Analysis of Modified Least Significant Bit Polynomial Function Algorithm For Securing Digital Image

Dian Rachmawati*, Amer Sharif*, and Muhammad Adrinta Abdurrazzaq*

1Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia

*Email: dian.rachmawati@usu.ac.id, amersharifdjamin@gmail.com, adrintaabdurrazzaq@gmail.com

Abstract. Nowadays, there are many types of information, among which is the digital image. Some information must be secured, so it has to be made confidential. One of the methods to secure data is by steganography, wherein information is hidden inside another media called cover. Least Significant Bit (LSB) is the smallest bit in the binary array from the value of a byte. The polynomial function is the total of the power product in one or more variables with coefficients. The digital image can be used as a cover in steganography. The pixel selection for containing the message may be determined by the polynomial function so that the polynomial function can be used as the embedding and extraction key. The larger the ratio between the dimensions of the cover image to the image message, the better values of PSNR and MSE obtained. It can be concluded that the combination of the LSB method and polynomial function may serve as a modification to the LSB method in steganography.

1. Introduction

The word steganography combines the Greek words steganos (στεγανός), meaning "roof or cover" and graphein meaning "writing or drawing"[3]. Steganography is a method to hide information in which information is hidden inside a media called cover[1] so that anyone can access the cover without noticing that the media has been embedded with secret information. A polynomial function is the total sum of multiplication between the coefficient and the powered variable[2].

One of the steganography methods which can be used is the Least Significant Bit (LSB) method. LSB is part of the binary array with the smallest value that can be valued 0\text{\textsubscript{2}} or 1\text{\textsubscript{2}} at the binary column that is valued 2\text{\textsuperscript{0}}. Steganography with LSB method uses the last bit or the rightmost bit from the binary array of the cover as a container for binary data from the secret information[6].

The result of the steganography is stego data. In this research, the researcher uses the digital image as stego data so it will be called the stego image. Stego image is vulnerable to editing and compression operations in the case to maintain the data integrity[4]. The image is one of multimedia in the form of visual information. Images are created from pixels. These pixels consist of colors namely: red, green and blue referred to as RGB [11]. Digital image embedding to RGB image can do by using several digital image file types, among others are JPG, PNG, and BMP. Hence, the stego image must be BMP or PNG to maintain the data integrity[7].

Least Significant Bit steganography is one such technique in which least significant bit of pixels or the last bit pixels of the image is replaced with data bits from secret message[10]. The LSB method is already known to many, so this may lessen the strength of this method. However, the LSB method is
also ideal for steganography because the quality of the cover is only reduced slightly by the hidden information. In this research, the researcher used a modification to the LSB method where a polynomial function is used as the reference for embedding and as key for extracting the secret information. Data embedding to digital image needs the very large cover image if there is no limitation to the result of polynomial function [8]. To determine the quality of the cover image after embedding parameters PSNR and MSE were utilized.

The processing time to embed and extract the image message is affected by the image dimensions and the order of the polynomial function (n). The larger the dimensions, the larger the processing time. Also the higher the order of the polynomial function, the larger the processing times.

2. Method

In this implementation there are three main menus, they are Embed for embedding the secret message, Extract to extract the secret message, Compare to benchmark the stego image and the cover image or the secret image and the image from the extraction.

2.1 Steps for embedding the secret image:
1. Input the cover image, secret image, and polynomial function.
2. Convert width and length value of the secret image into 11 bits binary array by each value and categorize the secret image into 2 bits binary array. The category of the secret image is:
   a. 00 for binary image or black and white image.
   b. 01 for 8 bits grayscale image.
   c. 10 for 24 bits RGB image.
   d. 11 for 32 bits ARGB image.
   Combine the binary arrays into one binary array as a header.
3. Convert the secret image into a binary array.
4. Combine the header and secret image’s binary array into the secret binary array.
5. Embed the value of the secret binary array into R, G, B of the cover image’s pixel which has been chosen before with polynomial function. A pixel can contain 3 bits of the secret binary array’s value. The polynomial function as follows:

\[ y = (f(x_0…q-1) + c) \mod p \]

