Hiding and Data Safety Techniques in Bmp Image with LSB and RPrime RSA Algorithm

S M Hardi 1*, M Masitha1, M A Budiman1, I Jaya1
1Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan 20155, Indonesia
*Corresponding Email: vani.hardi@usu.ac.id

Abstract. Image is a form of visual data or information that is often used today. Sending data or information can be done in various ways and anyone. However, not all contents of existing information are public. It is not uncommon that information is misused by irresponsible parties. For this reason, security is needed for confidential information. This image security can be done with Steganography Technique. Steganography technique is a method for hiding information in an image. The method used is the LSB (Least Significant Bit) method. Where every bit lowest in the image media bytes will be replaced with the message bits that will be inserted. The bit used is the last bit of the RGB binary. The goal is that the message sent by the sender cannot be known by a third party, and can arrive safely to the recipient of the message. To make the confidentiality of messages more secure, it is necessary to hide the existence of messages to overcome the weaknesses of the LSB method used, so the authors combine them with the RSA RPrime algorithm (Rivest Shamir Adleman) for the encryption and decryption process. The security of the RSA RPrime algorithm lies in the level of difficulty in factoring non prime numbers into prime factors. The longer the key bits, the more difficult it is to solve because of the difficulty of factoring these two very large numbers. The results showed the test using a document file 2.93 MB with cover image size of 5760 x 3840 pixels, where in the larger the file inserted, then the time of encryption and decryption process is also long.

1. Introduction
Image is a form of data or visual form of information that is often used today. Data sending or information can be done in various ways and by anyone. However, not all contents of existing information are public. It is not uncommon that information is misused by irresponsible parties. For this reason, security is needed for confidential information. This image security can be done with Steganography Technique. Steganography is a method for hiding information on a media, can be in the form of media images, sound or video. The level of security in which information is concealed is the most important aspect of steganography, which refers to how much the inability of a third party to detect the existence of hidden information [1].

In general, the technique of hiding information in steganography on image media, where text information inserted into pixel bits of the image. But the method often used is still quite simple so that third parties can still get hidden information. Therefore, the writer discusses an implementation that makes text steganography on image media stronger and secure using the LSB (Least Significant Bit) algorithm. The LSB method only changes the value of bytes one higher or one lower from the previous value, so the changes that occur are not very meaningful. Moreover, the human eye cannot distinguish the small changes that occur [2]. In this case the authors use the RSA RPrime Algorithm for the encryption and decryption process. Rprime RSA algorithm is a development algorithm of the RSA variant algorithm. In the journal Ceaser introduced the RSA variant by combining the R prime-RSA algorithm and the Rebalanced RSA algorithm. The combination of the two algorithms is Rprime RSA. RSA bases its encryption and decryption process on the concepts of prime numbers and modulo arithmetic. Both encryption and decryption keys are integers. The encryption key is not kept secret and...
given to the public (so called the public key), but the key for decryption is confidential (private key). To find the decryption key, it is done by factoring an integer into its prime factors. In fact, factoring integers into prime factors is not easy. Because of not found any efficient algorithm for factoring. The way that can be done in factoring is to use a factor tree. If the greater the number to be factored, the longer it will take. So the larger the factored number, the more difficult the factoring is, the stronger the RSA algorithm [3].

2. Method
This research using Least Significant Bit and RSA R Prime Algorithm for hiding and safety data.

2.1 Least Significant Bit (LSB)
A binary data insertion method that works by replacing the last bits of each cover pixel value with secret message bits. Before inserting a secret message into the cover, of course both must be changed into binary form so that the insertion process can be done.

2.2 RSA RPrime Algorithm
The RSA algorithm has many variants. One variant of the RSA algorithm is the RSA RPrime. In the standard RSA algorithm, the selected prime numbers are only 2, while in the RPrime algorithm the chosen prime numbers can be more than 2.

