

One time pad for enhanced steganographic security using least significant bit with spiral pattern

Rihartanto¹, Didi Susilo Budi Utomo¹, Ansar Rizal¹, Dwi Agus Diartono², Hery Februariyanti²

¹Department of Information Technology, Politeknik Negeri Samarinda, Samarinda, Indonesia

²Department of Information System, Universitas Stikubank, Semarang, Indonesia

Article Info

Article history:

Received Jan 5, 2024

Revised Apr 12, 2024

Accepted May 12, 2024

Keywords:

Exclusive-OR

Least significant bit

One time pad

Spiral pattern

Steganography

ABSTRACT

Data is an important commodity in today's digital era. Therefore, data needs to get adequate security to prevent misuse. A common data security practice in the transmission of information is cryptography. Another approach is steganography, which hides secret messages in other media that are not confidential and can be accessed by the public. In this study, the spiral pattern is used for data placement using the least significant bit (LSB) method. Modifications were made to the 2-bits LSB to increase the data capacity that can be hidden. In order to increase security, the data is first converted into a datastream using random numbers as one time pad (OTP). exclusive-OR (XOR) operation is performed on datastream and OTP to get encrypted data to be hidden. The results showed that the image quality of the steganography results at a capacity close to 100% was still fairly good, as indicated by a peak signal-to-noise ratio (PSNR) value greater than 46 dB. Visually, the steganographic image does not look different from the original one. Likewise, the use of random numbers as OTP succeeded in changing the hidden data significantly, as indicated by the avalanche effect value above 50%.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Rihartanto

Department of Information Technology, Politeknik Negeri Samarinda

Samarinda, Indonesia

Email: rihart.c@gmail.com

1. INTRODUCTION

The internet has become a staple in today's modern life. Various types of data are transmitted over the internet. Starting from unimportant data, hoax content to important and confidential information traveling on the internet all the time. Depending on the purpose and point of view of its users, the existence of the internet can be used for positive and beneficial things, and can also be used to do negative and harmful things.

Data security is an important issue in internet use. Data security can be interpreted as security from unauthorized users, security to maintain the authenticity of information, or security for other purposes. Methods that are widely used for securing data and information include cryptography and steganography. Cryptography is the process of changing or encoding information so that only the intended recipient of the message can read or know the information. Cryptography can also be interpreted as a method of protecting information and communications through the use of (secret) codes, so that only those who are intended to receive the information able to read and process them.

Steganography is another approach that can be used for data security [1]. In steganography, information is hidden in other media that are not confidential. The information is camouflaged in other media while keeping the media relatively unchanged so that only the intended person is aware of or knows the existence of the confidential information.

Steganography can be defined as an attempt to hide multimedia data (text, image, audio, video or files) into other media of the same or different type but with a larger size [2]. The methods used include least significant bit (LSB) [3]–[5], discrete wavelet transform [6], [7], pixel value differencing [8], [9], spread spectrum [10], [11], invertible neural network [12], as well as other new methods based on deep learning. The LSB method have some variation in its implementation such as k-LSB [13] and LSB matching revisit [14].

The LSB method is one of the most widely used and researched methods. One of the reasons is that LSB is a simple method and easy to implement. For the purpose of increasing data security, this method is often combined with cryptographic techniques [15] exclusive-OR (XOR) gate [16], [17], or XNOR gate [18]. The use of pixel reading patterns such as dual layer LSB-matching [19], adaptive pattern [20], or odd-even patterns [2] aims to complicate steganalysis efforts.

Many studies on steganography have focused on imperceptible criteria and steganographic image quality, but not a few have attempted to increase the capacity of hidden data [5], [9], [12], [21], [22]. This large storage capacity still strives for steganographic image results to remain in good quality. Also, keeping the file size in the storage does not experience a significant increase.

This study uses steganography to hide secret messages encrypted with the one time pad (OTP) approach [16], [17], [23] using random numbers. Data hiding is conducted by modifying the 2-bit of LSB and using a spiral pattern in its placement. The use of 2-bits aims to increase the amount of information that can be hidden while maintaining the quality of the steganographic image. Tests are carried out on steganography results that store the amount of data close to the maximum capacity the cover image can accommodate.

2. RESEARCH METHOD

The methods used in this research can be grouped into four parts. Namely the use of LSB to hide messages, spiral patterns in placing data into the cover, and implementing XOR operations to generate datastreams to be hidden in the cover. Each section also includes appropriate methods for testing research results.

