ISSN 2347 - 3983
Volume 8. No. 7, July 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter62872020.pdf
https://doi.org/10.30534/ijeter/2020/62872020

LSB Image Steganography with Data Compression Technique Using Goldbach G0 Code Algorithm

Jan Carlo T. Arroyo[1], Allemar Jhone P. Delima[2]
[1]College of Computing Education, University of Mindanao, Davao City, Davao del Sur, Philippines
[2]College of Engineering, Technology and Management, Cebu Technological University-Barili Campus, Cebu, Philippines
jancarlo_arroyo@umindanao.edu.ph[1], allemarjpdjca@yahoo.com[2]

ABSTRACT

In this paper, a new method of image steganography under the spatial domain is introduced and added to the body of learning. This study used a compression technique in securing text data using the Goldbach code algorithm. Image steganography is also performed with the use of the Least Significant Bit (LSB) technique. To execute, the message is first compressed using the Goldbach code, and the compressed data is embedded inside an image using the LSB. Simulation results revealed that the combination of the two methods had paved the way to more secure storage and data transmission, as evident in file size, Peak Signal to Noise Ratio (PSNR), and MSE parameters used.

Key words: Data security, Goldbach code algorithm, hybrid algorithms, least significant bit algorithm, steganography

1. INTRODUCTION

The present cyber environment has long identified the need for secure communication for the exchange of information in today's digital era. With these, there is a need for a secure channel with solid security in such a way that a third party can't access one's crucial data. The primary concern of most security systems is the safekeeping of secret information. To safeguard information, the three most common techniques are widely used, which are steganography, watermarking, and cryptography [1].

To differentiate, cryptography is the science of hiding information by transforming data into an unintelligible format using encryption technique [2]–[8]. Watermarking is a technique used for copyrighting one's property by embedding watermark such that counterfeit measures are detected [9]–[11]. Steganography, however, is a technique of hiding data and information into different file types such as images, video, audio, or text files, among others [12]–[14]. The indexed comparison of the three commonly used techniques for security obtained from [1] is shown in Table 1.

This study focused on the implementation of steganography in handling text data with the use of spatial domain technique, particularly the Least Significant Bit (LSB) algorithm. LSB is one of the widely used methods in steganography due to the simplicity of its method where a lesser chance of degradation of the original image is observed. However, steganography alone cannot protect data [15]. To leverage the use of technology in strengthening information and data security while maintaining the secrecy of information, the hybridization of two algorithms is performed. To protect the storage and transmission of data, the compression of the text file is performed using the Goldbach code algorithm prior to the embedding of the data to the image file using LSB.

The rest of the paper is structured as follows: Section 2 presents the existing methodology used in this study. The proposed hybrid method is discussed in Section 3, while Section 4 presents the results and discussion. The conclusion is shown in Section 5.

2. EXISTING SYSTEM

2.1 Goldbach G0 Code

The Goldbach G0 code was developed by Peter Fenwick in 2001. The G0 code is based on the Goldbach conjecture by Christian Goldbach, which states that every even number larger than four can be expressed as the sum of two odd primes [16], [17]. For example, number 14 can be produced and represented by adding primes 3 and 11. The G0 code number system encodes an integer twice with an offset value, such that 2(n+3), to generate an equivalent codeword [17]. To identify the corresponding Goldbach G0 codeword for a number, its two primes are mapped with an array of primes. For example, let array A=[3,5,7,11,13,17,19,23,29,31] contain the values of the first 10 prime numbers and array B=[0,0,0,0,0,0,0,0,0,0] as map for the indices of the primes. Suppose the integer 7 is encoded in G0. First, set n = 7 and then compute for 2(n+3), wherein 2(7+3) = 20. Next, identify from A the first two distinct primes, which are summed to get the value 20. Based on the array A, the first two prime numbers for the integer 20 are 7 and 13. These two primes are mapped to B according to their relative indices represented by the value 1, such that B=[0,0,1,0,1,0,0,0,0,0]. Finally, all trailing zeroes from B are removed to generate the codeword. As a result, the equivalent codeword for the digit 7 is 00101. The first 15 integers and their equivalent Goldbach G0 code are presented in Table 2.

