Implementation of the Gauss-Circle Map for encrypting and embedding simultaneously on digital image and digital text

Suryadi MT*, Yudi Satria, and Azzam Hadidulqawi

Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Depok 16424, Indonesia

E-mail*: yadi.mt@sci.ui.ac.id

Abstract. This paper discusses implementation of Gauss-Circle Map (GCM) in cryptography and steganography process simultaneously. Cryptography is used for securing data confidentiality, while steganography is used to protect the existence of data. The objects that used in this thesis are digital text and digital images. This research was conducted by designing algorithms for encryption and embedding simultaneously. Results obtained from the observation shows that GCM had randomness level 100% using NIST test with chosen parameter $x_0^{(1)} = x_0^{(2)} = 0, \alpha^{(1)} = \alpha^{(2)} = 9, \beta^{(1)} = \beta^{(2)} = 0.481, K^{(1)} = K^{(2)} = 1000000$, and $\Omega^{(1)} = \Omega^{(2)} = 0.5$. Algorithm that have been designed have varying degrees of sensitivity according to different parameters, and high key spaces that reaches $2.6244 \times 10^{1269}$. Encrypted image is uniformly distributed since it passes goodness of fit test. Correlation coefficient values of the stego image are at interval $[0.89,1]$ and very close to correlation coefficient values of the cover image. However, Peak Signal to Noise Ratio (PSNR) of encrypted image is above 40 dB. Here, the extracted-decrypted stego image have perfect similarity with the original image.

1. Introduction
Rapid development of technology today cannot be separated from invention of computer. Computer only works on digital data. Delivery of information on digital data has high vulnerability to abuse. Therefore fast, safe, and reliable security techniques are needed for digital data [1]. There are several methods to increase the data security. In general, there are three systems of confidentiality [2]. The first is steganography, a science that deals with hiding confidential data in a cover [3]. The second is privacy system, namely the use of special tools to recover messages. The third is cryptography, converts confidential data into random data so that the data cannot be understood by uninvolved parties. The uses of cryptography and steganography system simultaneously are recognized that it will increase the security of digital data [4]. The implementation of chaos function on cryptography and steganography become an interesting topic lately because its property described in [5]. The most important property is sensitive dependence on initial conditions, will cause a significant difference in the encryption and decryption result, protecting from brute force attacks. Another one is the random-like behavior which causes encryption result to have uniform histogram and high entropy, protecting it from statistical attacks [6].

Suryadi et al had introduced and analyzed the performance of digital image encryption algorithm by using logistic map as the key stream generator and DNA encoding [6], followed by [7] using MS Map...
as the key stream generator. On the other hand, [8] has further introduced an encryption and embedment algorithm simultaneously using edge adaptive based chaos cryptography.

In this paper, will try to implement encryption and embedment algorithm simultaneously with the chaos function proposed by Suryadi et al [9]. Here, Gauss-Circle Map (GCM) used as key stream generator for encrypting and embedding an object. Steganography technique used is Least Significant Bit (LSB) 3-3-2 because it has high perceptual transparency with low complexity [3]. Then, we will further analyze the performance of the encryption and embedment algorithm on digital image and text using National Institute of Standards and Technology (NIST), key sensitivity, key space, histogram, uniformity test, correlation coefficient, mean squared error, peak signal to noise ratio (PSNR), and entropy [3-10].

2. Experimental Methods

Gauss-Circle Map is a recursive function that is a composition of Circle Map [11] and Gauss Map [12]. This function is defined as follows [9]:

\[
    x_{n+1} = e^{-\frac{5}{4} \left( x_n + \frac{K}{2\pi} \sin \left( 2\pi x_n \right) \right)} \mod 1 + \beta, x \mod 1 = x - \lfloor x \rfloor
\]

(1)

where \( \alpha, \beta, K, \) and \( \Omega \) are real numbers with \( \alpha > 0, \) and \( \Omega \) can be limited to \( 0 \leq \Omega < 1. \) The parameters with \( \alpha = 9, \beta = 0.481, K = 1000000, \) and \( \Omega = 0.5 \) is one of the best maps for its chaotic behavior based on Lyapunov Exponent and Diagram [9].

