Implementation of Linear Congruential Generator (LCG) Algorithm, Most Significant Bit (MSB) and Fibonacci Code in Compression and Security Messages Using Images

M Elveny†, R Syah†, I Jaya† and I Affandi†

† Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Padang Bulan 20155 USU, Medan, Indonesia

*Email: marischaelveny@usu.ac.id

Abstract. The use of technology makes information spread more quickly compared to conventional methods that humans used before they knew technology. However, there is information that is only intended for certain people or institutions because the information is very confidential. The obstacle in exchanging information is the problem of information / data security. Because of that, an electronic message security algorithm which is quite well-developed is cryptographic (cryptography) and steganographic (steganography) algorithms. The development of increasingly advanced technology and the addition of more and more computer users also cause data to accumulate and transfer data from one device to another. The data is generally compressed first so that the exchange process does not take too long. Therefore, we need a system that can secure messages while reducing image data to be more efficient in the process of transmission and delivery. To implement this problem the system is designed using the Linear Congruential Generator (LCG) algorithm and the Most Significant Bit (MSB) as a security medium and using the Fibonacci Code algorithm as compression of the image. In implementation the system has a weakness in compressing images that not dominate colors because the size of the data becomes larger. In addition, the compression ratio on the variation of text length has decreased and has increased in the image variation experiments where if the compression ratio decreases, indicating better compression.

1. Introduction
Nowadays technological developments have greatly influenced human life, especially in the field of information data. There are various types of data that can be loaded, such as text, images, audio (sounds, sounds, music) and videos. These four types of data are generally known as multimedia. Exchange of information by utilizing technology enables information to spread more quickly compared to conventional methods that were used by humans before they knew technology. This is a strong reason for informants to use Internet media [12].

However, sometimes there is information that is only intended for certain people or institutions because the information is very confidential. And the obstacle in exchanging information like this, of course, is the problem of information / data security. For this reason, a well-known electronic message security algorithm was developed, namely the cryptographic (cryptography) and steganographic (steganography) algorithms.
The development of increasingly advanced technology and the addition of more computer users has caused the accumulation of data and the transfer of data from one device to another. The data is generally compressed first so that the exchange process does not take too long.

Data compression is one of the processes used to reduce file size thereby reducing data storage space. This is very useful because the better file quality requires a very large size, so that with the use of data compression methods, file size can be reduced without reducing the quality of the file [11].

Fibonacci code compression method is a type of compression method based on coding with a universal code system where the positive integer value of the data forms a binary code word using Fibonacci numbers [7].

The method used to insert messages into an image file is the Most Significant Bit (MSB) method. This method will replace the leftmost (largest) bit of the image pixel with the insertion bits. To insert a message into a color image, the image and the insertion message must be changed to binary. Converting a color image into a binary image is used to get one value at one pixel image, where one pixel of the binary image will be replaced by one bit from eight bits of one insertion message character.

2. Related Works

Linear Congruential Generator (LCG) is the most basic and popular PRNG. This algorithm was created by D.H. Lehmer in 1951. It simply formed the formula (Lehmer, D. H., 1951):

\[ X_{n+1} = (aX_n + b) \mod m \] …………………………….. (1)

where:
- \( X_{n+1} \) = random number \((n + 1)\) of rows of random numbers
- \( X_n \) = \(n\)th random number
- \( a \) = multiplier constant
- \( b \) = increase constant
- \( m \) = modulus constant

The initial values of \( X_0 \) or \( X_{n+1} \) and the constants \((a, b, \text{ and } m)\) determine the quality of the random numbers obtained. A good category of random numbers is the repetition of the same random number after many random numbers are generated and it is not predicted when the repetition will occur. The key to generating the number is \( X_0 \), which is called seed. The LCG period is not greater than the value of \( m \). If \( a, b, \text{ and } m \) are chosen correctly with \( b \) relatively prime to \( m \) and \( b < m \), then LCG will have a maximum period, \( m - 1 \).