Table 1: Indexed comparison of the three commonly used technique for security

Criteria	Steganography	Watermarking	Cryptography
Objective	Secret communication	Copyright protection	Data protection
Secret Information	Can be used with any file type	Watermarks are used	Text files are used
Secret Key	Key is optional to use	Key is optional to use	Key is necessary
Carrier Object	Any media can be used	Digital image or audio files can be used	Text or image files can be used
Selection of cover	Any type of cover can be used	Restriction in cover selection	No cover needed
Visibility	Never perceptible to the normal human vision.	May or may not be visible to human eye.	File is noticeable due to encryption but deciphering is difficult
Detection and retrieval	Full retrieval of data is possible and cover is not needed for recovery.	Cross-correlation helps in data retrieval.	Full retrieval of data can be done without the need of the cover
Capacity	High	Low	High
Attacks	Replacement of watermarks	Steganalysis	Cryptanalysis
Security	Very High	High	High

Table 2: Goldbach G0 Code

Value *n*	Encode 2(*n*+3)	Sum of Primes	Equivalent Codeword
1	8	3 + 5	11
2	10	3 + 7	101
3	12	5 + 7	011
4	14	3 + 11	1001
5	16	5 + 11	0101
6	18	7 + 11	0011
7	20	7 + 13	00101
8	22	5 + 17	010001
9	24	11 + 13	00011
10	26	7 + 19	0010001
11	28	11 + 17	000101
12	30	13 + 17	000011
13	32	13 + 19	0000101
14	34	11 + 23	00010001
15	36	5 + 31	100000001

2.2 LSB in Image Steganography

The LSB is one of the renowned and simplest methods of embedding sensitive data in other structures that works by replacing some of the least significant bits of a cover file [18]–[22]. Like other steganographic algorithms, the LSB performs in such a way that small image alterations are not obvious in the eyes of casual observers. LSB works by looking at each pixel of the image through the RGB colorspace. Since every RGB component is composed 8 bits of memory, LSB can be used to manipulate the last bit of each component to embed secret data. For example, a 9-bit binary message 101001101 is encoded into a group of 3 neighboring pixels, as shown in Figure 1.

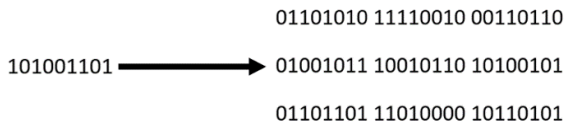

Figure 1: Embedding message to pixels

Each bit from the message is substituted to the least significant bit of each RGB component. If the LSB bit is equal to the message bit, it is skipped; otherwise, it is replaced. Based on the given, 9-bits of data was embedded in the sequence at the expense of masking 4 of bits (shown in red) as presented in Figure 2.

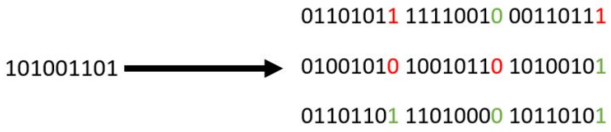

Figure 2: Embedding message to pixels

3. PROPOSED METHODOLOGY

The proposed method involves the use of the Goldbach G0 code for text compression, and then the text equivalent is embedded in an image using LSB. Figure 3 shows the flowchart of the proposed process.

