Analysis on Combination of Watermarking Algorithm: Modified Least Significant Bit Algorithm with Least Significant Bit+1

R R A Lubis, S M Hardi, M Zarlis, I Jaya, J T Tarigan

1,2,3,4, 5 Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. Universitas No.9, Padang Bulan, Medan Baru, Sumatera Utara 20155

E-mail: rifkiashari@gmail.com, vani.hardi@usu.ac.id, m.zarlis@yahoo.co.id

Abstract. To protect someone’s copyright on digital media such as digital image, watermarking technique can be used, which is a technique to insert copyright marks into digital media. The algorithms that commonly used in the watermark insertion is the Least Significant Bit (LSB). But there is deficiency in LSB algorithm. LSB algorithm requires one byte image pixel to insert one bit watermark. The more byte pixels are inserted, the higher the difference between watermarked image with the original image. Embedded watermark can also easily be known because it is definitely placed on the LSB bits of the pixel image. From these problems, in this research we analyze the combination of two watermarking algorithms: Modified Least Significant Bit (MLSB) with Least Significant Bit (LSB+1). MLSB have the advantage in compressing the watermark to be inserted from eight bit/character into five bit/character. LSB+1 algorithm will cover the deficiency of MLSB algorithm, which the location of the watermark bit is easy to be discovered. Added in this research, the watermark bit had inserted in random image pixel. The Combined algorithm tested by using five image that have different sizes and dimensions, which each of the image inserted by text watermark that have different length. The purpose is to find the value of these three comparison parameters: Mean Squared Error value (MSE), the size of the result image and processing time. It concluded that for the three parameters, the combined algorithm of MLSB-LSB+1 have done better than the LSB+1algorithm but slightly worse than the MLSB algorithm. This happens due to MLSB-LSB+1 besides inheriting the advantages of each algorithm, also inherited deficiency of theLSB+1 algorithm which is because of the insertion is in the one bit before the least significant bit of the image pixel, the value of the pixel is greater for two point(in decimal).

1. Introduction

In the current era of information and technology, data and information are presented in digital format, in the form of text, image, audio and video. This digital product has several characteristics, among others: easily duplicated, easy to distribute and changes can not be perceived by the sense of sight. Problems arise when this digital product is a protected work such as electronic book writing, photographic artwork, audio art, and video of one's important event (Munir, 2006).

The above problems can be solved by using watermarking method. Watermarking is a technique for inserting certain information called watermarks into digital media. The watermark insertion is done in
such a way that the watermark does not damage the protected digital data. Inserted watermarks can not be perceived by the human senses, but can be detected by the computer using the correct key (Munir, 2006).

The commonly used algorithm for inserting watermarks into digital media is the Least Significant Bit (LSB). The basic principle of this method is to replace the last bit of each data with insert bits. In other words every single bit of the insertion message requires one byte of media cover data, so for every single insert text character it takes eight bytes of media cover data because a byte insert consists of eight data bits (Seyyedi, 2013). Messages that are hidden with this algorithm can also be easily known, because the message bits are definitely in the LSB bit of the digital media (Utami, 2009).

Modified Least Significant Bit Algorithm (MLSB) is a modification of the LSB algorithm. This algorithm works by replacing the text message bits that 1 character must have an 8 bit value ASCII code (American Standard Code for Information Interchange) will be modified to 5 bits. Modification is done by converting insert bits with ASCII values. After the insertion bits are encoded with ASCII, they are combined with the ASCII code of control symbols (Control Symbols). Subsequent reduction of all insertion bits with the lowest bit value. Before being inserted into the image, the insertion bits are converted into binaries that generate 5 bits of each message value. The MLSB algorithm is more efficient, since the number of insert character bits is modified to five bits only, so for one insertion character requires only five bytes of media cover (Zaher, 2011). But the MLSB method in the study, has the disadvantage that the bit position inserted on the media cover is easy to read because it is placed in the last bit position and not random.

To strengthen the watermark insertion technique with MLSB, it can be done by inserting insert bits into the last 2 bit number. This method is called Least Significant Bit +1 (LSB +1) (Nath, 2011). The insertion process can also be done randomly. In the Saefullah (2012) study, for example, if there are 50 bytes and 6 bits of data to be hidden, then the byte replaced by the LSB +1 bit is randomly selected, for example bytes 36, 5, 21, 10, 18, 49. Numbers This random can be generated with Pseudo-Random-Number-Generator (PRNG). Research Laskar (2013), insertion is done randomly to pixel image with Random Number Generator. After the pixel is specified to be inserted, the message bit will be inserted in the RED pixel byte. By applying the Least Significant Bit +1 algorithm and the PRNG method to determine the bytes to be inserted in the watermarking process, the presence of the insertion is difficult to find because the insertion bits are placed on the second LSB bit for each byte cover and the location is randomized.