- f(x) : Function value
- p : Cover image height
- q : Cover image width
- c : The cycle of embedding, it can be valued between 0 until p-1.
- x : Variable, it can be valued between 0 until q-1
- y : Row coordinate

6. Do Step 5 until all the secret binary array’s value have been embedded.

2.2 Steps for extraction the secret image:
1. Input the right pair of the stego image and the polynomial function.
2. Get LSB’s value from the stego image's pixel that is chosen with a polynomial function, get the first 24 bits of it to recognize the dimensions and the category of the secret image.
3. Get LSB’s value from the stego image's pixel that is chosen with a polynomial function to get the secret image's binary array value.
4. Convert the secret image's binary array value into the byte.
5. Convert the byte into Image and use information from the header extraction to help reconstruct the image.

2.3 Steps for comparing:
1. Get the right pair of images, images that are the right pairs are:
   a. Stego image and cover image.
b. Secret image and the result of extraction's image.

2. Compare each pixel at the same coordinate by calculating the sum of the pixel’s value differences.

3. Calculate MSE and PSNR

4. If the MSE’s value is 0 that means the pair of images is identical.

Figure 1 shows the cover image and the secret image, the dimension of the cover image is 10x10 while the secret image is 1x4.

![Figure 1. Cover image (a) and secret image (b)](image)

From Figure 1, the steps for embedding and extraction are as follows:

1. Embedding Process
   a. Create a binary array of the secret image, as shown in table 1.

| Table 1. Secret Message’s Binary Array Structure |
|-----------------------------------------------|
| Information | Binary Array |
|-------------|--------------|
| Width Image | 00000000100  |
| Height Image| 00000000001  |
| Image Category| 10           |
| Secret Image| 0000000000000000000000000000000000011111111 |
| Binary Array | 11111110000000000000000000000000000000000 |
| Combination  | 00000000000000000000000000000000000000000 |

b. Embed the secret message’s binary array which is combination’s value to the selected pixel that is selected by the polynomial function. In this example, the polynomial function is \( f(x) = 1 + 2x + 3x^2 + 4x^3 + 5x^4 + 6x^5 \). The pixel selection can be seen in table 2.

