows that a high quality of invisibility and imperceptibility in the stego-image. Another bit inversion technique is suggested in [21], which uses a bit inversion based on specific patterns to improve the quality of the stego-image. The idea is to conceal the secret message after a lossless compression of smoother areas of the image, resulting in fewer pixels being modified in the stego-image.

The authors in [22] proposed a novel steganographic algorithm in the spatial domain using the concept of pixel

modulation, which diminishes the changes that occur in the stego-image generated from the cover-image. In this method, the LSB of the image's intensity values is modified according to the secret data. The secret data is converted into a binary string based on each character's ASCII value that is represented in the 8-bit notation. Then, all bits are embedded in a matrix containing the adjacent pixel values. Their result shows a lower average embedding capacity, but the PSNR values of the stego-images produced are high. In order to improve the security of the hidden secret message, the XOR transposition encryption based on the LSB approach is proposed in [23]. The embedded secret message is encrypted by transposing the message based on the order of the key. The result of this process is converted into binary and embedded into the cover-image using the XOR operation.

An LSB method is developed by [24] for embedded secret message into a 24-bit colour image, where the last two bit of LSB of each channel (red, green and blue) of the cover-image, are substituted by two bits of the secret message. This means that the last two bits of LSB of the red channel are substituted by the first two bits of the secret message; the last two bits of LSB of the green channel are substituted by the second two bits of the secret message, and the last two bits of LSB of the blue channel are substituted by the third two bits of the secret message. Therefore, a total of six bits of the secret image can be concealed in a 24-bit colour image. Another approach in [25] proposed a new technique intending to keep secure communication intact. The proposed method merged the advantage of the two bits of LSB and the XOR operation. the 8th bit of the cover-image is XORed with the 1st bit of the secret data, and the 7^{th} bit is XORed with the 2nd bit of the secret data. The bits from the final result are inserted into the last two LSBs of the image pixel. The results proved that this approach achieved high capacity and high security. An efficient LSB steganography method is proposed in [26] for transferring secret data in the digital format using the number theory. This research focuses on the spatial domain using the Fibonacci sequence and Zeckendorf theorem, which increase the embedding rate from 8-bit to 12-bit, on a grayscale image. This method achieves a high embedding capacity.

An adaptive LSB replacement was suggested by [27] for several colour images. In this method, the RGB colour channels are converted into the YCbCr form [28], where the Cr channel is selected and divided into an 8*8 size block. Each block is then transformed into discrete cosine, and a coefficient is selected where the bit sequence of the secret message is embedded to this coefficient. The image quality is preserved and robust against stego-attacks. An image steganography method was developed by [29], which is based on the use of two secret keys to randomize the embedding process of secret messages. Randomization increases the security of the secret message. The proposed method used some colour such as Red, Green, and Blue to enhance the pixel's values and calculate the location of the pixels' random position to embed the secret message. This approach provides high data hiding capacity and a high-security level compared with the standard LSB substitution technique.

A method of using a modified LSB algorithm is proposed in [30], based on the three RGB channels to increase the security level of the hidden message. This method relied on encryption of secret text message using the encryption key and the XNOR operation before embedding it in a colour image using LSB. The idea of concealing the message depends on the extraction of chromatic channels of the three RGB channels for each pixel and specifying the channel in which the bit of the encryption message is hidden.

A highly secure chaos-based image steganography method is proposed in [31]. Encryption to the secret message is performed using Caesar cipher and Chaos theory. The ciphertext obtained after the encryption process is embedded in a cover image using a 3, 3, 2 LSB replacement algorithm, which gave better security and performance than the traditional LSB technique. Another approach that combines the cryptography and steganography methods was presented in [32]. The Vigenere Cipher and Huffman Coding techniques are used to encrypt and compress the secret message. This method improved the security level and ensured the message content could not be recovered without knowledge of the decrypting key and the Huffman Dictionary table.

An approach for data hiding in RGB images based on the grey level modification (GLM) and multi-level encryption (MLE) is proposed in [33]. The secret key and secret messages are encrypted using the MLE algorithm before mapping it to the grayscale cover-image. A transposition function is applied to the cover-image before data embedding. The use of a secret key, MLE, and GLM adds various security levels to the algorithm, making it very complicated for a malicious user to recover the original secret data.

Based on the literature, many solutions have been proposed to solve standard LSB's main problems, which are related to security, capacity, and imperceptibility. Security focuses on the embedding methods, which is considered secure if an attacker finds it challenging to extract secret data from the cover medium. On the other hand, high capacity means that there are larger spaces to store secret messages in the cover-image. Lastly, imperceptibility is related to the high quality of the stego-image generated by the algorithm that is close to the original cover-image. In this study, the focus is on increasing the security level of a stego-image (maintaining imperceptibility) while improving the embedding capacity by utilizing the standard LSB image technique.

III. Proposed Methodology

The proposed methodology is explained in Fig. 1. It consists of four different stages as following:

- Data Preparation stage.
- Data Embedding stage.
- Evaluating stag.
- Extracting stage.

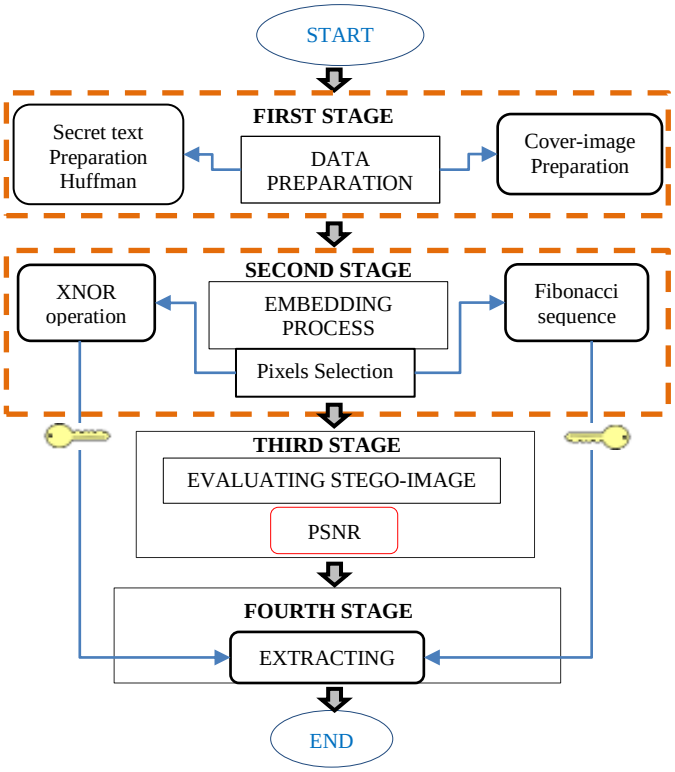

Fig. 1. Proposed Method.

A. First Stage: Data Preparation

1) Huffman coding: Before proceeding with the embedding stage, the Huffman algorithm is applied to the secret message to compress the message and then transform it into streams of bits. Later, each stream is converted into a secret code using the Huffman table. The result after the compression process is the vector of secret codes. The advantages of Huffman encoding are 1) lossless compression that improves the embedding capacity and 2) security because the encoded bitstream does not really disclose any information and only can be decoded using the Huffman table [34]. However, if the Huffman coded bitstream changes with just one bit, the Huffman table could not decode it. Huffman coding also satisfies the desired condition of high embedding capacity [21].