For encrypting and decrypting algorithm, the GCM sequence generated will be transformed into pixel values in the equation (2). The special function shown in equation (3) will be used to find the location for embedding the binary data.

\[
    f(x_i) = \lfloor 10^6 (x_i) \rfloor \mod 256
\]

\[
    g(x_i) = \lfloor 10^{12} (x_i) \rfloor \mod (p)
\]

(2) (3)

\( x_i \) is generated by Gauss-Circle Map with \( i = 1,2, \ldots, n \) and \( p \) is three times total pixel of the cover image. For encrypting and decrypting process, we use \( n = wh \) where \( w \) is the width size and \( h \) is the height size of the original image. \( f(x_i) \) and \( g(x_i) \) is discretization function from the sequence \((x_1, x_2, \ldots, x_n) \in R \) into new sequence with integer numbers.

For embedding and extracting process, to decide the location where it should be embedded or extracted, a vector with unique elements was necessary. After we generate a number with GCM, convert the result into integer number with equation (3), then append it to a vector \( V \) if there is no same value on vector \( V \). Do the iteration until we have vector length \( V = 3wh \).

Regarding LSB 3-3-2, we first describe and image as pixel values in matrix form based on RGB color. The pixel values can be represented as equation (4)

\[
    f(x, y) = (R(x, y), G(x, y), B(x, y))
\]

(4)

where \( R(x, y) \) represent intensity of red colour at location \((x, y)\), \( G(x, y) \) and \( B(x, y) \) represent green colour and blue colour, respectively. Before we apply the embedding process, we must change pixel values to binary number.

The embedding process with LSB 3-3-2 take byte data from pixel values, then split it to three parts respectively contains 3 bit-3 bit-2 bit. Based on location calculated (example \((a, b)\)), replace three last bit of \( R(a, b) \) with the first part, three last bit of \( G(a, b) \) with the second part, and two last bit of \( B(a, b) \) with the third part.

Securing digital image, the encrypting-embedding process shown in figure 1, and the extracting-decrypting process shown on figure 2.

For encrypting process, change plain image into matrix form based on (4). Then, generate key stream with equation (2) with the same size as plain image matrix size. Next, do XOR operation with plain image matrix form and key matrix based on corresponding position. The XOR operation result is the frequency of encrypted image in matrix form.
Before proceeding to embedding process, get a key stream to decide the location which each pixel of encrypted image will be embedded. Vector of the key stream can be generated using equation (3).

Then, apply LSB 3-3-2 method for embedding encrypted image into cover image to get a stego image.

Extracting process from a stego image start with generate vector $V$, based on prediction of original image size. If original size prediction is $w \times h$, then generate chaos sequence with length = $(w \times h)$. After that, apply equation (2) to get a sequence with integer numbers to get the vector for deciding location. Now, apply invers LSB to extract $w \times h$ pixels from the stego image. The result is frequency of encrypted image in matrix form. For decrypting process, it is similar with encrypting process because the characteristics of XOR operation.

We can do encrypting-embedding and extracting-decrypting process for digital text based on figure 1 and figure 2, respectively. The difference is the original text must be converted into numbers with American Standard Codes for Information Interchange (ASCII) so we will have frequencies of original image in vector form.

3. Result and Discussion
In section 2, it has been explained about the algorithm for encrypting-embedding and extracting-decrypting. Furthermore, the algorithm will be implemented in Python software version 3.7 on a computer with following specification:

- Processor: AMD A4-9125 RADEON R3
- Memory: 4096 RAM
- OS: Windows 10 Home Single Language 64-bit

3.1. Testing data and encryption result
We use two images with different size and two different text as our testing data. For the cover image, we use three images with different size. The plain and cover image shown in figure 3 and 4 respectively, while the data testing size shown on Table 1, 2, and 3. Selection of the data trials is random original image without being resized. Total pixel of the cover image chosen must be more than three times of total pixel of the plain image chosen.
The keys used through all experiment are $x_0^{(1)} = x_0^{(2)} = 0$, $\alpha^{(1)} = \alpha^{(2)} = 9$, $\beta^{(1)} = \beta^{(2)} = 0.481$, $K^{(1)} = K^{(2)} = 1000000$, and $\Omega^{(1)} = \Omega^{(2)} = 0.5$. Encryption of original image data test result shown on figure 5. Selection of the parameters is based on [9] where it is proven that this parameters have the best exponential with lyapunov exponent comparison.
For the result of the encrypting-embedding process, the result was always the same as the cover image used by visual. We named the stego image file as the number of original image and cover image used. For example, ‘stego 11.png’ means the result of encrypting-embedding using ‘binary.png’ as original image and ‘tower.png’ as cover image.