| Table 2. Embedding information |
|-------------------------------|
| No.  | x | c | \( y=f(x)+c \mod p \) | Embedded Bit |
|------|---|---|------------------|--------------|
| 1    | 0 | 0 | 1                | R:0, G:0, B:0 |
| 2    | 1 | 0 | 1                | R:0, G:0, B:0 |
| 3    | 2 | 0 | 1                | R:0, G:0, B:1 |
| 4    | 3 | 0 | 5                | R:0, G:0, B:0 |
|   |   |   |   |   |
|---|---|---|---|---|
| 5 | 4 | 0 | 7 | R:0, G:0, B:0 |
| 6 | 5 | 0 | 1 | R:0, G:0, B:0 |
| 7 | 6 | 0 | 1 | R:0, G:0, B:0 |
| 8 | 7 | 0 | 1 | R:1, G:1, B:0 |
| 9 | 8 | 0 | 5 | R:0, G:0, B:0 |
|10 | 9 | 0 | 7 | R:0, G:0, B:0 |
|11 | 0 | 1 | 2 | R:0, G:0, B:0 |
|12 | 1 | 1 | 2 | R:0, G:0, B:0 |
|13 | 2 | 1 | 2 | R:0, G:0, B:0 |
|14 | 3 | 1 | 6 | R:0, G:1, B:1 |
|15 | 4 | 1 | 8 | R:1, G:1, B:1 |
|16 | 5 | 1 | 2 | R:1, G:1, B:1 |
|17 | 6 | 1 | 2 | R:0, G:0, B:0 |
|18 | 7 | 1 | 2 | R:0, G:0, B:0 |
|19 | 8 | 1 | 6 | R:0, G:0, B:1 |
|20 | 9 | 1 | 8 | R:1, G:1, B:1 |
|21 | 0 | 2 | 3 | R:1, G:1, B:1 |
|22 | 1 | 2 | 3 | R:1, G:0, B:0 |
|23 | 2 | 2 | 3 | R:0, G:0, B:0 |
|24 | 3 | 2 | 7 | R:0, G:0, B:0 |
|25 | 4 | 2 | 9 | R:1, G:1, B:1 |
|26 | 5 | 2 | 3 | R:1, G:1, B:1 |
|27 | 6 | 2 | 3 | R:1, G:1, B:0 |
|28 | 7 | 2 | 3 | R:0, G:0, B:0 |
|29 | 8 | 2 | 7 | R:0, G:0, B:0 |
|30 | 9 | 2 | 9 | R:0, G:0, B:0 |
|31 | 0 | 3 | 4 | R:0, G:0, B:0 |
|32 | 1 | 3 | 4 | R:0, G:0, B:0 |
|33 | 2 | 3 | 4 | R:0, G:0, B:0 |
|34 | 3 | 3 | 8 | R:0, G:0, B:0 |
|35 | 4 | 3 | 0 | R:0, G:0, B:1 |
|36 | 5 | 3 | 4 | R:1, G:1, B:1 |
|37 | 6 | 3 | 4 | R:1, G:1, B:1 |
|38 | 7 | 3 | 4 | R:1, G:1, B:1 |
|39 | 8 | 3 | 8 | R:1, G:1, B:1 |
|40 | 9 | 3 | 0 | R:1, G:1, B:1 |
Figure 2 shows the stego image and the manipulated pixel in the cover image mark with the white pixel.

![Stego image and manipulated pixel](image)

Figure 2. Stego image (a) and manipulated pixel (b)

2. Extraction Process
   a. Extract first 24 bits to get secret image information; the selected pixel can be seen in table 2. From the extraction, we get width = 4, height = 1, and image category is 24 bit RGB image. So that there are 4x1x24 bits to be extracted.
   b. Extract bits until total bits that be gotten is equal 96 exclude header bits.

3. Comparison Process
   a. Compare each pixel at the same coordinate by calculating the sum of the pixel's value differences. In this example, it will compare stego image and cover image before embedding. The tabulation of the comparison can be seen in table 3.

| No. | Pixel Coordinate (x,y) | (R'−R)^2 | (G'−G)^2 | (B'−B)^2 | [f'(x,y)−f(x,y)]^2 |
|-----|------------------------|-----------|-----------|-----------|---------------------|
| 1   | (0,1)                  | 0         | 0         | 0         | 0                   |
| 2   | (1,1)                  | 0         | 0         | 0         | 0                   |
| 3   | (2,1)                  | 0         | 0         | 1         | 1                   |
| 4   | (3,5)                  | 0         | 0         | 0         | 0                   |
| 5   | (4,7)                  | 0         | 0         | 0         | 0                   |
| 6   | (5,1)                  | 0         | 0         | 0         | 0                   |
| 7   | (6,1)                  | 0         | 0         | 0         | 0                   |
| 8   | (7,1)                  | 1         | 1         | 0         | 2                   |
| 9   | (8,5)                  | 0         | 0         | 0         | 0                   |
| 10  | (9,7)                  | 0         | 0         | 0         | 0                   |
| 11  | (0,2)                  | 0         | 0         | 0         | 0                   |
| 12  | (1,2)                  | 0         | 0         | 0         | 0                   |
| 13  | (2,2)                  | 0         | 0         | 0         | 0                   |
| 14  | (3,6)                  | 0         | 1         | 1         | 2                   |
15 (4,8) 1 1 1 3
16 (5,2) 1 1 1 3
17 (6,2) 0 0 0 0
18 (7,2) 0 0 0 0
19 (8,6) 0 0 1 1
20 (9,8) 1 1 1 3
21 (0,3) 1 1 1 3
22 (1,3) 1 0 0 1
23 (2,3) 0 0 0 0
24 (3,7) 0 0 0 0
25 (4,9) 1 1 1 3
26 (5,3) 1 1 1 3
27 (6,3) 1 1 0 2
28 (7,3) 0 0 0 0
29 (8,7) 0 0 0 0
30 (9,9) 0 0 0 0
31 (0,4) 0 0 0 0
32 (1,4) 0 0 0 0
33 (2,4) 0 0 0 0
34 (3,8) 0 0 0 0
35 (4,0) 0 0 1 1
36 (5,4) 1 1 1 3
37 (6,4) 1 1 1 3
38 (7,4) 1 1 1 3
39 (8,8) 1 1 1 3
40 (9,0) 1 1 1 3