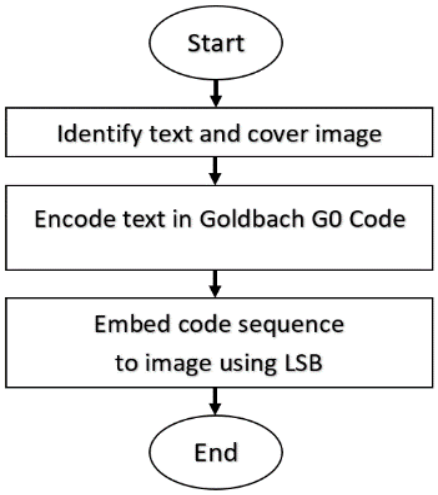

Figure 3: Process of the proposed methodology

To perform, the following steps must be executed:

a. Identify a text and a cover image.
b. Using the text, count the frequency of each character.
c. Arrange the characters according to the frequency and its order of appearance in the text. The most frequent character shall be *n*=1 in G0.
d. For each character in the G0 list represented by n, compute for *2(n+3)* and find the first two prime numbers of the result.
e. Map the two primes to the list of prime numbers > 2 to retrieve the equivalent codeword in binary format.

f. Substitute equivalent codeword of each character in G0 to every character in the text.
g. Embed the result of step f to the image by traversing through each pixel and replacing the LSB

The proposed method is implemented in Python. Three 512x512 sample test images from [23], [24] were downloaded to wit: Lena, Peppers, and Zelda. These images were converted into grayscale. The secret message to be embedded varies to sizes where 16kb, 32kb, and 48kb plaintext messages are used. The simulation was performed in an i7-7000HQ 2.8 GHz 16GB RAM 4GBVRAM Windows 10 laptop computer. The Peak Signal to Noise Ratio (PSNR) test and file size comparison were executed to validate the feasibility of the proposed method. The PSNR is used to assess the quality of an image by comparing the similarity between the original and altered images [25]. If the value of the PSNR is high, the higher the image quality and the more it is closer to the actual image. A PSNR of 100 means there is no significant noise detected between the two images. The PSNR is defined by a mean squared error (MSE), which finds the magnitude of error between the images. The equation used to find PSNR value is:

$$PSNR = 10\,log\left[\frac{MAX_C^2}{MSE}\right] \quad (1)$$

where MAX_C refers to the maximum possible value of the pixel in the image and MSE is expressed as:

$$MSE = \frac{1}{m \times n}\sum_{a=0}^{m-1}\sum_{b=0}^{n-1}[C(a,b) - S(a,b)]^2 \quad (2)$$

where, m and n are the number of rows and columns respectively, $C(a,b)$, and $S(a,b)$ are the pixels located at index a and b given cover image C and stego image S.

4. RESULTS AND DISCUSSION

The histogram, PSNR, MSE, and file size analyses for the images Lena, Peppers, and Zelda encoded using the lone LSB, and the proposed method is presented in this section.

4.1 Image Steganography using the Lena Image Dataset

A 16kb, 32kb, and 48kb secret messages were embedded in the Lena image dataset and were tested using the proposed methodology. Simulation results revealed that the stego images generated using LSB, and the proposed method have no visible trace of modifications. However, it is evident in the histogram results shown in Figures 4-6 that significant changes were made to the images wherein the lone LSB method had more noise since it obtained lower PSNR value as compared to the proposed methodology.

