ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse124932020.pdf
https://doi.org/10.30534/ijatcse/2020/124932020

An Efficient Least Significant Bit Image Steganography with Secret Writing and Compression Techniques

Jan Carlo T. Arroyo[1], Jenny A. Espadero[2], Marife A. Ganas[3], Randy F. Ardeña[4], Ramcis N. Vilchez[5], Allemar Jhone P. Delima[6]
[1-6]College of Computing Education, University of Mindanao, Davao City, Davao del Sur, Philippines
[6]College of Engineering, Technology and Management, Cebu Technological University-Barili Campus, Cebu, Philippines
jancarlo_arroyo@umindanao.edu.ph[1], jespadero@umindanao.edu.ph[2], marife_ganas@umindanao.edu.ph[3], randy_ardena@umindanao.edu.ph[4], ramcis_vilchez@umindanao.edu.ph[5], allemardelima@umindanao.edu.ph[6]

ABSTRACT

This paper improves information security on the basis of the traditional LSB steganography by the integration of cryptography and data compression techniques. The proposed method improves the security of information hiding by introducing multiple layers of security processes such as encryption and decryption, compression, and data embedding technique, respectively. This paper employed the Vigenere cipher for the encryption and decryption of the secret file where the generated ciphertext is compressed using the Huffman coding algorithm. The proposed method is tested on three image couriers embedded with 16kB, 32kB, and 48kB secret messages. The empirical results show that the proposed methodology is more competent compared to the traditional LSB image steganography alone with respect to imperceptibility by Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM), stego image file size, and Mean Square Error (MSE) metrics.

Key words: Cryptography, Huffman coding algorithm, least significant bit algorithm, Vigenere cipher, steganography

1. INTRODUCTION

The number of internet users increases over time. With this, the increase of information shared over wireless media is inevitable; hence, the upsurge of cybercrimes and threat of malicious access [1]. The two of the most regarded techniques used for information security are cryptography and steganography [2]. The combination of both technologies can be a prime solution to strengthen security and maintain the confidentiality of data [3].

Cryptography is the method of secret writing while the steganography is the scientific discipline of data hiding over other forms of media [4]. In cryptography, important data are transformed into an unintelligible format so that only the intended user can access the file only after the decryption process is successful [5], [6]. Meanwhile, steganography does not keep important data secret, but it provides secrecy of data by embedding the secret file to non-secret files like images, text, audio, and video [7], [8].

In this study, both cryptography and steganography were utilized in order to come up with a more amplified information security measures. The use of Vigenere cipher [9] was observed in order to transform the secret message into ciphertext. This study also uses a compression technique using the Huffman coding algorithm [10] in order to save storage costs. The ciphertext generated using the Vigenere cipher is compressed. The combination of both techniques ensures the secrecy of data as contents are imperceptible without earlier knowledge of decrypting rules and the compression technique. Subsequently, the compressed file is embedded in an image using the Least Significant Bit (LSB) algorithm [11]. The rest of the paper is outlined as follows: Section 2 discusses the existing algorithms used in this study. Section 3 includes a discussion of the proposed methodology. Section 4 presents the results and discussion, while Section 5 highlights the conclusion.

2. RELATED LITERATURE

2.1 Vigenere Cipher

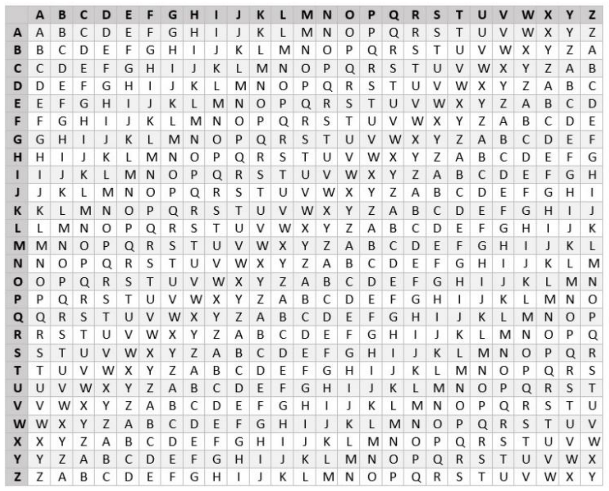
The Vigenere cipher is a simple polyalphabetic encryption technique which is based on a series of 26 Caesar ciphers [12], [13]. The algorithm uses a 26x26 matrix wherein rows and columns are represented by the characters A to Z. Each row of the matrix has the 26 letters of the alphabet, which is shifted once to the right in a cyclic manner. A sample Vigenere matrix is shown in Figure 1. Encryption and decryption using the algorithm require a secret keyword matched to each character of the plaintext.

