Computer Engineering and Applications Vol. 13, No. 1, February 2024

Analysis and Implementation of Blowfish and LSB Algorithm on RGB Images using SHA-512

Ilham Firman Ashari*, Mugi Praseptiawan, Randi Baraku, Edinia Rosa Filiana

Informatics Engineering, Institut Teknologi Sumatera
**firman.ashari@if.itera.ac.id*

ABSTRACT

The growth of the internet globally keeps increasing as time goes. There's a big amount of data type saved there too. Those data need to be secured so anyone who doesn't have the right to access them can access it. The purpose of this article is to secure text information into image media using the Blowfish method for encrypting text information and securing it using the Hash function SHA-512 and then embedded it in image media using the Least Significant Bit (LSB) method. The result of implementing those methods using image media sized 138Kb and 39.85Kb with plaintext measuring 27 and 85 characters shows that integrity data is secured with SHA-512 method. The test result using PSNR method to get the score of image quality after embedding information to the image shows that the average number of PSNR's score is 70,74 dB which means the quality is good and has less difference from the original image.

Keywords: Analysis, Blowfish, LSB, PSNR, SHA-512

1. INTRODUCTION

The development of information technology is currently accompanied by an increasing amount of information and/or digital data in various forms stored on the internet. The internet itself is public so it can be accessed by many people from various circles. This causes digital data stored on the internet to be vulnerable to eavesdropping and theft by unauthorized parties [1]. To prevent theft and access of data by third parties, it is necessary to protect the data. By protecting the data, the data can only be accessed by those who have the authority to access or modify the data [2]. One way to protect digital data is to use the application of cryptography and steganography.

There are several studies that are related and become a reference for this research, both in terms of topic and method. Related research written by Angga Aditya Permana in 2018 [3]. Research conducted by Angga applies the Blowfish cryptographic algorithm and the Least Significant Bit steganography algorithm in the preparation of the application being built. The research shows that the use of the Least Significant Bit method has a great vulnerability to be extracted by people who are not authorized to use tools to perform steganization. However, adding an encryption method using Blowfish can help improve security. That's because even if someone else tries to

extract the message, they need to decrypt the message first to be able to get the information inside. The difference between this research and the research conducted is the security of information in the form of text messages that we use plus a hash function to ensure message integrity.

The second related research compiled by Rizki and Sri Mulyati in 2020 [4]. In his research, the researcher implemented the SHA-512 function to improve authentication security by creating a One Time Password system. The results showed that the application of SHA-512 was declared feasible to be applied because the Media Expert score reached 82% and the Material Expert score reached 75%. Third study by Haneen Alabdulrazzaq and Mohammed N. Alenezi in 2022 [5]. The aim of this research is to compare the speed of block cipher encryption algorithms. The results of the study based on several variations of comparative variables showed that of the 5 types of algorithms, namely DES, 3DES, Blowfish, Twofish, and Threefish, the Blowfish algorithm was superior to other algorithms.

Fourth related research by Jai Verma in 2021[6]. This study aims to review how well the SHA-512 Algorithm can be applied in information security at this time. The results of research conducted by researchers by comparing 6 studies that apply SHA-512 get the conclusion that SHA-512 is the safest type of SHA Hash function algorithm for encrypting user data. Based on the Penetration Test against brute force attacks, it shows that SHA-512 has good resilience and strength because the length of the hash is 512-bit so there are 2256 possible combinations that are very difficult to solve.

The existing methods in cryptography can be classified into 3 groups, namely symmetric, asymmetric, and hybrid key cryptography. The encryption algorithms that are usually used are symmetric keys such as Advance Encryption Standard (AES), Data Encryption Standard (DES), GOST, Blowfish, Twofish, Advance AES (AAES), and many more [7][8]. Cryptography is essentially a technique to maintain the confidentiality of information so that the information sent can be ensured to be safe so that unauthorized or unauthorized parties cannot access the information [6][9].

There are two ways to encrypt and describe messages based on the key, namely symmetrical and asymmetrical [10]. The use of a symmetric key means that the key used for encryption and decryption is the same. The algorithms for symmetric keys are further divided into two types, namely Stream Cipher and Block Cipher. The difference between Stream cipher and Block Cipher is in their encoding orientation. Stream Cipher encodes each bit of data, while Block Cipher encodes each bit block [11]. The popular symmetric key algorithms used are DES, AES, RC4, Blowfish, Twofish, and many more [12]. The asymmetric key is also known as the public key. As the name implies, the keys used for encryption and decryption are different. The key used for encryption is called the public key. Meanwhile, the key used for decryption is called the private key [13]. Asymmetric key algorithms that are often used are RSA, Diffie-Hellman, Elgamal, and many more [11]. Steganography is the science or art of hiding messages by inserting them into the media in such a way that

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

unauthorized people do not realize that there is a hidden message in an inserted media [3].

While steganography is the science or art of hiding messages by inserting them into the media in such a way that unauthorized people do not realize that there is a hidden message in the inserted media [14]. The media used to hide messages is called the carrier file. There are several forms of media such as text, images, audio, and video. The number of media in the form of images or digital images scattered on the internet makes this type of media a medium that is often used as a carrier file [15]. The image formats commonly used are the Joint Photographic Expert Group (JPEG), Portable Network Graphics (PNG), Graphic Interchange Format (GIF), and Bitmap (BMP). There are two domains in steganography, namely spatial and transformation. Popular domains used are spatial domains with Least Significant Bit (LSB), Pixel Value Difference (PVD), Exploiting Modification Direction (EMD) methods, and etc [16].