Steganography comes from the Greek word steganos, which means "hidden / veiled," and graphein which means "writing." Herodotus is known as a Greek historian who first introduced the word steganography [16]. The process of inserting messages into a covertext media is called encoding, while the extraction process from stegotext is called decoding. There is a steganography algorithm that may require a secret key (called a stego-key) so that only certain parties are entitled to the process of inserting and extracting messages. [19]

![Figure 1. Schematic diagram of message insertion and extraction (Munir, 2006).](image-url)
The steps in making Fibonacci codes are as follows:

- Determine a positive integer $n$ which is greater or equal to 2.
- Determine the biggest Fibonacci number that is smaller or equal to $n$, subtract the value of $n$ by $f$, and note the remainder of the subtraction of $n$ value by $f$.
- If the number subtracted is the number contained in the Fibonacci F series (i), add the number "1" to i-2 in the Fibonacci code to be formed.
- Repeat step 2, exchange the value of $n$ with the remainder of the subtraction of $n$ value by $f$ until the remainder of the subtraction of $n$ value by $f$ is 0.
- Add the number "1" to the rightmost position of the Fibonacci code to be formed.

In the process of inserting a message into a digital image using the Most Significant Bit (MSB) method, the message is inserted at bit 1.

3. Material and Method

The process in the general architecture diagram of the system above has 2 actors, namely the sender and receiver, where the sender will first insert the character and compress the image then send it to the receiver.

- On the sender side, they will first input text from a .txt or .doc file that is uploaded on the dialog box and the .bmp / .png / .jpg image that will be processed into the system.
- After that the system will process the insertion of characters into the image with the MSB method by randomly using a Linear Congruential Generator (LGC).
- Then the image will be compressed with a Fibonacci code. After this process a stego-compressed file will be generated and sent to the receiver.
- On the receiver side, they will receive images that have been sent and they will input stego-compressed files into the system.
- At the beginning of input, the system will decompress the image with Fibonacci code and then extract the characters that are inserted into the image with MSB.
- After extraction, the receiver will get the text or characters sent by the sender.

The message will be inserted in the last bits of each element of the cover image binary in the order of R, G, B. Cover pixels to be manipulated are pixels (3), (7), (4), (6), (0). The insertion process is in Table 1 below:

### Table 1. The insertion results using MSB+2

| R:  | R:  | R:  |
|-----|-----|-----|
| 10011111 | 10011111 | 10011111 |
| G:  | G:  | G:  |
| 00011001 | 00011001 | 00011001 |
| B:  | B:  | B:  |
| 01100010 | 01000010 | 01000010 |

| R:  | R:  | R:  |
|-----|-----|-----|
| 10011111 | 10011111 | 10011111 |
| G:  | G:  | G:  |
| 00111001 | 00111001 | 00111001 |
| B:  | B:  | B:  |
| 01100010 | 01000010 | 01000010 |

| R:  | R:  | R:  |
|-----|-----|-----|
| 10111111 | 10011111 | 10011111 |
| G:  | G:  | G:  |
| 00111001 | 00111001 | 00111001 |
Table 2. Table of Fibonacci Codes Conversion from Image Color Values.

| Character | Frequency | Fibonacci Code | Bit | Bit x Frequency |
|-----------|-----------|----------------|-----|-----------------|
| 159       | 8         | 11             | 2   | 16              |
| 66        | 7         | 011            | 3   | 21              |
| 25        | 5         | 0011           | 4   | 20              |
| 57        | 4         | 1011           | 4   | 16              |
| 98        | 2         | 00011          | 5   | 10              |
| 191       | 1         | 10011          | 5   | 5               |

Table 3. Results of Fibonacci code conversion.

| R: 11 | R: 11 | R: 11 |
|-------|-------|-------|
| G: 0011 | G: 0011 | G: 0011 |
| B: 00011 | B: 011 | B: 011 |

| R: 11 | R: 11 | R: 11 |
|-------|-------|-------|
| G: 1011 | G: 1011 | G: 0011 |
| B: 00011 | B: 011 | B: 011 |

| R: 10011 | R: 11 | R: 11 |
|----------|------|------|
| G: 1011 | G: 1011 | G: 0011 |
| B: 011 | B: 011 | B: 011 |

If the rearrangement of the cover image pixels has been performed, the compressed bit string will become

11111111 11111001 11111001 10011001 11011101 10011101 11011001 10001101 10110001 10110110 11011011

After distributing 8 bits as above, each block will be converted into decimal form which can be seen in the results below

255 249 249 153 221 157 217 141 177 182 219

The results will be saved into a file with the .fb format which will then be sent to the receiver of the message.

