Digital image steganography by using edge adaptive based chaos cryptography

To cite this article: A Delmi et al 2020 J. Phys.: Conf. Ser. 1442 012041

View the article online for updates and enhancements.
Digital image steganography by using edge adaptive based chaos cryptography

A Delmi, S Suryadi and Y Satria
Department of Mathematics, Faculty of Mathematics and Natural Sciences (FMIPA), Universitas Indonesia, Kampus UI Depok, Depok 16424, Indonesia

Corresponding author’s e-mail: yadi.mt@sci.ui.ac.id

Abstract. Steganography and Cryptography is the method used to secure the information. Cryptography is used the principle of a Chaos by using Cat Arnold Map function to assure randomness. While in steganography, the method is used Least Significant Bit Matching Revisited (LSBMR). Embedding region was on edge digital imagery to ensure the message was not detected in the image by visual. The method used to detect the edge region by using Canny edge Detection. The test results obtained by Peak Signal to Ratio (PSNR) are 72.44 dB for data messages with a size of 10 % of the media cover.

Keywords: cryptography, edge, LSBMR, steganography.

1. Introduction
The era of digital technology has bought all of information into digital form. Digitalization of information easier for users to access information in the case and maintains the information longer than non-digital information [1]. However, the digitization also brought new problems, namely the information is very easy to plow or duplicated without permission of the owner. This can lead to such information could be misused by irresponsible. On that basis, security experts vying with the hackers to create a variety of methods to secure data both during storage and when transmitted difficult hijacked.

One method used to secure data is steganography. Steganography is a method to hide a secret message into a larger media in a way that other people do not know the contents or the existence of the secret message [2]. In steganography method, the medium used as the cover is usually in the form of audio files, digital images, and video. But in this study will be focused only on digital image steganography. In the digital image steganography, digital image messages to be hidden so-called secret image and media digital image is used as a place to store secret image called the cover image and the results of the secret image storage to the cover image is called stego image.

This research will focus on embedding a secret message to the Region of Interest (ROI) that is specific to the digital image. ROI can be any object in a digital image to produce minimum distortion when embedding data. ROI is one-area ledges or the so-called edge. Edge area is perfect for hiding data for humans are visually less sensitive to see the distortion at the edge area of the digital image and the area also has a random pixel position. Steganography based ROI can be considered as an alternative steganography algorithm better.

In this paper, will try and analyse the implementation of the method proposed by Roy et al. [3], which uses Canny Edge Detection (CED) to search for Edge on digital image messages and hide a digital image into pixels selected. The use of CED has a high immunity to noise and can detect weak edge. Then to improve security, methods Arnold Cat Map (ACM) is used to prevent the message can be peeped by others. ACM method is used because of the ease in returning a message that has been converted into shape just by doing a few iterations only. Then for the embedding process will use the method of Least Significant Bit Matching Revisited (LSBMR).
Figure 1. Example transition image with ACM function.

2. Proposed method
This proposed mechanism is using steganography and cryptography for more securing information. Cryptography is a method to securing data to a cipher data. Cipher data is a data that cannot be seen what information contains inside [2]. Steganography is a method to securing information by embeds the information to a media, which is bigger than the eavesdropper cannot see what the information inside on media. Before discussing how to securing data, the Chaos function, Canny Edge Detection will be explained first.

2.1. Chaos function
Chaos function is a function that always used in Cryptography method because of its chaotic mapping of the function. Hirsch et al. [4] explain on his book that the definition of Chaos function is $f: X \to X$ chaos in $X$ if

1. $f$ sensitive dependence on initial value.
2. $f$ topologically transitive.
3. Periodic points dense in $X$.

Arnold Cat Map function (ACM) is a Chaos function discovered by Russian mathematician Vladimir I Arnold in 1960. At the time of this function is found, he uses the image of a cat so named as Arnold Cat Map. Initially, this function is defined as in equation (1) for image size $N \times N$ [5].