The application of either cryptography or steganography methods can secure information. But both still have gaps if they stand alone. Cryptography that produces meaningless forms of information can arouse suspicion by others and increase the possibility of the message being modified so that the recipient does not get the real information [17]. Therefore, the application of cryptography can be followed by applying steganography to hide messages that have been encrypted into the media to minimize the possibility of being known by other parties. Steganography requires media to hide messages or information. The media to hide the message or information vary from images, audio, text, and video.

Broadly speaking, there are only two processes in steganography, namely embedding and decoding. Embedding is the process of embedding a secret message into a medium. While Decoding is the process of extracting messages that have been hidden in the media. The insertion of messages into the media can change the quality of the media. Therefore, there are several criteria for a good steganographic assessment, namely Fidelity, Robustness, and Recovery [3]. In image media, the objective evaluation method used to measure the quality of the inserted image is to use Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) [18][9].

Blowfish is a block cipher symmetric key algorithm. The key size used ranges from 32 to 448-bit with a block size of 64 bits. First published in 1993 by Bruce Scheiner as an alternative to the Data Encryption Standard (DES) encryption algorithm because the time for encryption using Blowfish is much faster [19]. A hash function is a function that takes a message as input and compresses it into a fixed-size string [20]. This function is usually implemented to ensure the security of information by detecting changes or modifications to information. One of the hash function algorithms is Secure Hash Algorithm (SHA)-512. SHA-512 is a hashing algorithm that has a maximum input value of less than 2128 and is processed in 1024-bit blocks. Meanwhile, the output of this algorithm is a message digest with a size of 512 bits.

From the results of the above description, the researchers decided to use the Blowfish symmetric key block cipher method to encrypt the message, then hide it into

digital image media using the Least Significant Bit (LSB) method in the spatial domain. The message is secured again using the SHA2-512 hash function so that if there is a change in the message before and after message extraction it will be detected so that the security of the data is more optimal.

2. METHODS

The method used in this research is the encryption of messages in text form using a modern cryptographic algorithm with a symmetric key block cipher, namely Blowfish. Messages that have been encrypted will be secured by ensuring the integrity of the data using the SHA512 hash function. While hiding messages into the image using the Least Significant Bit (LSB) method. Messages that have been extracted in the form of bytes will be hashed and then compared with the hashed results of messages that have been previously encrypted. An overview of the method can be seen in FIGURE 1.

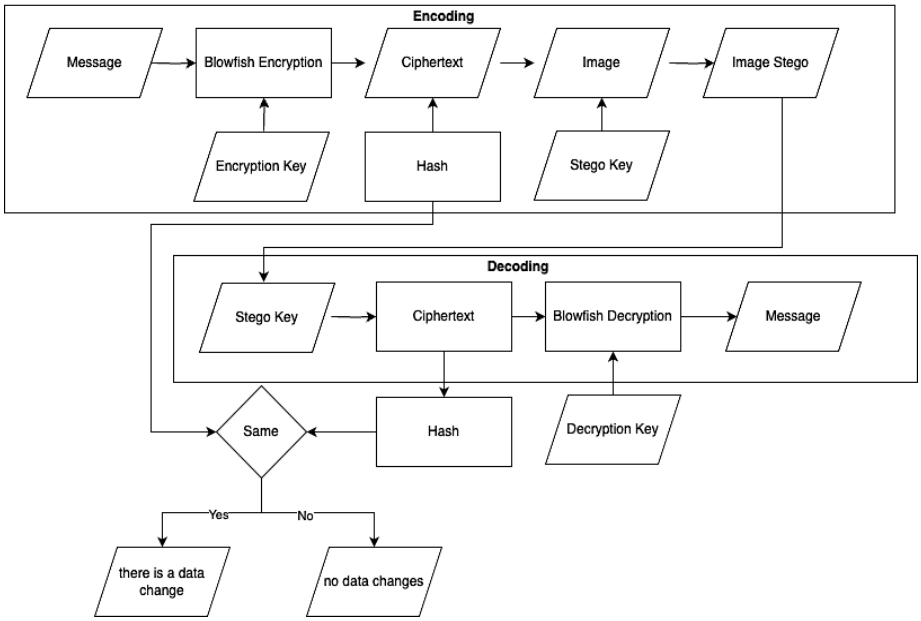


FIGURE 1. Overview of the method

2.1 *BLOWFISH ENCRYPTION*

The blowfish encryption process can be seen in FIGURE 3.

1) For example, X is 64-bit plaintext. The results of the conversion to binary and bit padding can be seen in FIGURE 2.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