For instance, the plaintext EXECUTIVE is encrypted using the secret keyword KEY. Every character from the keyword is matched repeatedly with each character in the plaintext. Next, each plaintext character paired with its corresponding keyword character is used as row and column lookup values, respectively. Based on the matrix, the first plaintext character E is substituted as O by referring to row E and column K. Similarly, for the second character X, it is substituted as B by referring to row X and column E., The results of encrypting the rest of the plaintext is shown in Table 1.

Table 1: Encryption using Vigenere cipher

Plaintext	E	X	E	C	U	T	I	V	E
Keyword	K	E	Y	K	E	Y	K	E	Y
Ciphertext	O	B	C	M	Y	R	S	Z	C

To decrypt, each ciphertext character is paired with the characters in the keyword. Each keyword character is used to locate the corresponding row, and the heading of the column that contains the ciphertext letter is the equivalent plaintext character. For example, the first ciphertext character O is translated as E by referring to row K and the cell with the character O (column E).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 1: Vigenere matrix

2.2 Huffman Coding

David A. Huffman developed an optimal prefix code used for lossless data compression called Huffman coding [14], [15]. The algorithm introduces variable-length codewords in character substitution, based on a table derived from the frequency rate of characters from a plaintext. With Huffman coding, the symbols which frequently occur are represented with fewer bits, while those less frequent symbols are represented with more bits. For instance, a file containing 100,000 characters A, B, C, D, E, and F is encoded. The frequency count and their equivalent codewords are presented in Table 2.

Table 2: Character frequency and equivalent codewords

	A	B	C	D	E	F
Frequency (in thousands)	49	17	12	10	6	4
Fixed-length codeword	000	001	010	011	100	101
Variable length codeword	0	100	101	110	1110	1111

When the file is encoded using a 3-bit fixed-length codeword representation, it uses 300,000 bits. However, if it is encoded using variable-length codewords, the message is encoded in 212,000 bits only, such that (49 * 1 + 17 * 3 + 14 * 3 + 10 * 3 + 6 * 4 + 4 * 4) * 1,000 = 212,000 bits. The use of Huffman coding in this example allows a saving of approximately 29% of space.

In generating codewords, the Huffman coding algorithm uses a binary tree based on the frequency count of symbols. First, a leaf node is created for every symbol and is added to the queue. Next, new internal nodes are created from the two nodes called child with a frequency equal to the sum of the two nodes' frequency. After, new nodes are added to the queue. The process is repeated while there are still nodes in the queue. The last remaining node is the root node, and the binary tree is complete. Based on the given example, the tree and the generated codewords are shown in Figure 2.

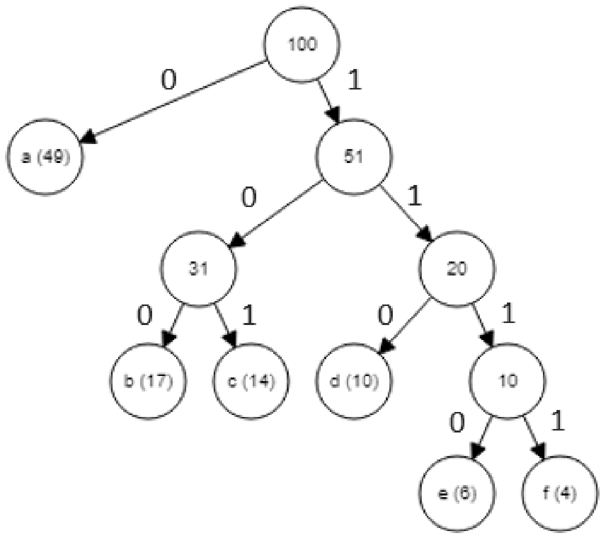


Figure 2: Huffman binary tree

2.3 Least Significant Bit in Image Steganography

LSB is a renowned technique in steganography known for its straightforward process in embedding crucial data in other objects by replacing some of the least significant bits of a cover file [16]–[20]. LSB in image steganography functions in a way that slight alterations done to images are not obvious using the naked eye.