\[ \sum (f(x,y) - f'(x,y))^2 = 43 \]

\[ \text{MSE} = \frac{1}{10 \times 10 \times 3} \times 69 = 0.14333 \]

\[ \text{PSNR} = 10 \times \log_{10} \frac{255^2}{43} = 56.5673316000799 \]

3. Results and Discussions

The experiments were performed on Windows 8.1 Notebook with Intel Celeron N2840 processor, 64-bit architecture, and 4096MB RAM. The Integrated Development Environment (IDE) used for coding is SharpDevelop and the programming language used is C#. The results of the experiments of each set are presented in Tables 4, 5 and 6 as follows.

**Table 4. Experiments with the same cover image and polynomial function**

| No | Image Dimension (Width x Height) | Comparison I | Comparison II | Running Time (millisecond) |
|----|---------------------------------|--------------|---------------|---------------------------|
|    | MSE (dB) | PSNR (dB) | MSE (dB) | PSNR (dB) | Embedding | Extraction |
| 1  | 500x500 | 0.4418253 | 51.6782975 | 0 | ∞ | 328 | 512 |
| 2  | 600x500 | 0.3681878 | 52.4701099 | 0 | ∞ | 333 | 516 |
| 3  | 700x500 | 0.3155895 | 53.1395778 | 0 | ∞ | 337 | 519 |
| 4  | 800x500 | 0.2454585 | 53.7194973 | 0 | ∞ | 335 | 516 |
In table 4, it shows process' running time doesn't depend on the cover image dimension. However, it shows the better value of MSE and PSNR if the cover image dimension is larger.

Table 5. Experiments with the same secret image and polynomial function

| No | Image Dimension (Width x Height) | Comparison I | Comparison II | Running Time (millisecond) |
|----|---------------------------------|--------------|---------------|---------------------------|
| 1  | 240x160                         | 0.0183399    | 65.4968454    | Embedding: 426 Extraction: 341 |
| 2  | 280x187                         | 0.0249858    | 64.1538749    | Embedding: 569 Extraction: 508 |
| 3  | 320x213                         | 0.0324404    | 63.0199357    | Embedding: 741 Extraction: 637 |
| 4  | 360x240                         | 0.0410594    | 61.9966768    | Embedding: 946 Extraction: 872 |
| 5  | 400x267                         | 0.0506502    | 61.0849901    | Embedding: 1157 Extraction: 1092 |
| 6  | 440x293                         | 0.0609831    | 60.2787079    | Embedding: 1397 Extraction: 1332 |
| 7  | 480x320                         | 0.0726794    | 59.5166899    | Embedding: 1658 Extraction: 1593 |
| 8  | 520x347                         | 0.0852385    | 58.8244447    | Embedding: 1958 Extraction: 1893 |
| 9  | 560x373                         | 0.0985299    | 58.1951220    | Embedding: 2252 Extraction: 2187 |
| 10 | 600x400                         | 0.1131493    | 57.5942836    | Embedding: 2617 Extraction: 2552 |

In table 5, process' running time get longer when embedding or extracting the larger secret image.