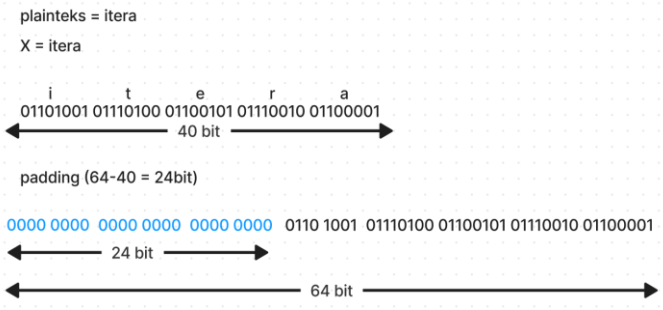


FIGURE 2. Input plaintext blowfish

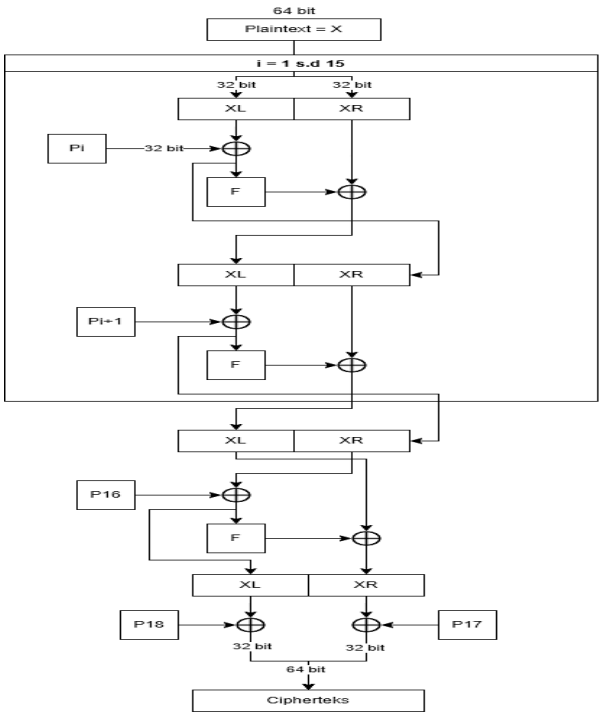


FIGURE 3. Blowfish Encryption Flow

2) The key size range is 32-64 bits can be seen in FIGURE 4.

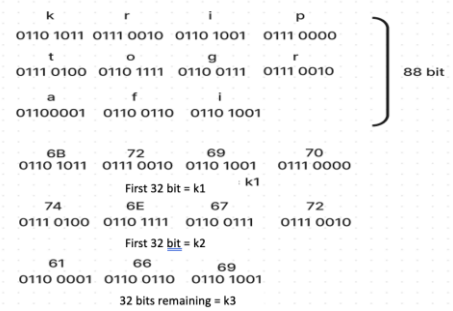


FIGURE 4 . Convert key to binary form

3) Subkey Pi generation using Pbox can be seen in FIGURE 5.

i	Pi		Ki	Pi
1	243F6A88		6B726970	4F4D03F8
2	85A308D3		746E6772	F1CD6FA1
3	13198A2E		616669	1378EC47
4	03707344		6B726970	3707344
5	A4093822		746E6772	A4093822
6	299F31D0		616669	299F31D0
7	082EFA98		6B726970	82EFA98
8	EC4E6C89		746E6772	EC4E6C89
9	452821E6	XOR	616669	452821E6
10	38D01377		6B726970	38D01377
11	BE5466CF		746E6772	BE5466CF
12	34E90C6C		616669	34E90C6C
13	C0AC29B7		6B726970	C0AC29B7
14	C97C50DD		746E6772	C97C50DD
15	3F84F5B5		616669	3F84D5B5
16	B5470917		6B726970	B5470917
17	9216D5D9		746E6772	9216D5D9
18	8979FB1B		616669	8979FB1B

FIGURE 5. Generation of Pi

4) Divide X into 2 parts of 32 bits each: XL and XR which can be seen in FIGURE 6.

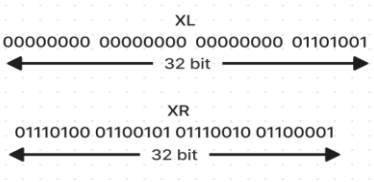


FIGURE 6. Dividing XL and XR

5) In each iteration from the 1st to the 16th iteration.

- XOR the value of XL with Pi (XL = XL XOR Pi)

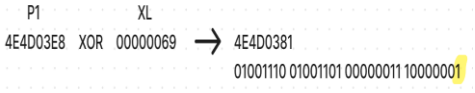


FIGURE 7. XOR XL with Pi

- Shift Left XL

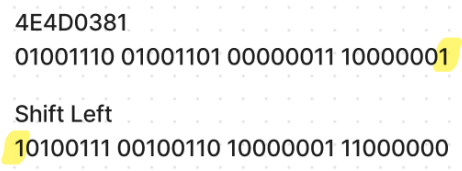


FIGURE 8. Shift Left value XL

- Processed on the F function
- XOR returns the output of Function F with XR (XR = XL XOR XR)

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

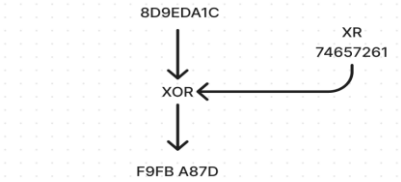


FIGURE 9. XR calculation

- Change order of XL and XR to XR then XL

FIGURE 10. XL and XR round i+1

6) After 16 iterations, exchange XL and XR again to cancel redemption at the end of the iteration.

