The implementation of trithemius algorithm and modified least significant bit (mlsb) blue channel for bitmap image security

S M Hardi*, D Rachmawati, A Wulandari, I Jaya, J T Tarigan

Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia

Email: vani.hardi@usu.ac.id

Abstract. In the delivery of information confidentiality message is one of the things that are very important, because the image data is very span to the actions of computer crimes such as modified or stolen. The combination of cryptography and steganography can be one solution to prevent such crime. Here the author uses the Trithemius algorithm. Trithemius algorithm is a symmetric algorithm that has advantages that is in the process of encryption or decryption of symmetric cryptography takes a short time and key size relatively shorter. While the weakness is the key must be changed frequently, every time implementing communication for the security of the message so that the authors combine with steganography technique that is Modified Least Significant Bit (MLSB), so that messages will be inserted not visible clearly or visible. In this research, the message or image file will be sent is first encrypted with Trithemius cryptography algorithm, then encryption result will be hidden in an image file with bitmap format using Blue Channel MLSB steganography method. Then do the process of extraction of messages for further decryption to return to the original message as before.

1. Introduction

Image is an optical representation of an object illuminated by a radiation source [6]. Images have several formats, namely Bitmap, PNG, JPEG and others. Here the author uses bitmap (bmp) format to secure images because this format stores digital image data without processing images. Representation of image data in this format is the same as image data used for image data representation on computer memory [2]. In sending messages via digital image media and when it reaches the recipient of the message, the information must remain confidential and maintain authenticity so cryptographic methods are needed to maintain the security of the data[5].

One of the symmetry cryptography methods used is the Trithemius algorithm. Trithemius is an algorithm that uses the same key for encryption and decryption. The advantage is that the process of encryption or decryption of symmetrical cryptography requires a short time and the key size is relatively shorter. While the weakness is the key must be changed frequently, every time to carry out communication for the sake of security of the message [4]. Then steganography techniques are needed to maintain the security of the data contained in the image file.

Steganography on digital media image files is used to exploit the limitations of the power of the human vision system by reducing the color quality of image files that have not been inserted in secret messages. So with these limitations, it is difficult for humans to find a gradation of color quality degradation in image files that have been inserted secret messages [3].

One method of digital image steganography is the LSB method with message hiding techniques at the lowest bit location in digital images [1]. LSB is the simplest and easiest method of steganography
to be implemented into an application. This method uses digital images as convertext. In the array of bits in a byte (1 byte = 8 bits), there is the most significant bit (MSB) and the least significant bit (LSB) [8]. MLSB or modification of the LSB algorithm is used to encode an identity into the original image [6]. MLSB's advantage is that it can modify where the message bits that should be 1 character have an 8 bit ASCII value modified to 5 bits (31 decimal places). From the modification of the MLSB, it only inserts an image message on the blue channel section because none of the previous research has discussed the insertion of the image on the blue channel and so that later this research can help as a reference for the next researchers.

2. Method
This research uses the Trithemius algorithm and the MLSB Blue Channel for securing bitmap image.

2.1 Trithemius Algorithm
The Trithemius algorithm is a continuation of the polyalphabetic cipher by Alberti, where Johannes Trithemius uses an alphabet table to assist in the encryption and decryption of polyalphabetic ciphers, which was explained in his book Polygraphia, where Polygraphia was the first cryptographic book of the period [7].

• Trithemius algorithm encryption formula:
  \[ C_R = (R + K_i) \mod 256 \]
  \[ C_G = (G + K_i) \mod 256 \]
  \[ C_B = (B + K_i) \mod 256 \]

• Trithemius algorithm decryption formula:
  \[ R = (C_R - K_i) \mod 256 \]
  \[ G = (C_G - K_i) \mod 256 \]
  \[ B = (C_B - K_i) \mod 256 \]

Where:
C: Ciphertext
R: Channel Red
G: Gree Channel
B: Blue Channel
K: Key

2.2 Modified Least Significant Bit Blue Channel
For improving the effectiveness and safety of the LSB method, it is necessary to modify the method. There have been many modification methods for LSB[8], but the MLSB Blue Channel algorithm is an improvement algorithm from the LSB algorithm, where only one color channel is selected to store information. The bit used is the last bit of the binary value of the selected color channel which is blue or blue.

Steps for embedding the secret image’s:
1. Input the cover image and file cipher.
2. Previously cipher files whose contents in the first decimal are changed to binary.
3. Convert the value of the cipher file into binary.
4. Embed the secret binary value to B (blue channel) of the cover image’s pixel. A pixel can contain just 1 bit of the secret binary value.
5. Do step 3 until all the secret binary values have been pinned.

