Chaos-based Modified "EzStego" Algorithm for Improving Security of Message Hiding in GIF Image

Rinaldi Munir
Informatics Research Group, School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
E-mail: rinaldi@informatika.org

***Abstract*—EzStego is a steganography algorithm to embed the secret message in the GIF images. The message is embedded into indices of sorted color palette of the images. EzStego is a sequential embedding type of stego-algorithm. There is no key required for embedding, so anyone that know the algorithm can extract the message. For improving security, a modified EzStego algorithm is proposed. Bits of the message are embedded randomly in the image. Locations of embedding is generated from a random permutation which need an initial value as stego key. Before embedding, the message is encrypted with random bit which is generated by a chaos map. Based on experiments, the modified EzStego is more secure than the original EzStego, because anyone who has no information on the key can not extract the message from the stego-images.**

Keywords—EzStego, GIF images, chaos, random, secure.

I. INTRODUCTION

Besides of cryptography, information security can be done using steganography. Steganography means hidden writing. Steganography is the art and science of hiding message in the communication by embedding the secret message into a cover media (usually digital data such as image, video, or audio). Goal of steganography is hiding existence of message in the cover. By using steganography, transmission of secret information can be done securely, so that the presence of the information can't be known from the third party.

Images are common cover in message hiding. The message can be embedded into the image in spatial domain or transform domain. In the spatial domain, the message is embedded into pixel values, meanwhile in the transform domain the message is embedded into coefficient values such as discrete cosine transform (DCT) coeficients.

One of popular steganography technique in spatial domain is the least significant bit (LSB) embedding. In this technique, bits of the message are embedded into LSB of pixel values. Majority of based-LSB algorithms use images in bitmap (BMP) format. In the bitmap format, pixel values represent graylevel of the pixel. The message is embedded directly by replace LSB of pixel values with bits of the message.

Besides of bitmap images, there are another popular image formats such as GIF images. GIF (Graphics Interchange Format) image, a kind of indexed image, was introduced as an image format by Compuserve in 1987. An indexed image uses a palette of up to 256 colors from the 24-bit RGB color space with values in the range [0,1]. The pixel values represent index to a palette row. Color of the pixel is combination of each channel red (R), green (G), and blue (B) in the palette row.

One of the most popular steganography algorithm for GIF images has been proposed by Machado [1]. Her algorithm is called *EzStego*. In order to minimize color degradation, the palette is sorted so that the difference between two adjacent color is minimized. EzStego embeds message into the LSB of indices (pixel values) pointing to the sorted palette. Besides of EzStego, there is another steganographic algorithm for GIF images, i.e. *S-Tool*. S-Tool was developed by Andy Brown. S-Tool encrypt the message before embedding with various encryption algorithm such as DES and IDEA [2].

The disadvantage of EzStego is there is no key required in embedding process, so anyone who know that the stego-image is made using *EzStego* can extract the message. In this paper, we present a modified EzStego for improve security. In the modified EzStego, the pixels for message embedding are chosen randomly using a random permutation that seeded with a secret key. To make the embedding more secure, the secret message is encrypted before it is inserted in the image. The secret message is encrypted by XOR-ing it with random bits that is generated from a chaos system. Thus, there are two keys needed, one for a seed for the random permutation, and another key for encryption. The chaos system is chosen because it is sensitive to very little change of initial values. This characteristics is important on security, because it makes the exhaustive key search becomes more difficult.

This paper is organized into five sections. The first section is introduction. The second section will explain some study of literatures such as GIF images, chaos system that called logistic map, and an original EzStego algorithm. In the third section, we propose a modified EzStego algorithm that improve security of original EzStego. The fourth section describe the experiments and discuss the results. Finally, in last section we give conclusion and suggest future works.

II. LITERATURE STUDY

A. Review of GIF Images

The GIF images is a kind of indexed images. A GIF image consists of a pixel matrix and a color palette. There is a direct mapping from pixel values to the color in the palette. The pixel values represent indices to the palette. The color of the pixel is

determined by mixing of each component RGB in the palette. Fig. 1 shows the structure of an GIF image. In the figure, the pixel value 5 represents the fifth row of the palette. In the row, R = 0.2902, G = 0.0627, B = 0.0627. Thus, the color perception of the pixel 5 is combination of the color component.