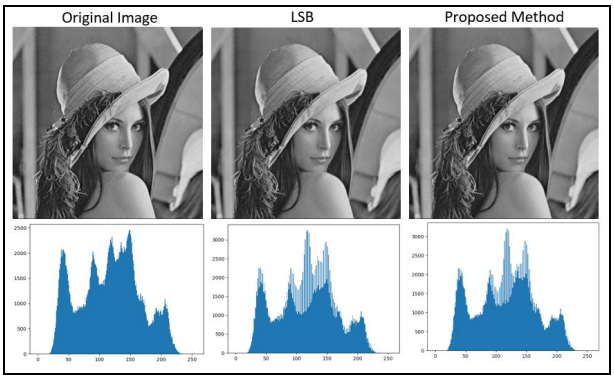

Figure 4: Histogram plot for Lena image dataset with 16kb secret message

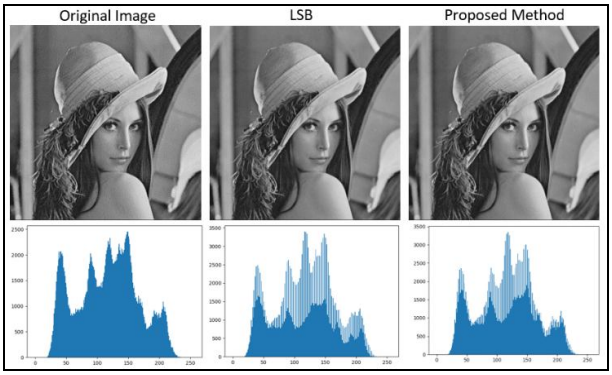

Figure 5: Histogram plot for Lena image dataset with 32kb secret message

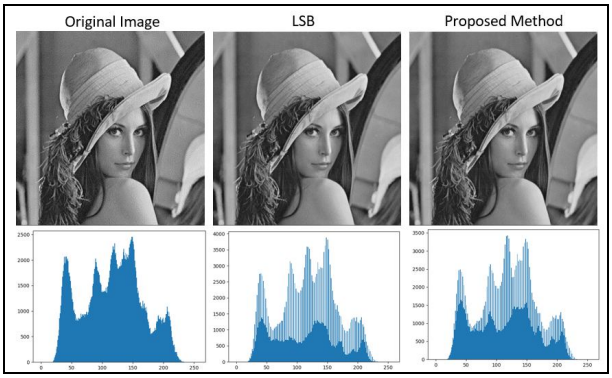

Figure 6: Histogram plot for Lena image dataset with 48kb secret message

When embedded with a 16kb secret message, the lone LSB method has obtained a PSNR value of 58.94 decibels (dB) as against the proposed method with 60.72 dB, which is 3.02% higher than the former. Extent on the file size generated, the proposed method is 0.19% lower in size, with 488,056 bytes as against the lone LSB method with 489,002 bytes. Also, the proposed method revealed a lower MSE value with a difference of approximately 37% as compared to the lone LSB method. The summary of results is presented in Table 3.

When a 32kb secret message is encoded to the Lena image dataset, the stego image generated using LSB, and the proposed method shows no visible trace of modifications.

However, it is evident in the histogram results that significant changes were made to the image wherein the lone LSB method had more noise as compared to the proposed method. The image steganography technique using the lone LSB method produced a stego image with 55.92 dB PSNR. In contrast, a 57.71 dB PSNR is depicted on the stego image generated using the hybrid method. Extent on the size of the stego images, the proposed method generated a smaller file size due to the compression technique with a size of 490,590 bytes as against the lone LSB method with 492,862 bytes. Further, the proposed method revealed a lower MSE value with a difference of approximately 31% as compared to the lone LSB image steganography technique. The summary of simulation results is presented in Table 4. When a 48kb message is encoded in the Lena image dataset, the lone LSB method gained 54.15 dB PSNR, while the proposed method obtained 56.96 dB, which is 5.18% higher than the former. This denotes that the lone LSB method has a higher noise. As for the file size, the proposed method generated a stego image that is 0.67% lower than the LSB method alone. Extent on error measures, the proposed method revealed to have a lower MSE with a value of 0.16 as compared to the lone LSB method, as evident in Table 5.