Steps for extraction the secret image’s:
1. Input the stego pair and key image’s.
2. Get MLSB’s value from the stego image’s pixel which selected from blue channel, to get the binary file cipher value.
3. Convert the cipher file’s binary to byte.
3. Results and Discussions

This experiment were performed on Windows 7 Notebook with Intel Inside Core i3 processor, 32-bit architecture, and 4000MB RAM. The Integrated Development Environment (IDE) used for coding is SharpDevelop 4.3 and the programming language used is C#.

3.1 Encryption Process

![Illustration of 2x2 Pixel RGB Image](image)

Figure 1. Illustration of 2x2 Pixel RGB Image

| Pixel (0,0) | Pixel (1,0) |
|------------|------------|
| CR         | CR         |
| CG         | CG         |
| CB         | CB         |

Piksel (0,0):

1. \( CR = (R + K_i) \mod 256 \)
   \[ = (29 + A) \mod 256 \]
   \[ = (29 + 65) \mod 256 \]
   \[ = 94 \mod 256 \]
   \[ = 94 \]

2. \( CG = (G + K_i) \mod 256 \)
   \[ = (100 + Y) \mod 256 \]
   \[ = (100 + 89) \mod 256 \]
   \[ = 189 \mod 256 \]
   \[ = 189 \]

3. \( CB = (B + K_i) \mod 256 \)
   \[ = (159 + U) \mod 256 \]
   \[ = (159 + 85) \mod 256 \]
   \[ = 244 \mod 256 \]
   \[ = 244 \]

Piksel (1,0):

1. \( CR = (R + K_i) \mod 256 \)
   \[ = (0 + A) \mod 256 \]
   \[ = (0 + 65) \mod 256 \]
   \[ = 65 \mod 256 \]
   \[ = 65 \]

2. \( CG = (G + K_i) \mod 256 \)
   \[ = (0 + Y) \mod 256 \]
   \[ = (0 + 89) \mod 256 \]
   \[ = 89 \mod 256 \]
   \[ = 89 \]

3. \( CB = (B + K_i) \mod 256 \)
   \[ = (0 + U) \mod 256 \]
   \[ = (0 + 85) \mod 256 \]
   \[ = 85 \mod 256 \]
   \[ = 85 \]

Piksel (0,1):

1. \( CR = (R + K_i) \mod 256 \)
   \[ = (0 + A) \mod 256 \]
   \[ = (0 + 65) \mod 256 \]
   \[ = 65 \mod 256 \]
   \[ = 65 \]

2. \( CG = (G + K_i) \mod 256 \)
   \[ = (0 + Y) \mod 256 \]
   \[ = (0 + 89) \mod 256 \]
   \[ = 89 \mod 256 \]
   \[ = 89 \]

3. \( CB = (B + K_i) \mod 256 \)
   \[ = (0 + U) \mod 256 \]
   \[ = (0 + 85) \mod 256 \]
   \[ = 85 \mod 256 \]
   \[ = 85 \]

Piksel (1,1):

1. \( CR = (R + K_i) \mod 256 \)
   \[ = (56 + A) \mod 256 \]
   \[ = (56 + 65) \mod 256 \]
   \[ = 121 \mod 256 \]
   \[ = 121 \]

2. \( CG = (G + K_i) \mod 256 \)
   \[ = (180 + Y) \mod 256 \]
   \[ = (180 + 89) \mod 256 \]
   \[ = 269 \mod 256 \]
   \[ = 13 \]

3. \( CB = (B + K_i) \mod 256 \)
   \[ = (6 + U) \mod 256 \]
   \[ = (6 + 85) \mod 256 \]
   \[ = 91 \mod 256 \]
   \[ = 91 \]
3.2 Embedding Process

Piksel (0,0):

| R | G | B |
|---|---|---|
| 94(10) = 01011110(2) | 189(10) = 10111101(2) | 244(10) = 11110100(2) |

Piksel (0,1):

| R | G | B |
|---|---|---|
| 121(10) = 01111001(2) | 13(10) = 00001101(2) | 91(10) = 01011011(2) |

Piksel (1,0):

| R | G | B |
|---|---|---|
| 65(10) = 11000001(2) | 89(10) = 01011001(2) | 85(10) = 01010101(2) |

Piksel (1,1):

| R | G | B |
|---|---|---|
| 176(10) = 10110000(2) | 33(10) = 00100001(2) | 28(10) = 00011100(2) |

The message bits inserted from the leftmost first and the message is inserted in the last bit of pixel value on each RGB element of the cover image. The pixel element B is needed to insert a message, the message will be inserted in the last bits of the element in each pixel B. From the results of the embedding process above, we can see the comparison between the cover image and the stego image in Figure 3 (a) and (b) below.

Figure 2. Illustration of Encrypted 2x2 Pixel RGB Image

Figure 3 (a) Image Cover
Figure 3 (b) Stego Image

Based on figure 3 (a) and (b) Shows the difference between the cover image and the stego image, which is only 1 in the decimal number. Other pixel elements remain, which changes only in the last bit.

