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/ijatcse137932020.pdf
https://doi.org/10.30534/ijatcse/2020/137932020

An Improved Image Steganography through Least Significant Bit Embedding Technique with Data Encryption and Compression Using Polybius Cipher and Huffman Coding Algorithm

Jan Carlo T. Arroyo[1], Charisse P. Barbosa[2], Meljohn V. Aborde[3], Fe B. Yara[4], Allemar Jhone P. Delima[5]
[1-5]College of Computing Education, University of Mindanao, Davao City, Davao del Sur, Philippines
[5]College of Engineering, Technology and Management, Cebu Technological University-Barili Campus, Cebu, Philippines
jancarlo_arroyo@umindanao.edu.ph[1], charisbarbosa@umindanao.edu.ph[2], mjaborde@umindanao.edu.ph[3], fe_yara@umindanao.edu.ph[4], allemardelima@umindanao.edu.ph[5]

ABSTRACT

This paper proposed an enhanced data handling technique by introducing multiple layers of security in securing sensitive text data, which is realized by using both cryptography and steganography techniques. The proposed method is cost-effective as the data compression technique is observed. This study uses Polybius cipher to transform plaintext containing vital information into an unintelligible format called ciphertext. The ciphertext is compressed using the Huffman coding algorithm, where output is embedded in an image file using the Least Significant Bit (LSB) algorithm for the steganography technique. Simulation results show that the proposed methodology produced stego images with better performance as to file size, Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM), and error metrics as against the stego image generated using the lone LSB steganography technique.

Key words: Cryptography, Huffman code, least significant bit, Polybius square, steganography

1. INTRODUCTION

In today's digital age, the concern for information security has become more significant as the exchange of essential information through computer and internet media has gained the attention and interest of attackers. The rapid increase of attacks on the electronic exchange of information has drawn concern that calls for a more robust method of data transfer and communication security [1].

The regarded solution for the abovementioned concern has paved the way for the development of cryptography and steganography [2]. Cryptography is the science and mathematics of hiding and obscuring information into an unintelligible format as protection to adversaries. Cryptography is categorized into two: symmetric and asymmetric key cryptography [3], [4]. The former introduces a single key concept that is instrumental for the cipher process using different cipher techniques while the latter uses a separate key for both encryption and decryption processes. The graphical representation of cryptographic types and their key distribution concepts are shown in Figures 1-3.

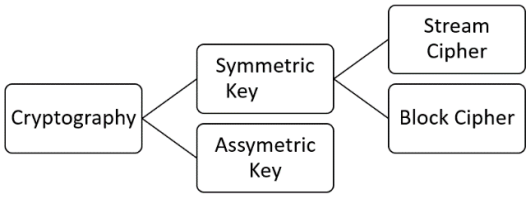

Figure 1: Categories of cryptography

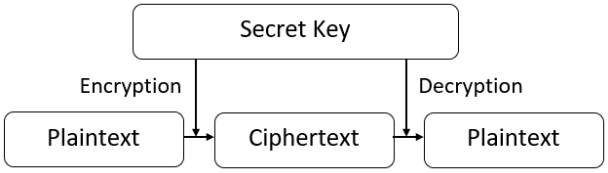

Figure 2: Symmetric key cryptography concept

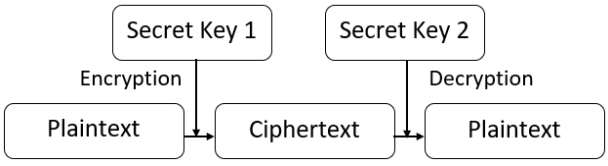

Figure 3: Asymmetric key cryptography concept

Steganography, on the other hand, is a technique of covered writing that embeds secret files to other non-secret files. The file that contains the secret data is called carriers. Carriers of secret data can be text, image, audio, and video files. Modified carrier, despite having embedded files on it, shows no trace of alteration and looks like the original carrier [5], [6]. Steganography is divided into different types. Some of the following techniques include (1) text steganography,

where information is hidden in the text file; (2) audio steganography, where secret information is attached to audio files; (3) video steganography, where secret information is hidden in digital video format; and (4) image steganography, where secret information is hidden behind pixels of colored or gray images [7], [8].