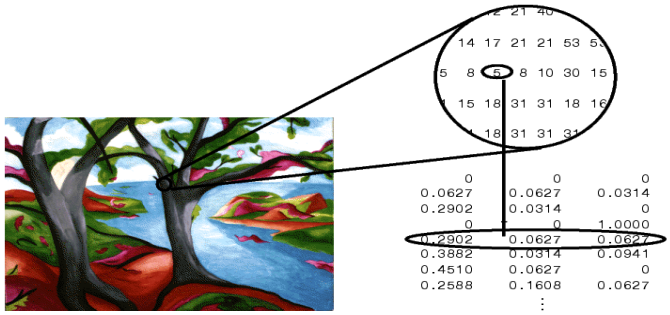


Fig. 1 GIF image structure (Source: Matlab)

The palette size is up to 256 entries (rows). The limited number of colors in the palette makes GIF format efficient only for images with low dept color such as cartoon or animation image.

B. Logistic Map

Study of the chaos systems for increasing information security is very interesting in recent years [3]. The Chaos systems is interesting because they have a characteristics of sensitivity to initial values. It means that the bit changes to the initial values will produce the chaos values that differ significantly. This characteristics is required to information security.

A popular and the simplest chaos system is a Logistic Map, described by a iteration equation,

$$x_{k+1} = \mu x_k (1 - x_k) \quad (1)$$

The initial values is μ, where $0 < \mu \leq 4$, and x_0 for starting the iteration. The map is in chaotic state when $3.57 < \mu \leq 4$ [4], and in this chaotic the behavior of systems appears to be random.

If a logistic map is used as as a pseudo-random generator, then the initial values, x_0, and constant μ, behave as the secret keys. For example, by changing x_0 slightly becomes $x_0 + \Delta$, the chaos values generated, after iterated several times are significantly different from the previous chaos values with initial value x_0.

C. EzStego Algorithm

EzStego can be referred as a name of steganographic tool or as a name of algorithm. EzStego is a sequential embedding type of stego system. Bits of the message are embedded sequentially in the LSBs of the pixels values. However, if we replace LSB of the pixel value with a message bit, the pixel value maybe increase or decrease. Because of the pixel value is a pointer to the palette, this new pixel value will point to a previous or next entry in the palette. The color difference between two adjacent entries in the palette maybe significant, so that the color of the pixels before and after embedding maybe different significantly. This makes distortion in the stego-image.

In order to minimize the distortion, the palette is first sorted by intensity values so that the difference between two adjacent colors is minimized. In the sorted palette, the colors are near to each other. During the embedding process, the message bits are embedded to LSBs of color indices to the sorted palette. It replace color by its neighboring color in the sorted palette, if necessary.

Fig. 2 illustrates the embedding process in EzStego. Suppoe there are eight different colors in the palette which the pixel values (pointer to the palette) are 0, 1, 2, …, 7. First, the palette is sorted so that the two adjacent colors is minimized. Next, we assign the new indexs to the palette (000, 001, …, 111). Suppose we embed a message bit '1'to pixel 7 whose the palette index is '100'. We replace the LSB of the '100' by '1' (100 → 101) which it points to pixel 3. Thus, the color of pixel 7 is replaced by color of pixel 3. This technique is applied to all pixels sequentially until the message is exhaustive. By this technique, we get a stego-image with minimal distortion.

The extraction process is very easy. Before extracting, first we sort the palette with a same way in embedding process. Next, the message bits are extracted from the LSB of the sorted indexes.