3.3 The experimental results processing time from each process are presented in Tables 1, 2, 3 and 4 below.

| Table 1. Encryption |
|--------------------|
| Encryption         |
| Size Image (Pixel) | Processing Time (ms) |
|-------------------|----------------------|
| 100 x 100 pixel    | 155,0088 ms          |
| 150 x 112 pixel    | 225,0129 ms          |
| 200 x 149 pixel    | 444,0254 ms          |

Based on table 1 the average results from the testing of process time carried out by the system during the encryption process using image files measuring 100x100 pixels, 150x112 pixels, 200x149 are 274,682367 ms.

| Table 2. Embedding |
|-------------------|
| Embedding         |
| Size Image (Pixel) | Processing Time (ms) |
|-------------------|----------------------|
| 525 x 700 pixel   | 4901,2804 ms         |
| 1000 x 1333 pixel | 14003,3009 ms        |
| 1080 x 1440 pixel | 19825,1339 ms        |

Based on table 2 the average results of the process time testing carried out by the system during the embedding process using an image file measuring 525x700 pixels, 1000x1333 pixels, 1080x1440 pixels are 12,9100717 ms.

| Table 3. Extraction |
|---------------------|
| Extraction          |
| Size Image (pixel)  | Time Processing(ms) |
|---------------------|---------------------|
| 525 x 700 pixel     | 54,0031 ms          |
| 1000 x 1333 pixel   | 83,0048 ms          |
| 1080 x 1440 pixel   | 147,0084 ms         |

Based on table 3 the average results of the process time testing carried out by the system during the extraction process using an image file measuring 525x700 pixels, 1000x1333 pixels, 1080x1440 pixels are 94.6721 ms.
Table 4. Decryption

| Size Image (pixel) | Time Processing (ms) |
|-------------------|----------------------|
| 100 x 100 pixel   | 1 ms                 |
| 150 x 112 pixel   | 1.0001 ms            |
| 200 x 149 pixel   | 3.0002 ms            |

Based on Table 4 the average result of the process time testing carried out by the system during the decryption process that uses image files measuring 100x100 pixels, 150x112 pixels, 200x149 pixels with an average of 1.67 ms.

Based on the tables 1,2,3,4 relationship between the time of the encryption and embedding process and image extraction and decryption with the pixel size of the image indicates that the pixel size of the image is directly proportional to the time. The bigger the pixel size of the image to be secured, the longer the time needed to process the image security.

4. Conclusions

The conclusion of this research, MSE and PSNR test results show that the use of the Blue Channel Trithemius and MLS algorithms is relatively safe to maintain image files. The process of encryption and embedding as well as extraction and decryption on image security using Trithemius and MLSB Blue Channel algorithms is able to provide security on messages and can restore the image file from the encryption – embedding process as before. Time process between the process of encryption, embedding, extraction and decryption of images with the pixel size of the image shows that the pixel size of the image is directly proportional to time. The larger the pixel size of the image to be secured, the longer it takes to secure the image.

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] Harjo, Teguh B., Marly K., dan Dwi. 2016. Aplikasi Steganografi Menggunakan LSB (Least Significant Bit) dan Enkripsi Triple DES Menggunakan Bahasa Pemrograman C#. Universitas Budi Luhur. ISSN: 2008 – 1762, Vol.6, No.1.
[2] Hendri. 2014. Kompresi Citra dari Format BMP ke Format PNG. Jurnal TIME, Vol. III No 1 : 27-31. ISSN : 2337 – 3601.
[3] Iswahyudi, Catur, dan Iwan R. 2008. Penyisipan Pesan Rahasia Pada Citra Digital Dengan Teknik Steganografi.
[4] Kamil, Fardian. 2016. Implementasi Kriptografi dengan Menggunakan Algoritma Advanced Encryption Standard (AES 256) dan Lempel Ziv Welch (LZW) sebagai Pengamanan Data pada PT. Lea Sanent. Skripsi. Tangerang: STMIK Raharja.
[5] Mollin, Richard A. 2005. Codes: the guide to secrecy from ancient to modern times : United State of America.
[6] Papp, J., 2006. Quality Management in The Imaging Science, Thrid Edition. Saint Louis : Mosby.
[7] Rakhmat, B. Dan Fairuzabadi, M. 2010. Steganografi menggunakan Metode Least Significant Bit dengan Kombinasi Algoritma Kriptografi Vignere dan RC4. Jurnal Dinamika Informatika. Vol.5, No.2, 2010.
[8] Zaher, M.A. 2011. Modified Least Significant Bit (MLSB), Kanada, Jurnal CCSENET Vol.4, No.1.