On top of that background, the author intends to do a combination of both algorithms by taking advantage of each of the above algorithms and give the title of this thesis research with Combination Algorithm Algorithm Watermarking Modified Least Significant Bit (MLSB) with Least Significant Bit +1 (LSB +1 ).

2. Method of MLSB-LSB+1 Combined Algorithm
The MLSB-LSB+1 algorithm will conversing the insertion bit to ASCII value then the insertion bit in ASCII will merge with ASCII Control Symbols. After that process, the algorithm performed the reduction of the insertion bit with the lowest bit value. After this process, the insertion bit will have five bit per character from eight bit per character.

Since length of the insertion bit is five bit per character, we will insert that bit into one bit before least significant bit of cover image pixel. The name of this method is Least Significant Bit+1 (LSB +1). This insertion process is done randomly, so we use Pseudo-Random-Number-Generator (PRNG) method to generate the chosen pixel that will be inserted.

The above process are described as follows:
1. Convering the insertion bit to ASCII value. Example of message: “STEGO with 05 bits”, if it change to binary message: 18 character x 8 bit per character = 144 bit length. That message if conversed to ASCII (hex) will be:

   53_{16}, 54_{16}, 45_{16}, 47_{16}, 4F_{16}, 20_{16}, 77_{16}, 69_{16}, 74_{16}, 68_{16}, 20_{16}, 30_{16}, 35_{16}, 20_{16}, 62_{16}, 69_{16}, 74_{16}, 73_{16}

2. The message in ASCII will normalize with Control Symbol Table 1.
5. 

4. 

watermarked image pixel will be showed as follows:

If we inserting this line of messages:

That binary message will be inserted to defining the pixel that will inserted by bit of message. Before the insertion process, this algorithm will generate random number as much as number of length.

This message line has 115 bit length that will be inserted to cover image pixels to defining the pixel that will inserted by bit of message.

11110, 00000, 00101, 11101, 11011, 01001, 10100, 01000, 11101, 11110, 00000, 00101, 11111, 11011, 01001, 10100, 01000, 11101, 11110, 00000, 00101, 11111, 11011, 01001, 10100, 01000, 11101, 11111.

This group merge with the second group and adding Control Symbol 1D_{16} for space and 1E_{16} for number. The result line: 1D_{16}, 1B_{16}, 02_{16}, 09_{16}, 14_{16}, 13_{16}F_{16}.

e. The fourth group of line message, which is 62_{16}, 69_{16},74_{16}, 73_{16}, will XOR with the lowest tens from the line which is 60_{16}. So it obtained: 62_{16} \oplus 60_{16} = 02_{16}, 69_{16} \oplus 60_{16} = 09_{16}, 74_{16} \oplus 60_{16} = 14_{16}, 73_{16} \oplus 60_{16} = 13_{16}. This group merge with the lowest group and adding Control Symbol 1D_{16} for space, 1B_{16} for small letter and 1F_{16} at the end of the text. The result line: 1D_{16}, 1B_{16}, 02_{16}, 09_{16}, 14_{16}, 13_{16}F_{16}.

3. The message after normalization will be:

1C_{16}, 13_{16}, 14_{16}, 05_{16}, 07_{16}, 0F_{16}, 1D_{16}, 1B_{16}, 17_{16}, 09_{16}, 14_{16}, 08_{16}, 1D_{16}, 1E_{16}, 00_{16}, 05_{16}, 1D_{16}, 1B_{16}, 02_{16}, 09_{16}, 14_{16}, 13_{16}, 1F_{16}.

That message transform to binary line will be:

11100, 10011, 10100, 00101, 00111, 01111, 11011, 11011, 01111, 01001, 10100, 01000, 11101, 11110, 00000, 00101, 11111, 11011, 01001, 10100, 01000, 11101, 11111.

This message line has 115 bit length that lesser than the original message line which has 144 bit length.

4. Before the insertion process, this algorithm will generate random number as much as number of image pixels to defining the pixel that will inserted by bit of message.

