Application of The Modified *EzStego* Algorithm for Hiding Secret Messages in The Animated GIF Images

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

***Abstract*—Animated GIF images consist of a number of frames that displayed in succession as a video. Capacity of hiding messages (or payload) in an animated GIF file is much greater than in a single GIF image. A modified EzStego steganography algorithm has been proposed for GIF images. The modified EzStego is a improved version of the original EzStego so that bits of the message can be embedded randomly in the image. The modified EzStego is based on chaos theory. In this paper we applied the modified EzStego algorithm for hiding the secret messages in the animated GIF images. Bits of the message are embedded randomly in each frame. Frames also can be selected randomly for increasing security. In order to more secure, the message is encrypted before embedding. The bits of the message is encrypted with random bit which is generated by a chaos map. Based on experiments, the modified EzStego algorithm can be applied to the animated GIF images with the minimal visual distortions .**

Keywords—Modified EzStego, animated GIF images, frame, random, chaos.

I. INTRODUCTION

Steganography is one of the techniques in information security. Aim of steganography is for hiding message in the communication by embedding the secret message in a cover media so that the presence of the message can't be known from the third party. The cover media can be anything, but the cover usually are the images. Hiding the message in the images exploits the weakness of the human visual that can not detect small changes in the images.

A popular method for image steganography is the least significant bit (LSB) embedding. In this method, LSBs of pixel values of the cover image are replaced by bits of the message. Majority of based-LSB algorithms use images in bitmap (BMP) format. The BMP format is rarely used in *World Wide Web* because of their large size (The BMP files are not be compressed). The another popular image formats are JPEG, PNG, GIF, etc.

In this paper we focused on GIF images. GIF (Graphics Interchange Format) was introduced by Compuserve in 1987 and come into widespread usage on the World Wide Web because of its wide support and portablility [1]. GIF image is a kind of the indexed image. It 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. 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.

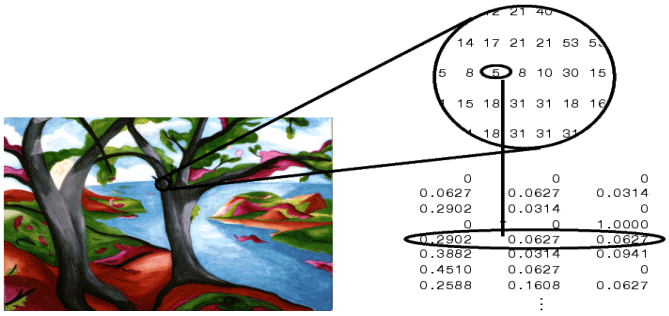


Fig. 1 GIF image structure (Source: Matlab)

GIF format supports animaton and we call it the animated GIF images. The animated GIF images comprises a number of frames. Each frame was displayed in succession as a video. The animated GIF images is usually used for displaying cartoon, funny images, or other interesting images.

A popular steganography algorithm for GIF images is *EzStego*. EzStego algorithm has been proposed by Machado [2]. It embed bits of the message to LSBs of indices pointing to the palette. In order to minimize color degradation due to changes in the indices, at first the palette is sorted so that the difference between two adjacent color is minimized. EzStego embeds message into the LSBs of indices pointing to the sorted palette. Besides of EzStego, there is another steganographic algorithm for GIF images, i.e. *S-Tool* and *Gifshuffle*. *S-Tool* was developed by Andy Brown. S-Tool encrypt the message before embedding with various encryption algorithm such as DES and IDEA [3]. *Gifshuffle* is developed by Matthew Kwan. It works with all GIF images, including those with transparency and animation, by shuffling the palette. In addition it provides compression and encryption of the concealed message [4].

Back to EzStego. Unfortunately, the original EzStego didn't use the key in embedding process, so anyone who know that the stego-image is made using EzStego can extract the

message. The modified version of EzStego has been proposed so that both embedding and extaction process need the same key [5]. In the modified EzStego, the pixels for message embedding are chosen randomly by 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.

When we hide a message in an animated GIF image, we will get more capacity (or payload) to embed, because there are more frames than only a frame in a single GIF image. We needn't develop a new steganography algorithm for the animated GIF images. We can improve the modified EzStego so that it can be used for embedding the message in the animated GIF images.

This paper will present application of the modified EzStego for animated GIF images. The paper is organized into five sections. The first section is introduction. The second section will review some study of literatures such as animated GIF images, chaos system, and the modified EzStego algorithm. In the third section, we explain a scheme of application of the modified EzStego for the animated GIF images. 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. *Animated GIF Images*

The animated GIF images consist of a number of frames, each frame may has independent palette itself. Fig. 2 shows six frames of an animated GIF image (*jogging.gif*).