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \mod N$$

(1)

$[X_n \ Y_n]$ is the location of two pixels of the image are each worth positive integers with $N \times N$ is size of data and mod N is an unary operation mapping from integer N into integer $d \in \{0,1,2,\ldots,N - 1\}$ [6]. Figure 1 is an example how ACM works.

n subsequent developments, ACM can be generalized [3] using two parameter $p$ and $q$ as follows

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq + 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \mod N$$

(2)

where $p$ and $q$ is positive integer.

2.2. Canny edge detection
Canny Edge Detection is an algorithm developed by John Canny to determine the edge on the digital image [7]. The processes of Canny Edge Detection algorithm are shown in figure 2. Figure 2 shows the process of Canny Edge detection that starts with Filter Gaussian, search the gradient, and then do the non-maximum suppression, then sought edge by comparing with Maximum Value and Minimum Value of Canny Edge Detection, then if the edge is located between Maximum Value and Minimum Value, the edge will be traced by the edge tracking using a gradient value that has been searched before.

2.3. Embedding
In this paper we will discuss how to hide secret messages in the form of digital image into a digital image by using an implementation of cryptography and steganography digital image-based adaptive
Chaos. Secret messages and cover digital image digital image represented a collection of pixels. Each pixel contains three RGB colour components i.e. each component is organized into 8-bit. Thus, each pixel is composed as a data 24 bits or 3 bytes. Figure 3 shows the flow of cryptography and steganography algorithms.

In figure 4, input process in the form of a digital image colour as the secret image with a size $M_1 \times N_1$ with $M_1$ is the length of the digital image confidential and $N_1$ is wide digital image confidential and digital image color as the cover image with a size $M_2 \times N_2$ with $M_2 > M_1$ is length of cover digital imagery and $N_2 > N_1$ is wide digital image of the cover. Then will do encryption Arnold Cat Map so as to produce a digital image of the cipher. Next, the cover image is detected by Canny Edge Detection to obtain the edge.

In this paper, the process of embedding and extracting would use the concept proposed by [8]. This concept is called the Least Significant Bit Matching Revisited (LSBMR). Before further discussion
regarding LSBMR, will be explained first concept of Least Significant Bit (LSB) and the functions required. LSB is a bit position that declares an even or odd value. At this writing LSB declared value of the rightmost bit positions. Suppose an integer is represented by the bit string consisting of \((b_0b_1b_2b_3b_4b_5b_6b_7)\) where \(b_i \in \{0,1\}\) and \(i = 0,1,\ldots,7\). Then LSB is LSB\((b_0b_1b_2b_3b_4b_5b_6b_7)\) = \(b_7\). Then the binary function \(f\) is defined as follows,

\[
 f(x_i,x_{i+1}) = \text{LSB}\left(\frac{x_i}{2}\right) + x_{i+1}
\]

(3)

where \(x_i, x_{i+1} \in \mathbb{Z}^+\) and output of this function is bit form and \(i = 0,1,\ldots,k\) where \(k\) is message length which will in embedding. The process of embedding case LSBMR follow the rules as follows:

\begin{align*}
\text{Input} & : m_i, m_{i+1} \\
& x_i, x_{i+1} \\
\text{Output} & : y_i, y_{i+1} \\
\text{Case 1 : If } \text{LSB}(x_i) = m_i \text{ and } f(x_i, x_{i+1}) = m_{i+1} \text{ then} \\
& y_i = x_i \\
& y_{i+1} = x_{i+1} \\
\text{Case 2 : If } \text{LSB}(x_i) = m_i \text{ and } f(x_i, x_{i+1}) \neq m_{i+1} \text{ then} \\
& y_i = x_i \\
& y_{i+1} = x_{i+1} + 1 \\
\text{Case 3 : If } \text{LSB}(x_i) \neq m_i \text{ and } f(x_i - 1, x_{i+1}) = m_{i+1} \text{ then} \\
& y_i = x_i - 1 \\
& y_{i+1} = y_{i+1} \\
\text{Case 4 : If } \text{LSB}(x_i) \neq m_i \text{ and } f(x_i - 1, x_{i+1}) \neq m_{i+1} \text{ then} \\
& y_i = x_i + 1 \\
& y_{i+1} = y_{i+1}
\end{align*}

where \(y_i, y_{i+1}\) are embedded pixel value, \(x_i, x_{i+1}\) is pixel yang which will be embed with secret message \(m_i\) dan \(m_{i+1}\) is secret bit message.