However, cryptography or steganography is not capable of protecting data alone. With the combination of both technologies in one system, enhanced data security and transmission processes are ensured [9], [5], thus, this study. To reduce storage cost and transmission time, a technique called data compression is done. The remaining sections of this paper are organized as follows: Section 2 introduces the existing algorithms to be used in this study. The proposed methodology is outlined in Section 3. Section 4 presents the results and discussion of the proposed work, and Section 5 shows the conclusion.

2. RELATED LITERATURE

2.1 Polybius Cipher

The Polybius Cipher uses a 5x5 square matrix wherein characters are placed in alphabetical order from left to right, then top to bottom [10]–[13], as shown in Table 1. Cells in the matrix are identified corresponding to their relative indices in the grid represented by the combination of the row and column number.

Table 1: Traditional Polybius square matrix

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Encryption and decryption using the Polybius cipher are relatively easy since no key is used for this technique. In encrypting a plaintext, characters are matched with the matrix to retrieve their equivalent bigrams. Bigrams represent the character coordinates in the matrix based on the intersection of rows and columns. For instance, the plaintext COMPUTING is encrypted as 133432354544243322, wherein C is found in row 1 column 3; thus, C is represented as the bigram 13. The ciphertext is presented in Table 2.

Table 2: Encryption using traditional Polybius square

Plaintext	C	O	M	P	U	T	I	N	G
Position	0	1	2	3	4	5	6	7	8
Ciphertext	13	34	32	35	45	44	24	33	22

The decryption process is done by comparing each bigram to the grid to retrieve its equivalent plaintext value. For example, the ciphertext 133432354544243322 is translated into plaintext COMPUTING. The results of the decryption process are shown in Table 3.

Table 3: Decryption using traditional Polybius square

Ciphertext	13	34	32	35	45	44	24	33	22
Position	0	1	2	3	4	5	6	7	8
Plaintext	C	O	M	P	U	T	I	N	G

2.2 Huffman Coding

The Huffman coding, developed by David A. Huffman, is an optimal prefix code used for lossless data compression [14], [15]. The algorithm uses variable-length codewords in substitution, based on a table derived from the occurrence frequency of characters from the data. The most frequent symbols are represented with fewer bits. For instance, a 100,000-character data file containing the letters A to F is encoded. The frequency count and equivalent codewords are presented in Table 4.

Table 4: Frequency and codewords

	A	B	C	D	E	F
Frequency (in thousands)	49	17	14	10	6	4
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	100	101	110	1110	1111

If a 3-bit fixed-length codeword representation is used, the file can be encoded in 300,000 bits. However, using a variable-length codeword allows the message to be encoded in only 212,000 bits, wherein (49 * 1 + 17 * 3 + 14 * 3 + 10 * 3 + 6 * 4 + 4 * 4) * 1,000 = 212,000 bits. This optimal method saves approximately 29% of space.

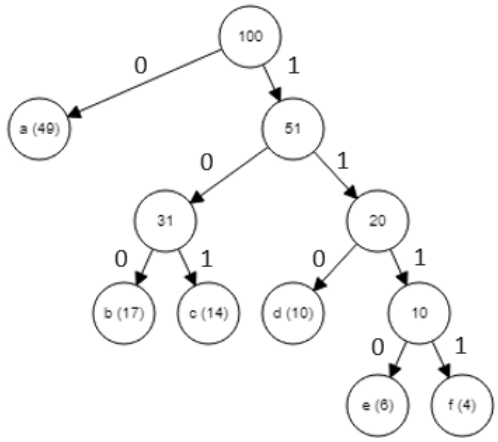

Figure 4: Huffman binary tree

The Huffman coding algorithm uses a binary tree to generate a codeword for a specific symbol based on the character frequency count. First, create a leaf node for each symbol and add it to the queue. Next, create a new internal node with these two nodes as children and with a frequency equal to the sum of the two nodes' frequency. After, add the new node to the queue. Repeat the process while there is are still nodes in the queue. The remaining node is the root node, and the Huffman binary tree is complete. With the given example, the tree and the generated code words are shown in Figure 4.

2.3 Least Significant Bit in Image Steganography

LSB in steganography is a renowned technique known for its simplicity in embedding sensitive data in other objects by replacing some of the least significant bits of a cover file [16]–[20]. Like other steganographic algorithms, LSB used

in image steganography performs in such a way that minor modifications made to pictures are not noticeable using the naked eye.

LSB works by altering each pixel of the image through its RGB color space. Since every RGB component is composed 8 bits of memory, LSB manipulates 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 5.