3.2. Algorithm performance

3.2.1. Randomness Key Stream Analysis

The procedure of the test is described in [13] with encrypting parameters that we used, and generated by 200,000 terms of key stream with equation (1). The test result shown on Table 4.

![Figure 5](image)

**Figure 5.** Display of encrypted image (a) binary.png (b) castle.png

Here, we can see that the Gauss-Circle Map have passed all NIST test. So, we can conclude that Gauss-Circle Map generate random sequence.

Entropy of a chaotic sequence can be measured with `skimage.measure.shannon_entropy`. Now we calculate the entropy using the same sequence generated for NIST test, we got $\text{Entropy} = 7.99907817634102$. It shows that its entropy values range from 7.999-8 bits, close to 8 bits. Thus, it can be implied that the encryption algorithm has a good resistance against entropy attack [6].
3.2.2. Key sensitivity
We will extract ‘stego 21’ with \( x_0^{(1)} \) then decrypt it with \( x_0^{(2)} \). In encryption process we used \( x_0^{(1)} = 0 \), and \( x_0^{(2)} = 0 \). In this part we will extract-decrypt image with three different \( x_0^{(1)} \) and \( x_0^{(2)} \) as shown on Table 5 and Table 6.

### Table 5. Key sensitivity test result for encrypting parameter

| \( x_0^{(1)} \) | Image Result | Histogram |
|----------------|--------------|-----------|
| \( 10^{-17} \) | ![Image Result](image1) | ![Histogram](histogram1) |
| \( 10^{-22} \) | ![Image Result](image2) | ![Histogram](histogram2) |
| \( 10^{-23} \) | ![Image Result](image3) | ![Histogram](histogram3) |

### Table 6. Key sensitivity test result for embedding parameter

| \( x_0^{(2)} \) | Image Result | Histogram |
|----------------|--------------|-----------|
| \( 10^{-12} \) | ![Image Result](image4) | ![Histogram](histogram4) |
| \( 10^{-15} \) | ![Image Result](image5) | ![Histogram](histogram5) |
| \( 10^{-16} \) | ![Image Result](image6) | ![Histogram](histogram6) |

From Table 5 we seen the original image will be produced when \( x_0^{(1)} = 10^{-23} \), means that key sensitivity reaches \( 10^{-23} \) for \( x_0^{(1)} \) and from Table 6 the key sensitivity reaches \( 10^{-16} \) for \( x_0^{(2)} \)

3.2.3. Key space
In the encrypting-embedding algorithm with GCM, there are four parameters for encrypting and four parameters for embedding, a total eight. \( \alpha, \beta, K, \) and \( \Omega \) are real numbers with \( \alpha > 0 \), and \( x_0, \Omega \in [0,1) \). In Python, the maximum value of float is \( 1.7976931348623157e+308 \approx 1.8 \times 10^{308} \). So, based on key sensitivity the key space of our algorithm is,
\[
(1.8 \times 10^{308})^4(0.9 \times 10^{308})^2(10)^{22}(10)^{15} = 2.6244 \times 10^{1269}
\]

3.2.4. Histogram analysis and uniform test
We will test the distribution of encrypted image pixel values using goodness of fit test [6]. The test result shown in Table 7 and Table 8.
Table 7. Histogram analysis

| Data | R Channel (plain) | G Channel (plain) | B Channel (plain) | R Channel (encrypted) | G Channel (encrypted) | B Channel (encrypted) |
|------|------------------|------------------|------------------|----------------------|----------------------|----------------------|
| 1    | ![Histogram](image1) | ![Histogram](image2) | ![Histogram](image3) | ![Histogram](image4) | ![Histogram](image5) | ![Histogram](image6) |
| 2    | ![Histogram](image7) | ![Histogram](image8) | ![Histogram](image9) | ![Histogram](image10) | ![Histogram](image11) | ![Histogram](image12) |

To prove the uniformity of the encrypted image data histogram, *Goodness of Fit Test* is performed.

Table 8. Goodness of fit test

| Data | R Channel | G Channel | B Channel |
|------|-----------|-----------|-----------|
| 1    | 262.24355555555570 | 253.69600000000014 | 279.21066666666690 |
| 2    | 256.08000000000004 | 263.136000000000010 | 240.46400000000006 |

With degree of freedom is $256 - 1 = 255$ and 1% level significance, the critical value of Goodness of fit test is 310.4. It can be seen from Table 8 that all data sample are smaller than 310, so we can conclude that the distribution of encrypted image pixel values are uniformly distributed.