Table 3: Lena image dataset with 16kb secret message summary of results

	LSB	Proposed Method	Variance
PSNR	58.94 dB	60.72 dB	+3.02%
MSE	0.08	0.05	-37.50%
File Size	489,002 B	488,056 B	-0.19%

Table 4: Lena image dataset with 32kb secret message summary of results

	LSB	Proposed Method	Variance
PSNR	55.92 dB	57.71 dB	+3.20%
MSE	0.16	0.11	-31.25%
File Size	492,862 B	490,590 B	-0.46%

Table 5: Lena image dataset with 48kb secret message summary of results

	LSB	Proposed Method	Variance
PSNR	54.15 dB	56.96 dB	+5.18%
MSE	0.24	0.16	-33.33%
File Size	496,640 B	493,284 B	-0.67%

These findings show that the proposed method gains higher PSNR values, lower error rates, and smaller file sizes in all test cases, which denotes effectiveness as it produced higher quality images with lesser noise and better file size reduction.

4.2 Image Steganography using the Peppers Image Dataset

The histogram, PSNR, MSE, and file size analyses for the Peppers image dataset encoded using LSB and the proposed method are presented in Figures 7-9, and Table 6. The histogram and actual images generated using LSB and the proposed method show no visible trace of modifications. However, it is evident in the PSNR values and file sizes that the proposed method performed better when compared to LSB image steganography alone. Also, the proposed method revealed lower MSE values as compared to the lone LSB method.

With the integration of a 16kb secret message to the cover image, the lone LSB method produced a stego image with 58.93 dB PSNR. The proposed method produced a stego image that is 3.03% higher than the former. Extent on the size of the file, the stego image provided by the proposed method has a file size of 494,760 bytes. By embedding a 32kb secret message, the lone LSB method produced a stego image with peak signal to noise ratio value of 55.91 dB, while the proposed method produced a stego image that is 3.18% higher at 57.69 dB. Further, simulation results revealed that the stego image generated by the proposed method has a lower error rate.