5. That binary message will be inserted to cover image pixel, exemplified as follows:

|Hex Representation| Operation       |
|------------------|----------------|
|1B_{16}           | Define Small Letter |
|1C_{16}           | Define Capital Letter |
|1D_{16}           | Define Space        |
|1E_{16}           | Define Number       |
|1F_{16}           | Define end of text  |

\begin{tabular}{|c|c|}
\hline
010010010 & 010010010 \\
000010000 & 000000000 \\
010101110 & 010010010 \\
011111011 & 000110101 \\
000000011 & 011111100 \\
011010000 & 000000000 \\
\hline
\end{tabular}

If we inserting this line of messages: 11100, 10011, 10100, 00101, 00111, 01111, 11011, 11011, 01111, 01001, 10100, 01000, 11101, 11110, 00000, 00101, 11111, 11011, 01001, 10100, 01000, 11101, 11111 (STEGO), then watermarked image pixel will be showed as follows:

3
The test use LSB+1, MLSB and ML-LSB+1 method to comparing each other. The first block is control symbol 1C_{16} that represented capital letter. The next block that is not control symbol will XOR with 40. Will be: 53
showed as follows:

|    |    |    |    |    |
|----|----|----|----|----|
| 52 | 4B | 46 | 44 | 28 |
| 0A | 00 | 00 | 57 | 43 |
| 56 | 45 | 46 | 15 | 2D |
| 3D | 1D | 2A | 00 | 4F |
| 01 | 7C | 03 | 12 | 47 |
| 68 | 02 | 03 | 13 | 07 |

For the extraction process will described as follows:
1. Read the watermarked image and it random number file
2. Read the watermarked image pixel that pointed by the random number file
3. The pixel value transform to binary value
4. Get the one bit before least significant bit in each pixel and group it into five bits per group/block
5. Converting each block into ASCII value (hexadecimal)
6. If we found control symbol, than do:
   a. If 1B_{16} then each next block that is not control symbol will XOR with 60
   b. If 1C_{16} then each next block that is not control symbol will XOR with 40
   c. If 1E_{16} then each next block that is not control symbol will XOR with 30
   d. If 1D_{16} then it represent space character
7. Step 3 to 5 repeated after we found control symbol end of text (1F_{16})
8. Reconstruct each block as watermark message

If we use the insertion example before, the extraction process will described as follows:

|    |    |    |    |    |
|----|----|----|----|----|
| 01010010 | 01001011 | 01000110 | 01000100 | 00101000 |
| 00001010 | 000000000 | 00000000 | 01010111 | 01000011 |
| 01010110 | 01000101 | 01000110 | 00101010 | 00101100 |
| 00111010 | 00011101 | 00101010 | 00000000 | 00100111 |
| 00000001 | 01111100 | 00000111 | 00100101 | 01000111 |
| 01101000 | 00000010 | 00000111 | 00010011 | 00000111 |

We get one bit before least significant bit in each pixel and group it into five bits per group/block, showed as follows: 11100 10011 10100 00101 00111 01111. If converting each block into ASCII value (hexadecimal): 1C_{16}, 13_{16}, 14_{16}, 05_{16}, 07_{16}, 0F_{16}. The first block is control symbol 1C_{16} that represent capital letter. The next block that is not control symbol will XOR with 40. Will be: 53_{16}, 54_{16}, 45_{16}, 47_{16}, 4f_{16}. This hexadecimal value represent the watermark message: STEGO

3. Test and Result
The test is processed by using five digital images of different size in which each image will be inserted with one text as watermark. The parameters to be obtained are the MSE, the length of the process and the result of file size. The test use LSB+1, MLSB and MLSB-LSB+1 method to comparing each other method.
The result of this test showed in the below table:
From the comparison graphics, the authors make conclusions into three parts: MSE Value, Process Time and File Size.

3.1. Results of Comparison of MSE Values
In the graph above can be seen that the algorithm Modified Least Significant Bit (MLSB) superior in the comparison of MSE parameters. This is because the MLSB algorithm gets the binary watermark into five bits per character then inserts the message bit in the Least Significant Bit position. The bit changed in the least significant bit positions do not have a big effect on the inserted image so that the MSE value is not larger.

This is not found in the use of LSB+1 algorithm. The LSB+1 binary watermark algorithm consists of 8 bits of each character, requiring more pixels to accommodate the binary watermark. Additionally, because it is inserted in the first bit before the least significant bitpixel, it causes the pixel changes more than if it is inserted at least significant bit, which causes the MSE value to be greater than using MLSB.

MLSB-LSB+1 combination algorithm is in the second position. The MSE value obtained by this algorithm is not too bad compared to the MLSB algorithm and is much better than the LSB+1 algorithm. This is because the MLSB-LSB+1 algorithm has the ability to compress the watermark to five bits per character so that the required pixels are less than the LSB+1. Higher MSE values than MLSB are obtained because this combination algorithm has the ability to insert messages on one bit before least significant bitpixel. But it make more difficult for certain parties to do message extraction and change the watermark.