3.2.5. *Correlation coefficient*

This test is performed based on [6] for the original image and the *stego* image. As a note, (C) and (S) is the cover and *stego* image respectively.

Table 9. Correlation coefficient of ‘binary.png’

| Data | Channel | Horizontal (C/S) | Vertical (C/S) | Diagonal (C/S) |
|------|---------|------------------|----------------|----------------|
| Stego 11 | R | 0.95685 / 0.95617 | 0.93921 / 0.93855 | 0.91344 / 0.91282 |
|       | G | 0.94732 / 0.94647 | 0.92459 / 0.92372 | 0.89350 / 0.89267 |
|       | B | 0.97036 / 0.97024 | 0.95680 / 0.95668 | 0.93937 / 0.93926 |
| Stego 12 | R | 0.97977 / 0.97804 | 0.98931 / 0.98755 | 0.96969 / 0.96798 |
|       | G | 0.96906 / 0.96755 | 0.98249 / 0.98093 | 0.95554 / 0.95405 |
|       | B | 0.93274 / 0.93194 | 0.95760 / 0.95677 | 0.91828 / 0.91747 |

Table 10. Correlation coefficient of ‘castle.png’

| Data | Channel | Horizontal (C/S) | Vertical (C/S) | Diagonal (C/S) |
|------|---------|------------------|----------------|----------------|
| Stego 21 | R | 0.95685 / 0.95623 | 0.93921 / 0.93862 | 0.91344 / 0.91287 |
|       | G | 0.94732 / 0.94653 | 0.92459 / 0.92383 | 0.89350 / 0.89273 |
|       | B | 0.97036 / 0.97025 | 0.95680 / 0.95669 | 0.93937 / 0.93927 |
| Stego 22 | R | 0.97977 / 0.97821 | 0.98931 / 0.98774 | 0.96969 / 0.96816 |
|       | G | 0.96906 / 0.96771 | 0.98249 / 0.98112 | 0.95554 / 0.95421 |
|       | B | 0.93274 / 0.93202 | 0.95760 / 0.95688 | 0.91828 / 0.91757 |
| Stego 23 | R | 0.98878 / 0.98785 | 0.98193 / 0.98092 | 0.97630 / 0.97526 |
|       | G | 0.98386 / 0.98248 | 0.97650 / 0.97515 | 0.96852 / 0.96718 |
|       | B | 0.97323 / 0.97276 | 0.96663 / 0.96612 | 0.95297 / 0.95246 |
Table 9 and Table 10 shows that correlation coefficient of cover image before and after embedding have difference lower than ‘0.1’. Values at table 9 and 10 are close to one which implied that both cover and stego image have a strong linear correlation.

3.2.6. Quality of stego image analysis

This test will be performed using Mean Squared Error (MSE) and Peak-Signal to Noise Ratio (PSNR) [7]. The results are shown in Table 11 and Table 12.

Table 11. MSE and PSNR result between cover image and stego image.

| Plain Image | Cover Image | Stego image | MSE    | PSNR   |
|-------------|-------------|-------------|--------|--------|
| 1           | Stego 11.png| 1.109047    | 47.3735|
| 2           | Stego 12.png| 3.23557     | 43.0312|
| 3           |             | -           | -      |
| 1           | Stego 21.png| 1.05736     | 47.8885|
| 2           | Stego 22.png| 2.88563     | 43.5283|
| 3           | Stego 23.png| 6.93587     | 39.7197|

Table 12. MSE and PSNR result between cover image and stego image.

| Plain Text  | Cover Image | Stego image | MSE    | PSNR   |
|-------------|-------------|-------------|--------|--------|
| 1           | Stego t 11.png| 0.00374     | 72.3955|
| 2           | Stego t 12.png| 0.01041     | 67.9529|
| 3           | Stego t 13.png| 0.02515     | 64.1238|
| 1           | Stego t 21.png| 0.01664     | 65.9181|
| 2           | Stego t 22.png| 0.04536     | 61.5640|
| 3           | Stego t 23.png| 0.11244     | 57.6212|

MSE is used to search the average of difference of two images pixel values. On the other hand, PSNR is the ratio of largest possibility power of a signal and distortion that influence the credibility of a digital data. The larger PSNR image quality means the greater of stego image fidelity means lower level distortions. PSNR expected to have greater value than 40 for high fidelity of stego image. Based on the Table 11, there is a data sample that did not meet that value.