To test the system, user does the "embed user" process by pressing the "Embed - compression" menu then "Browse text" to find the text file to be inserted into the image. After that the image will be a cover object. After doing that user will be asked to press the "Embed" button to embed the text into the image. The embed process is complete and will be displayed in the "MSB Results" picture box. The process will display the image, the running time and the original size of the image after the insertion process. Once the insertion process is done, user can do the compression process in order to reduce the size of the embed image.

To do compression, user must press the compression button and they will get some information, including compression ratio, compression time, compress ratio, space saving, and compress size. After performing the compression process, user can do the "save" process to save the results of the compression in the *.fb extension.
The results obtained from this study are the results of the embed-compression process and the extract-decompression process of images and text files with 5 variations of resolution for each image and 5 length variations for each text file.

**Table 4.** Experiment with text variations.

| No | Length of Text | Running Time | Original size | Ratio Compression | Compression Time | Compression Ratio | Space Saving | Compress Size |
|----|----------------|--------------|---------------|-------------------|-----------------|------------------|--------------|--------------|
| 1  | 100            | 3.42 s       | 1162          | 54.15%            | 8.26s           | 1.85             | 45.85%       | 629 Kb       |
| 2  | 200            | 3.44 s       | 1162          | 54.17%            | 8.27s           | 1.85             | 45.83%       | 629 Kb       |
| 3  | 300            | 3.60 s       | 1162          | 54.2%             | 8.24s           | 1.85             | 45.8%        | 630 Kb       |
| 4  | 400            | 3.71 s       | 1162          | 54.22%            | 8.49s           | 1.84             | 45.78%       | 630 Kb       |
| 5  | 500            | 3.77 s       | 1162          | 54.24%            | 9.79s           | 1.84             | 45.76%       | 630 Kb       |

**Figure 2.** Chart of Inter-Time Experiment - Experiment of text length variations.

On the chart above, it can be seen that the compression time of each process and types of time always increases. The chart always goes up because when more texts are inserted, the longer is the running time needed.

**Figure 3.** Chart of Inter Compress Experiment - Experiment of text length variations.

On the chart above, it can be seen that on the compression comparisons of each process and types of compresses, the compression ratio increases but the space string decreases. Compression ratio increases because the size of the data experiences a greater difference between before and after compression.
In this experiment, 5 variations of image resolution were tested, including 100x100 pixels, 200x200 pixels, 300x300 pixels, 400x400 pixels, and 500x500 pixels. Each pixel is inserted with text that has a length of 100 characters. The test details are made in the form of the following table:

Table 5. Experiment with variations in image resolution.

| No | Image Resolution | Running Time | Original Size | Ratio Compression | Compression Time | Compression Ratio | Space Saving | Compress Size |
|----|------------------|--------------|---------------|-------------------|------------------|-------------------|--------------|---------------|
| 1  | 100x100          | 0.07 s       | 29Kb          | 48.27%            | 0.03 s           | 2.07              | 51.73%       | 14Kb          |
| 2  | 200x200          | 0.30 s       | 117Kb         | 48.34%            | 0.18 s           | 2.07              | 51.66%       | 56Kb          |
| 3  | 300x300          | 0.57 s       | 263Kb         | 47.94%            | 0.49 s           | 2.09              | 52.06%       | 123Kb         |
| 4  | 400x400          | 1.22 s       | 468Kb         | 46.85%            | 0.65 s           | 2.13              | 53.15%       | 224Kb         |
| 5  | 500x500          | 1.74 s       | 732Kb         | 43.2%             | 0.98 s           | 2.31              | 56.8%        | 316Kb         |

Figure 4. Chart of Inter-time - Experiment with variations in the size of the image resolution.

On the chart above it can be seen the comparison of time of each process and types of time, where time consumed always increases. It is clearly seen on the chart that the time taken always rises because the more number of pixels chosen, the longer the time taken. More pixels mean more frequency process to calculate and order during Fibonacci code conversion.

Figure 5. Inter-Compress Chart - Experiment with variations in the size of the image resolution.