Next, a numerical example is provided to explain how Huffman will work within the proposed system. This example used five different symbols as B, E, A, D, and C with their frequencies of 14, 8, 56, 10, 12, respectively, as shown in Table I. The following seven steps are performed:

- **Step 1:** Construct a table that contains the source symbols and their respective frequency numbers.
- **Step 2:** Arrange the source symbols in ascending order according to their frequency numbers.
- **Step 3:** Merge the first two frequency numbers and then rearrange the table.
- **Step 4:** Repeat Step 3 until a single frequency number is obtained.
- **Step 5:** Construct a Huffman tree by assigning the value of 0 or 1 to each pair of branches in the tree.
- **Step 6:** Construct the final table (Huffman coding) that contains the leaf nodes and their respective codes according to the Huffman tree.
- **Step 7:** Create the compressed secret text by rewriting the output codes using the table of Huffman coding [1, 011, 010, 001, 000], as mentioned in Table II (in the 'Code' column).

As illustrated in the Huffman tree (Fig. 2), the two-parent nodes have the frequencies of 18 and 26, from accumulating the frequency of their children (8,10) and (12, 14), respectively. The high-frequency letter 'A' will be created at a high level to construct the final tree will connect both children in one parent with a frequency of 100.

For the output code in Step 7, each leaf has a path to reach the main node (100), so the paths' numbers and direction refer to these values. For example, the code for 'C' is 010, from following the path from 'C' to 100. For 'A', the code is 1, as it has only one path to the main node. After completing the Huffman tree to the text, we obtain (188 bits) while the text in ASCII code is represented by 800 bits (100 characters × 8 bits), as shown in Table II.

Huffman: 56*1+14*3+12*3+10*3+8*3=188 bits

ASCII: (56+14+12+10+8)*8=800 bits

TABLE I. CONSTRUCTION OF THE HUFFMAN TREE

Step 1	Symbol	**B**	**E**	**A**	**D**	**C**
	Frequency	14	8	56	10	12
2	Symbol	E	D	C	B	A
	Frequency	8	10	12	14	56
3.1	Symbol	ED	C	B	A	
	Frequency	18	12	14	56	
3.2	Symbol	ED	CB	A		
	Frequency	18	26	56		
3.3	Symbol	EDCB	A			
	Frequency	44	56			
3.4	Symbol	EDCBA				
	Frequency	100				

TABLE II. THE SEQUENCE OF SYMBOLS AND CODES USING THE HUFFMAN TREE

Symbols	**Code**	**Length**	**Frequency**
A	1	1	56
B	011	3	14
C	010	3	12
D	001	3	10
E	000	3	8

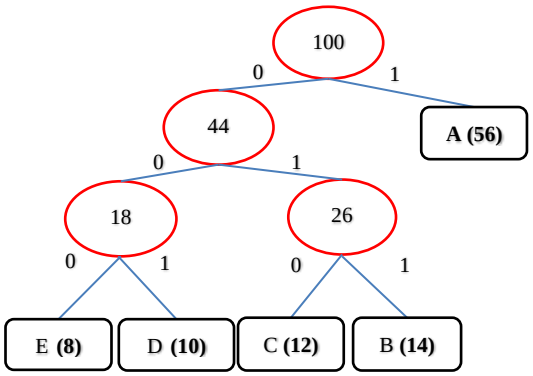

Fig. 2. Huffman Tree.

Therefore, 76.5% of space can be saved using Huffman coding.

2) Image preparation: The cover-image is the original image that is selected to host the secret messages before embedding. RGB colour images or grayscale images are used as a cover-medium in this research. The images can be downloaded from http://sipi.usc.edu/database/database.php?volume=misc (USC-SIPI) [32] with different sizes as a data set. The size of the cover-images is 512*512 pixels to hide the secret messages. Lena.bmp, Baboon.bmp, Pepper.bmp, Airplane.bmp, Camera.bmp, Barbara.bmp, Tiffany.bmp, and Tree.bmp are used to cover-images.

B. Second Stage: Proposed Embedding Process

1) Pixel selections: The embedding method deals with the process of selecting and preparing pixels for embedding. Fig. 3 shows an example of pixel selections. The text message represents the secret message, and it will be encoded with the Huffman algorithm. The encoded message will be embedded into a cover-image. In this example, Lena.bmp is chosen as a cover-image. The embedding process starts at the top left of the row and column (the x and y) position of Lena.bmp. The pixel position of *x* and *y* is compared to determine whether to embed using the XNOR operation or using the Fibonacci sequence. If the *x* value is higher than the *y* value, the Fibonacci sequence is used to embed it; otherwise, the XNOR operation is applied. Whereas, in the case where the value of *x* is equal to *y*, the embedding is skipped. Therefore, half of the pixels use the XNOR operation while the other half use the Fibonacci algorithm. After all secret messages are embedded, the output will be a stego-image.

2) XNOR operation: In embedding with XNOR, the idea is to use the last bit of MSB of the red channel as a secret key, which is agreed by the sender and recipient. The red channel will specify which channel (either green or blue) that the bit of the secret text message will be hidden inside it. The red channel is chosen as the secret key, while the embedding process happens in either green and blue. The reason is that human eyes are most sensitive to red, followed by green and blue [40-41], and therefore we avoid embedding secret data in the red channel.

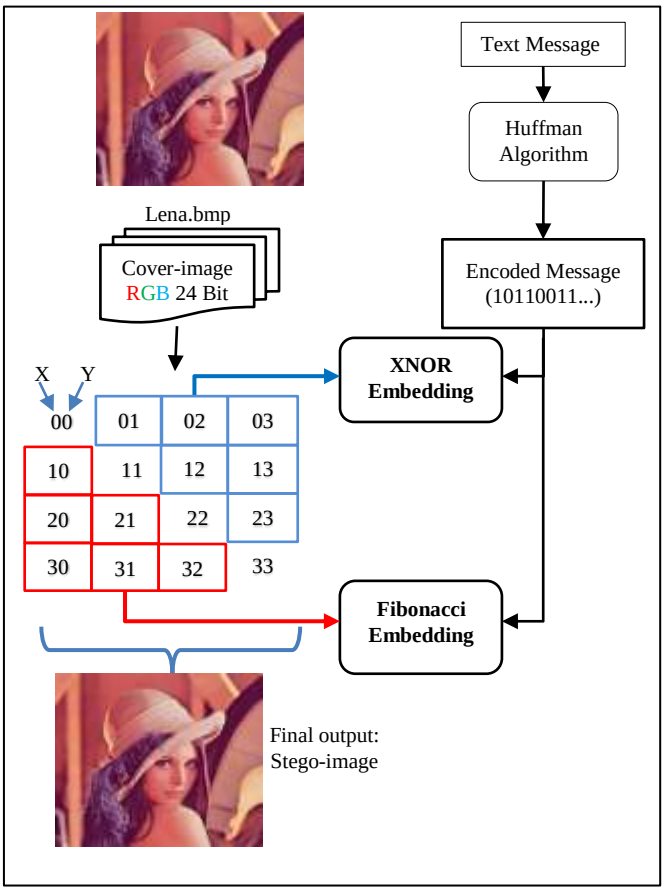

Fig. 3. Pixel Selections Flowchart.