The following are the stages of the key generation process in the RSA Rprime:

- a) Find the value of k which is the number of prime numbers to be used, where k> 2.
- b) determine the prime number k as long as the condition for the gcd value of each combination if the prime value minus 1 is 2. For example p1, p2, p3, ...
- c) Calculate the value of N which is a multiplication of a predetermined prime number, N = p1.p2.p3.p4
- d) Take the values dp1, dp2 ... dpk with condition that gcd (dp1, p1-1) = gcd (dp2, p2-1) = ... = gcd (dpk, pk-1) = 1 and dp1, dp2 ... dpk is odd numbers and dp1 < p1, dp2 < p2 ... dpk < pk
- e) Determine the value d with the terms:
  dp1 (mod p1-1) ≡ d
  dp2 (mod p2-1) ≡ d
  ...
  dpk (mod pk-1) ≡ d
- f) Determine the value φ(N) : φ(N) = (p1 -1 ) . (p2 -1 ) . (p3 -1 ) . (p4 -1 )
- g) Calculate the value of d with the following conditions:
  e.d ≡ 1 (mod φ(N)) atau e.d (mod φ(N)) = 1

by trying the value d = 1,2,3,4, .. n so that it satisfies the equation.

The encryption and decryption process for the RSA RPrime algorithm is as follows.

1. Message recipient public key (e) and mod N.
2. Plaintext is represented by blocks m1, m2, ..., mx until each block presents the value [0, N-1]
3. Each block ni is encrypted into a block with the formula : ci = pni mod N

The decryption process is done using a private key. Where the encrypted message will be returned as the original message. Each ciphertext block will be described by the formula: ci = pni mod N

2.3 Image
In an RGB image, one pixel is 24 bits with details of 8 bits for RED, 8 bits for GREEN and 8 bits for BLUE. The number of bits with a value of 8 for each color segment, represents the range of possible values in the color segment, 0 (2) to 11111111 (2), if converted to decimal it will be 0 to 255. To make it easier to see the illustrations in Figure 2.
In Figure 2, can be in view illustration 3x3 pixel RGB image, which has a different color where the column and row index runs from 0 to 2. In each pixel can we know the combination of the RGB color value of the image. For example, pixels (0,1) the RGB values of these pixels are R = 100, G = 200, and B = 255, so the resulting color matches the combination of these RGB values.

3. Result and Discussion
The key used for the encryption and decryption process uses the RPrime RSA algorithm at random at random classes in visual studio C#.

3.1 Encryption and Insertion Process
After randomizing the key, the encryption process will be performed as the following table 1.

![Table 1. Calculation of the Encryption Process](image)

| Index length of message (i) | Plaintext (p) document file | N   | e   | C = P^e mod N |
|-----------------------------|-----------------------------|-----|-----|---------------|
| 0                           | 19                          | 1155| 143 | 94            |
| 1                           | 32                          | 1155| 143 | 758           |
| 2                           | 35                          | 1155| 143 | 140           |
| 3                           | 25                          | 1155| 143 | 940           |
| 4                           | 20                          | 1155| 143 | 1070          |
| ...                         | ...                         | ... | ... |               |
| 3079243                     | 25                          | 1155| 143 | 940           |
| 3079244                     | 21                          | 1155| 143 | 21            |
| 3079245                     | 20                          | 1155| 143 | 1070          |
| 3079246                     | 41                          | 1155| 143 | 776           |
| 3079247                     | 13                          | 1155| 143 | 1042          |

After encrypting the document file, the system will then insert the encrypted document file into the cover image. The encryption results are converted into hexadecimal numbers then into binary form and then inserted into the last bit of the cover image. Insertion process calculation can be seen in Table 2.

![Table 2. Calculation of the Insertion Process](image)

| Index length of message (i) | Cipher (C) | N   | d   | P = C^d mod N |
|-----------------------------|------------|-----|-----|---------------|
| 0                           | 94         | 1155| 47  | 19            |
| 1                           | 758        | 1155| 47  | 32            |
| 2                           | 140        | 1155| 47  | 35            |
| 3                           | 940        | 1155| 47  | 25            |
| 4                           | 1070       | 1155| 47  | 20            |
| ...                         | ...        | ... | ... |               |
| 3079243                     | 940        | 1155| 47  | 25            |
| 3079244                     | 21         | 1155| 47  | 21            |
| 3079245                     | 1070       | 1155| 47  | 20            |
3.2 Extraction and Decryption Processes
Before decrypting, the system first extracts the stego image by using a key in the form of the message length (length) that has been entered. Illustration of this extraction calculation can be seen in Table 3.