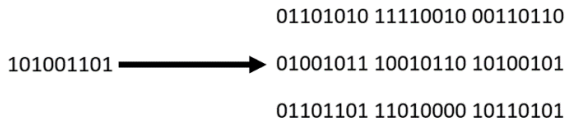

Figure 5: Embedding message to pixels

The bits from the message replace the least significant bits of each RGB component. If the LSB is equal to the message bit, it is skipped; otherwise, it is substituted. Based on the example, 9-bits of data was embedded in the sequence at the expense of masking 4 of bits (shown in red) as presented in Figure 6.

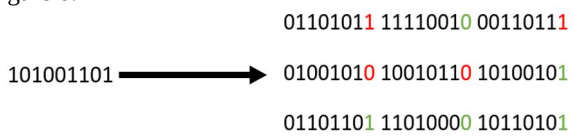

Figure 6: Embedded message using LSB

3. PROPOSED METHODOLOGY

The proposed method follows the encrypt-compress-embed technique. It involves the use of the Polybius cipher for encryption and the Huffman Coding algorithm for text compression. After, the result is embedded in an image using the LSB method. The flowchart of the proposed process is presented in Figure 7.

To perform the proposed method, the following steps are executed as follows:

a. Identify a plaintext, cover image, and key.
b. Using the key, generate a Polybius square.
c. Encrypt the plaintext using Polybius cipher and the generated matrix
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 8 and detailed as follows:

a. Identify the image and key.
b. Using the LSB method, retrieve the embedded binary sequence.
e. Decompress the sequence using Huffman coding
f. Generate the Polybius square using the key input
h. Decrypt ciphertext using Polybius cipher and the generated matrix.

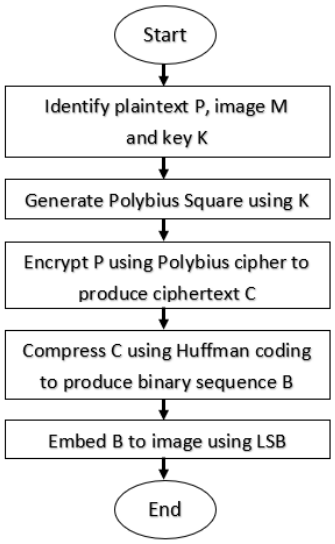

Figure 7: Encoding a message using the proposed method

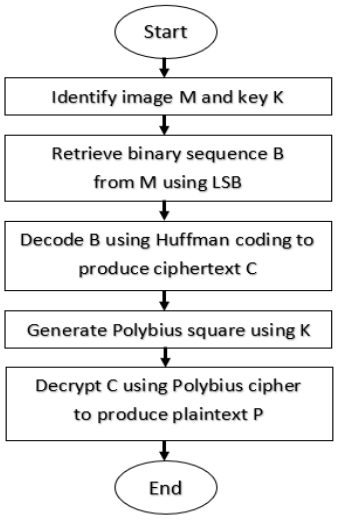

Figure 8: Decoding a message using the proposed method

The proposed method was implemented in Python. The sample Lena, Peppers, and Plane images shown in Figure 9, hereto referred as dataset 1, dataset 2, and dataset 3, respectively, downloaded from [21], [22] were utilized. The specifications of each dataset are presented in Table 5. The sizes of the messages embedded are 16kb, 32kb, and 48kb. The key used to generate the Polybius is CIPHER. 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), Structural Similarity Index (SSIM), and compression rate tests were executed to validate the feasibility of the proposed method.

Table 5: 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

Figure 9: Testing dataset

The PSNR is used to assess the quality of an image by comparing the amount of distortion between the original and the altered images [23], [24]. If the value of the PSNR is high, it means that there are lesser noise and good image restoration quality. A PSNR of 100 denotes 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 expressed as:

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

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

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

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*.

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 changes 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 calculated as:

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

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

4.1 Image Steganography Using Dataset 1

Figures 10-12 show the histogram of the original and stego images embedded with 16kb, 32kb, and 48kb secret messages generated using the lone LSB and the proposed method. In plain view, results show that there is no significant difference between the original image and the stego image. However, it is evident in the histogram results shown in Tables 6-8 that significant changes were made to the images wherein the lone LSB method had more noise since it obtained lesser PSNR value as compared to the proposed method.