2.1. Steganography uses LSB

In contrast to encryption, where secret messages are secured by changing the message into a form that is different or unrecognizable from its original form. In steganography, secret messages are hidden in other media, so their existence is unrealized. Steganography implementation can be performed on various forms of data, including text, images, audio, or video. Several criteria often used to assess steganography results are imperceptible, fidelity, and recovery. Imperceptible, that is, the existence of a secret message cannot be sensed. It means that the stego-media that contains messages isn't easy to distinguish from the original cover-media, visually or audio-visually. Fidelity means that the quality of the container media does not change significantly after hiding the message, and recovery means that hidden messages can be retrieved correctly.

LSB is a widely used method of hiding messages in the cover. Hidden messages can be in the form of text, images, or other digital data, as well as the cover used as a place to hide the message. The general assumption that is widely used is that the size of the media used as cover is larger than the hidden message. For example, text is hidden in an image, or an image is hidden in audio or video.

For example, the cover used is an image. Steganography with LSB is carried out by modifying the bits in the LSB part for each pixel contained in the cover. These LSB bits are modified by replacing them with the message bits we want to hide. Depending on the need, the number of bits replaced can vary from 1 to 4. The fewer bits replaced, usually the image quality of the steganography results will be better, while the more bits replaced, the larger the message size that can be hidden.

Suppose, the letter K with an ASCII value of 75 will be hidden in white pixels starting from the 20th to the 23rd pixel. Modifications are made to the last 2-bits of each pixel. Figure 1 illustrates the placement of each bit of the letter K into 4 pixels in the cover image. The first two bits of the letter K will be placed in the 20th pixel, the next two bits will be placed in the 21st pixel, and so on, until all the message bits have been hidden. If the hidden data is two bits per pixel, each character in the message will require four pixels on the cover.

$$MSE = \frac{1}{m \times n} \sum_{i=0}^m \sum_{j=0}^n (X_{ij} - X'_{ij})^2 \quad (1)$$

$$PNSR = 10 \log_{10} \frac{I^2}{MSE} \quad (2)$$

LSB, which modifies the last bits in the image pixels, visually looks the same as the original image. However, this method has drawbacks, including in terms of reliability. The LSB method is susceptible to filtering, scaling, rotation, and cropping processes, which can damage hidden messages. The image quality measurement of the steganography results is carried out using the peak signal-to-noise ratio (PSNR). Where to get the PSNR value, the mean squared error (MSE) value is first calculated. MSE is calculated using (1), and PNSR is calculated using (2).

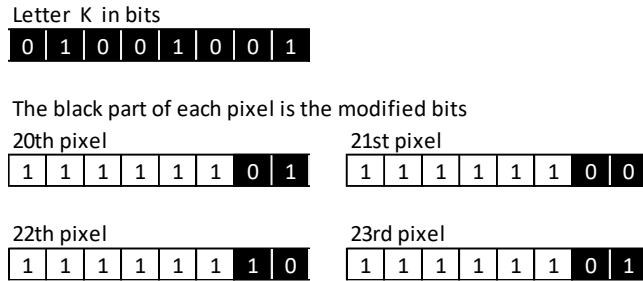


Figure 1. Illustration of hiding the letter K into 4 white pixels

The X_{ij} is the pixel intensity of the i -th row and j -th column of the cover-image, X'_{ij} is the pixel intensity of the i -th row and j -th column of the stego-image, m and n are the number of the row and column of the cover-image, and I is the maximum pixel intensity. For an 8-bit image, $I = 255$. The greater the PSNR (the smaller the MSE), the better the quality of the stego-image. The expected PSNR value is higher or equivalent compared to previous studies [5], [13], [24], [25].

2.2. Spiral pattern

The spiral pattern data placement algorithm that will be built is data placement starting from the center point of the spiral towards the exit and moving away from the spiral point in a clockwise direction. What is considered the midpoint or center point is the position of the element in the middle of the array. A representation of filling an array with a spiral pattern is shown in Figures 2 and 3. The position of the center point of the spiral will relatively adjust to the shape and size of the array. Figures 2(a) and 3(a) are 5x5 square arrays, Figures 2(b) and 3(b) are 5x8 landscape arrays, and Figures 2(c) and 3(c) are 8x6 portrait arrays. These differences in size and shape are aimed at showing the square’s position filled using a spiral pattern according to the size of the given array. The difference between Figures 2 and 3 lies in the direction outward from the center point of the spiral. The initial direction of Figure 2 is rightward, while Figure 3 is downward, then moves spirally in a clockwise direction.