The algorithm for pixel selections is shown in Fig. 4.

Algorithm 1: Pixel selections for embedding

Input:
Image Pixel points (x, y)
Output:
Selected Points for embedding

Steps:
1. Image size = 512*512
2. For x, y in a range of (image_size)
3. Select the points of x as a key
4. If (x > y) **then**
   Apply Fibonacci Embedding
5. If (x < y) **then**
   Apply XNOR Embedding

Fig. 4. Pixel Selections based on Values of *x* and *y*.

There are two processes inside the embedding stage, which are channel selections and data insertion. Firstly, in the channel selections for embedding, the last bit of the red channel's LSB is specified, either '0' (to select BLUE channel) or '1' (to select GREEN channel). Secondly, in the data insertion, it performs the XNOR operation of the last MSB red channel with one bit of the secret text. Finally, the result (0 or 1) will be carried to the selected channel and replace it with LSB's first bit in the cover-image (as shown in Table III). Fig. 5 and Fig. 6 provides an example of embedding using the XNOR operation.

TABLE III. INPUT, OUTPUT AND CHANNEL SELECTIONS

XNOR operation			First bit LSB for channel selection
Last bit of MSB red channel of the cover-image	Secret bit	Result	
0	0	1	BLUE
0	1	0	BLUE
1	0	0	GREEN
1	1	1	GREEN

Fig. 5 explains an example of embedding one bit of secret data using the XNOR operation inside the Green channel.

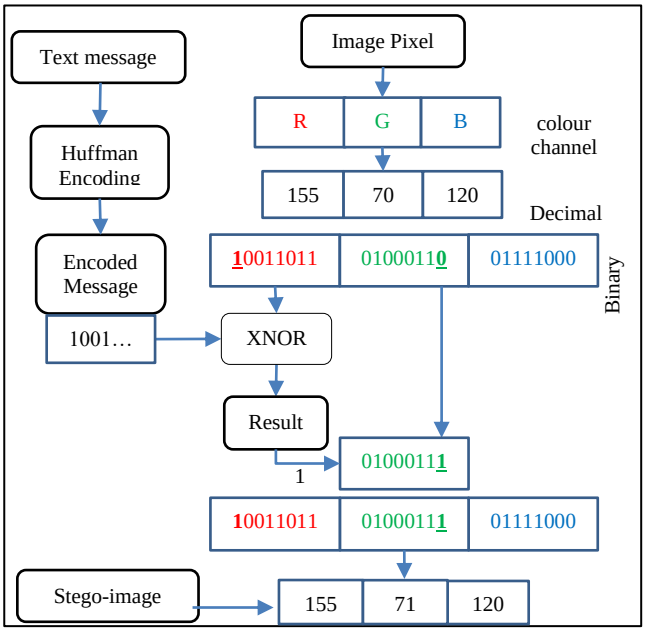

Fig. 5. Embedding One Bit with XNOR Operation Inside Green Channel.

The text message in Fig. 5 represents the secret message, which is encoded with the Huffman algorithm. Consider an image pixel with Red, Green, and Blue colour channels, which has decimal values of 155, 70, and 120, respectively. These decimal values are converted to binary values. Consider also that the encoded message starts with 1001 and so on.

The MSB of the red channel, which is '1', will be XNORed with the secret message bit '1', and the results, in this case, is also '1'. Therefore, this result will be embedded in the green channel, according to Table III. As a result, the final decimal values has been changed to 155, 71, and 120, respectively.

On the other hand, Fig. 6 has '0' for the MSB red channel, XNORed with '0' of the secret bit, and the result is '1'. Therefore, it will be embedded in the blue channel. The decimal values have changed to 27, 70, and 121, respectively.

The embedding process using the XNOR operation is summarized in Fig. 7.

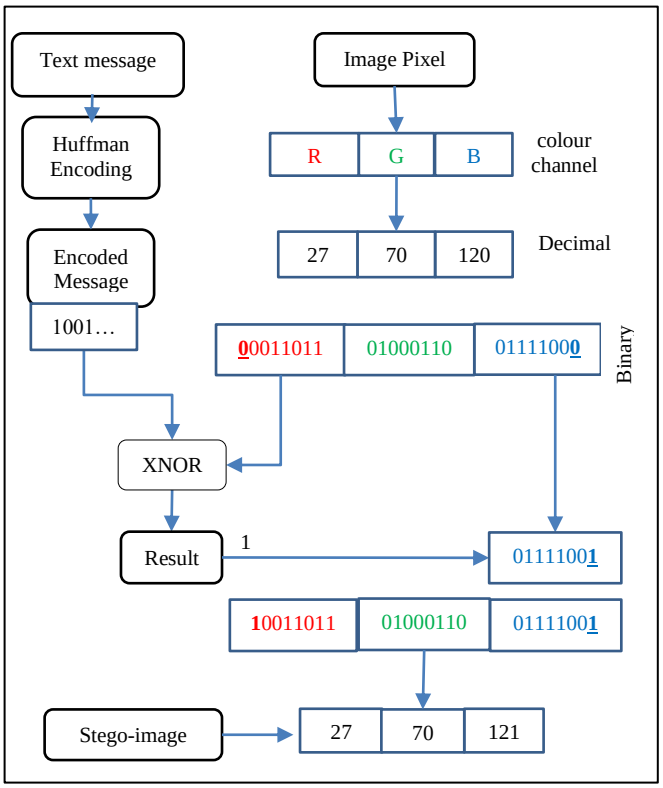

Fig. 6. Embedding One Bit with XNOR Operation Inside Blue Channel.

Algorithm 2: XNOR Image Embedding

Input:
Image Pixels of RGB:
$R = \{P_{r1}, P_{r2}, \ldots, P_{rn}\}$,
$G = \{P_{g1}, P_{g2}, \ldots, P_{gn}\}$
$B = \{P_{b1}, P_{b2}, \ldots, P_{bn}\}$
Message (Mx)
Output: Stego-image

Steps:
1. Apply Huffman encoding on the message $H(Mx)$
2. Get the binary representation of the encoded message $H(Mx)=\{b_{m1}, b_{m2}, \ldots, b_{mh}\}$
3. Get Binary values of each Red, Green and Blue pixels
   $P_{ri} = \{b_{r1}, b_{r2}, \ldots, b_{r8}\}$,
   $P_{gi} = \{b_{g1}, b_{g2}, \ldots, b_{g8}\}$,
   $P_{bi} = \{b_{b1}, b_{b2}, \ldots, b_{b8}\}$
4. Get last bit **MSB** of Red array (br8)
5. Get first bit **LSB** of Green and Blue arrays (bg1,bb1)
6. Get first bit of the message binary array (bm1)
7. **If** (br8) == 1 **then**
   (br8) **XNOR** (bm1) == (bg_1)
8. **If** (br8) == 0 **then**
   (br8) **XNOR** (bm1) == (b_{b1})
9. Initiate stego-image with the new values of RGB pixels

Fig. 7. Embedding Algorithm of the XNOR.