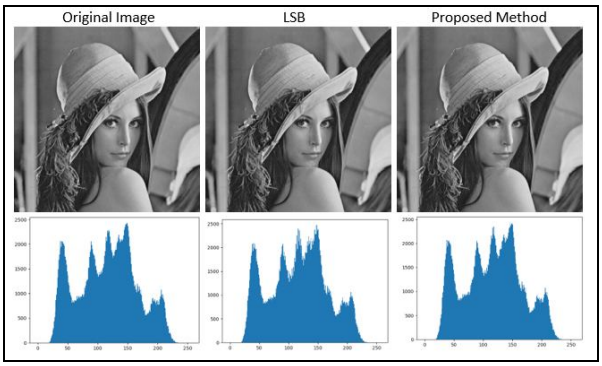

Figure 10: Dataset 1 with 16kb secret message

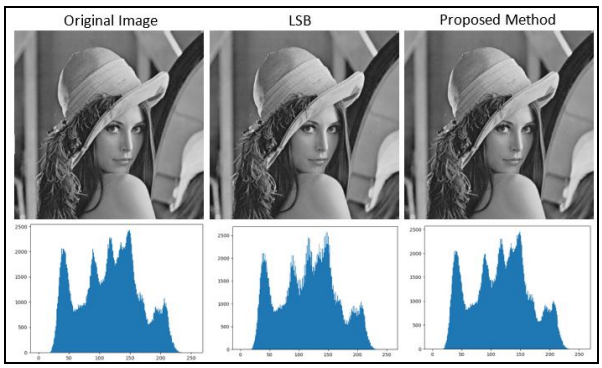

Figure 11: Dataset 1 with 32kb secret message

By embedding a 16kb message on the image, the lone LSB method produced a PSNR of 58.62 decibels (dB), which is 10% lower than the 61.27 dB generated using the proposed method. This goes to show that the lone LSB image steganography has more noise than the proposed method. Further, the SSIM value of the proposed method is closer to 1 as against the lone LSB steganography technique, which means that the generated stego image is almost identical to the original image despite being embedded with a secret file. Extent on the size of the files, the proposed method generated a stego image with a smaller file size as against the method that uses LSB alone. Both methods generated stego images with 300,005 bytes and 314,615 bytes, respectively. Further, the MSE statistical tool used revealed a 45% difference with 0.0485 and 0.089 error rates for the proposed method and the lone LSB, respectively. The summary of results is presented in Table 6.

Table 6: Dataset 1 with 16kb message indexed result

	LSB	Proposed Method	Variance
PSNR	58.62	61.27	+10.15%
MSE	0.089	0.0485	-45.50%
File Size	314,615	300,005	-4.64%
SSIM	0.99954	0.99970	+0.16%

When a 32kb message is encoded in dataset 1, the stego image generated using LSB, and the proposed method shows no visible trace of modifications when inspected by the naked eye. However, significant changes were made to the images wherein the lone LSB method gained more noise since it obtained a PSNR value of 55.52 dB as compared to the proposed method with 57.98 dB. As for the file sizes, the proposed method generated a stego image with 321,444 bytes, which is 8.69% lower as against the lone LSB method with 352,049 bytes stego file size. The SSIM value of the proposed method is also closer to 1, indicating similarity to the original image. Based on the statistical error test, the lone LSB method obtained an error rate that is 43.40% higher than the proposed methodology, as evident in Table 7.

Table 7: Dataset 1 with 32kb message indexed result

	LSB	Proposed Method	Variance
PSNR	55.52	57.98	+4.43%
MSE	0.182	0.103	-43.40%
File Size	352,049	321,444	-8.69%
SSIM	0.99920	0.99948	+0.02%

After embedding a 48kb message in dataset 1, the proposed method still shows no visible trace of modifications that is perceivable by the naked eye. However, extent on the noise of the two stego images, the stego image generated by the lone LSB method shows higher noise as against the proposed method. Further, the proposed method generated an output file with lesser file size having 346,227 bytes as against the 389,784 bytes stego image of the lone LSB. Furthermore, the MSE and SSIM value of the stego image generated using the proposed methodology show better results against the LSB method alone. The summary of results is presented in Table 8.