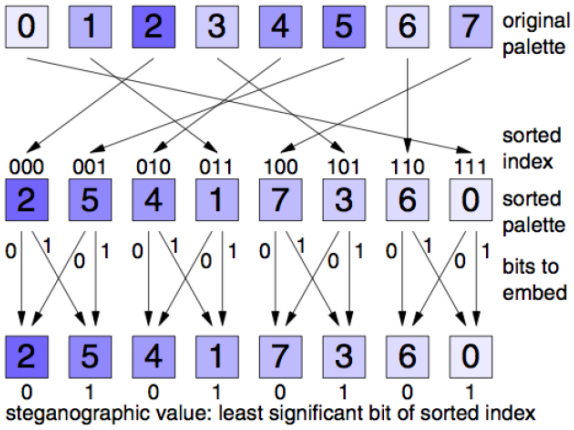


Fig 2. The embedding process of EzStego [5]

III. THE PROPOSED ALGORITHM

We can see in the original EzStego that no key(s) required for embedding the message. Thus, anyone who get a stego-image that maked by using EzStego, he or she can extract the message from the stego-image.

So, in this section we propose a modified EzStego where the message bits are embedded in random order of pixels..A random permutation for the random positions of embedding is generated. For increasing security, before embedding, we encrypt the message with the random bits that generated by a logistic map. The algorithm of embedding and extraction is described follows.

A. *Embedding Algorithm*

We can resume the steps of embedding message in the modified EzStego as follow:

1. Sort the palette of the original image by distance between color of the pixels. The distance between the color (R_1, G_1, B_1) dan (R_2, G_2, B_2) is calculated by Euclidean distance:

$$d = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \qquad (2)$$

   Starting from the first entry of the palette, we calculate the distance between the first entry and the next entries and choose the closest distance. Repeat for the second entry, the third entry, etc. Next, sort the palette by the distances in the ascending order.

2. Assign the new index of the sorted palette by numbering 0, 1, 2, … etc.

3. Encrypt the message bits by XOR-ing them with the random bits that generated by a Logistic Map with initial values x_0 and constant μ.

   Because of a Logistic Map yields real numbers between 0 and 1, we convert them to integers and extract the LSB of the integers as the random bits. Conversion x_i to integer is obtained as follows: x_i multiplied by 10 repeatedly until it reach a desired long number (size), and then truncate to take the integer part. Mathematically, this process is described by function T as follows [6]:

$$T(x, size) = \left\| x * 10^{count} \right\|, x \neq 0 \qquad (3)$$

   where *count* is begined from 1 until $x * 10^{count} > 10^{size-1}$ and symbol || || represents truncation. The least significant bit of binary representation of the integer is then extracted to get the random bits [7].

   Suppose the set of message bits is M dan the set of random bits is K, we encrypt M by XOR-ing M with K to get the encrypted message, C:

$$C = M \oplus K \qquad (4)$$

   Embedding bits of C is performed by next steps.

4. Generate a random permutation with initial key y that represent the random position of embedding.
5. Based on the random position, replace the LSB of indexs of the sorted palllete by bits of the encrypted message, C. Finally we get a stego-image.

B. *Extraction Algorithm*

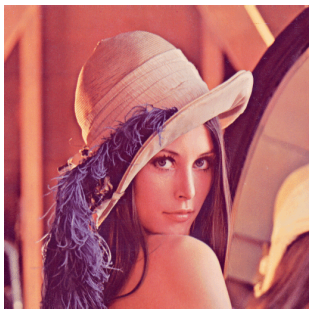
1. Sort the palette of the stego-image by distance between color of the pixels.
2. Assign the new index of the sorted palette by numbering 0, 1, 2, … etc.
3. Generate a random permutation with initial key y that represent the random position of embedding.
4. Extract the LSB of the index of the sorted palette. We will get the encrypted message, C
5. Generate the random bits K by iterating the Logistic Map with initial values x_0 and constant μ.
6. Decrypt the encrypted message by XOR-ing C with K to yield the original message, M:

$$M = C \oplus K \qquad (5)$$

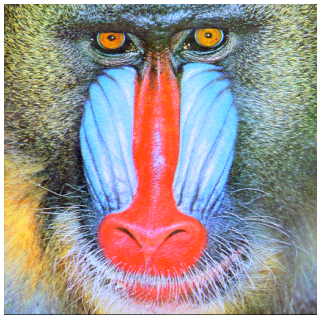
IV. Experiment Results

We have performed some experiments to measure the performance of the proposed algorithm. We use some test images from natural images until cartoon image. The test images are ‘roman’ (a grayscale image), ‘lenna’ and ‘mandrill’ (the color images), and ‘disney’ (a cartoon image). All of them are images in GIF format (Fig. 3).