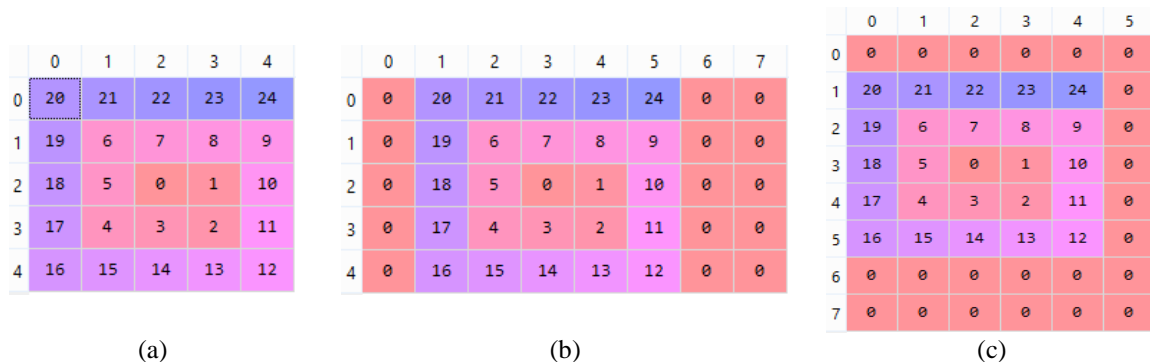


Figure 2. Rigthward array filling with a spiral pattern: (a) 5x5 array, (b) 5x8 array, and (c) 8x6 array

The value 0 in the middle of the array is the center point of the spiral, while the value 0 on the outside is the part of the array that is not filled or whose value has not changed. The direction of filling the array follows the increase in the value filled in for each element, starting from 0, 1, 2, 3, 4, and so on.

The filling direction refers to the first direction when exiting the center point, namely to the right, left, up, or down. At the same time, the part of the array filled with data is part of the array in the square shape with an odd size. This odd size is the maximum length in an odd number that can form a square within the given array. The center point of the spiral [x, y] is obtained from the integer division of the array height and array width by 2. If the height or width of the array is even, then the result of the division is subtracted by 1 to get the center point. Following this approach, Algorithm 1 shows how to determine the square size in the image to be filled. Determination of the center point of the spiral and how to change the X and Y values as the coordinates of the array whose data will be modified.

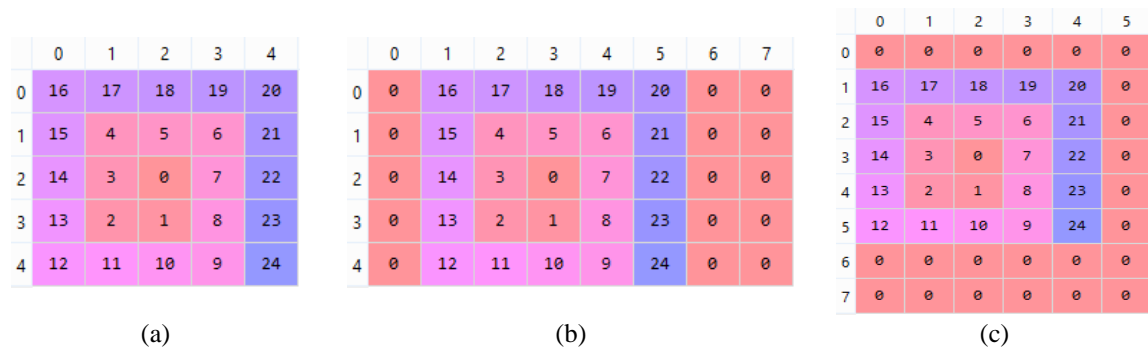


Figure 3. Downward array filling with a spiral pattern: (a) square shape, (b) landscape shape, and (c) portrait shape

2.3. Formation of datastream using random numbers

Datastream usually refers to the formation of data in the process of sending or receiving data through certain communication channels. The original data is converted into data packets of a certain size and then sent continuously. The formation of these packets sometimes also involves an encryption process for the purpose of securing the transmitted data. The encryption used can use a block cipher or stream cipher approach.