3) Fibonacci decomposition: Fibonacci numbers are a sequence of numbers that begin with zero or one, and then the next value is the summation of the two previous numbers [35]. In this research, we convert pixel values from binary into Fibonacci representation for the image. Binary consists of 8-bit planes, and the pixel values will occupy these bit. However, the bit planes with Fibonacci decomposition consists of 12-bit planes, therefore manipulating the LSB of Fibonacci is more flexible and efficient [26], as shown in Fig. 8.

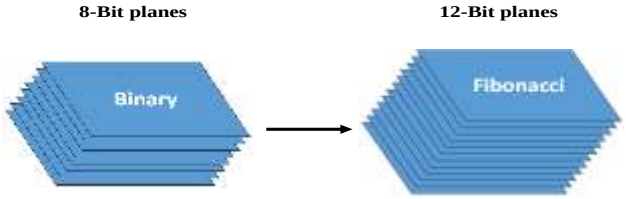

Fig. 8. Bit Plane Distribution in Binary and Fibonacci Representation.

In the 13th century, Leonardo of Pisa introduced the classical Fibonacci number [36]. The general definition of the Fibonacci sequence can be explained by using equation (1).

$$F_{(N)} = F_{(N-1)} + F_{(N-2)} \quad (1)$$

Where $F_0 = 0$, $F_1 = 1$, and $F_2 = 1$, N Fibonacci number. The sequence of Fibonacci is as follows:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,… and so on; such that any numeric value can be represented as binary representation [35]–[37]. Table IV distinguishes between two representations, Binary and Fibonacci for 8-bit and 12-bit planes. In the binary representation, a large range of numbers is available for 8 bits that are [0-255], whereas Fibonacci representation provides the same range [0-255] to represent 12 bits, like the following:

- Binary representation for 255=128 +127 = 11111111
- Fibonacci binary representation for 255 = 233+21+1 = 100001000001.

It is important to note that the binary representation does not introduce redundancy. On the other hand, the Fibonacci representation is redundant, which means that different sequences may represent a single number. For instance, using the Fibonacci representation, a number 70 can have different representations such as the following:

1) 55+13+2
2) 34+21+13+2
3) 55+8+5+2
4) 34+21+8+5+2

Thus, it could be represented as follows (shown in Table V):

In order to obtain the unique representation of the given number, the Zeckendorf theorem can be applied, which states that "each positive integer m can be represented as the sum of distinct numbers in the sequence of Fibonacci numbers, using no two consecutive Fibonacci numbers" [38]. Consequently, any positive number can be represented using equation (2) [25].

$$\alpha_F = \alpha_0 + \alpha_1 F^1 + \alpha_2 F^2 + \cdots = \sum_{i=0}^{N} b_i F^I \quad (2)$$

As can be seen in Table V, there are four representations for number 70. However, only one representation should be selected. Thus, to choose only one of them, the one with the lexicographically lowest value of 1 will always be selected. In the case of number 70, the one which has a minimum number of '1' will be selected as the Fibonacci representation, which is the first rows of Table V (which has three '1'). Thus, a sequence produced using Fibonacci numbers would be valid if there are *no two repeating '1'* in a sequence. Therefore, the probabilities for the first two bits LSBs of a cover-image pixel in Fibonacci representation are '00', '01', or '10' with no repeating of '1'. For embedding purposes, one bit from the secret message (either 0 or 1) can be hidden in the first bit of Fibonacci LSB in the cover-image. Depending on these probabilities, our mapping is proposed, as illustrated in Table VI.

TABLE IV. BIT PLANE REPRESENTATION IN BINARY AND FIBONACCI

Bit plane	Binary	Fibonacci
1	1	1
2	2	2
3	4	3
4	8	5
5	16	8
6	32	13
7	64	21
8	128	34
9	-	55
10	-	89
11	-	144
12	-	233

TABLE V. REPRESENTATION NUMBER 70 IN FIBONACCI

Number	Fibonacci	Binary
70	0*1+1*2+0*3+0*5+0*8+1*13+0*21+0*34+1*55+0*89+0*144+0*233	010001001000
	0*1+1*2+0*3+1*5+1*8+0*13+0*21+0*34+1*55+0*89+0*144+0*233	010110001000
	0*1+1*2+0*3+1*5+1*8+0*13+1*21+1*34+0*55+0*89+0*144+0*233	010110110000
	0*1+1*2+0*3+0*5+0*8+1*13+1*21+1*34+0*55+0*89+0*144+0*233	010001110000

TABLE VI. MAPPING ALGORITHM FOR FIBONACCI

Secret bits / Cover bits	0	1
00	00	01
01	00	01
10	10	01

In the proposed method, the idea of embedding using Fibonacci is by using the green channel of each pixel for the cover-image as a secret key. The secret bit will be inserted to the first bit of the LSB of the green channel, according to Table VI. For example, if the first two bits from the cover-image is '00', and the secret bit is '0', then the stego-image bit is not changed, which is '00'. Otherwise, the stego-image is changed to '01'. Only the green channel is selected to make it harder for the attacker to guess the embedding channel.

Another example is shown in Fig. 9, with the cover-image in the Fibonacci code is represented as 000100100010 in the Green channel. The secret bit can be either '0' or '1'. According to the mapping in Table VI, if the secret bit is '1', and the first two bits is '10', the result after embedding will become '01', which means that the Fibonacci code now becomes 000100100011.

However, this Fibonacci code is not valid according to the Zeckendorf theorem because it has two consecutive '1'. Therefore, a condition is given where the second bit is changed to 0, to avoid the consecutive ones. The final result will be 00010010000**1**. Meanwhile, if the secret bit is '0', and the first two bits is '10', the green bits on the cover-image is not changed. If there is no change to the cover-image, it means less distortion on the stego-image after the embedding process, which results in a high level of security for the hidden message.

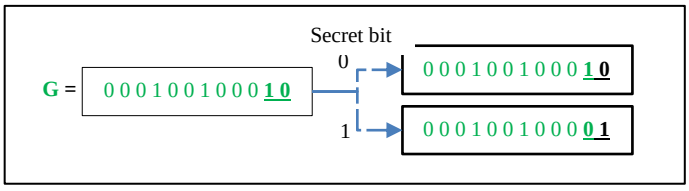

Fig. 9. An Example of Fibonacci Pixel Embedding in the Proposed Method.