Table 6. Experiments with different polynomial function order

| No | Ordo | f(x)          | Running Time (millisecond) |
|----|------|---------------|----------------------------|
|    | 0    | 1             | Embedding: 86 Extraction: 230 |
|    | 1    | 1 + 2x        | Embedding: 127 Extraction: 266 |
|    | 2    | 1 + 2x + 3x²  | Embedding: 169 Extraction: 308 |
|    | 3    | 1 + 2x + 3x² + 4x³ | Embedding: 199 Extraction: 335 |
|    | 4    | 1 + 2x + 3x² + 4x³ + 5x⁴ | Embedding: 232 Extraction: 376 |
|    | 5    | 1 + 2x + 3x² + 4x³ + 5x⁴ + 6x⁵ | Embedding: 261 Extraction: 407 |

In table 6, process’ running time get longer when embedding or extracting using the higher order of the polynomial function.

4. Conclusions

It can be concluded that the secret message integrity is verified and maintained. Process running time depends on the size of the secret image and the order of the polynomial function; larger values make it longer. PSNR and MSE depend on the ratio between cover image and secret image; it gets better value when the ratio is larger. LSB and polynomial function combination may serve as a modification for the LSB method in steganography.

5. Acknowledgments

The authors gratefully acknowledge that the present research is supported by Fund Dissemination IPTEKS Research Results for Lecturers / Researchers Universitas Sumatera Utara.
References

[1] Akhtar, Nadeem, Vasim Ahmad and Hira Javed 2017 A Compressed LSB Steganography Method. Institute of Electrical and Electronics Engineers DOI: 10.1109/CIACT.2017.7977371

[2] Barbeau, E.J 2003 Polynomials New York Springer

[3] Kothari, Lipi, Rikin Thakkar and Satvik Khara 2017 Data Hiding on Web Using Combination of Steganography and Cryptography. Institute of Electrical and Electronics Engineers. DOI: 10.1109/COMPTELIX.2017.8004011

[4] Muadzani, Alim, Oky Dwi Nurhayati and Ike Pertiwi Windasari 2016 Jurnal Teknologi dan Sistem Komputer ISSN: 2338-0403 DOI: 10.14710/jtsiskom.4.3.2016.470-478

[5] Nurhayati, & Syukri Sayyid Ahmad 2016 Steganography for Inserting Message on Digital Image Using Least Significant Bit and AES Cryptography Algorithm. Institute of Electrical and Electronics Engineers DOI: 10.1109/CITSM.2016.7577468

[6] Rahmani, V., & Mostafa MohammadPour 2017 High Hiding Capacity Steganography Method Based On Pixel Indicator Institute of Electrical and Electronics Engineers DOI: 10.1109/CFIS.2017.8003673

[7] Sannawira, Rahmandhita Fikri, and Agus Sidiq Purnomo 2016 Penyisipan Citra Pesan Menggunakan Metode Least Significant Bit dan Redundant Pattern Encoding. Informatics Journal Vol. 1 No.1:39-46 ISSN: 2503-250X

[8] Sari, Mega Kartika 2014 Modifikasi Least Significant Bit (LSB) Menggunakan Persamaan Kuadrat Pada Kunci Steganografi Skripsi Medan Indonesia Universitas Sumatera Utara.

[9] Sugandhi, G., and C.P. Subha. Efficient Steganography Using Least Significant Bit And Encryption Technique 2016 Institute of Electrical and Electronics Engineers DOI: 10.1109/ISCO.2016.7726971.

[10] D. Rachmawati and M. A. Budiman New approach toward data hiding by using affine cipher and least significant bit algorithm 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT) Kuta Bali 2017 pp 1-6 doi: 10.1109/CAIPT.2017.8320737

[11] D Rachmawati et al 2018 IOP Conf. Ser.: Mater. Sci. Eng. 308 012003