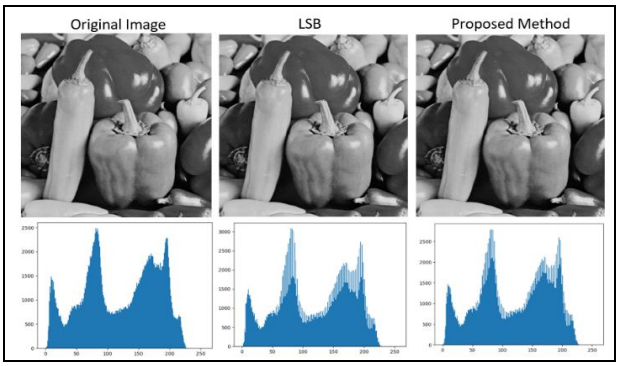

Figure 7: Histogram plot for Peppers image dataset with 16kb secret message

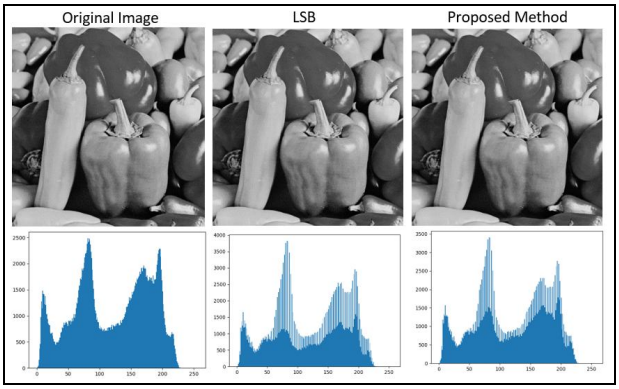

Figure 8: Histogram plot for Peppers image dataset with 32kb secret message

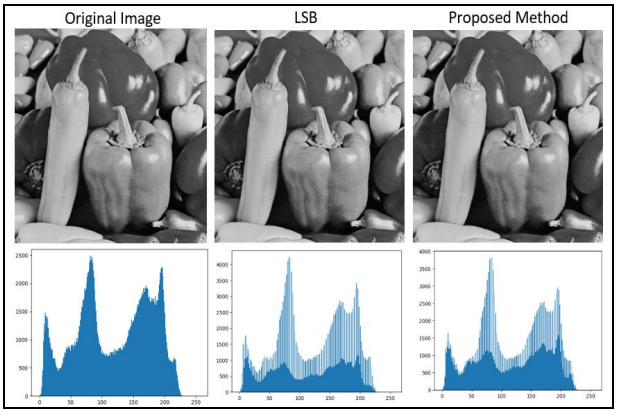

Figure 9: Histogram plot for Peppers image dataset with 48kb secret message

With a 48kb secret message, the stego image produced by the lone LSB method gained 54.15 dB PSNR, while the proposed method is 3.30% higher at 55.94 dB. Extent on the size of the file, the proposed method produced a stego image with a smaller file size of 498,932 bytes as against the lone LSB method with 501,526 bytes. The results show that the proposed method produced stego images with higher PSNR values, lower error percentages, and smaller file sizes, which denotes higher quality images with lesser noise and better storage management.

Table 6: Peppers image dataset summary of results

	LSB	Proposed Method	Variance
Peppers image dataset with 16 kb secret message			
PSNR	58.93 dB	60.72 dB	+3.03%
MSE	0.08	0.05	-37.50%
File Size	495,598 B	494,760 B	-0.16%
Peppers image dataset with 32 kb secret message			
PSNR	55.91 dB	57.69 dB	+3.18%
MSE	0.16	0.11	-31.25%
File Size	498,465 B	496,811 B	-0.33%
Peppers image dataset with 48 kb secret message			
PSNR	54.15 dB	55.94 dB	+3.30%
MSE	0.24	0.16	-33.33%
File Size	501,526 B	498,932 B	-0.51%

4.3 Image Steganography using the Zelda Image Dataset

The histogram, PSNR, MSE, and file size analyses for the Zelda image dataset encoded with 16kb, 32kb, and 48kb secret messages using LSB and the proposed method are presented in Figures 10-12, and Table 7. The stego images generated using LSB, and the proposed method shows no visible trace of modifications. However, it is evident in the histogram results that there are significant changes made to the cover images wherein the stego images produced by the lone LSB method had more noise as compared to the stego images generated by the proposed method. The PSNR values and file sizes affirm that the proposed method produced better stego images than that of the lone LSB image steganography technique. With a 16kb message, the lone LSB method produced a stego image with 58.91 dB PSNR value as against the proposed method with peak signal to noise ratio value of 60.73 dB. Further, the stego image produced by the latter has a smaller file size as against the former.