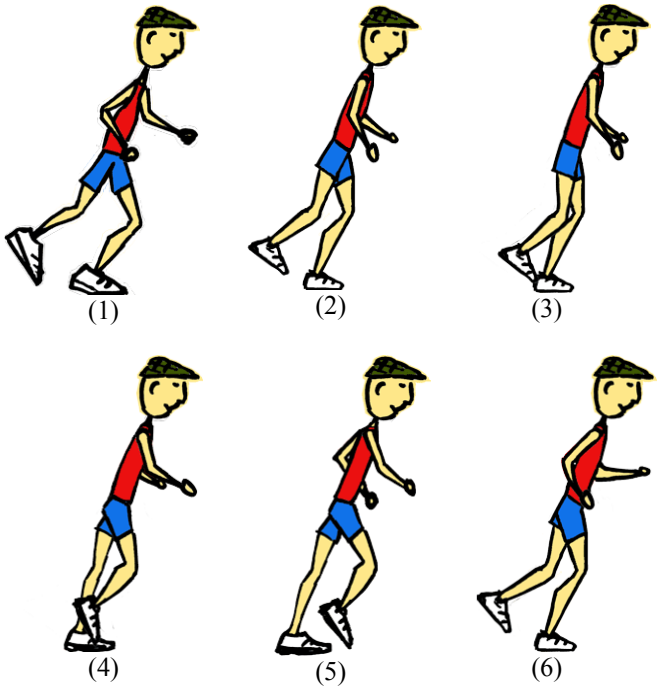


Fig. 2. Six frames of an animated GIF image, jogging.gif

B. *Chaos System*

Chaos system have a characteristics of sensitivity to initial values. It means that a very small changes to the initial values will produce the chaos values that differ significantly. This characteristics is required to information security.

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

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

The initial values is μ, where $0 < \mu \leq 4$, and x_0 for starting the iteration. The map is in chaotic state when $3.57 < \mu \leq 4$, and in this chaotic the behavior of systems appears to be random [7]. Thus, a logistic map can be used as as a pseudo-random generator. The initial values of Logistic Map, x_0 and constant μ, behave as the secret keys. If we change 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. *Modified EzStego Algoritm*

The original EzStego is a sequential embedding type of stego system. It means that bits of the message are embedded sequentially in the LSBs of the pixels values. No key required for embedding and extraction the message [5]. The modified EzStego has been proposed where the message bits are embedded in random order of pixels [6]. A random permutation for the random positions of embedding is generated by a secret key. 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.

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

Embedding Algorithm

1. In order to minimize the distortion, the palette is first sorted by intensity values so that the difference between two adjacent colors is minimized. The palette of the original image is sorted 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:

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

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.
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.

Extraction Algorithm

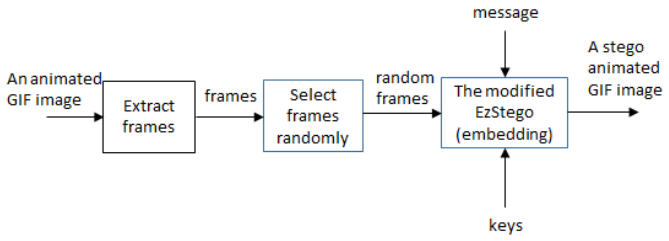
1. Sort the palette of the stego-image by distance between color of the pixels.
2. Assign the new index of the sorted palette by numbering 0, 1, 2, … etc.
3. Generate a random permutation with initial key y that represent the random position of embedding.
4. Extract the LSB of the index of the sorted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

III. The Proposed Scheme

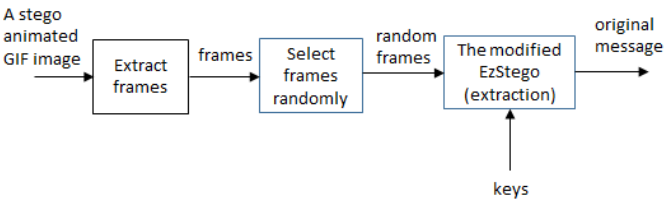
The modified EzStego algorithm can be improved so that it can be used for embedding the message in the animated GIF images. In the proposed sheme, the frame can be selected randomly, and for each selected frame, bits of the message are embedded randomly in the pixels of the frame. The messages will be embedded to the images can be anything: text, image, music, video, etc. Fig. 3 shows the embedding scheme and the extraction scheme.

In the embedding scheme, frames are extracted from an animated GIF image. Next, the frames for embedding are selected randomly. Finally, run the modified EzStego with input are message and the secret keys to result a stego animated GIF image.

In the extraction scheme, the scheme is similar, except the inputs are an animated stego GIF image and the secret keys, and the output is the original message.



(a) The embedding scheme



(b) The extraction scheme

Fig. 3. Embedding and extraction scheme of the modified EzStego for the animated GIF image

IV. Experiment Results and Discussion

Some experiments have been done to ensure the proposed schemes are correct and quality of the stego animated GIF images was measured by PSNR. We use two animated GIF images as the cover images, one is a animated cartoon image (*SnowWhite.gif*), and another one is animated natural image (*CuteKitten.gif*) (see Fig. 4). The *SnowWhite.gif* image consist of 32 frames, each frame has size 225 × 300 pixel. The *CuteKitten.gif* image consist of 9 frames, each frame has size 274 × 500 pixel.