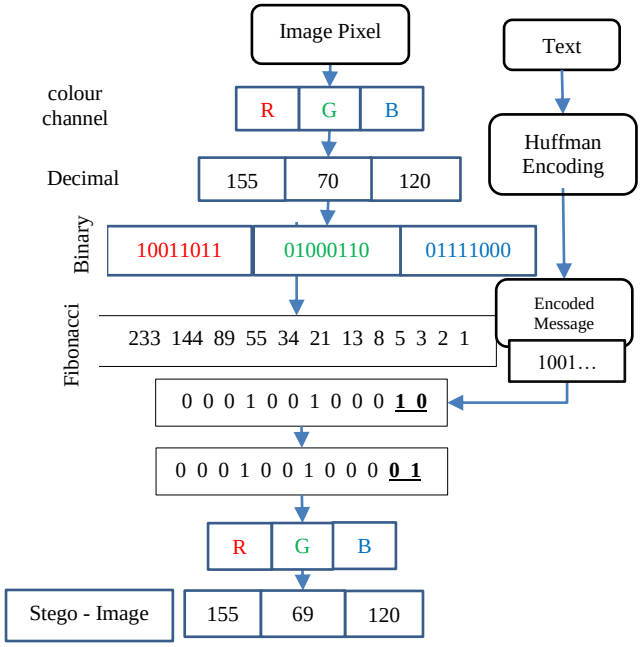

Fig. 10. Embedding One bit in the Fibonacci Algorithm.

Fig. 10 explains another example of the Fibonacci embedding process. Similar to the previous example, we choose 155, 70, and 120 as the values for Red, Green, and Blue channels. The Fibonacci value for 70 is given as a sequence, as well as its binary representation, which is 000100100010. The first two bits of the Fibonacci representation is '10' (Bold and underlined), and the secret message is '1'. Therefore, the embedding results, according to Table VI, is 000100100011, but this representation is not valid according to the Zekendorf theorem. So, we change the second bit to '0': 000100100001, which is also representing 69 in decimal value.

The algorithm for the embedding process is summarized in Fig. 11.

Algorithm 3: Image Embedding using Fibonacci

Input:
Image Pixels of RGB where:
R ={P_{r1},P_{r2} …,P_{rn}
G ={P_{g1},P_{g2}, …,P_{gn}}
B ={P_{b1},P_{b2}, …,P_{bn}}
Message(Mx)

Output: Stego-image

Steps:

1. Apply Huffman encoding on the message $H(Mx)$
2. Get the binary representation of the encoded message $H(Mx)$= {b_{m1}, b_{m2}, …, b_{mh}}
3. Select the Green pixels G = {P_{g1}, P_{g2}, …, P_{g3}}
4. Get the Binary values of each Green pixel P_{gi} = {b_1, b_2, …, b_8}
5. Transform the binary array of Green pixel into Fibonacci representation $Fib(P_{gi})$ = {b_{f1}, b_{f2}, …, b_{f11}, b_{f12}}
6. Get the first two bits LSB of green pixel Fibonacci (b_{f1}, b_{f2})
7. Get the first bit of the encoded message (b_{m1})
   - **7.1** **If** b_{f1}=0 && b_{f2}=0 && b_{m1}=0 b_{f1}=0, b_{f2}=0
   - **7.2** **Else If** b_{f1}=0 && b_{f2}=0 && b_{m1}=1 b_{f1}=1, b_{f2} = 0
   - **7.3** **Else If** b_{f1}=1 && b_{f2}=0 && b_{m1}=0 b_{f1}=0, b_{f2}=0
   - **7.4** **Else If** b_{f1}=1 && b_{f2}=0 && b_{m1}=1 b_{f1} = 1, b_{f2} = 0
   - **7.5** **Else If** b_{f1}=0 && b_{f2}=1 && b_{m1}=0 b_{f1} = 0, b_{f2} = 1
   - **7.6** **Else** b_{f1} = 1, b_{f2} = 0
8. Get the decimal value of the new green pixel Fibonacci array $Fib(P_{gi})$ = {**NEW**b_{f1}, **NEW**b_{f2},…, b_{f11}, b_{f12}} ➔ D
9. Initiate the stego-image with the new values of RGB pixels

Fig. 11. Embedding Process using Fibonacci.

C. Third Stage: Performance Evaluation

Any steganography system aims to build a secure communication system that cannot be detected by a third party. The attacker can use statistical methods to identify the stego-image such as histogram analysis and chi-square attack, even if the cover-image is unknown [26],[32]. As mentioned before,

after the embedding stage, the image produced is called stego-image. The PSNR values are used to evaluate the quality of this stego-image, and the desired value for PSNR is as similar as possible with the cover-image. When the value of PSNR is high, that indicates the proposed method does not damage or distort the cover-image. The PSNR value is calculated like the following equation (3).

$$PSNR = 10 \log_{10} \frac{(255)^2}{MSE} \quad (3)$$

Equation (4) is used to calculate the mean squared error. MSE is calculated by obtaining the average square error. Then its result will be used to calculate PSNR to evaluate the resolution and quality of the stego-image.

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - y_{ij})^2 \quad (4)$$

Here, *m* and *n* are the images sizes while *x* and *y* are the cover and stego-images, respectively.

D. Fourth Stage: Extracting Stage

The information hiding process always follows specific procedures, which are an embedding method, and the main steps in this method with information of the embedding process should also be known to the receiver. The receiver uses the secret keys, which contain the embedding information. In order to extract from the secret message embedded in a stego-image, the same procedure for embedding is used but in a reverse manner. The pixel selection process summarized in Fig. 12.

Algorithm 4: Pixel Selections for Extracting Process

Input:
Stego-Image Pixel points (x, y)
Output:
Selected Points for extracting

Steps:
1 Image size = 512*512
2 For x,y in a range of (image_size)
3 Select the points of x as a key
4 If (x > y) **then**
Apply Fibonacci Extracting
5 If (x < y) **then**
Apply XNOR Extracting

Fig. 12. Pixel Selection for the Extracting Process.

Fig. 13 and Fig. 14 explains the extracting processes for XNOR and Fibonacci, respectively.

Algorithm 5: Image Extracting using XNOR

Input: Stego-image Pixels of RGB
R ={P_{r1},P_{r2},…,P_{rn}}
G ={P_{g1},P_{g2},…,P_{gn}}
B ={P_{b1},P_{b2},…,P_{bn}}

Output: Message (*Mx)*

Steps:
1. Initialize an empty array of bits *Ex*
2. Get Binary values of each Red, Green and Blue pixels P_{ri} ={b_{r1},b_{r2},…,b_{r8}}, P_{gi} ={b_{g1},b_{g2},…,b_{g8}}, P_{bi} ={b_{b1},b_{b2},…,b_{b8}}
3. Get last bit MSB of Red array (br8)
4. Get first bit LSB of Green and Blue arrays (bg1,bb1)
5. **If** (br8) == 1
(br8) XNOR (bg1) == (bm1) **then**
Add (bg1) to H(Mx)
6. **If** (br8) == 0 && (bb1)== 0
(br8) XNOR (bb1) == (bm1) then
Add (1) to H(Mx)
7. **If** (br8) == 0 && (bb1)== 1
(br8) XNOR (bb1) == (bm1) then
Add (br8) to H(Mx)
8. Get binary representation of the encode message H(Mx) = {bm1, bm2, …, bmh}
9. Apply Huffman to decode H(Mx)

Fig. 13. Extracting Process with XNOR Operation.