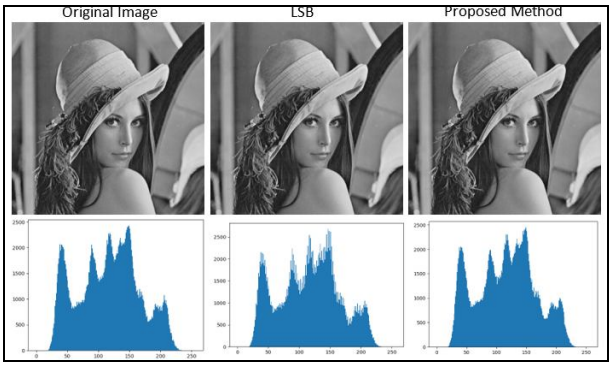

Figure 12: Dataset 1 with 48kb secret message

Table 8: Dataset 1 with 48kb message indexed result

	LSB	Proposed Method	Variance
PSNR	55.52	56.11	+1.06%
MSE	0.275	0.159	-42.18%
File Size	389,787	346,277	-11.16%
SSIM	0.99889	0.99927	+0.03%

These findings show that the proposed method gains higher PSNR and SSIM values, lower error percentages, and smaller file sizes in all of the test cases, which therefore denotes higher quality images with lesser noise and a cost-effective method.

4.2 Image Steganography Using Dataset 2

The histogram, PSNR, SSIM, MSE, and file size analyses for the Peppers image dataset embedded with 16kb, 32kb, and 48kb secret messages encoded using LSB, and the proposed method are presented in Figures 13-15 and Table 9. The histogram of both stego images generated using LSB and the proposed method shows no visible trace of modifications when compared to the carrier. However, it is evident in the PSNR and SSIM values, and stego image file sizes that the proposed method performed better than the LSB alone. Further, the proposed method has produced lower MSE values as compared to the lone LSB method.

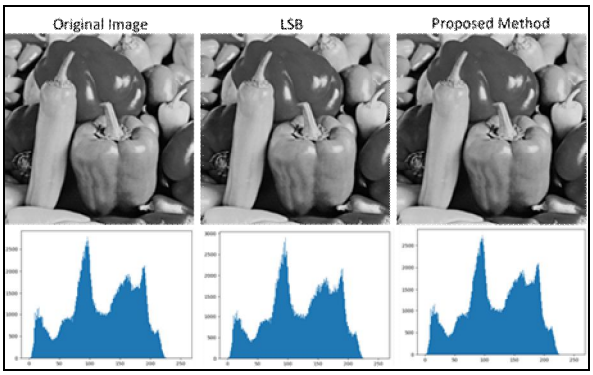

Figure 13: Dataset 2 with 16kb secret message

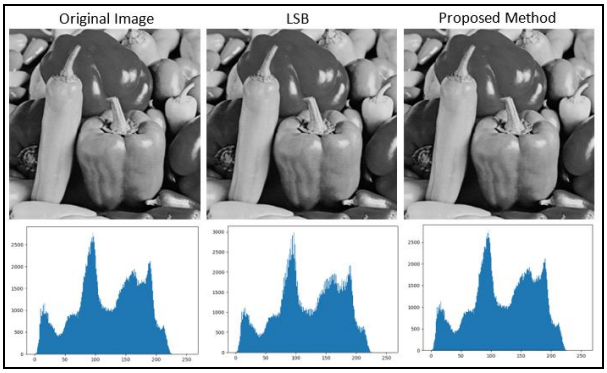

Figure 14: Dataset 2 with 32kb secret message

With a 16kb secret message embedded to the carrier, a stego image with a file size of 303,958 bytes was generated using the proposed method, which is 4.57% lower than the 318,531 bytes stego image output of the lone LSB. Further, the lone LSB image steganography technique produces a stego image with 58.42 dB PSNR, while the proposed method produces an output that has a PSNR value that is 3.95% higher. As for the SSIM value, the proposed method produced an output that is very close to the original image as it obtained an SSIM value of 0.99980 as against the 0.99963 of the lone LSB method.

When embedded with a 32kb secret message, the lone LSB method produced a stego image with 55.40 dB PSNR, while the proposed method has a stego image that has lesser noise with PSNR value of 55.70 dB which is 4.15% higher than the former.

By adding a 48kb secret message on the dataset 2, the lone LSB method produced an SSIM value of 0.99899. However, the SSIM value of the stego image generated by the proposed method is much closer to 1 with a value of 0.99939, which denotes insignificant change compared to the carrier. The MSE values of stego images generated by the lone LSB image steganography and the proposed method when embedded with 32kb and 48kb secret messages show minimal error rates with 41.17% variance for the former and 40.71% for the latter. The index comparison of the stego images with their corresponding metric values is shown in Table 9.