FIGURE 11. Switch XL and XR posisi positions

7) XOR the value of XR with P17

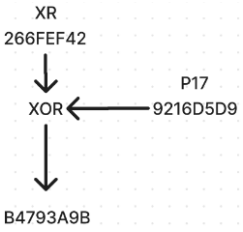


FIGURE 12. XOR XR with P18

8) XOR the value of XL with p18

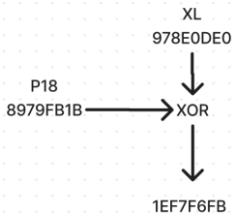


FIGURE 13. XOR P18 with XL

9) Recombine XL and XR

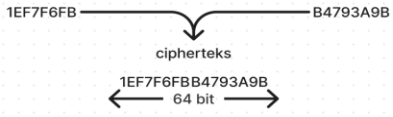


FIGURE 14. Getting ciphertext

2.2 *BLOWFISH FUNCTION*

The process in the blowfish algorithm F function can be seen in FIGURE 15:

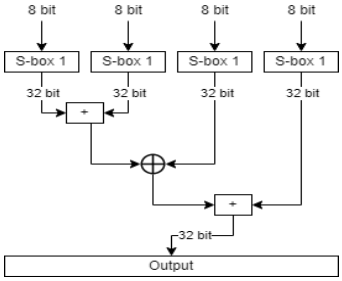


FIGURE 15. F Function Flow in Blowfish

1) The XL function is divided into 4 sections of 8 bits each.

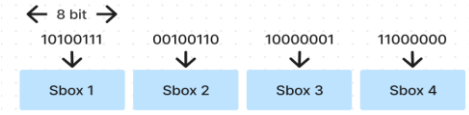


FIGURE 16. Divide the XL into 4 parts

2) $F(XL) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \text{ XOR } S_{3,c}) + S_{4,d} \bmod 2^{32}$.

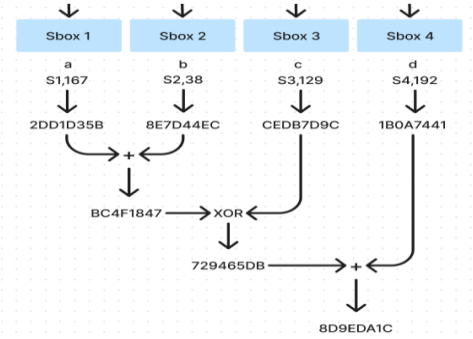


FIGURE 17. Calculation of the function F

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

2.3 EMBEDDING USING LSB

The process of inserting messages into the image using the LSB:

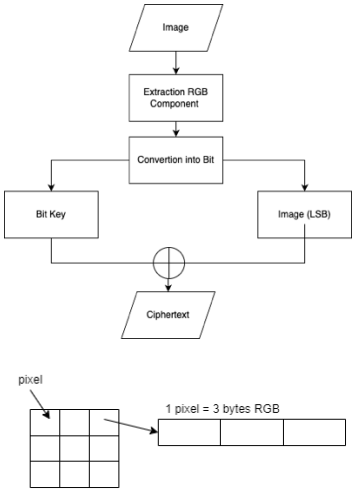


FIGURE 18. LSB Em Embedding Flow

1. In the form of an encrypted message, namely Ci then convert it to binary form

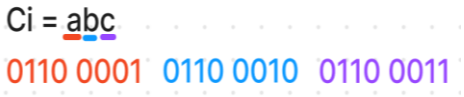


FIGURE 19. Convert message to binary form

2. Convert pixels to binary

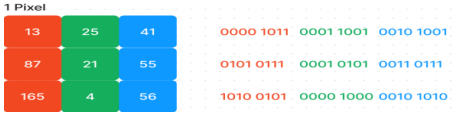


FIGURE 20. Conversion of message pixels to binary

3. Replacing the rightmost bit in the pixel with binary in Ci

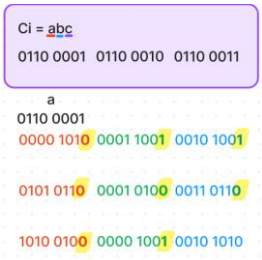


FIGURE 21. Binary embedding of messages in pixels

2.4 EXTRACTING

The process of extracting messages from images:

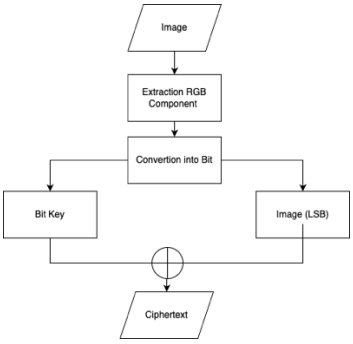


FIGURE 22. LSB Extracting Flow

1) Transform image pixels into RGB components (Red, Green, and Blue)

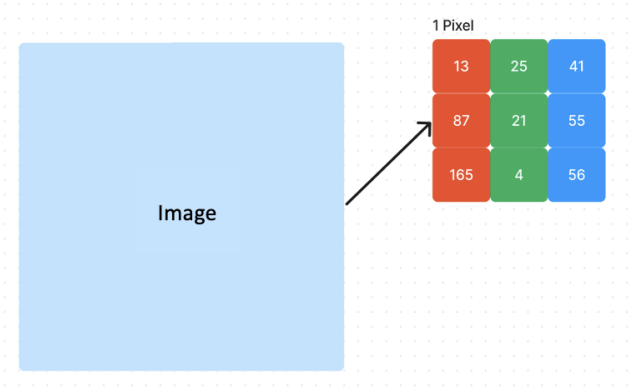


FIGURE 23. Image transformation to RGB components

2) Convert the value of each RGB component into binary form

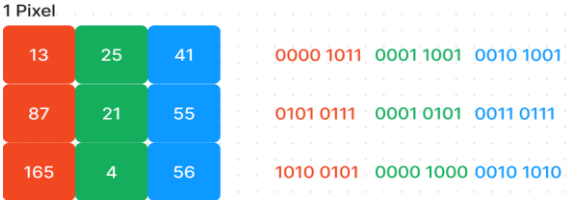


FIGURE 24. Conversion of message pixels to binary

3) Take the value in the first or rightmost bit of each component and store it in a one-dimensional array

FIGURE 25. Retrieving bit values for each component

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

4) Convert the bit values in the array into characters.
5) Show messages.