LSB works by modifying each pixel of the image through its RGB color space. Since every RGB component is composed of 8 bits of memory, LSB modifies 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 3.

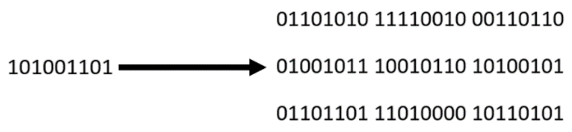


Figure 3: Embedding message to pixels

The bits from the message replace the least significant bits of each RGB component. If the LSB is identical to the message bit, it is skipped; otherwise, it is replaced. For instance, the

9-bit message was embedded in the RGC component sequence at the cost of replacing 4 bits (shown in red), as shown in Figure 4.

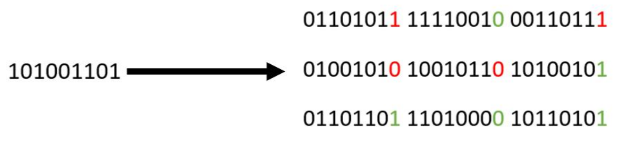


Figure 4: Embedded message using LSB

3. PROPOSED METHODOLOGY

The encrypt-compress-embed technique proposed in this study involves the use of the Vigenere cipher for encryption, Huffman coding algorithm for compression, and LSB method for data insertion. The flowchart of the proposed process is shown in Figure 5.

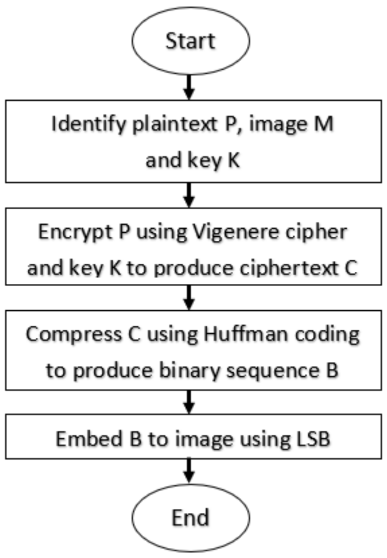

Start
Identify plaintext P, image M and key K
Encrypt P using Vigenere cipher and key K to produce ciphertext C
Compress C using Huffman coding to produce binary sequence B
Embed B to image using LSB
End

Figure 5: Encoding a message using the proposed method

To encode a message using the proposed method, the following steps are detailed as follows:

a. Identify a plaintext, cover image, and key.
b. Encrypt the plaintext using Vigenere cipher and the key
d. Compress ciphertext using Huffman Coding
e. Embed the binary sequence result to the image by traversing through each pixel and replacing the LSB.

In decoding a hidden message using the proposed method, the steps are presented in Figure 6 and detailed as follows:

a. Identify the image and key.
b. Using the LSB method, retrieve the embedded binary sequence.
c. Decompress the sequence using Huffman coding
d. Decrypt ciphertext using Vigenere cipher and the key

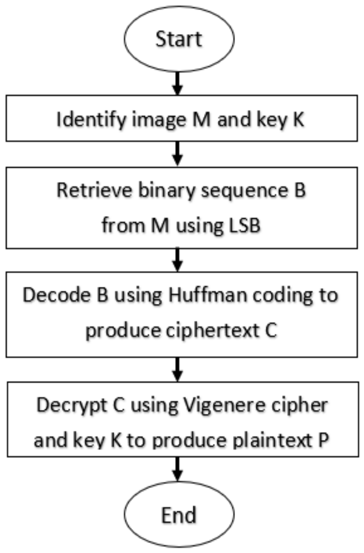

Start
Identify image M and key K
Retrieve binary sequence B from M using LSB
Decode B using Huffman coding to produce ciphertext C
Decrypt C using Vigenere cipher and key K to produce plaintext P
End

Figure 6: Decoding a message using the proposed method

The proposed method was implemented and tested using Python 3. Three sample images, namely Plane, Lena, and Peppers, shown in Figure 7, hereto referred as dataset 1, dataset 2, dataset 3, respectively, obtained from [21], [22] were used as the cover images. The specifications of each dataset are presented in Table 3. The size of the messages embedded was 16kB, 32kB, and 48kB. The key used for encryption and decryption is CIPHER. The simulation was executed in an i7-7000HQ 2.8 GHz 16GB RAM 4GBVRAM Windows 10 laptop computer. To assess the viability and performance of the proposed method, it was tested with the following metrics: Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM), and resulting file size. Results were then compared against the performance of using LSB alone.