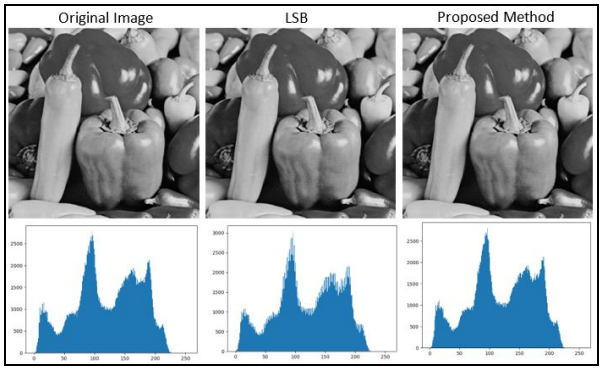

Figure 15: Dataset 2 with 48kb secret message

Table 9: Indexed simulation results using dataset 2

	LSB	Proposed Method	Variance
Dataset 2 embedded with 16kb secret message			
PSNR	58.42	60.73	+3.95%
MSE	0.093	0.054	-41.93%
File Size	318,531	303,958	-4.57%
SSIM	0.99963	0.99980	+0.01%
Dataset 2 embedded with 32kb secret message			
PSNR	55.40	57.70	+4.15%
MSE	0.187	0.110	-41.17%
File Size	353,934	325,437	-8.05%
SSIM	0.99932	0.99957	+0.02%
Dataset 2 embedded with 48kb secret message			
PSNR	53.65	55.92	+4.23%
MSE	0.280	0.166	-40.71%
File Size	394,067	349,686	-11.26%
SSIM	0.99899	0.99939	+0.04%

Findings show that the proposed method gains higher PSNR and SSIM values with lower error percentages and smaller file sizes in all test cases where dataset 2 is embedded with 16kb, 32kb, and 48kb secret messages.

4.3 Image Steganography Using Dataset 3

Figures 16-18 show the histogram of the original and stego images embedded with 16kb, 32kb, and 48kb secret message generated using the proposed method and the lone LSB image steganography. In plain view, results show that there is no significant difference between the carrier and stego images. However, it is evident in the histogram analysis shown in Table 10 that significant changes were made to the images wherein the lone LSB method had more noise since it obtained lesser PSNR value as compared to the proposed method.

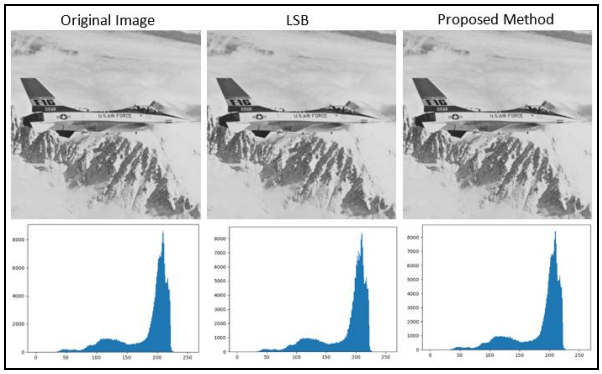

Figure 16: Dataset 3 with 16kb secret message

With a 16kb message embedded to dataset 3, the lone LSB method revealed a peak signal to noise ratio value of 58.41 dB. In comparison, the proposed method shows a 3.95% variance at 60.71 dB from both stego images generated. As for the file size, the proposed method produced a stego image with a smaller file size against the stego image generated by the lone LSB technique with a variance of 4.75%.

Further, the stego image with a 32kb secret message generated by the proposed method shows a structural similarity index value that is much closer to 1 as compared to the SSIM value of the stego image generated using the lone LSB method with 0.99936 and 0.99884 SSIM values, respectively. Both stego images obtained a lower mean square error with 0.187 and 0.110 error rates using the lone LSB and the proposed method.

Furthermore, by embedding a 48kb secret message on the carrier, the stego image generated by the lone LSB method revealed a 53.62 dB peak signal to noise ratio value. The stego image generated using the proposed method obtained a PSNR value of 55.95 dB. This denotes that the stego image generated by the lone LSB has more noise since it obtained lesser PSNR value when compared to the proposed method. The indexed comparison of the file sizes, PSNR, SSIM, and MSE values of the stego images generated by both techniques are shown in Table 10.