(a) Roman, 512 × 512



(b) Lenna, 512 × 512



(c) Mandrill, 512 × 512

(d) Disney, 397 × 280

Fig 3. The cover images

We embed some text file from vary size so that maximaze a payload of the original image. Because of we only can embed one bit in one index of the sorted palette, so the payload of the GIF image is limited. If the payload is less than size of the message, then only a part of the message that enbedded into the original image. Suppose the original GIF image has the size of $M \times N$ pixel. The payload is MN bits or $MN/8$ bytes. The ‘roman’ image has the size of 512 × 512, thus the payload is 512×512 = 262144 bits or 32768 bytes. The same payload is for ‘lenna’ and ‘mandrill’ image with the same size with ‘roman’ image. Finally, the last image is ‘disney’ whose size is 397 × 280.

Quality of the stego-image is by calculating PSNR. PSNR is calculated by

$$PSNR = 20 \times \log_{10}\left(\frac{255}{rms}\right) \quad (6)$$

where *rms* is abbreviation of *root mean square* of two images, I and $\hat{I}$, of size $M \times N$ pixels, that has a formula:

$$rms = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(I_{ij} - \hat{I}_{ij})^2}$$

(a) Roman, PSNR = 94.5974

(b) Lenna,PSNR = 77.8391

(c) Mandrill, PSNR = 69.5183

(d) Disney, PSNR = 73.06400

Fig 4. The stego-images

TABLE I. RESULTS OF EXPERIMENT

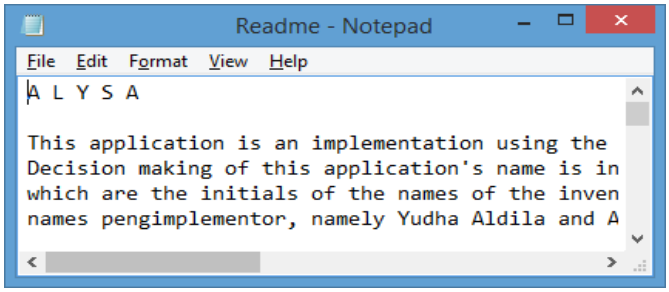
No.	Cover image	Text File	Size of File	PSNR (dB)
1.	Roman.gif	Readme.txt	3.25 Kb	+104.4527
2	Roman.gif	Snow.txt	17,3 Kb	+97.1699
3	Roman.gif	License.txt	75.Kb	+94.4974
1	Lenna.gif	Readme.txt	3.25 Kb	+87.7391
2	Lenna.gif	Snow.txt	17,3 Kb	+80.5117
3	Lenna.gif	License.txt	75.Kb	+77.8391
1	Mandrill.gif	Readme.txt	3.25 Kb	+79.4378
2	Mandrill.gif	Snow.txt	17,3 Kb	+72.1956
3	Mandrill.gif	License.txt	75.Kb	+69.5183
1	Disney.gif	Readme.txt	3.25 Kb	+79.2028
2	Disney.gif	Snow.txt	17,3 Kb	+73.0012
3	Disney.gif	License.txt	75.Kb	+73.0640

The higher PSNR represent a fine quality after embedding, the lower PSNR represent a big degradation after embedding. For the convenience, the quality is still can be tolerance if PSNR > 30.

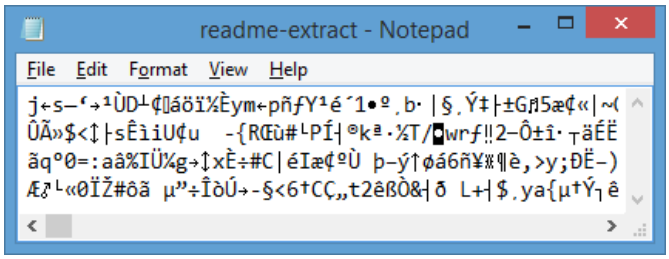
The initial values for the Logistic Map is x_0 = 0.675, μ = 3.9762, and a seed for the random permutation is y = 0.81. All of them behave the secret keys in the stego-system.