Fig. 4. The cover images. Top: *SnowWhite.gif*; Bottom: *CuteKitten.gif*

The message embedded to the cover images is a text file *story.txt* which size is 35,546 bytes or 284,368 bits (a beginning part of the the text file is shown in Fig.5). Each frame in *SnowWhite.gif* image can be embedded 225 × 300 = 67,500 bits. Thus, to embed 142048 bits of message it needs $\lceil 284{,}368 / 67{,}500 \rceil = 5$ frames. Next, each frame in *CuteKitten.gif* image can be embedded 274 × 500 = 137000 bits. It needs $\lceil 284{,}368 / 137{,}000 \rceil = 3$ frames for embedding all of bits.

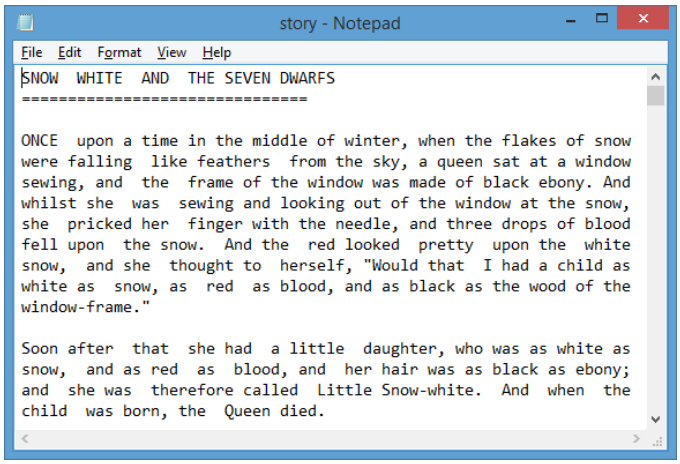


Fig. 5. The beginning part of the message (*story.txt*)

The initial values for the Logistic Map for generating random bits in these experiments is $x_0 = 0.1234$, $\mu = 3.9762$, and a seed for the random permutation for location of embedding is $y = 0.5678$. All of them behave the secret keys in the stego-system.

Quality of the stego-image is measured by PSNR. PSNR (in dB) is calculated by

$$PSNR = 20 \times \log_{10}\left(\frac{255}{rms}\right) \quad (3)$$

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:

$$rms = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(I_{ij} - \hat{I}_{ij})^2} \quad (4)$$

Before calculating PSNR, the GIF images have to be transformed from indexed images to RGB images. The higher PSNR represent a fine quality after embedding, the lower PSNR represent a big degradation after embedding. For the convenience, the quality of an image is still can be tolerance if PSNR > 30.

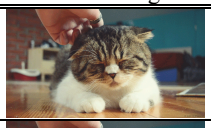
Table. I and Table II respectively show the frames of the stego animated GIF images of *SnowWhite.gif* and *CuteKitte.gif* that was embedded bits of messages. PSNRs of each frame are displayed in right side. We can see that visually the stego frames is very similar with their cover frames. The stego animated GIF images can be displayed in webpage perfectly as fine as the cover images. Nobody knows that the animated GIF images contain the hidden message inside.

In the extraction process, the hidden messages could be extracted back from the stego animate FIF images exactly. The original messages were same exactly with the extacted messages, both size and content. It means the proposed scheme worked very well.

TABLE I. RESULTS OF EXPERIMENT (SNOWWHITE.GIF)

Frame	Cover Image	Stego Image	PSNR
8			79.2926 dB
13			72.7790 dB
16			72.8334 dB
19			72.6843 dB
20			72.4812 dB

TABLE II. RESULTS OF EXPERIMENT (CUTEKITTEN.GIF)

Frame	Cover Image	Stego Image	PSNR
3			74.3899 dB
6			85.2463 dB
9			74.2097 dB

Based on results of the experiments we found that the proposed scheme worked very well. The stego animated GIF images were similar with the cover animated GIF images. We also got that PSNR of every stego-frames were very high (all are > 70 dB). The results represented that the embedding of messages didn't affect quality of the images significantly. The stego animated GIF images can be displayed as well as the original animated GIF images.

Bits of the message can be inserted in random frames of the animated GIF images successfully. The message can be recovered exactly from the stego images.

Compared with results of the previous work [6], the payload (size of data embedded) in an animated GIF image is more larger than the payload in a single GIF image. Therefore, the proposed scheme is better than the previous scheme for embedding more messages in an animated GIF image.

V. CONCLUSION

In this paper we have presented application of the modified EzStego algorithm for hiding message in the animated GIF images. Quality of the stego-images are very fine and no degradation significantly. The message was embedded in random frames and in random pixels in each frame.

REFERENCES

[1] Navin, A. H, Sadighi, A., Fesharaki, M. N., Teshnelab, M., Keshmi, R., Data Oriented Model of Image: as a Framework for Image Processing, World Academy of Science, Engineering and Technology 34, 2007

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

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

[4] The Gifshuffle homepage, http://www.darkside.com.au/gifshuffle/

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

[6] Munir, R., Chaos-based Modified "EzStego" Algorithm for Improving Security of Message Hiding in GIF Image, Proceeding of Proceeding of "2015 International Conference on Computer, Control, Informatics and Its Applications" (IC3INA 2015), LIPI Bandung, 5-7 Oktober 2015.

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