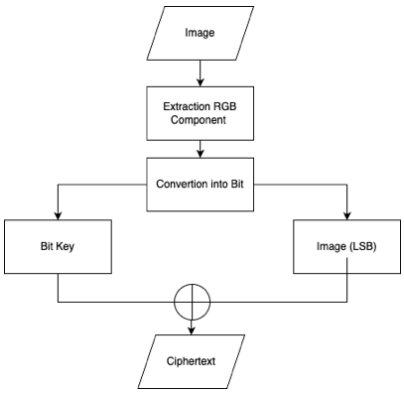


FIGURE 26. LSB Extracting Flow

2.5 *BLOWFISH DESCRIPTION*

The decryption process is almost the same as encryption, except that the subkey value used is reversed from P18 to P1.

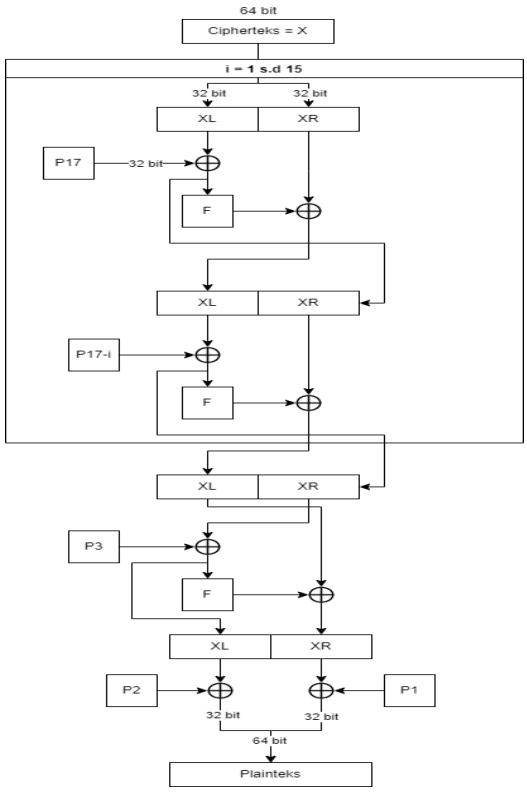


FIGURE 27. Blowfish decryption flow

1) On each iteration from the 1st to the 16th iteration:

- Divide the ciphertext X into 2 parts measuring 32 bits, XL and XR

FIGURE 28. Divide the ciphertext into 2 parts

- XOR XL with P18

FIGURE 29. XOR XL with P18

- XOR XR with P17

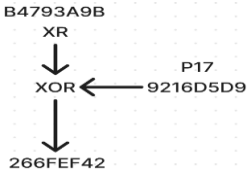


FIGURE 30. XOR XR with P17

- XL processed on Function F
- The output of Function F becomes the new XL value. While the XR value is the result of the initial XOR XL with P18.

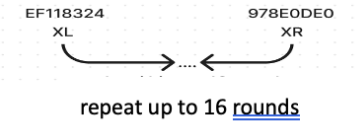


FIGURE 31. Process results on the F . function

- Change the order of XL and XR to XR then XL

2) After 16 iterations, exchange XL and XR again to cancel redemption at the end of the iteration.

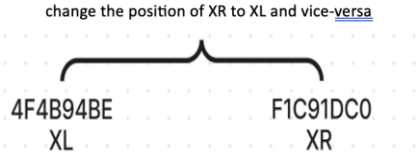


FIGURE 32. Exchange results 16 iterations

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

3) XOR the value of XR with P2

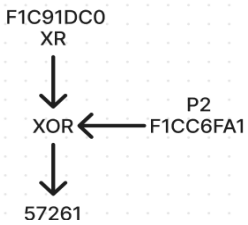


FIGURE 33. XOR XR with P2

4) XOR the value of XL with P1

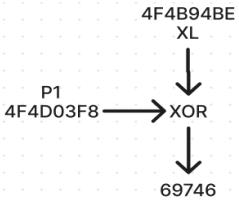


FIGURE 34. XOR XL with P1

5) Recombine XL and XR

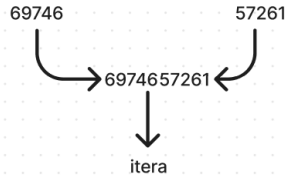


FIGURE 35. The result of the ciphertext decryption

2.6 *SHA512 HASH FUNCTIONS*

The hashing process using SHA-512 can be seen in FIGURE 35:

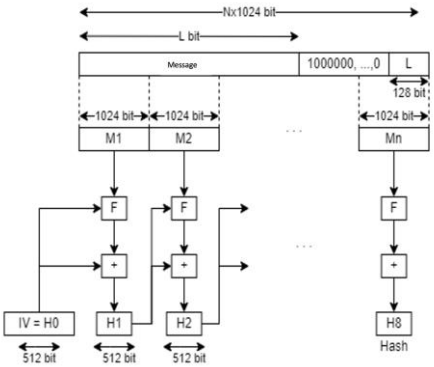


FIGURE 36. SHA-512 Flow

1) Input in the form of messages in various sizes, for example the message input is "abc"

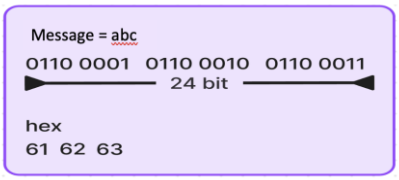


FIGURE 37. Input hash

2) Added a padding bit with a 1 followed by 0 bits (100…) until the message is congruent with 896 mod 1024.