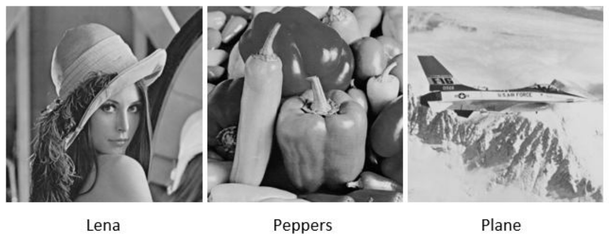


Figure 7: Testing dataset

Table 3: Dataset specifications

	Dimension	File type	Color mode	File Size
Dataset 1	512x512	PNG	Grayscale	290kB
Dataset 2	512x512	PNG	Grayscale	159kB
Dataset 3	512x512	PNG	Grayscale	240kB

The PSNR is used to assess the restoration quality of an image by identifying the amount of noise distortion between the original and the modified images [23], [24]. Having high PSNR means that there are lesser noise and good image restoration quality. The PSNR is defined by a mean squared error (MSE), which finds the magnitude of error between the images. To find the PSNR, the following equation is used:

$$PSNR - 10 \log \left[\frac{MAX_C^2}{MSE} \right] \qquad (1)$$

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

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

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*.

Another measure to test the viability of the proposed method is the use of the Structural Similarity Index (SSIM). SSIM is a metric that measures perceived alterations or degradation in the quality of images caused by modifications [25], [26]. Basically, this measure identifies how similar one image is to another. The SSIM is also known as an improvement to the PSNR metrics. To find the SSIM, the equation used is:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \qquad (3)$$

where μ_x is the average of *x*, μ_y is the average of *y*, σ_x^2 is the variance of *x*, σ_y^2 is the variance of *y*, σ_{xy} is the covariance of *x* and *y*, $c_1 = k_1L^2$ $c_2 = k_2L^2$ are two variables to stabilize the division with the weak denominator, *L* is the dynamic range of the pixel-values, $k_1 = 0.01$ $k_2 = 0.03$ by default. The closer the value of SSIM to 1, the more identical the two images are.

4. RESULTS AND DISCUSSION

The simulation results using both traditional LSB image steganography and the proposed method applied on Plane, Lena, and Pepper datasets are shown in this section. The histogram, PSNR, SSIM, MSE, and file size analyses for the used dataset are also presented.

The histogram of the original carrier and the stego images are shown in Figures 8-9. It is evident in the histograms shown in Figure 9 that the lone LSB method produced more noise as opposed to those of the proposed method. It can be observed that the noise becomes noticeable as the message size increases, as evident in the stego image generated using LSB alone.

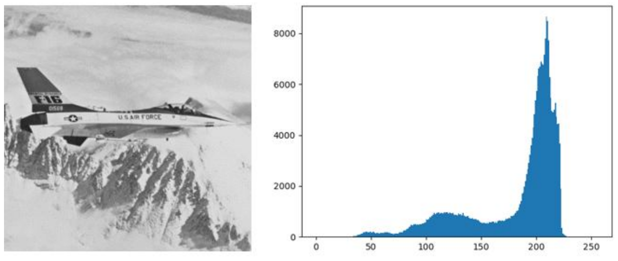


Figure 8: Dataset 1 original image carrier and its histogram

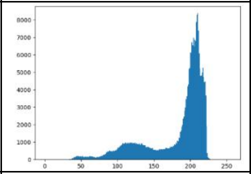
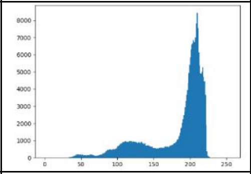
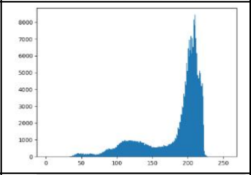
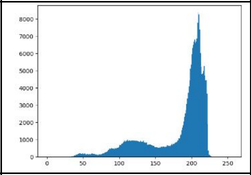
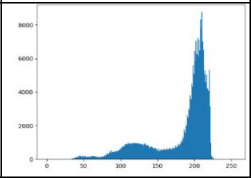
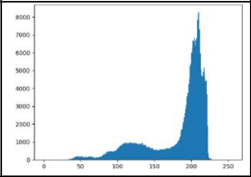
Message Size	Lone LSB	Proposed Method
16kB		
32kB		
48kB		

Figure 9: Histograms of dataset 1 stego images

The histograms of the original image carrier and the stego images generated using the dataset 2 are shown in Figures 10-11. Based on the empirical outcomes, it is noticeable that the lone LSB method showed more noise compared to the histogram of images produced by the proposed method. The proposed method generated better histograms, which are likely similar to the original image carrier despite being added with a secret message with increased size.