2.4. Extracting

Figure 4 is a mechanism for extraction algorithm. Figure 4 shows the extraction process. The input of this algorithm is Stego Image with \(M_2 \times N_2\). First of all, take edge that has been detected previously and from edge obtained do extraction to obtain the cipher image. Then by using Arnold Cat Map will
be obtained secret original image. Before discussing the design and implementation of steganography and cryptography algorithms based adaptive Chaos more, will first be discussed each on the basics of the theory used as Arnold Cat Map, Canny Edge Detection and embedding and extraction mechanism used.

For the process of extracting of pixels that have been embedding will be done with the following steps

\[
m_i = LSB \left( y_i \right) \\
m_{i+1} = f \left( y_i, y_{i+1} \right)
\]

3. Results and discussion

In section 2, it has been explained about the design of steganography and cryptography algorithms. Further embedding and extracting from the program will be implemented in Python software version 2.7.6 on a computer with the following specifications:

- Processor : Intel(R) Core(TM) i3 31217 CPU (4 CPU) -1.8GHz
- RAM : 6.00 GB
- System : Windows 10 64-bit Operating System

Image for the cover image used in this trial is a colour image that is pesawat.tif, moscow.tif, and merak.tif data size of each is 878 kB, 805 kB, 994 kB with the size of 1920 × 1080 pixels as shown in figure 5.

Selection of the three images is based on different types of images taken. In Pespur.tif image looks sharp enough image and focuses on the centre of the image so that the resulting hard edge. In Moscow.tif image has many areas but not too focused edge. As for the image of Merak.tif have much edge and sharp enough for the secret image picture and imagery used also fotokucing.tif colour image is shown in figure 6. Then the size of the secret image is shown in table 1.
Table 1. List of test data

| Data test | Image name        | Image size (pixel) | Image size (kB) |
|-----------|-------------------|--------------------|-----------------|
| 1         |                 | 32 × 32            | 15.9            |
| 2         |                 | 40 × 40            | 22.6            |
| 3         | Fotokucing.tiff  | 64 × 64            | 35.7            |
| 4         |                 | 80 × 80            | 26.8            |
| 5         |                 | 100 × 100          | 52.1            |
| 6         |                 | 128 × 128          | 72.2            |
| 7         |                 | 150 × 150          | 92.8            |
| 8         |                 | 200 × 200          | 157             |

Table 2. Average time embedding test data

| Data test | Image size (pixel) | Average embedding time (second) |
|-----------|--------------------|---------------------------------|
|           |                    | (a) Pespur .tiff                |
|           |                    | (b) Moscow .tiff                |
|           |                    | (c) Merak .tiff                 |
| 1         | 32 × 32            | 22.6430                        |
| 2         | 40 × 40            | 22.9988                        |
| 3         | 64 × 64            | 28.2072                        |
| 4         | 80 × 80            | 37.1458                        |
| 5         | 100 × 100          | 40.3398                        |
| 6         | 128 × 128          | 53.8660                        |
| 7         | 150 × 150          | 59.7212                        |
| 8         | 200 × 200          | 89.6250                        |

Table 3. Average time extracting data test

| Data test | Image size (pixel) | Average embedding time (second) |
|-----------|--------------------|---------------------------------|
|           |                    | (a) Pespur .tiff                |
|           |                    | (b) Moscow .tiff                |
|           |                    | (c) Merak .tiff                 |
| 1         | 32 × 32            | 1.2202                          |
| 2         | 40 × 40            | 2.0818                          |
| 3         | 64 × 64            | 5.2202                          |
| 4         | 80 × 80            | 9.3614                          |
| 5         | 100 × 100          | 13.5624                         |
| 6         | 128 × 128          | 21.7776                         |
| 7         | 150 × 150          | 28.2266                         |
| 8         | 200 × 200          | 48.2090                         |

3.1. Time behaviour
In the existing image in table 1 do embedding and extracting process. Each process is executed for 5 times and then seen the average time for each process of embedding and extracting. Table 2 shows the results of the average time of the embedding process and table 3 show the results of the average time extracting process.