FIGURE 38. Adding padding bit

3) Increase the message length by 128 bits which represents the original message length so that the current message length is 1024 bits

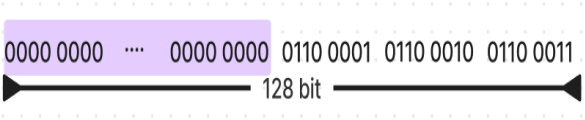


FIGURE 39. Increasing the message size to 128 bits

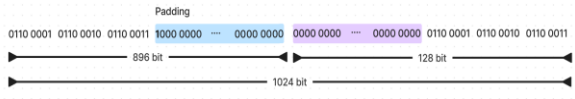


FIGURE 40. Total message length

4) Initialize 8 buffers (a, b, c, d, e, f, g, h) 64 bit hexadecimal:
   a) H_0 = 6A09E667F3BCC908 = a
   b) H_1 = BB67AE8584CAA73B = b
   c) H_2 = 3C6EF372FE94F82B = c
   d) H_3 = A54FF53A5F1D36F1 = d
   e) H_4 = 510E524FADE6C1F = e
   f) H_5 = 9b05688C2B2E6C1F = f
   g) H_6 = 1F83D9ABFB41BD6B = g
   h) H_7 = 5BE0CD19137E2179 = h

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

So, the total length of the buffer is 8 x 64 = 512 bits

5) Processing messages with 80 rounds consisting of:

a) $W_t = M_t \qquad 0 < t < 15$

$= \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$

$16 < t < 79$

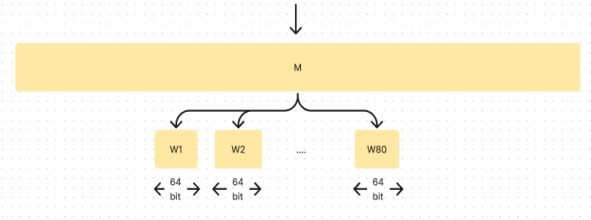


FIGURE 41. Generating 80 words

b) Using a 64-bit K_t adder number

a. Round $0 < t < 19$ K_t = 5A827999
b. Round $20 < t < 39$ K_t = 6ED9EBA1
c. Round $40 < t < 59$ K_t = 8F1BBCDC
d. Round $60 < t < 79$ K_t = CA62C1D6
e. $T_1 = h + (x \wedge y) \oplus (\sim x \wedge z) K_t + W_t$
f. $T_2 = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
g. h = g
h. g = f
i. f = e
j. $e = d + T_1$
k. d = c
l. c = b
m. b = a
n. $a = T_1 + T_2$

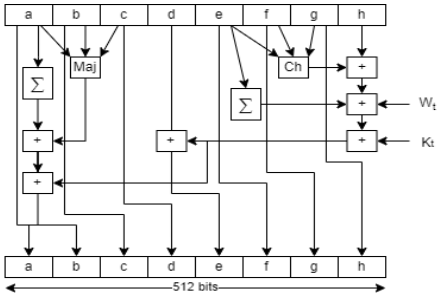


FIGURE 42. SHA-512 . message-digest flow

3. RESULTS

The following is the result of inserting an encrypted message into an image in PNG format. The first experiment, the process of inserting a message with the number of plaintext 27 characters on an image with dimensions of 525x525 pixels. Prior to insertion, the image was 137,826 bytes in size. After inserting an encrypted message, the image dimensions remain 525x525 pixels, but the image size changes to 137,830 bytes. The time it takes to encrypt a 27-character message using Blowfish is 0.004 seconds. Meanwhile, the time required to embed the encrypted message into the image is 0.005 seconds. In the second experiment, the insertion of a message with the number of plaintext 235 characters on an image with dimensions of 720 x 720 pixels. Prior to insertion, the image was 39,849 bytes in size. After inserting an encrypted message, the image dimensions remain 720 x 720-pixel, but the image size changes to 394,968 bytes. The time it takes to encrypt a 27-character message using Blowfish is 0.005 seconds. Meanwhile, the time required to embed the encrypted message into the image is 0.004 seconds. Cover image information can be seen in table 1.

TABLE 1.
The Information of Cover-Image

File	Size	Dimension
Tes1.png	137.826 bytes	525 x 525 pixels
Tes2.png	39.849 bytes	720 x 720 pixels

The results of the experiment using 2 plaintexts with different sizes are shown in table 2.

TABLE 2.
Results of The Encryption Process

Plaintext	Key	Character Length	Encryption Time
Institut Teknologi Sumatera	iteraaa	27	0.004 seconds
Kriptografi pada intinya merupakan sebuah teknik untuk menjaga kerahasiaan informasi	kriptografi2022	85	0.005 seconds

From the test results, the greater the message capacity, the longer it will take to perform the encryption process. The results of the insertion using LSB can be seen in table 3.