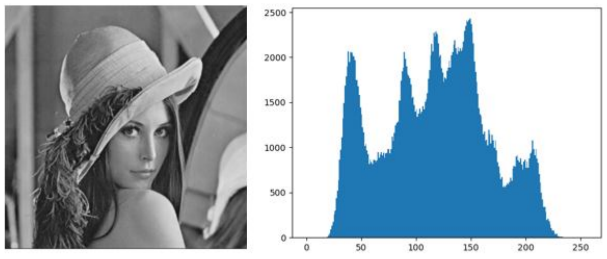


Figure 10: Dataset 2 original image carrier and its histogram

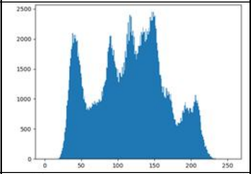
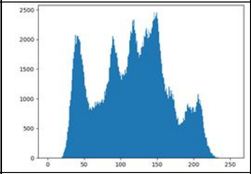
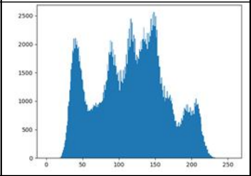
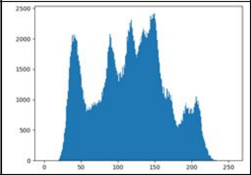
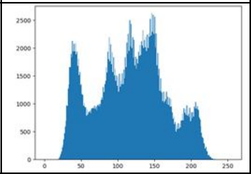
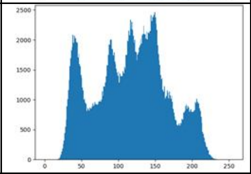
Message Size	Lone LSB	Proposed Method
16kb		
32kb		
48kb		

Figure 11: Histograms of dataset 2 stego images

The histograms of the stego images using dataset 3 and the original image carrier are shown in Figures 12 and 13, respectively. The results show that is proposed method generated a better stego image with lesser noise regardless of message size being embedded as compared to the stego image generated using the lone LSB.

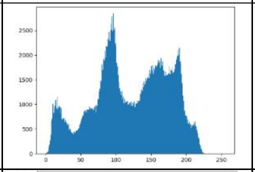
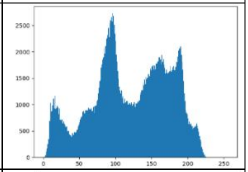
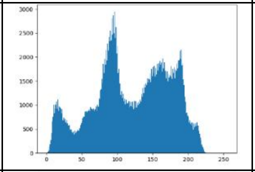
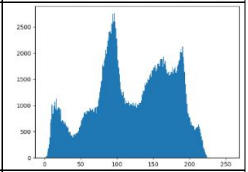
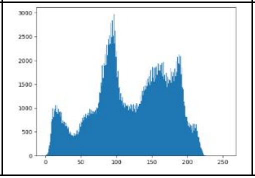
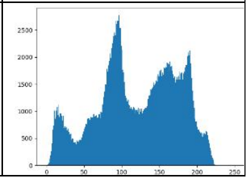
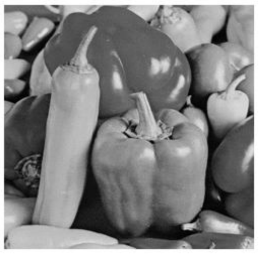
Message Size	Lone LSB	Proposed Method
16kb		
32kb		
48kb		