The results of experiments is resumed in Tabel I, while the stego images for embedding *License.txt* are shown in Fig. 4. The hidden messages could be extracted back from the stego images exactly. The original messages were same exactly with the extacted messages, both size and content.

The last experiment was sensitivity testing. We know that one of characteristics of chaos is the sensitivite to a little bit changes in initial values. In the embedding algorithm above, Logistic Map is used to generate random bits with initial values x_0 and constant μ and then encrypt the message with the random bits. In this experiment, we changed x_0 = 0.675 to x_0 = 0.67499999. The cover image was 'roman' image and the message was Readme.txt. As a result, the extracted message was different significantly if compared with the original message. Fig. 5 shows the original message (a) and the extracted message when x_0 was changed from 0.675 to 0.67499999.



(a) The original message



(b) The extracted message

Fig 5. Result of the sensitivity testing.

V. DISCUSSION

We have performed some experiments and the results have been shown in the above section. We find that the stego-images are similar with the cover images, For maximum payload, we get that PSNR of every stego-images are very high (all are > 30

dB). The results represent that the embedding of messages didn't affect quality of the images significantly. We observed that the greater the size of the message, getting down the PSNRs. The grayscale image, in this experiment 'roman' image, has the highest PSNR among the others. This result can be explained as follows. In the grayscale image, the value of R, G, and B are identic. The distance of two consecutive entry in the palette is always same, i.e the distance is equal to one. It implies no distortion arise from the sorting of palette.

The sensitivity testing of chaos shows this charcteristics provide a strong security from exhaustive attack. The very small change of the initial values caused the extraction process produced wrong message. Therefore, the modified EzStego algorithm is more secure than the original EzStego.

Next, we calculate space key. The space key states of different number of keys that can be used to do embedding and extraction [8]. In order to make brute-force attack not effective, then the key space should be made large enough. The secret keys used in the proposed algorithm is x_0, μ, and y. According to standard 64-bit IEEE floating-point, computation precision of the floating point is 10^{-15} [8], so number of possible values of x_0 is 10^{15} as well as μ and y. Thus, key space is $10^{15} \times 10^{15} \times 10^{15} = 10^{45}$. This key space is large enough so that the algorithm can be resistant to exhaustive attack.

VI. CONCLUSION

In this paper we have presented a modified EzStego algorithm based on chaos. Quality of the stego-images are very fine and no degradation significantly. The algorithm is secure because the key space is enough large.

The future research is how the proposed algorithm can be implemented for animated GIF. The animated GIF have a special format that differ from the stil GIF image.

REFERENCES

[1] R. Machado, EZStego, http://www.stego.com

[2] N.F. Johnson and S. Jajodia (1998), 'Exploring Steganography: Seeing the Unseen, George Mason University', IEEE Computers.

[3] Dawei, Z., Guanrong, C., Wenbo, L., A Chaos-Based Robust Wavelet-Dmain Watermarking Algorithm, *Chaos Solitons and Fractals 22* (2004) 47-54.

[4] Bose, R., Banerjee, A., Implementing Symmetric Cryptography Using Chaos Function, Indian Institute of Technology.

[5] A. Westfeld and A. Pfitzmann (1999). "Attack on Steganographic System", Lecture Notes in Computer Sciences, vol. 1768, pp. 61-76.

[6] Lampton, J., Chaos Cryptography: Protecting Data Using Chaos, Mississippi School for Mathematics and Science.

[7] Munir, R., A Chaos-based Fragile Watermarking Method in Spatial Domain for Image Authentication, Proceeding of ISITIA 2015.

[8] C. Fu, J. Chen, H. Zou, W. Meng, Y. Zhan, Y. Yu. 2012. A Chaos-based Digital Image Encryption Scheme with an improved Diffusion Strategy. Journal Optic Express 2363, Vol. 20. No. 3