TABLE 3.
Results of The Embedding Process

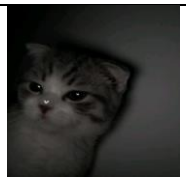
File	Size	Dimension	Extraction Time	Hidden Messages
Hasil1.png	137.830 bytes	525 x 525 pixels	0.957s	a2eeea28d29c622b27753b358bbdc3808203fc678511080d8786b6

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

File	Size	Dimension	Extraction Time	Hidden Messages
Hasil2.png	394.937 bytes	720 x 720 pixels	1.983 s	079e1a561f2bf57d473a57e82e1b8394240150a8583aa851637a44a6f4409c4c8b8a9fffd392ebfda54ff0072c6f5220ec63814d3bbdb986e2d5a45eb76a5816e01c87f0ff0e85101214fd1b1169a988947e689b36

The experimental results of inserting encrypted text messages can be seen in table 4.

TABLE 4.
Comparison of Experimental Images

Test	Before	After
1	 1st image test before inserting a message.	 2nd image test after inserting a message
2	 1st image test before inserting a message	 2nd image test after inserting a message

Visually, there is no difference between the message that has been inserted and the original image. The test results for the effect of message capacity with PSNR can be seen in table 5.

TABLE 5.
Aspects of Measuring Capacity

File	Image Resolution	File Size (Bytes)	maximum inserted characters	PSNR (dB)
Tes1.png	525 x 525	137826	51990	51.2
Tes2.png	720 x 720	39849	95382	49.2

From the test results, it was found that the larger the message size, the smaller the PSNR value. The results of testing the computational time of encryption, decryption, insertion, and extraction can be seen in table 6.

TABLE 6.
Aspects of Time Measurement

Number of character Message	Encryption time (seconds)	Decryption time (seconds)	Embedding Time (seconds)	Extraction Time (seconds)
27	0.004	0.004	0.005	1.248
85	0.005	0.005	0.007	2.268

From the test results in table 6 it is found that the larger the message character, the longer the time required for message extraction.

3.1 THE RESULT OF SHA-512 TESTING

Tests using the SHA2-512 hash function method were carried out to check the integrity of the message by ensuring that there were no changes or modifications to the message after it was extracted. Message integrity checking is done by comparing the hash value of the message before insertion and after extraction. If there are no changes, it can be ascertained that the encryption, embedding, extraction, and decryption processes have been successfully carried out. The following is a comparison table for the SHA value test results from messages. The test results from Fig 1 and Fig 2 from Table 4 can be seen in Fig 7.

TABLE 7.
Results of SHA-512

Plaintext	SHA-512 (Fig 1)	SHA-512 (Fig 2)
Order before insert	e4ab1b63d0766f58b88e6dd82dc6f77126f2a437b8975838f7809f9ae78ae14a332940529d70621a4171c030245f09a637a38af0f14d15fdd8d4447dcbb25d6a	14e6aea639e9f832035c7766b04e1490da59e825450a8c4f8a55b90c5b195e9d49866a695371ed5b1abe6076dffa55d00b23e6e4cbea8736bf5e9294d8c1527c
Message after pasting	e4ab1b63d0766f58b88e6dd82dc6f77126f2a437b8975838f7809f9ae78ae14a332940529d70621a4171c030245f09a637a38af0f14d15fdd8d4447dcbb25d6a	14e6aea639e9f832035c7766b04e1490da59e825450a8c4f8a55b90c5b195e9d49866a695371ed5b1abe6076dffa55d00b23e6e4cbea8736bf5e9294d8c1527c

From the results of testing the integrity of the message, this is evidenced by the hash value of the same message, before and after it is inserted.

3.2 PSNR TEST RESULTS

Testing using the Peak Signal to Noise Ratio (PSNR) method is carried out to check the quality of the image after inserting a message or information. This

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

needs to be done because differences in quality can arouse suspicion from other parties. Before getting the PSNR value, the Mean Square Error (MSE) value needs to be searched first. The smaller the MSE value, the smaller the error value. After getting the MSE value, then the PSNR value can be searched. The maximum PSNR value is 100 which means there is no change at all. Therefore, the greater the PSNR value, the better the image quality. The following is a table of results for the comparison of MSE and PSNR values in experiments 1 and 2. The test results on PSNR from 2 images can be seen in table 8.

TABLE 8.
Image PSNR Results

File	MSE (dB)	PSNR (dB)
Tes1.png	0.0001656840514	85.93799655
Tes2.png	0.1814351852	55.54358849

4. CONCLUSION

By using the Least Significant Bit (LSB) method to insert a message into a binary pixel array, testing with the SHA2-512 hash function method shows that the message can be extracted again without changing the hash value. The insertion of this message can affect the image quality depending on the size of the hidden message; the more messages that are inserted, the more visible the difference with the original image. Even though a message of 235 characters has been inserted, the difference between the stego image and the original image cannot be seen directly. In addition, the test results using the PSNR value show that the difference in quality between the original image and the stego image can still be tolerated, with an average PSNR of 70.74 dB and an average MSE of 0.09.

REFERENCES

[1] H. Hendy and H. Akbar, "Pengembangan Aplikasi Android Belajar Kriptografi Sebagai Media Pembelajaran Mata Kuliah Kriptografi," *J. Inform.*, vol. 8, no. 1, pp. 17–25, 2021, doi: 10.31294/ji.v8i1.9420.