Figure 12: Histograms of dataset 3 stego images



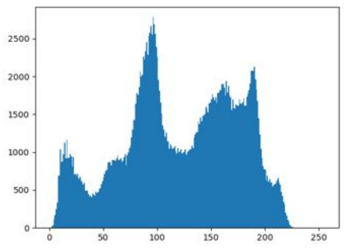


Figure 13: Dataset 3 original image carrier and its histogram

For dataset 1 image embedded with 16kB, 32kB, and 48kB messages, the proposed method gained better PSNR values at 61.3, 57.98, and 56.16 decibels (dB). The abovementioned values are higher than those of the stego image generated using the lone LSB, wherein the stego image with 16kB message obtained 58.66 dB PSNR, while 55.53 dB and 53.72 dB PSNR values for the images with 32kB and 48kB messages, respectively. Results show that the proposed method generated lesser noise against the lone LSB method. Moreover, the SSIM value of the proposed method is much closer to 1 compared to the lone LSB steganography technique, which means that the stego images generated using the proposed method is very close to the original image despite being embedded with a hidden message. On the extent of the file size, the proposed method produced stego images with smaller file sizes as against the method that uses LSB alone at around 4.9 to 11.7% savings. Further, the MSE statistical tool used revealed approximately 42-45% difference with error rates for the proposed method and the lone LSB, respectively. The summary of the results for dataset 1 is presented in Table 4.

Table 4: Dataset 1 PSNR, SSIM, MSE and file size results

Message Size	Metric	Lone LSB	Proposed Method	Variance
16kB	PSNR	58.66 dB	61.3 dB	4.50%
	SSIM	0.99954	0.99970	0.016%
	MSE	0.08835	0.04813	-45.52%
	File Size	315,430 B	299,967 B	-4.90%
32kB	PSNR	55.53 dB	57.98 dB	4.41%
	SSIM	0.9992	0.99948	0.028%
	MSE	0.18196	0.10332	-43.22%
	File Size	354,300 B	321,453 B	-9.27%
48kB	PSNR	53.72 dB	56.16 dB	4.54%
	SSIM	0.99899	0.99928	0.029%
	MSE	0.27568	0.15732	-42.93%
	File Size	392,125 B	346,024 B	-11.76%

In dataset 2, the LSB method gained more noise with lower PSNR values than the output of the proposed method at 3.9% to 4.3% variance. On the other hand, the SSIM of the proposed method is relatively higher than the lone traditional LSB steganography, which means that the resulting images are very similar to the original image. The proposed method also generated stego images, which are 8.9-11.85% smaller in terms of file size against the lone LSB method. Based on the statistical error test, the traditional LSB steganography obtained an error rate of at least 41.30% higher than the proposed methodology. The summary of the results for dataset 2 is shown in Table 5.

Table 5: Dataset 2 PSNR, SSIM, MSE and file size results

Message Size	Metric	Lone LSB	Proposed Method	Variance
16kB	PSNR	58.42 dB	60.73 dB	3.95%
	SSIM	0.99936	0.99980	0.044%
	MSE	0.09353	0.05490	-41.30%
	File Size	319,719 B	304,001 B	-4.92%
32kB	PSNR	55.41 dB	57.73 dB	4.19%
	SSIM	0.99932	0.99957	0.025%
	MSE	0.18701	0.10944	-41.48%
	File Size	357,277 B	325,346 B	-8.94%
48kB	PSNR	53.65 dB	55.96 dB	4.31%
	SSIM	0.99899	0.99940	0.041%
	MSE	0.28058	0.16450	-41.37%
	File Size	396,543 B	349,549 B	-11.85%

The embedding of 16kB, 32kB, and 48kB secret messages in dataset 3 affirmed that the proposed method performed better in terms of PSNR, SSIM, MSE, and file size. For the PSNR metric, the proposed method gained a higher score of 60.75 dB, 57.74 dB, and 55.97 dB as compared to the lone LSB with 58.40 dB, 55.39 dB and 53.63, respectively. In terms of the similarity between the images, the proposed method gained values that are closer to 1; thus, the images are more identical to the original image. The files generated by the proposed method also has smaller file sizes than that of the lone LSB with around 4.7 to 12.5% diminution. As for the MSE metric, the lone LSB has a higher error rate at 41% as compared to the proposed method. The summary of the results for dataset 3 is presented in Table 6.