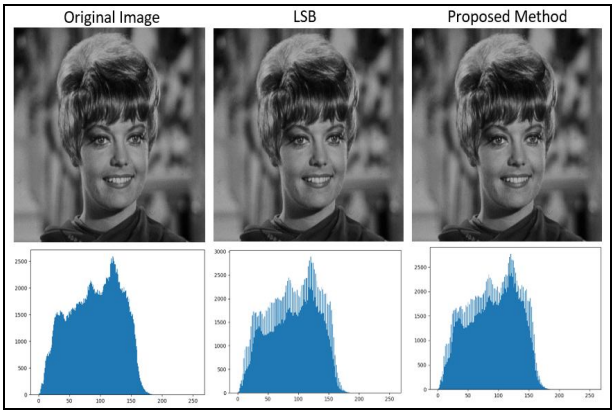

Figure 10: Histogram plot for Zelda image dataset with 16kb secret message

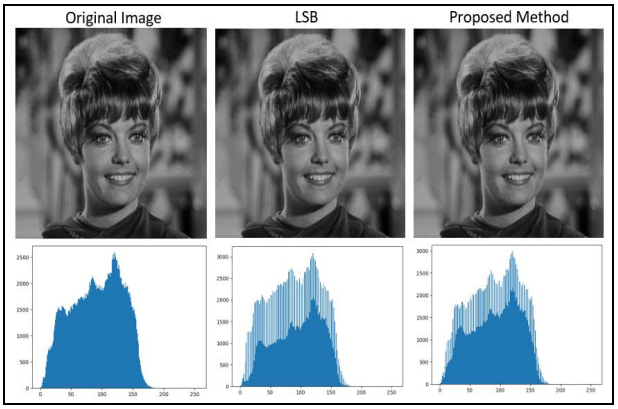

Figure 11: Histogram plot for Zelda image dataset with 32kb secret message

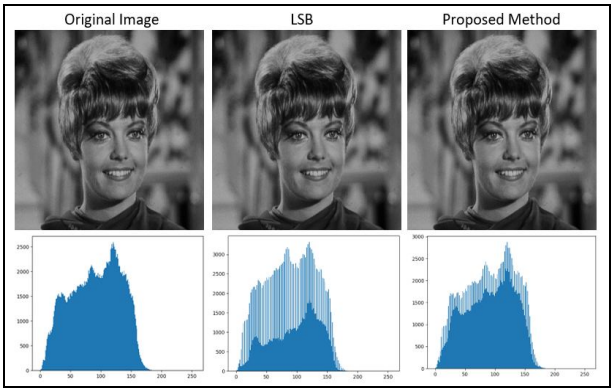

Figure 12: Histogram plot for Zelda image dataset with 48kb secret message

With a 32kb secret message, the stego image produced by the proposed method revealed a peak signal to noise ratio value of 57.70 dB, which is 3.22% higher than the stego image generated using the lone LSB method. Furthermore, the proposed method produced a stego image that is 0.61% smaller, with 452,656 bytes as against the stego image generated using the LSB method alone. With a 48kb secret message embedded to the cover image, the lone LSB method produced a stego image with peak signal to noise ratio value of 54.15 dB. The stego image generated using the proposed method revealed a PSNR value of 55.94 dB, which is 3.30% higher against the latter. Extent on the error metric used, the stego image from the proposed method show a lesser error rate against the stego image generated using the lone LSB image steganography technique.

Table 7: Zelda image dataset summary of results

	LSB	Proposed Method	Variance
Zelda image dataset with 16 kb secret message			
PSNR	58.91 dB	60.73 dB	+3.08%
MSE	0.08	0.05	-37.5%
File Size	451,173 B	450,075 B	-0.24%
Zelda image dataset with 32 kb secret message			
PSNR	55.90 dB	57.70 dB	+3.22%
MSE	0.16	0.11	-31.25%
File Size	454,499 B	452,656 B	-0.61%
Zelda image dataset with 48 kb secret message			
PSNR	54.15 dB	55.94 dB	+3.30%
MSE	0.24	0.16	-33.33%
File Size	457,726 B	454,910 B	-0.61%

The simulation result shows that the proposed method produced stego images that has higher PSNR values, lower error percentages, and smaller file sizes.

5. CONCLUSION

In this paper, the use of the LSB and Goldbach code algorithms as a new method of hiding data is presented. The proposed approach introduces a more secure technique of obscuring data away from perpetrators as data are hidden twice. First, the data is compressed to hide meaningful information that may be blatantly used by unwanted users. Second, the compressed data is hidden in an image for safeguarding during data transmission or when simply hidden as protection against the unauthorized person. The simulation results revealed that the proposed method produced stego images with smaller file sizes, lower MSE percentages, and higher PSNR values, which denotes higher quality images with lesser noise and better storage management.

REFERENCES

[1] A. Saini, K. Joshi, and S. Allawadhi, "A Review On Video Steganography Techniques," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 3, pp. 1015–1020, 2017.

[2] R. Rahim and A. Ikhwan, "Cryptography Technique with Modular Multiplication Block Cipher and Playfair Cipher," *Int. J. Sci. Res. Sci. Technol.*, vol. 2, no. 6, pp. 71–78, 2016.

[3] O. Reyad, "Cryptography and Data Security: An Introduction," 2018.

[4] W. Stallings, *Cryptography and Network Security Principles and Practices*. Prentice Hall, 2015.