In this study, datastreams were formed using the XOR operation by utilizing random numbers generated using a pseudo random number generator (PRNG). Encrypted data is obtained from the results of the XOR operation between plaintext and random numbers. In contrast, the decryption results are generated from XOR operations between ciphertext and random numbers. The number of random numbers taken is as many as the characters in the plaintext. For example, the message’s contents to be encrypted is “cloud” while the random numbers between 0 and 255 obtained from the PNRG are [56, 103, 30, 67, 200], an illustration of the encryption and decryption operations shown in Figure 4.

Encryption					
Plaintext	C	l	o	u	d
ASCII	67	108	111	117	100
Rnd numbers	56	103	30	67	200
Plaintext (bit)	01000011	01101100	01101111	01110101	01100100
Rnd number (bit)	00111000	01100111	00011110	01000011	11001000
XOR (bit)	01111011	00001011	01110001	00110110	10101100
ASCII	123	11	113	54	172
Ciphertext	{		q	6	~

Decryption					
Ciphertext	{		q	6	~
ASCII	123	11	113	54	172
Rnd numbers	56	103	30	67	200
Ciphertext (bit)	01111011	00001011	01110001	00110110	10101100
Rnd number (bit)	00111000	01100111	00011110	01000011	11001000
XOR (bit)	01000011	01101100	01101111	01110101	01100100
ASCII	67	108	111	117	100
Decryption result	C	l	o	u	d

Figure 4. Illustration of XOR operation for encryption and decryption

Algorithm 1. Spiral pattern**Input:** array, direction**Output:** array

```

1  Function spiralFill(array, direction)      29           right ← Y
2     row, column ← get array size           30           direction ← downward
3     size ← MIN(row, column)                31           else if direction == rightward
4                                           32           and Y > right:
5     #center point determination           33           X ← X + 1
6     if size % 2 == 0:                     34           right ← Y
7         size ← size - 1                   35           direction ← downward
8                                           36           else if direction == downward
9     if column % 2 == 0:                   37           and X < bottom:
10        left = right = Y ← INT(column /   38           X ← X + 1
11    2) -1                                   39           else if direction == downward
12        else:                               40           and X == bottom:
13            left = right = Y ← INT(column / 41           X ← X + 1
14    2)                                       42           bottom ← X
15        if row % 2 == 0:                   43           direction ← leftward
16            top = bottom = X ← INT(row /   44           else if direction == leftward
17    2) -1                                   45           and Y > left-1:
18        else:                               46           Y ← Y - 1
19            top = bottom = X ← INT(row /   47           else if direction == leftward
20    2)                                       48           and Y == left-1:
21        #data placement with spiral pattern 49           X ← X - 1
22        rightward, leftward, upward,       50           direction ← upward
23        downward ← 0, 1, 2, 3              51           left ← Y
24                                           52           else if direction == upward and
25        for i in range(size^2):            53           X > top-1:
26            array[X, Y] ← modify pixel     54           X ← X - 1
27            value                           55           else if direction == upward and
28            if direction == rightward and Y 56           X == top-1:
29            < right:                         57           Y ← Y + 1
30                Y ← Y + 1                   58           direction ← rightward
31            else if direction == rightward 59           top ← X
32            and Y == right:                 60           endfor
33                Y ← Y + 1                   61           return array

```

The use of random numbers between 0 and 255 aims to match ASCII values accordingly. The result of encryption is obtained from the XOR operation between the plaintext and the random number. At the same time, the result of decryption is obtained from the XOR operation between the ciphertext and the random number.

The PNRG used to get random numbers is activated using a certain seed value obtained from the entered encryption key. The seed is calculated by adding up the ASCII value of each key character multiplied by the square of its respective position. Suppose the given key is “ab12”. The ASCII values for each character in the key are 97, 98, 49, and 50, respectively. Then the seed value obtained from the key is $((97 \times (1^2)) + (98 \times (2^2)) + (49 \times (3^2)) + (50 \times (4^2)))$ is 1,730. This method of determining the seed aims to obtain a different seed value if the given key has the same character, but has a different order. So “12ab” will produce a value of 2690, and “a1b2” will produce a value of 1975.

The formation of a datastream or, in this case, the encrypted text is shown in Algorithm 2. As input is plaintext (in the encryption process) or ciphertext (in the decryption process) and the seed value for random generation. The XOR operation is performed on the ASCII value of each character with the generated random number. The generated random values are in the range 0 to 255.