Algorithm 6: Image Extracting using Fibonacci

Input:
Image Pixels of RGB where:
R = {P_{r1},P_{r2},…,P_{rn}}
G = {P_{g1},P_{g2},…,P_{gn}}
B = {P_{b1},P_{b2},…,P_{bn}}

Output: Message Mx

Steps:
1. Initialize an empty array of bits Ex
2. Select the Green pixels, G={Pg1,Pg2,…,Pgn}
3. Get the Binary values of each Green pixel Pgi = {b1,b2,…,b8}
4. Transform the binary array of Green pixel into Fibonacci representation Fib(Pgi) = {bf1, bf2, …, bf11, bf12}
5. Get the first two LSB bits of green pixel Fibonacci (bf1, bf2)
6. If (bf1 = 1 | bf2 = 0)
add 1 to H(Mx)
7. Else:
add 0 to H(Mx)
8. Get the binary representation of the encoded message H(Mx)= {bm1, bm2 ,…, bmh}
9. Apply Huffman to decode H(Mx)

Fig. 14. Extracting Process with the Fibonacci Algorithm.

IV. Experimental Results

A. Dataset

A dataset is defined as a group of images that is used to benchmark the observation of this research with the existing literature. The study follows the same strategy to evaluate and benchmark the experimental results obtained using the proposed method. RGB colour images with size 512*512 have been used as benchmarks in the evaluation stage. The following images are selected as cover-images, as shown in Fig. 15.

B. PSNR Comparison and Analysis

The PSNR is an expression for the ratio between the maximum possible value of a signal and the distorting noise that affects its representation quality. If the PSNR values are greater than 50 dB when tested on several images, the images are high in quality [37]. This type of evaluation is used to measure the quality of the image after embedding. PSNR is most commonly used to measure the quality of reconstruction of lossy presentation codecs (e.g., for image hiding). In this case, the signal is the original data, and the noise is the error introduced by the embedding process. When comparing image representation codecs, PSNR approximates the human perception of reconstruction quality. Although a higher PSNR generally indicates that the reconstruction is of higher quality, in some cases, it may not [38]. PSNR is most easily defined via the mean squared error (MSE), as mentioned before, which is based on the dimensions of the image.

The MSE is calculated by obtaining the average square error. Then its result will be used to calculate PSNR to evaluate the resolution and quality of the stego-image, as shown in Table VII.

Table VIII shows the comparison stages of the results with four other methods in [30], [32], [26], and [33], with 8Kb, 10Kb, 13Kb, and 16Kb of secret messages, and 512*512 size of cover images. We note that the bigger the secret messages' size, the lower the PSNR value as more significant data will cause more distortion to the stego-image. From the results, high values of PSNR are obtained compared with previous findings. This indicates that the proposed method is very efficient in hiding data, which means that this technique can keep changes to the stego-image to a minimum. Therefore, we can conclude that this technique has a good quality of imperceptibility.

Fig. 15. Dataset used as Cover-Images.

TABLE VII. PSNR and MSE Values for Stego-Images

Cover- image	Secret Message 8 KB		Secret Message 16 KB	
	MSE	PSNR	MSE	PSNR
Lena.bmp	0.002477	74.191525	0.014170	66.6170
Camera.bmp	0.006153	70.239859	0.019850	65.1531
Babbon.bmp	0.004360	71.735740	0.016741	65.8928
Airplane.bmp	0.004323	71.772626	0.016536	65.9463
Tiffany.bmp	0.004240	71.856456	0.016924	65.8456
Peppers.bmp	0.004189	71.908861	0.016565	65.9386
Tree.bmp	0.004057	72.048147	0.017345	65.7389
Barbara.bmp	0.003378	72.843499	0.016370	65.9902

TABLE VIII. Comparison of PSNR and MSE Values in (dB) with the Literature

Methods	Size of Secret Message (Kb)	PSNR		
		Lena	Baboon	Pepper
[26]	12.79	51.045	51.997	49.442
[32]	10	62.32	62.29	62.22
[33]	8	57.411	-	57.442
[30]	2.96	53.65	40.89	39.99
Proposed Method	8	74.192	71.736	71.909
	10	72.750	69.927	69.971
	13	70.301	67.64012	67.681
	16	66.617	65.893	65.939

C. Histogram Analysis

One way to discover a good steganography method is to analyze the histogram of all stego-images and then compare them with the original. It represents the number of pixels that have colours in the image's colour space. The histogram for the original cover-image and stego-image for Lena and baboon, with 16KB size of secret messages embedded, are shown in Fig. 16 to Fig. 19, respectively.

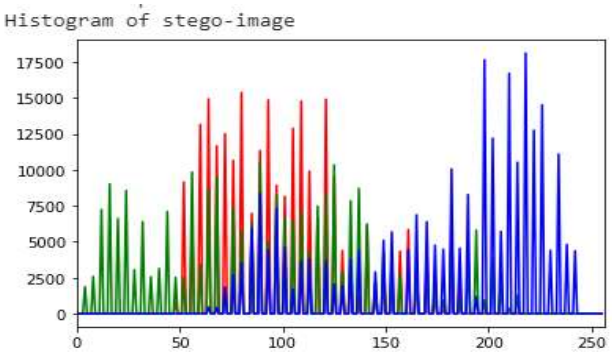

Fig. 16. Histogram of Cover-Image Lena.bmp.

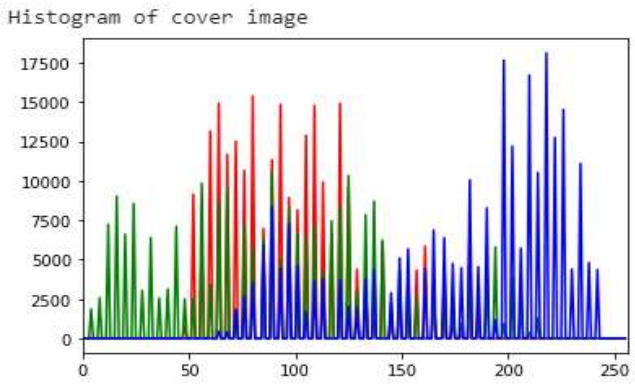

Fig. 17. Histogram of Lena.bmp Stego-Image.

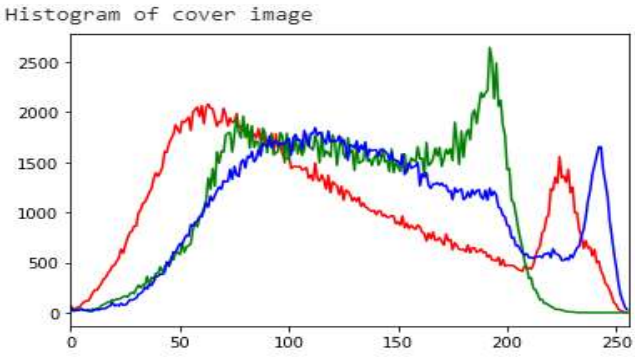

Fig. 18. Histogram of Cover-Image Baboon.bmp.