[5] J. F. Dooley, *History of Cryptography and Cryptanalysis*. 2018. https://doi.org/10.1007/978-3-319-90443-6

[6] S. A. Babu, "Modification Affine Ciphers Algorithm for Cryptography Password," *Int. J. Res. Sci. Eng.*, vol. 3, no. 2, pp. 346–351, 2017.

[7] O. Toshihiko, "Lightweight cryptography applicable to various IoT devices," *NEC Tech. J.*, vol. 12, no. 1, pp. 67–71, 2017.

[8] S. N. Kumar, "Review on Network Security and Cryptography," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 8, no. 6, p. 21, 2018.

[9] Q. Su, X. Zhang, and G. Wang, "An improved watermarking algorithm for color image using Schur decomposition," *Soft Comput.*, vol. 24, pp. 445–460, 2020.

[10] E. H. Rachmawanto, D. R. I. M. Setiadi, C. A. Sari, and N. Rijati, "Imperceptible and secure image watermarking using DCT and random spread technique," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 17, no. 4, pp. 1750–1757, 2019.

[11] R. Bairagee and M. Gupta, "Review of Digital Image Watermarking Techniques," *Int. J. Sci. Res. Eng. Trends*, vol. 5, no. 3, pp. 1075–1079, 2019.

[12] S. Karakus and E. Avci, "A new image steganography method with optimum pixel similarity for data hiding in medical images," *Med. Hypotheses*, 2020. https://doi.org/10.1016/j.mehy.2020.109691

[13] R. Cogranne, Q. Giboulot, and P. Bas, "Steganography by Minimizing Statistical Detectability : The cases of JPEG and Color Images," in *ACM Information Hiding and MultiMedia Security (IH&MMSec)*, 2020.

[14] S. M. A. Al-Nofaie and A. A. A. Gutub, "Utilizing pseudo-spaces to improve Arabic text steganography for multimedia data communications," *Multimed. Tools Appl.*, vol. 79, pp. 19–67, 2020.

[15] A. Baby and H. Krishnan, "Combined Strength of Steganography and Cryptography- A Literature Survey," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 3, pp. 1007–1010, 2017.

[16] M. A. Budiman and D. Rachmawati, "On Using Goldbach G0 Codes and Even-Rodeh Codes for Text Compression on Using Goldbach G0 Codes and Even- Rodeh Codes for Text Compression," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 180, 2017.

[17] P. Fenwick, "Variable-Length Integer Codes Based on the Goldbach Conjecture, and Other Additive Codes," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 2412–2417, 2002. https://doi.org/10.1109/TIT.2002.800483

[18] S. Goel, S. Gupta, and N. Kaushik, "Image Steganography -- Least Significant Bit with Multiple Progressions," in *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*, 2015, pp. 105–112.

[19] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, Second Edi. Burlington: Morgan Kaufmann, 2008.

[20] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Syst. J.*, vol. 35, no. 3.4, pp. 313–336, 1996. https://doi.org/10.1147/sj.353.0313

[21] C. C. Chang, J. Y. Hsiao, and C. S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy," *Pattern Recognit.*, vol. 36, pp. 1583–1595, 2003.

[22] K. Curran, X. Li, and R. Clarke, "An Investigation into the Use of the Least Significant Bit Substitution Technique in Digital Watermarking," *Am. J. Appl. Sci.*, vol. 2, no. 3, pp. 684–654, 2005.

[23] "Public-Domain Test Images for Homeworks and Projects," *Retrieved from https://homepages.cae.wisc.edu/~ece533/images/*. .

[24] "The USC-SIPI Image Database," *http://sipi.usc.edu/database/*.

[25] K.-H. Jung and K.-Y. Yoo, "Data hiding method using image interpolation," *Comput. Stand. Interfaces*, vol. 31, no. 2, pp. 465–470, 2009. https://doi.org/10.1016/j.csi.2008.06.001