Algorithm 2. The formation of datastream**Input:** plaintext, seed_number**Output:** ciphertext

```

1  Function streamData(plaintext, seed)
2     Activate random seed
3     cipherteks ← ""
4     for character in plaintext
5         nchr ← ASC( character )
6         nrand ← get random number (8 bit)
7         cchr ← CHAR(nchr ⊕ nrand)
8         cipherteks ← cipherteks + cchr
9     end for
10 return cipherteks

```

To know the magnitude of the change in the result of the encryption from the original text is measured using the avalanche effect (AE). The AE value indicates how significant the changes in the ciphertext are due to small changes in both the message and the key. Small changes are meant, for example, if one-character changes or two characters exchange their positions. AE is calculated using (3). AE is considered good if the occurring bit changes is greater than 45% [26]. The more bits that change indicate that the encryption algorithm is more difficult to solve.

$$AE = \frac{\text{Changed number of bits}}{\text{The number of ciphertext bits}} \times 100\% \quad (3)$$

In addition to measuring the magnitude of changes resulting from the encryption process, the influence of the plaintext on the encryption results is also assessed. The correlation coefficient is used to measure this relationship. A correlation value close to zero indicates that the original message does not affect the encryption results.

2.4. Implementation of datastream hiding with spiral pattern

Hiding encrypted messages is carried out following the flow in Figure 5 using the LSB method by replacing the last two bits of each modified pixel. This 2-bit LSB modification aims to increase the message storage capacity. Thus, every single character that is hidden will require four pixels as a placeholder. The process of hiding messages can only be done if the cover capacity is sufficient to accommodate the entire contents of the message. The message length is stored as a 16-bit integer. It is hidden first, then followed by the entire message content.

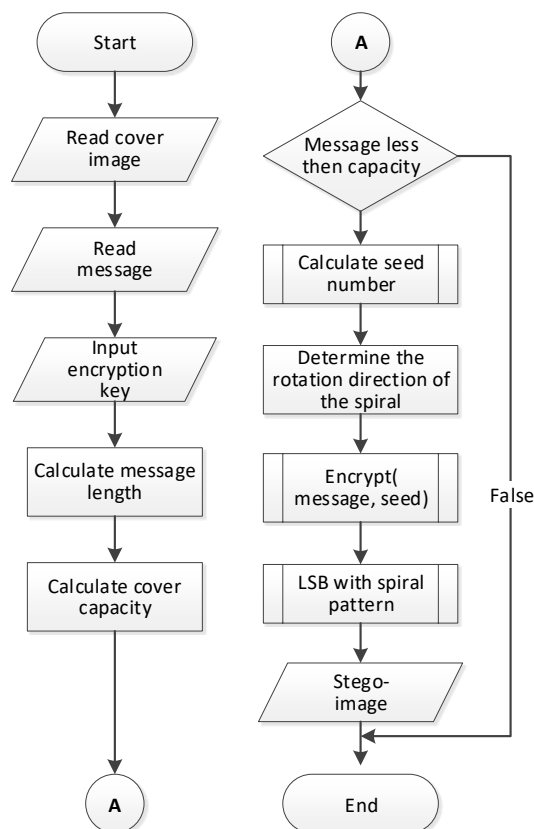


Figure 5. Steganography implementation with a spiral pattern

The encryption key entered determines the seed value the PNRG will later use to retrieve random numbers as many as the number of characters in the message. In simple terms, this seed value acts as an OTP so that each character is paired with a random number constantly changing in the XOR operation. Of course, the goal is that there are no repeated patterns to increase the security of the encryption results.

Using the same seed value, the direction of the spiral pattern is also randomly determined. The intended direction is the exit direction from the center point of the spiral to the left, right, up, or down to rotate clockwise. Message hiding starts from the center coordinates of the spiral in the randomly selected direction. The first eight pixels are used to store the length of the message and continue with the message's contents starting at the 9th pixel. Thus, the maximum possible message length stored in the cover is the maximum capacity minus two.

3. RESULTS AND DISCUSSION

The grayscale images used as covers are lena, peppers, and mandril with a size of 256×256, and a boat and barbara with a size of 256×375. These five images use the PNG format and have the same message storage capacity, which is 16,384 characters. The message text that will be hidden is taken from the prologue of Eragon's novel at [https://www.allfreenovel.com/Page/Story/13817/page-1-of-Eragon\(the-inheritance-cycle-1\)/1/50](https://www.allfreenovel.com/Page/Story/13817/page-1-of-Eragon(the-inheritance-cycle-1)/1/50). The text is taken except for the last paragraph. The text comprises 2,806 words, or 15,826 characters (including spaces and control characters). It is equivalent to 5 pages of A4 single-spaced text. The size of the hidden text is close to the maximum capacity of the five cover images, which is close to 96.59% of the capacity.