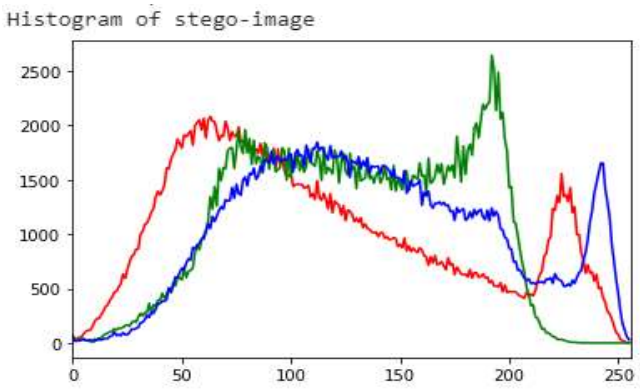

Fig. 19. Histogram of Baboon.bmp Stego-Image.

From the figures of Lena's and Baboon's stego-images, the histogram of the stego-images was similar to that of the cover images, resulting from alternately choosing either green or blue channels to embed the secret data, which does not give any significant difference between the two images. This means that the proposed technique can embed secret messages with minimum changes in the stego-image.

D. Embedding Capacity

The embedding capacity or ratio capacity depends on how much secret data bits can be hidden in the cover-image. It is calculated using equation (5).

$$EC = \frac{number\ of\ secret\ mesage\ bits}{number\ of\ cover\ image\ pixels}\ (bpp) \quad (5)$$

Table IX shows the comparison of the hiding capacity between our proposed method and four other methods. The results show that the proposed method yields similar results to the other methods in terms of capacity.

TABLE IX. COMPARISON OF HIDING CAPACITY

Methods	Capacity (Kb)	Ratio capacity
[30]	2.9	0.093
[33]	8	0.25
[32]	10	0.313
[26]	12.79	0.399
Proposed Method	**8**	**0.250**
	10	**0.313**
	13	**0.406**
	16	**0.501**

V. CONCLUSION

The standard LSB method is the most popular steganography technique, as it is more efficient to use. However, this method's main weakness is that it is easy to recover the secret text message from the image, which is always hidden in the LSB of each pixel in an image. The developed XNOR operation with the Fibonacci sequence presents several characteristics that have enhanced the LSB standard technique's limitation. In terms of security level, there are three criteria integrated for higher security. Firstly, the Fibonacci algorithm uses only the green channel for embedding instead of the three channels red, green, and blue. The advantage of this is that the red and blue channels will act as noise data, which makes the extraction process harder for any intruder. Secondly, similar to the Fibonacci embedding operation, the XNOR operation also uses only one channel (the green or blue channels), while the other two channels act as noise data. Lastly, this method exploits the pixel selection to conceal the secret message, either use the Fibonacci algorithm or XNOR operation, to make the extraction process more secure. In this paper, we exploited the characteristics of both the XNOR operation and the Fibonacci algorithm to obtain high security and capacity to embed the secret message. Another important thing is that using the green or blue channel for each pixel on the cover-image gives an advantage as almost all of the pixels will be exploited to conceal secret data, and thus, capacity will be increased.

VI. FUTURE WORK

For future work, the three channels of RGB can be exploited to use as an indicator for embedding with the XNOR operation. The proposed method used the red channel as the key or supplier for embedding to green or blue channels (that means the red channel that will select which channel for embedding) as well as exploit one channel of RGB with the Fibonacci algorithm. In the proposed method, the green channel is used only for embedding, and that will let the proposed method be applied on three-channel RGB for embedding instead of using only the green channel. Another recommendation is to apply the proposed method with other cover-media types, such as text or audio.

ACKNOWLEDGMENT

This work was supported by Universiti Kebangsaan Malaysia under research grant PP-FTSM-2020.

REFERENCES

[1] Ghosh, A. K. Chattopadhyay, and A. Nag, "A novel approach of image steganography with encoding and location selection," in Advances in Intelligent Systems and Computing, 2019, doi: 10.1007/978-981-13-1544-2_10.

[2] S. S. Baawi, M. R. Mokhtar, and R. Sulaiman, "New text steganography technique based on a set of two-letter words," Journal of Theoretical and Applied Information Technology, vol. 95, no. 22, pp. 6247–6255, 2017.

[3] M. A. Majeed and R. Sulaiman, "An improved LSB image steganography technique using bit-inverse in 24 bit colour image," Journal of Theoretical and Applied Information Technology, vol. 80, no. 2, pp. 342–348, 2015.

[4] S. Kamil, M. A. Authors, S. N. H. S. Abdullah, and Z. Ahmad, "Lightweight and optimized multi-layer data hiding using video steganography paper," International Journal of Advanced Computer Science and Applications, vol. 9, no. 12, pp. 256–262, 2018, doi: 10.14569/IJACSA.2018.091237.

[5] S. Kamil, M. Ayob, S. N. H. Sheikh Abdullah, and Z. Ahmad, "Optimized Data Hiding in Complemented or Non-Complemented Form in Video Steganography," Proceedings of the 2018 Cyber Resilience Conference, CRC 2018, pp. 1–4, 2019, doi: 10.1109/CR.2018.8626871.

[6] 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," Multimedia Tools and Applications, vol. 77, no. 23, pp. 31487–31516, 2018, doi: 10.1007/s11042-018-6213-0.

[7] F. Akhter and M. Selim, "A New Approach of Graph Realization for Data Hiding using Human Encoding," International Journal of Advanced Computer Science and Applications, vol. 7, no. 12, pp. 436–442, 2016, doi: 10.14569/ijacsa.2016.071256.

[8] S. Mavanai, A. Pal, R. Pandey, A. Prof, and D. Nadar, "Message transmission using DNA crypto-system," International Journal of Computer Science and Mobile Computing, vol. 8, no. 4, pp. 108–114, 2019.

[9] M. Cui and Y. Zhang, "Incorporating randomness into DNA steganography to realize secondary secret key, self-destruction , and quantum key distribution-like function," pp. 1–19, 2019, doi: https://doi.org/10.1101/725499.

[10] R. A. Watheq, F. Almasalha, and M. H. Qutqut, "A new steganography technique using JPEG images," International Journal of Advanced Computer Science and Applications, vol. 9, no. 11, pp. 751–760, 2018, doi: 10.14569/ijacsa.2018.0911107.

[11] A. Pradhan, K. R. Sekhar, and G. Swain, "Digital image steganography using LSB substitution, PVD, and EMD," Mathematical Problems in Engineering, vol. 2018, 2018, doi: 10.1155/2018/1804953.

[12] S. Jeevitha and N. Amutha Prabha, "A comprehensive review on steganographic techniques and implementation," ARPN Journal of Engineering and Applied Sciences, vol. 13, no. 17, pp. 4780–4791, 2018.

[13] R. Gupta, S. Gupta, and A. Singhal, "Importance and techniques of information hiding : A review," International Journal of Computer Trends and Technology, vol. 9, no. 5, pp. 260–265, 2014, doi: 10.14445/22312803/ijctt-v9p149.

[14] M. S. Subhedar and V. H. Mankar, "Current status and key issues in image steganography: A survey," Computer Science Review, vol. 13–14, pp. 95–113, 2014, doi: 10.1016/j.cosrev.2014.09.001.

[15] E. Satir and H. Isik, "A Huffman compression based text steganography method," Multimedia Tools and Applications, vol. 70, no. 3, pp. 2085–2110, 2014, doi: 10.1007/s11042-012-1223-9.