Table 6: Dataset 3 PSNR, SSIM, MSE and file size results

Message Size	Metric	Lone LSB	Proposed Method	Variance
16kB	PSNR	58.40 dB	60.75 dB	4.02%
	SSIM	0.99947	0.99971	0.024%
	MSE	0.09386	0.05463	-41.80%
	File Size	276,073 B	262,885 B	-4.78%
32kB	PSNR	55.39 dB	57.74 dB	4.24%
	SSIM	0.99884	0.99936	0.052%
	MSE	0.18776	0.10920	-41.84%
	File Size	311,348 B	284,679 B	-8.57%
48kB	PSNR	53.63 dB	55.97 dB	4.36%
	SSIM	0.99853	0.99899	0.046%
	MSE	0.28162	0.16437	-41.63%
	File Size	347,495 B	303,964 B	-12.53%

5. CONCLUSION

The data secrecy in this method is robust because multiple layers of security are introduced where cipher, compression, and file embedding processes were undertaken. Simulation results revealed that the proposed method generates stego images with higher PSNR and SSIM values with lower MSE rates and file sizes against the stego images generated using the lone traditional LSB steganography. In all test cases, the proposed method shows superiority when it comes to generating stego images, as evident in the results of the specified metrics used in this study.

REFERENCES

[1] S. Chauhan, Jyotsna, J. Kumar, and A. Doegar, “Multiple layer Text security using Variable block size Cryptography and Image Steganography,” in *3rd IEEE International Conference on Computational Intelligence and Communication Technology*, 2017, pp. 1–7. https://doi.org/10.1109/CIACT.2017.7977303

[2] D. Seth, L. Ramanathan, and A. Pandey, “Security Enhancement: Combining Cryptography and Steganography,” *Int. J. Comput. Appl.*, vol. 9, no. 11, pp. 3–6, 2010. https://doi.org/10.5120/1433-1932

[3] S. Singh and A. Singh, “An Information Security Technique Using DES-RSA Hybrid and LSB,” *Int. J. Emerg. Technol. Comput. Appl. Sci.*, vol. 14, no. 355, pp. 187–192, 2013.

[4] A. D. P. Ariyanto, E. H. Rachmawanto, D. R. I. M. Setiadi, and C. A. Sari, “Performance Analysis of LSB Image Steganography Combined with Blowfish-RC4 Encryption in Various File Extensions,” *Proc. 2019 4th Int. Conf. Informatics Comput.*, 2019.

[5] S. N. Gowda, “Innovative enhancement of the Caesar cipher algorithm for cryptography,” in *International Conference on Advances in Computing, Communication and Automation*, 2016.

[6] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre, “From identification to signatures via the Fiat-Shamir transform: Necessary and sufficient conditions for security and forward-security,” *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3631–3646, 2008. https://doi.org/10.1109/TIT.2008.926303

[7] M. O. Espina, A. C. Fajardo, B. D. Gerardo, and R. P. Medina, “Multiple level information security using image steganography and authentication,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 6, pp. 3297–3303, 2019. https://doi.org/10.30534/ijatcse/2019/100862019

[8] M. Espina, A. C. Fajardo, B. D. Gerardo, and R. P. Medina, “A novel puzzle-based image steganography technique,” in *Eleventh International Conference on Graphics and Image Processing*, 2020, vol. 1137317, no. 1.

[9] A. D. Achmad, A. A. Dewi, M. R. Purwanto, P. T. Nguyen, and I. Sujono, “Implementation of vigenere cipher as cryptographic algorithm in securing text data transmission,” *J. Crit. Rev.*, vol. 7, no. 1, pp. 76–79, 2020.

[10] A. Moffat, “Huffman coding,” *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–35, 2019. https://doi.org/10.1145/3342555

[11] X. Zhou, W. Gong, W. Fu, and L. Jin, “An improved method for LSB based color image steganography combined with cryptography,” in *IEEE/ACIS 15th International Conference on Computer and Information Science*, 2016, pp. 4–7.

[12] D. Kahn, *Codebreakers*. Macmillan and Sons, 1967.

[13] D. Salomon, *Coding for Data and Computer Communication*. Springer, 2005.

[14] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952. https://doi.org/10.1109/JRPROC.1952.273898

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Huffman codes,” in *Introduction to Algorithms*, 2009, pp. 428–437.

[16] 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.

[17] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, Second Edi. Burlington: Morgan Kaufmann, 2008. https://doi.org/10.1016/B978-012372585-1.50015-2

[18] 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.

[19] 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.

[20] 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. https://doi.org/10.3844/ajassp.2005.648.654

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

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

[23] 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

[24] G. Swain and S. Lenka, "Classification image steganography techniques in spatial domain: A study," *Int J Comput Sci Eng Tech*, vol. 5, pp. 219–232, Jan. 2014.

[25] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[26] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009. https://doi.org/10.1109/MSP.2008.930649