For the purpose of hiding messages in the cover, the key used for datastream formation is "abc123". It is a weak key which returns 5,250 as the seed value. The cover image and the result of steganography are shown in Figure 6, while the test results are shown in Tables 1 and 2.

Cover images:



Steganographic images:



Figure 6. Cover images and steganographic images (lena, peppers, mandril, boat, and barbara)

The test results show that visually the steganographic image looks similar to the cover image. This can be interpreted that the imperceptible criteria, namely the existence of a secret message that cannot be perceived visually, can be fulfilled. Likewise, the fidelity criterion, which states that the image quality after adding the message does not experience a significant change, can be fulfilled. This is supported by PSNR values which are in the range of 46 dB. This PSNR value is much higher than the results obtained by [13], [24], which obtained PSNR values in the range of 30 dB, lesser than [18] which obtained PSNR value between 40 and 54 and [21] which obtained PSNR nearly 50, and is in line with [5], [25] where steganographic images for near-maximum capacity are considered to still have good quality at the PSNR value is above 40 dB. The PSNR values for the boat and barbara images tend to be higher than the other three images because there are quite a number of pixels outside the spiral pattern square area, so the intensity values do not change.

The physical size (in bytes) of the image also changes. This slight size change is in line with [16]; there is a slight increase in the stego-image compared to its original cover. This size change varies between 0.03% to 3.1%, depending on the characteristics of the image used as the cover. Table 2 also shows that the number of bit changes in the five images is in the range of 63000, or approximately 50% of the total hidden bits. With an AE value of 12%, it indicates that the overall image does not experience significant data changes.

Table 1. The test results of steganographic images

Images	Image size (byte)			MSE	PSNR
	original	stego	increase		
lena	44,412	45,130	1.6%	2.423	46.209
peppers	41,910	43,195	3.1%	2.417	46.214
mandril	51,835	51,979	0.3%	2.427	46.206
boat	66,470	67,487	1.5%	1.650	47.044
barbara	69,939	71,200	1.8%	1.650	47.044

Table 2. Concealment test results with LSB

Images	Number of bits		AE (%)
	hidden	changed	
lena	126,608	63,868	12.182%
Peppers	126,608	63,336	12.080%
mandril	126,608	63,752	12.160%
boat	126,608	63,484	8.266%
barbara	126,608	63,388	8.254%

In order to increase the security of hidden data, the message is first encrypted using OTP with the XOR operation, similar to [16], [17]. The Avalanche Effect is used to measure the results of encryption on small changes that occur in the key used. For testing purposes, the keys used are “abc123”, “acb123”, “abc127”, and “123cba”, respectively. The results of the encryption test are shown in Table 3. The test is carried out by comparing the original text with the encrypted text.

Table 3. The encryption test results

Keys	Seed	AE	r
abc123	5,250	50.01%	0.007
acb123	5,245	50.15%	0.011
abc127	5,394	50.21%	0.010
123cba	8,234	50.08%	0.005

Table 3 shows that the AE values for all keys used are greater than 50%. This proves that a slight change in the key gives a significant difference in the encryption results. The correlation coefficient value, which is very close to zero, indicates that the encryption results are not correlated or not affected by the encrypted text. In this study, the key is used to obtain a seed as an OTP in the datastream formation process and determines the direction of the spiral pattern. Without knowing the key used, attempts to extract messages will only result in meaningless strings of characters. This research was conducted under the assumption that the hidden data must be able to be extracted perfectly. It is the same with the encryption results, which must be returned to their original form exactly. Departing from this assumption, the recovery criteria and decryption performance are not specifically examined and tested.

4. CONCLUSION

The more LSB bits that are modified, the greater the capacity of the information that can be hidden in the image. Modification of the 2-bit LSB with data close to the maximum capacity is still capable of producing good quality steganographic images as indicated by PSNR values in the range of 46 dB. Likewise, using different random seeds as OTP consistently results in significant changes to the hidden text. Using OTP and spiral patterns to place the LSB bit can reduce the chance of hacking data hidden in the image.