3.2. Results Process Comparison Time
In the graph above, for the time parameter of each algorithm process slightly adrift. This is because each algorithm has its advantages and disadvantages that affect the processing time in addition to each tested algorithm performs a randomization of numbers to determine the pixel position to be inserted watermark. The advantages of MLSB algorithm that require less pixel to insert watermark compared with LSB+1 algorithm shorten the processing time of this algorithm. But the advantages of LSB+1 algorithm that do not need to process the compression watermark into five bits per character but directly insert the watermark bits into the pixel also shorten the processing time.

The MLSB-LSB+1 combination algorithm also does not require much greater processing time than the two algorithms above. MLSB-LSB+1 uses almost the same time as the MLSB algorithm because it requires only a small pixel to be inserted compared to the LSB+1 algorithm.

3.3. Comparison of File Size
In the graph above, MLSB algorithm file size parameter has the smallest watermarked image file size. It is because the MLSB algorithm requires less pixels and the pixel values only increase or decrease for one (in decimal) when embedded, because it inserted in the least significant bit pixel position. The LSB+1 algorithm has the largest watermarked image file size. It is because the LSB+1 algorithm requires more pixels than MLSB and the pixel values had increase or decrease for two (in decimal) when embedded, because it inserted at position one after the least significant bit pixel.

The image file size of the watermark MLSB-LSB+1 algorithm is slightly larger than the MLSB algorithm but smaller than the LSB+1 algorithm. the MLSB-LSB+1 algorithm although it takes less
pixel to inserting the watermark, but the pixel values had increase or decrease for two (in decimal) when embedded, because it inserted at position one after the least significant bit pixel.

4. Conclusions
Based on the comparison of LSB+1, MLSB and MLSB-LSB+1 of the three parameters: MSE image, process time and image size, obtained that on MSE parameter and image size, MLSB algorithm has image with smaller MSE image and file size than the LSB+1 algorithm that has the largest MSE image and file size. For process time parameters, it is concluded that each algorithm has almost the same processing time because in addition to depending on the size of the image and the message to be inserted, each algorithm has its advantages and disadvantages that affect the processing time.

MLSB-LSB+1 combination algorithm data shown that based on MSE image parameter, process time and image size are above LSB+1 algorithm and not so much different from MLSB algorithm.

The combination algorithm MLSB-LSB+1 inherited the advantages and cover the deficiencies of each MLSB and LSB+1 algorithms. The advantages of the MLSB algorithm that compresses the message to be inserted closes the shortcomings of the LSB+1 algorithm that requiring the number of image pixels to accommodate large messages. The advantages of the LSB+1 algorithm in the difficulty of extracting messages covering the weaknesses of the MLSB algorithm that are easily extracted if the symbol table is known.

References
[1] Laskar, S.A & Hemachandran, K. 2013. Steganography Based On Random Pixel Selection For Efficient Data Hiding. *International Journal of Computer Engineering and Technology (IJCIET)*, Volume 4, Issue 2, March – 2013.
[2] Munir, Rinaldi. 2006. *Kriptografi*. Bandung. Penerbit: Informatika.
[3] Nath, Joyshree, Sankar Das, Shalabh Agarwal & Asoke Nath. 2011. A Challenge in Hiding Encrypted Message in LSB dan LSB+1 Bit Position in Various Cover Style. *Journal of Global Research in Computer Science*. ISSN: 2229-371X Vol. 2 No. 4 April 2011.
[4] Utami, E. 2009. Pendekatan Metode Least Bit Modification Untuk Merancang Aplikasi Steganografi Pada File Audio Digital Tidak Terkompresi. *Jurnal Dasi ISSN: 1411-3201 Vol. 10 No. 1 Maret 2009*. STMIK AMIKOM Yogyakarta.
[5] Saefullah, A., Himawan & Agani, N. 2012. Aplikasi Steganografi Untuk Menyembunyikan Teks dalam Media Image Dengan Menggunakan Metode LSB. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2012 (Semitk 2012)*. Semarang, 23 Juni 2012.
[6] Seyyedi, S.A. & Sadykhov, R.K. 2013. *Digital Image Steganography Concept and Evaluation*. International Journal of Computer Applications (0975 – 8887) Volume 66– No.5, March 2013.
[7] Zaher, Mazen Abu. 2011. Modified Least Significant Bit (MLSB). *Jurnal Computer and Information Science Vol. 4, No. 1, Januari 2011*. www.ccsenet.org/cis. Diakses tanggal 15 Maret 2012 : 60-67.