In table 2 and table 3 shows that the larger the test data used the process of embedding and extracting the longer time it takes. Notice in that the embedding processing time is longer than extracting processing time for each test data. This is because the process of embedding edge detection process is using Canny Edge Detection while in the extraction process of detection using available databases corresponding to the cover image used.

3.2. Distortion analysis
Distortion produced the cover images during the embedding process was measured by using Peak Signal to Noise Ratio (PSNR) is measured using the Mean Square Error (MSE) [1]. The measurement is defined as follows:

\[
MSE = \frac{1}{M \times N} \sum_{i=1}^{N} (x_{ij} - y_{ij})^2
\]
Based on the discussion, the analysis and the results of the test data, it was concluded and the advice is: The larger the test data the greater the time required to carry out the process of embedding and extracting based on table 2 and table 3. The process of extracting more quickly than the embedding process due to the process of extracting not perform edge detection again but take away from the edge image database that already exists. The larger the message data to be inserted, the smaller the value of PSNR obtained. Notice when the message size of 150 × 150 in which the size of the secret of 10% of the cover image, resulting PSNR value is between 76-44 dB which according to [3] PSNR value above 40 dB can be referred to have a good quality.

4. Conclusions

In the test results with PSNR of table 4 can be seen that the larger the message data to be inserted, the smaller the value of PSNR obtained. Notice when the message size of 150 × 150 in which the size of the secret of 10% of the cover image, resulting PSNR value is between 76-44 dB which according to [3] PSNR value above 40 dB can be referred to have a good quality.

\[
\text{PSNR} = 10 \log_{10} \left( \frac{255^2}{\text{MSE}} \right) \text{dB}
\]

(5)

with M and N is the size of the pixel dimensions of the cover image, \(x_{ij}\) and \(y_{ij}\) is the value of each pixel on the cover and stego image. In equation (5), a constant value of 255 indicates maximum value of a colour that can be represented in a pixel with 8-bit colour depth. For 24-bit RGB colour, each colour component has a depth of 8-bit. The larger the PSNR values indicating the greater of stego image fidelity means a more significant low levels of distortion. MSE for RGB image is calculated per the array of colours of the image and the average MSE on each array provides Mean Square Error between cover and stego image. PSNR expected to be greater than 40 dB for high fidelity stego image. If the value is above 40dB it can be said stego image of good quality [3] by the value of PSNR for each image. In table 3 shown results have been obtained.

In the test results with PSNR of table 4 can be seen that the larger the message data to be inserted, the smaller the value of PSNR obtained. Notice when the message size of 150 × 150 in which the size of the secret of 10% of the cover image, resulting PSNR value is between 76-44 dB which according to [3] PSNR value above 40 dB can be referred to have a good quality.

### References

1. Smith A 1999 Why Digitize? (Washington: Council on Library and Information Resources)
2. Stallings W 2006 Cryptography and Network Security: Principles and Practices 4th ed (Upper Saddle River: Pearson Prentice Hall)
3. Roy R, Sarkar A and Changder S 2013 Procedia Technol. 10 138–46
4. Hirsch M W, Smale S and Devaney R L 2004 Differential equations, dynamical systems, and an introduction to chaos Pure and Applied Mathematics Series vol 60, ed P A Smith and S Eilenberg (San Diego: Academic Press)
5. Peterson G 1997 Arnold’s Cat Map (New York: Cornell University) Math 45 – Linear Algebra pp 1–7
6. Rosen K H 2012 Discrete Mathematics and Its Applications 7th ed (New York: McGraw-Hill)
7. Canny J 1986 IEEE Trans. Pattern Anal. Mach. Intell. 8 679–98
8. Mielikainen J 2006 IEEE Signal Process. Lett. 13 285–7