REFERENCES

- [1] O. C. Abikoye and R. O. Ogundokun, “Efficiency of LSB steganography on medical information,” *International Journal of Electrical and Computer Engineering*, vol. 11, no. 5, pp. 4157–4164, 2021, doi: 10.11591/ijece.v11i5.pp4157-4164.
- [2] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, “Image steganography: a review of the recent advances,” *IEEE Access*, vol. 9, pp. 23409–23423, 2021, doi: 10.1109/ACCESS.2021.3053998.
- [3] M. M. Hashim, M. S. M. Rahim, F. A. Johi, M. S. Taha, and H. S. Hamad, “Performance evaluation measurement of image steganography techniques with analysis of LSB based on variation image formats,” *International Journal of Engineering and Technology(UAE)*, vol. 7, no. 4, pp. 3505–3514, 2018, doi: 10.14419/ijet.v7i4.17294.
- [4] S. S. Chaeikar, M. Zamani, A. B. A. Manaf, and A. M. Zeki, “PSW statistical LSB image steganalysis,” *Multimedia Tools and Applications*, vol. 77, no. 1, pp. 805–835, 2018, doi: 10.1007/s11042-016-4273-6.
- [5] S. Solak, “High embedding capacity data hiding technique based on EMSD and LSB substitution algorithms,” *IEEE Access*, vol. 8, pp. 166513–166524, 2020, doi: 10.1109/ACCESS.2020.3023197.
- [6] V. Kumar and D. Kumar, “A modified DWT-based image steganography technique,” *Multimedia Tools and Applications*, vol. 77, no. 11, pp. 13279–13308, 2018, doi: 10.1007/s11042-017-4947-8.
- [7] V. K. Sharma, P. Mathur, and D. K. Srivastava, *Highly secure DWT steganography scheme for encrypted data hiding*, vol. 106. Springer Singapore, 2019.
- [8] A. K. Sahu and G. Swain, “Pixel overlapping image steganography using PVD and modulus function,” *3D Research*, vol. 9, no. 3, 2018, doi: 10.1007/s13319-018-0188-5.
- [9] N. M. Ganguly, G. Paul, S. K. Saha, and D. Burman, “A PVD based high capacity steganography algorithm with embedding in non-sequential position,” *Multimedia Tools and Applications*, vol. 79, no. 19–20, pp. 13449–13479, May 2020, doi: 10.1007/s11042-019-08178-9.




- [10] A. Kuznetsov, O. Smirnov, A. Arischenko, I. Chepurko, A. Onikiychuk, and T. Kuznetsova, "Pseudorandom sequences for spread spectrum image steganography," *CEUR Workshop Proceedings*, vol. 2654, pp. 122–131, 2020.
- [11] A. Kuznetsov, A. Onikiychuk, O. Peshkova, T. Gancarczyk, K. Warwas, and R. Ziubina, "Direct spread spectrum technology for data hiding in audio," *Sensors*, vol. 22, no. 9, 2022, doi: 10.3390/s22093115.
- [12] S. P. Lu, R. Wang, T. Zhong, and P. L. Rosin, "Large-capacity image steganography based on invertible neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 10811–10820, 2021, doi: 10.1109/CVPR46437.2021.01067.
- [13] O. Elharrouss, N. Almaadeed, and S. Al-Maadeed, "An image steganography approach based on k-least significant bits (k-LSB)," *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020*, pp. 131–135, 2020, doi: 10.1109/ICIoT48696.2020.9089566.
- [14] M. Fateh, M. Rezvani, and Y. Irani, "A new method of coding for steganography based on LSB matching revisited," *Security and Communication Networks*, vol. 2021, 2021, doi: 10.1155/2021/6610678.
- [15] X. Chai, H. Wu, Z. Gan, Y. Zhang, Y. Chen, and K. W. Nixon, "An efficient visually meaningful image compression and encryption scheme based on compressive sensing and dynamic LSB embedding," *Optics and Lasers in Engineering*, vol. 124, no. August 2019, p. 105837, 2020, doi: 10.1016/j.optlaseng.2019.105837.
- [16] O. M. Osman, M. E. A. Kanona, M. K. Hassan, A. A. E. Elkhair, and K. S. Mohamed, "Hybrid multistage framework for data manipulation by combining cryptography and steganography," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 327–335, 2022, doi: 10.11591/eei.v11i1.3451.
- [17] Z. Ji et al., "Random shifting intelligent reflecting surface for OTP encrypted data transmission," *IEEE Wireless Communications Letters*, vol. 10, no. 6, pp. 1192–1196, 2021, doi: 10.1109/LWC.2021.3061549.
- [18] R. M. Neamah, J. A. Abed, and E. A. Abbood, "Hide text depending on the three channels of pixels in color 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.
- [19] A. K. Sahu and G. Swain, "Reversible image steganography using dual-layer LSB matching," *Sensing and Imaging*, vol. 21, no. 1, 2020, doi: 10.1007/s11220-019-0262-y.
- [20] S. Rustad, D. R. I. M. Setiadi, A. Syukur, and P. N. Andono, "Inverted LSB image steganography using adaptive pattern to improve imperceptibility," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 3559–3568, 2022, doi: 10.1016/j.jksuci.2020.12.017.
- [21] P. Puteaux, S. Member, M. Vialle, W. Puech, and S. Member, "Homomorphic encryption-based LSB substitution for high capacity data hiding in the encrypted domain," pp. 108655–108663, 2020, doi: 10.1109/ACCESS.2020.3001385.
- [22] Y. Liu and J. Zhang, "Large - capacity LSB information hiding scheme based on two - dimensional code," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2017, no. 1, pp. 528–532, doi: 10.1109/ICEIEC.2017.8076621.
- [23] X. Wang, Y. Wang, X. Zhu, and C. Luo, "A novel chaotic algorithm for image encryption utilizing one-time pad based on pixel level and DNA level," *Optics and Lasers in Engineering*, vol. 125, no. August 2019, p. 105851, 2020, doi: 10.1016/j.optlaseng.2019.105851.
- [24] P. Li and A. Lu, "LSB-based steganography using reflected gray code for color quantum images," *International Journal of Theoretical Physics*, vol. 57, no. 5, pp. 1516–1548, 2018.
- [25] N. Pandian, "An image steganography algorithm using huffman and interpixel difference encoding," *International Journal of Computer Science & Security*, vol. 8, no. 6, pp. 202–215, 2014.
- [26] H. Noura, L. Sleem, M. Noura, M. M. Mansour, A. Chehab, and R. Couturier, "A new efficient lightweight and secure image cipher scheme," *Multimedia Tools and Applications*, vol. 77, no. 12, pp. 15457–15484, 2018, doi: 10.1007/s11042-017-5124-9.