[2] J. Simarmata *et al., Sistem Keamanan Data*. 2022.

[3] A. A. Permana, "Aplikasi Penyisipan Teks pada Gambar Dengan Algoritma Blowfish dan Least Significant Bit," *J. Tek. Inform.*, vol. 59, pp. 11–17, 2017.

[4] R. Rizki and S. Mulyati, "Implementasi One Time Password Menggunakan Algoritma SHA-512 Pada Aplikasi Penagihan Hutang PT. XHT," *Edumatic J. Pendidik. Inform.*, vol. 4, no. 1, pp. 111–120, 2020, doi: 10.29408/edumatic.v4i1.2158.

[5] H. Alabdulrazzaq and M. N. Alenezi, "Performance Evaluation of Cryptographic Algorithms: DES, 3DES, Blowfish, Twofish, and Threefish,"

Int. J. Commun. Networks Inf. Secur., vol. 14, no. 1, pp. 51–61, 2022, doi: 10.54039/ijcnis.v14i1.5262.

[6] A. M. Qadir and N. Varol, "A review paper on cryptography," *7th Int. Symp. Digit. Forensics Secur. ISDFS 2019*, no. October, pp. 1–6, 2019, doi: 10.1109/ISDFS.2019.8757514.

[7] I. F. Ashari, "Graph Steganography Based On Multimedia Cover To Improve Security and Capacity," in *2018 International Conference on Applied Information Technology and Innovation (ICAITI)*, 2018, no. April 2019, pp. 194–201.

[8] I. F. Ashari, A. W. Bhagaskara, J. M. Cakrawarty, and P. R. Winata, "Image Steganography Analysis Using GOST Algorithm and PRNG Based on LSB," *Techno.Com*, vol. 21, no. 3, pp. 700–713, 2022, doi: 10.33633/tc.v21i3.6331.

[9] I. F. Ashari, "The Evaluation of Audio Steganography to Embed Image Files Using Encryption and Snappy Compression," *Indones. J. Comput. Sci.*, vol. 11, no. 2, pp. 318–336, 2022.

[10] Y. Ramesh and K. K. Reddi, "RK algorithm: stochastic parallel methodology for symmetric key cryptography," *APTIKOM J. Comput. Sci. Inf. Technol.*, vol. 2, no. 3, pp. 137–144, 2020, doi: 10.34306/csit.v2i3.73.

[11] T. H. Saputro, N. H. Hidayati, and E. I. H. Ujianto, "Survei Tentang Algoritma Kriptografi Asimetris," *J. Inform. Polinema*, vol. 6, no. 2, pp. 67–72, 2020, doi: 10.33795/jip.v6i2.345.

[12] I. F. Ashari, I. R. Siwi, H. Londata, and I. Wicaksono, "Steganographic Analysis of Audio and Image Media Using Lsb and Rc4 Algorithms," *ELTIKOM*, vol. 7, no. 1, pp. 67–78, 2023.

[13] M. Maxrizal and B. D. A. Prayanti, "Penerapan Matriks Persegi Panjang Sebagai Kunci Publik Dan Kunci Privat Pada Modifikasi Cipher Hill," *AdMathEdu J. Ilm. Pendidik. Mat. Ilmu Mat. dan Mat. Terap.*, vol. 7, no. 2, p. 151, 2017, doi: 10.12928/admathedu.v7i2.9156.

[14] M. Kurniawan and S. Agustini, "Algoritma Steganografi untuk Pengamanan Data Teks ke dalam Citra Digital Menggunakan XOR Sederhana," *INTEGER J. Inf. Technol.*, vol. 3, no. 2, pp. 63–68, 2018, doi: 10.31284/j.integer.2018.v3i2.416.

[15] A. P. Ratnasari and F. A. Dwiyanto, "Metode Steganografi Citra Digital," *Sains, Apl. Komputasi dan Teknol. Inf.*, vol. 2, no. 2, p. 52, 2020, doi: 10.30872/jsakti.v2i2.3300.

[16] M. Hussain, A. W. A. Wahab, Y. I. Bin Idris, A. T. S. Ho, and K. H. Jung, "Image steganography in spatial domain: A survey," *Signal Process. Image Commun.*, vol. 65, no. December 2017, pp. 46–66, 2018, doi: 10.1016/j.image.2018.03.012.

[17] A. Eko Setiawan and A. Pasaribu, "Penerapan Steganografi Pada Citra Digital Menggunakan Metode Least Significant Bit (LSB) Kombinasi RC4 Berbasis Mobile Android," *Aisyah J. Informatics Electr. Eng.*, vol. 2, no. 1, pp. 18–28, 2020, doi: 10.30604/jti.v2i1.27.

[18] C. Jatmoko, L. B. Handoko, C. A. Sari, and D. R. I. M. Setiadi, "Uji Performa

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

Penyisipan Pesan Dengan Metode Lsb Dan Msb," *Din. Rekayasa*, vol. 14, no. 1, pp. 47–56, 2018.

[19] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," *Procedia Comput. Sci.*, vol. 78, no. December 2015, pp. 617–624, 2016, doi: 10.1016/j.procs.2016.02.108.

[20] D. Ravilla and C. S. R. Putta, "Enhancing the Security of MANETs using Hash Algorithms," *Procedia Comput. Sci.*, vol. 54, no. December 2015, pp. 196–206, 2015, doi: 10.1016/j.procs.2015.06.022.