On the chart above it can be seen that in the compression comparison of each process and types of compresses, the compression ratio increases and the space string increases significantly. Compression ratio increases because the comparison between data sizes experiences a greater difference between before and after compression.
4. Conclusion
The conclusions that can be drawn based on testing the data compression system and message security using the Linear Congruential Generator (LCG), Most Significant Bit (MSB) and Fibonacci Code methods are as follows:

- The designed application is able to insert and compress images using the Linear Congruential Generator (LCG) algorithm, Most Significant Bit (MSB) and Fibonacci Code and decompress the compression results.
- The Fibonacci Code Algorithm has the disadvantage that if the input image does not have a predominant color value, the resulting compress can become larger than the original file.
- Compression ratio in the text variation experiment with text lengths of 100, 200, and 300 characters (1.85) decreases compared to text lengths of 400 and 500 characters (1.84), and in the experiment of variations in image resolution of 100x100 and 200x200 pixels (2.07) increases compared to the resolution of images 500x500 pixels (2.31).
- On file sizes, all experiments experience increases, with a slight increase in the text variation experiment and rapid increase in the image variation experiment.

References
[1] Andrian, Yudhi., fably, Sakkad. Analisis Ketahanan Citra Stego Metode LSB, LSB+1, LSB+2, MSB Terhadap Perubahan Kecerahan Citra. STMIK potensi Utama.
[2] Andrian, Yudhi. Perbandingan metode LSB, LSB+1, dan MSB pada steganografi citra digital. STMIK potensi Utama. Medan.
[3] Budiman, M. A. & Dian Rachmawati. (2017). On Using Goldbach Go Codes and Even-Rodeh Codes for Text Compression. Jurnal. IOP Conf. Series: Materials Science and Engineering, No. 012062.
[4] David, S. Dan Giovanni M. 2010. Handbook of Data Compression. Fifth Edition, Computer Science Departement California State University Northridge Spinger-verlag London. Spinger.
[5] Gabriel, Stephen, Waweru. 2012.An enhanced Least Significant Bit Steganographic Method for Information Hiding, Journal of Information Engineering and Applications, Vol 2, No.9.
[6] Gleave, Adam, Cristian Stein Steinruecken. 2017. Making Compression Algorithms For Unicode Text. University of Cambridge
[7] Gulo, Aktualitas. 2018. Implementasi Algoritma Kriptografi Rabin dan Fibonacci Codes dalam Pengamanan dan Kompresi File Citra. Skripsi. Universitas Sumatera Utara. Medan
[8] Nurhyati, D. Oky. (2010). Kompresi Data. Program Studi S1 Sistem Komputer Universitas Diponogoro.
[9] Rao, K. R. Dan Yip. P. C. 2000. The Transform And Data Compression Handbook. IEEE.
[10] Rinanda, Kevin. 2018. Implementasi Metode Modified Least Significant Bit Dengan Algoritma Linear Congruential Generator Pada Pengamanan Citra. Skripsi. Universitas Sumatera Utara. Medan
[11] Rahmat R F, Andreas T S M, Fahmi F, Pasha M F, Alzahrani M Y and Budiarto R. 2019. Analysis of dicom image compression alternative using huffman coding Journal of Healthcare Engineering 2019, 5810540.
[12] Siahaan, Antoni. 2011. Pengenalan Karakter Dan Manajemen Database Pada Formulir Isian Menggunakan Jaringan Syaraf Tiruan Dengan Metode Learning Vector Quantization (LVQ). Skripsi Universitas Sumatera Utara. Medan
[13] Siregar, Sari Ramadani. 2016. Perbandingan Algoritma Fibonacci Code dan Inverted Elias Gamma Code Dalam Kompresi File Teks. Program Studi Ilmu Komputer Fakultas Ilmu Komputer Dan Teknologi Informasi Universitas Sumatera Utara.
[14] Subada, M. Ali. 2018 Analisis Perbandingan Algoritma Even-Rodeh Code dan Algoritma Fibonacci Code untuk Kompresi File Teks. Skripsi. Universitas Sumatera Utara. Medan.
[15] Sutoyo dkk, (2009), Teori Pengolahan Citra Digital, Andi Publisher, Yogyakarta.
[17] Vijayakumar, Soniya. 2011. *Image Steganography Based On Polynomial Functions*, Journal of Global Research in Computer Science, Volume 2, No. 3, March.

[18] Wahyuni, Meri Sri. 2017 Implementasi Steganografi dalam menyembunyikan pesan teks dengan metode MSB (Most Significant Bit). Institut Teknologi Medan (ITM). Medan.

[19] Wijaya, Ermadi Satria., Prayudi, Yudi. 2004. Media Informatika, Vol. 2. Yogyakarta.