### Table 3. Calculation of the Extraction Process

| c     | hexadecimal | binary    | Cover Image Pixel Position     | Rcover | Rstego |
|-------|-------------|-----------|--------------------------------|--------|--------|
| 94(10)| 94(16)      | 01011110(2)| (0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0), (7,0) | 52, 33, 21, 30, 47, 68, 90, 57 | 52, 32, 20, 30, 46, 68, 90, 56 |
| FF(16)| FF(16)     | 11111111(2)| (8,0), (9,0), (10,0), (11,0), (12,0), (13,0), (14,0), (15,0) | 81, 99, 23, 44, 33, 79, 61, 37 | 80, 98, 22, 44, 32, 78, 60, 36 |

* until it does not exceed the limit on the number of cover image pixels.

After extracting, the system then decrypts the cipher from the extraction process. The results of this decryption will be saved in the form of a document file with the potential (*.Pdf). Illustration of this decryption calculation can be seen in Table 4.

### Table 4. Calculation of the Decryption Process

| Stego Image Pixel Position | Rstego  | Taken bit | Cipher |
|---------------------------|---------|-----------|--------|
| (0,0)                     | 11011000(2) | 0         | 94(16) |
| (1,0)                     | 01101101(2) | 1         | 94(16) |
| (2,0)                     | 10101010(2) | 0         | 94(16) |
| (3,0)                     | 01010101(2) | 1         | 94(16) |
| (4,0)                     | 01111111(2) | 1         | 94(16) |
| (5,0)                     | 11000001(2) | 1         | 94(16) |
| (6,0)                     | 01100001(2) | 1         | 94(16) |
| (7,0)                     | 00001000(2) | 0         | 94(16) |
| (8,0)                     | 01011111(2) | 1         | FF(16) |
The process time test conducted by the system during the encryption and insertion process that uses document files measuring 43.8 KB, 158 KB, 293000 KB is 2.9736274 sec, 10.05222543 sec, 158.1945197 sec. The test results of this processing time can be illustrated with a graph that can be seen in Figure 4.

![Graph of the Size of the Document File Against the Processing Time Encryption and Insertion](image)

**Figure 4.** Graph of the Size of the Document File Against the Processing Time Encryption and Insertion

Testing time processing performed by the system during the extraction process and decryption using the document file size of 43.8 KB, 158 KB, 293000 KB is 2.9736274 sec, sec 10.05222543, 158.1945197 sec. The test results of this processing time can be illustrated with a graph that can be seen in Figure 5.

![Graph of the Size of the Document File Against Extraction and Decryption Processing Time](image)

**Figure 5.** Graph of the Size of the Document File Against Extraction and Decryption Processing Time
4. Conclusions
The conclusion of this research that the result of testing from process of the encryption, insertion and extraction and decryption process show that the document file size with time processing is directly proportional, the large document file size to be secured, the longer time needed to secure the document file process.

5. Acknowledgments
The authors gratefully thank to all who have supported in the completion of this research. And to the Universitas Sumatera Utara that has given me the opportunity to publish this paper.

References

[1] Cheddad, A., J. Condell, K. Curran, P. Mc Kevitt. 2010. Digital Image Steganography: Survey and Analysis of Current Methods. Elsevier. Northern Ireland, UK, Signal Processing 90, 2010, pp.727-752

[2] Arifin, Rian. 2013. Implementasi Kriptografi dan Steganografi Menggunakan Algoritma RSA dan Metode LSB. Skripsi. Universitas Negeri Malang.

[3] Gumelar, Agum. 2017. Pengamanan File Citra Dengan Skema Hybrid Cryptosystem Menggunakan Algoritma RSA-CRT Dan Algoritma Affine Cipher. Skripsi. Medan: Universitas Sumatera Utara.

[4] Rahmat R F, Prayoga D F, Gunawan D, Sitompul O S, 2018. Boyer-Moore Algorithm in Retrieving Deleted Short Message Service in Android Platform. IOP Conference Series: Materials Science and Engineering 308(1),012016

[4] Prayudi, Yudi, Idham Halik. 2005. Studi Analisis Algoritma Rivest Code 6 (RC6) Dalam Enkripsi/Dekripsi Data. Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005), Yogyakarta.