[16] A. Gutub and N. Al-juaid, "Multi-bits stego-system for hiding text in multimedia images based on user security priority," Journal of Computer Hardware Engineering, vol. 1, no. April, pp. 1–9, 2018, doi: 10.63019/jche.v1i2.513.

[17] G. V. K. Murugan and R. Uthandipalayam Subramaniyam, "Performance analysis of image steganography using wavelet transform for safe and secured transaction," Multimedia Tools and Applications, vol. 79, no. 13–14, pp. 9101–9115, 2019, doi: 10.1007/s11042-019-7507-6.

[18] T. Rabie, M. Baziyad, and I. Kamel, "Enhanced high capacity image steganography using discrete wavelet transform and the Laplacian pyramid," Multimedia Tools and Applications, vol. 77, no. 18, pp. 23673–23698, 2018, doi: 10.1007/s11042-018-5713-2.

[19] Y. Yeung, W. Lu, Y. Xue, J. Huang, and Y.-Q. Shi, "Secure binary image steganography with distortion measurement based on prediction," IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 5, pp. 1423-1434, 2019, doi: 10.1109/tcsvt.2019.2903432.

[20] D. Laishram and T. Tuithung, "A survey on digital image steganography: current trends and challenges," Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT), 2018, Malaviya National Institute of Technology, Jaipur (India), March 26-27, 2018.

[21] N. Akhtar, "An LSB substitution with bit inversion steganography method," Springer India 2016. Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics, Smart Innovation, Systems and Technologies, vol. 43, pp. 515–521, 2016, doi: DOI 10.1007/978-81-322-2538-6_53.

[22] S. Das, S. Sharma, S. Bakshi, and I. Mukherjee, "A framework for pixel intensity modulation based image steganography," Advances in Intelligent Systems and Computing, vol. 563, pp. 3–14, 2018, doi: 10.1007/978-981-10-6872-0_1.

[23] A. Setyono and D. R. I. M. Setiadi, "Securing and hiding secret message in image using XOR transposition encryption and lsb method," Journal of Physics: Conference Series, vol. 1196, no. 1, 2019, doi: 10.1088/1742-6596/1196/1/012039.

[24] D. Rawat and V. Bhandari, "A steganography technique for hiding image in an image using LSB method for 24 bit colour image," International Journal of Computer Applications, vol. 64, no. 20, pp. 15–19, 2013, doi: 10.5120/10749-5625.

[25] K. Joshi, R. Yadav, and G. Chawla, "an enhanced method for data hiding using 2-bit XOR in image steganography," International Journal of Engineering and Technology, vol. 8, no. 6, pp. 3043–3055, 2017, doi: 10.21817/ijet/2016/v8i6/160806266.

[26] A. Rehman, T. Saba, T. Mahmood, Z. Mehmood, M. Shah, and A. Anjum, "Data hiding technique in steganography for information security using number theory," Journal of Information Science, vol. 45, no. 6, pp. 767–778, 2019, doi: 10.1177/0165551518816303.

[27] S. Maurya and V. Shrivastava, "An improved novel steganographic technique for RGB and YCbCr colourspace," IOSR Journal of Computer Engineering, vol. 16, no. 2, pp. 155–157, 2014.

[28] Y. G. Yang, L. Zou, Y. H. Zhou, and W. M. Shi, "Visually meaningful encryption for colour images by using Qi hyper-chaotic system and singular value decomposition in YCbCr colour space," Optik, vol. 213, p. 164422, 2020, doi: 10.1016/j.ijleo.2020.164422.

[29] S. Dagar, "Highly randomized image steganography using secret keys," International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2014, 2014, doi: 10.1109/ICRAIE.2014.6909116.

[30] R. M. Neamah, J. A. Abed, and E. A. Abbood, "Hide text depending on the three channels of pixels in colour images using the modified LSB algorithm," International Journal of Electrical and Computer Engineering, vol. 10, no. 1, pp. 809–815, 2020, doi: 10.11591/ijece.v10i1.pp809-815.

[31] G. S. Charan, S. S. V. Nithin Kumar, B. Karthikeyan, V. Vaithiyanathan, and K. Divya Lakshmi, "A novel LSB based image steganography with multi-level encryption," ICIIECS 2015 - 2015 IEEE International Conference on Innovations in Information, Embedded and Communication Systems, pp. 1–5, 2015, doi: 10.1109/ICIIECS.2015.7192867.

[32] Z. S. Younus and M. K. Hussain, "Image steganography using exploiting modification direction for compressed encrypted data," Journal of King Saud University - Computer and Information Sciences, 2019, doi: 10.1016/j.jksuci.2019.04.008.

[33] K. Muhammad, J. Ahmad, H. Farman, Z. Jan, M. Sajjad, and S. W. Baik, "A secure method for colour image steganography using gray-level modification and multi-level encryption," KSII Transactions on Internet and Information Systems, vol. 9, no. 5, pp. 1938–1962, 2015, doi: 10.3837/tiis.2015.05.022.

[34] A. Nag, S. Biswas, D. Sarkar, and P. P. Sarkar, "A novel technique for image steganography based on DWT and Huffman," International Journal of Computer Science and Security, (IJCSS), vol. 4, no. 6, pp. 73–82, 2013, doi: 10.1017/CBO9781107415324.004.

[35] M. N. Abdulwahed, "An effective and secure digital image steganography scheme using two random function and chaotic map," Journal of Theoretical and Applied Information Technology, vol. 98, no. 1, pp. 78–91, 2020.

[36] A. A. Abdulla, S. A. Jassim, and H. Sellahewa, "Efficient high-capacity steganography technique," Mobile Multimedia/Image Processing, Security, and Applications 2013, vol. 8755, no. February 2019, p. 875508, 2013, doi: 10.1117/12.2018994.

[37] M. Sherif, "StegoCrypt : Geometric and Rudin – Shapiro Sequence – Based Bit – Cycling and 3DES", Bachelor Thesis, 2019.

[38] Aroukatos N.G., Manes K., Zimeras S., "Social networks medical image steganography using sub-Fibonacci sequences," Springer International Publishing Switzerland 2016 A.A. Lazakidou et al. (eds.), mHealth Ecosystems and Social Networks in Healthcare, Annals of Information Systems 20, pp. 171–185, 2016, doi: 10.1007/978-3-319-23341-3.

[39] S. Kamil, M.Ayob, S. N. H. S. Abdullah, and Z. Ahmad, Challenges in multi-layer data security for video steganography revisited, Asia-Pacific Journal of Information Technology and Multimedia (APJITM). pp. 53-62, 2018, doi: dx.doi.org/10.17576/apjitm-2018-0702(02)-05

[40] S. Roy and A. K. Pal, "A blind DCT based color watermarking algorithm for embedding multiple watermarks," AEU - International Journal of Electronics and Communications, vol. 72, pp. 149–161, 2017, doi: 10.1016/j.aeue.2016.12.003.

[41] S. Rajagopala et al., "MSB Based Embedding with Integrity: An Adaptive RGB Stego on FPGA Platform," Information Technology Journal, vol. 13, no. 12, pp. 1945–1952, 2014, doi: 10.3923/itj.2014.1945.1952.