BIOGRAPHIES OF AUTHORS







Rihartanto    received the B.Sc. degree in computer engineering from Institute of Science and Technology "Akprind" Yogyakarta in 1996 and the M.Sc. degree in environmental science from Mulawarman University, Samarinda, Indonesia, in 2017. Currently, he is a lecturer at Department of Information Technology, State Polytechnic of Samarinda, Samarinda, Indonesia. His research interests are in the areas of information security, data compression and image processing. He can be contacted at email: rihart.c@gmail.com.







Didi Susilo Budi Utomo    get his diploma degree in power electronics from LuccasNule. GmbH in 1996, B.Sc. degree in electrical engineering from the Islamic University of Malang, in 1999 and M.Sc. degree in electrical engineering System design and technology from Fachhochschule Darmstadt Germany, in 2003. Currently, he is a lecturer at the Department of Information Technology, State Polytechnic of Samarinda, Samarinda, Indonesia. His research interests are in computer control and green energy. He can be contacted at email: dsbudiutomo10@gmail.com.







Ansar Rizal     get his B.Sc. degree in Electronics and Telecommunication from Muslim University of Indonesia in Makasar in 1995, Master degree in Computer Science from Gadjah Mada University in 2008. Currently, he is a lecturer at the Department of Information Technology, State Polytechnic of Samarinda, Samarinda, Indonesia. His research interests are in computer, electronic and telecommunications. He can be contacted at email: ansardeuy@gmail.com.



Dwi Agus Diartono     received his B.Sc. degree in Management of Informatics from Bina Nusantara University, Jakarta, Indonesia in 1998 and the M.Sc. degree in Computer Science from Gadjah Mada University, Yogyakarta, Indonesia in 2003. Currently, he is a lecturer at Department of Information System, Faculty of Information Technology and Industry, Stikubank University, Semarang, Indonesia. His research interests are in information systems and decision support system. He can be contacted at email: dwieagus@edu.unisbank.ac.id.



Herny Februriyanti     received her B.Sc. degree in Management of Informatics and Computer Engineering from Institute of Science and Technology "Akprind" Yogyakarta in 1998 and the M.Sc. degree in Computer Science from Gadjah Mada University, Yogyakarta, Indonesia, in 2010. Currently, she is a lecturer at Department of Information System, Faculty of Information Technology and Industry, Stikubank University, Semarang, Indonesia. Her research interests are in the areas of information retrieval and information security. She can be contacted at email: hernyfeb@edu.unisbank.ac.id.