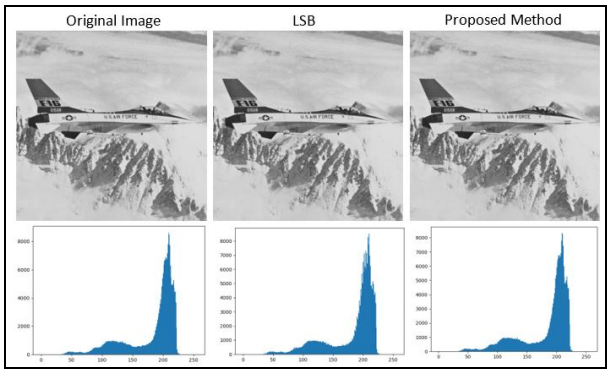

Figure 17: Dataset 3 with 32kb secret message

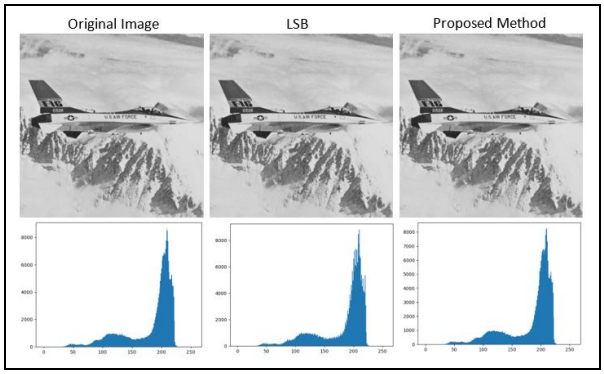

Figure 18: Dataset 3 with 48kb secret message

These findings show that the proposed method gains higher PSNR values, lower error percentages, and smaller file sizes in all cases, which therefore equates to higher quality images with lesser noise and better storage.

Table 10: Indexed simulation results using dataset 3

	LSB	Proposed Method	Variance
Dataset 3 embedded with 16kb secret message			
PSNR	58.41	60.71	+3.93%
MSE	0.093	0.055	-41.08%
File Size	276,049	262,949	-4.75%
SSIM	0.99947	0.99971	+0.02%
Dataset 3 embedded with 32kb secret message			
PSNR	55.39	57.67	+4.11%
MSE	0.187	0.110	-41.17%
File Size	309,434	284,626	-8.01%
SSIM	0.99884	0.99936	+0.05
Dataset 3 embedded with 48kb secret message			
PSNR	53.62	55.95	+4.34%
MSE	0.281	0.164	-41.63%
File Size	344,849	304,152	-11.80%
SSIM	0.99854	0.99898	+0.04%

5. CONCLUSION

In this paper, the combination of cryptography and steganography for increased data security and transmission efficiency is observed. To save storage costs, a lossless compression technique is done. The Polybius cipher was instrumental for the encryption and decryption of the secret message in the form of plaintext. The ciphertext generated by the Polybius square is compressed using the Huffman coding algorithm. The compressed unintelligible secret message is now embedded in an image using the least significant bit embedding technique. The proposed method has paved the way for a more secure data handling technique by introducing layers of security protocols. Simulation results revealed that the carrier, when applied with the proposed method, shows no trace of data alteration, hence embedding of the secret message is undetectable. Further, a reduction in file size was achieved with the use of the compression technique as against the image steganography using LSB alone.

REFERENCES

[1] J. V. Karthik and B. V. Reddy, "Authentication of secret information in image steganography," *Int. J. Latest Trends Eng. Technol.*, vol. 3, no. 1, pp. 97–104, 2013.

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

[4] 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

[5] 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.

[6] 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.

[7] R. J. Mstafa and K. M. Elleithy, "Compressed and raw video steganography techniques: a comprehensive survey and analysis," *Multimed. Tools Appl.*, vol. 76, pp. 21749–21786, 2017.

[8] S. M. Nasreen, G. Jalewal, and S. Sutradhar, "A Study on Video Steganographic Techniques," *Int. J. Comput. Eng. Res.*, vol. 5, no. 10, pp. 30–34, 2015.

[9] S. Mishra and P. Pandey, "A Review on Steganography Techniques using Cryptography," *Int. J. Adv. Res. Scinece Eng.*, vol. 4, no. Special Issue 1, pp. 1–4, 2015.

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

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

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

[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.

[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.

[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. https://doi.org/10.1147/sj.353.0313

[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.

[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.

[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. https://doi.org/10.1109/TIP.2003.819861

[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