3.2.7. Quality of decrypted object analysis

We will check the similarity of the result of extracting-decrypting image or text with the original image or text. The result shown on Table 13 and Table 14.

Table 13. MSE between decrypted image and original image.

| Plain Image | Stego image | MSE |
|-------------|-------------|-----|
| Stego 11.png| 0           |
| Stego 12.png| 0           |
| Stego 13.png| -           |
| Stego 21.png| 0           |
| Stego 22.png| 0           |
| Stego 23.png| 0           |
### Table 14. MSE between decrypted text and original text

| Plain Image | Stego image | MSE |
|-------------|-------------|-----|
| Stego t 11.png | 0 |
| 1 | Stego t 12.png | 0 |
| Stego t 13.png | 0 |
| 2 | Stego t 21.png | 0 |
| Stego t 22.png | 0 |
| Stego t 23.png | 0 |

From the Table 13 and Table 14 the result of extracting-decrypting process has 100% similarity with the original data.

### 4. Conclusion

From the result above, we can conclude that:

(i) Gauss-Circle Map can be used to generate chaotic sequence for encrypting and embedding purposes.

(ii) Performance of the algorithm with chosen parameters $x_0^{(1)} = x_0^{(2)} = 0, \alpha^{(1)} = \alpha^{(2)} = 9, \beta^{(1)} = \beta^{(2)} = 0.481, K^{(1)} = K^{(2)} = 1000000$, and $\Omega^{(1)} = \Omega^{(2)} = 0.5$ as follows:

(a) GCM sequence passes all 16 NIST test with parameters $x_0 = 0, \alpha = 9, \beta = 0.481, K = 1000000$, and $\Omega = 0.5$.

(b) Based on entropy test, GCM algorithm have good resistance against entropy attack.

(c) Key sensitivity of the algorithm reaches $10^{-23}$ and $10^{-16}$ respectively at encrypting and embedding process. The key space reaches $2.6244 \times 10^{1269}$, so that encrypting-embedding using GCM have a good resistant against brute-force attack.

(d) The encryption image is hard to be cracked by known plaintext attack due to distribution of cipher image values is uniform.

(e) Stego image results have strong linear correlation because the correlation coefficient value is above ‘0.89’ and the difference with its cover image is always below ‘0.1’.

However, it did not meet standard PSNR value.

(f) the result of extracting-decrypting process has 100% similarity with the original data.

### 5. Acknowledgements

This research is funded by Directorate of research and community engagements Universitas Indonesia in the Hibah PUTI Proceedings scheme with contract No. NKB-929/UN2.RST/HKP.05.00/2020.

### References

[1] Suryadi MT, Nurpeti E and Widya D 2014 performance of chaos-based encryption algorithm for digital image TELKOMNIKA 12 675-682

[2] Shannon C E 1949 Communication theory of secrecy systems The Bell system technical journal, 28(4), 656-715.

[3] Charan G S et al 2015 A novel lsb based image steganography with multi-level encryption IEEE Proceedings Internasional Conference on Innovations in Information Embedded and Communication Systems, 62-66.

[4] Nurluthfia Z 2016 Digital image protection using chaos based cryptography and steganography Skripsi FMIPA, Universitas Indonesia.

[5] Kocarev L and Lian S 2011 Chaos-based cryptography: theory, algorithm and applications, (Berlin: Springer-Verlag)

[6] MT Suryadi et al 2018 Implementation of digital image encryption algorithm using logistic function and DNA encoding J. Phys.: Conf. Ser. 974 012028
[7] MT Suryadi et al 2017 New modified map for digital image encryption and its performance J. Phys.: Conf. Ser. 893 012050
[8] A Delmi et al 2020 Digital image steganography by using the edge adaptive based chaos cryptography J. Phys.: Conf. Ser. 1442 012041
[9] MT Suryadi, Yudi Satria and Luqman N Prawadika 2020 An improvement on the chaotic behavior of the Gauss Map for cryptography purposes using the circle map combination J. Phys.: Conf. Ser. 1490 012045
[10] Suryanto Y, Suryadi MT and Ramli K 2017 A New Image Encryption using color scrambling based on chaotic permutation multiple circuler shrinking and Expanding Multimedia Tools and Applications 76 16831-16854
[11] Philip L B 1986 Bifurcations of circle maps: Arnol'd tongues, bistability and rotation intervals, (Berlin: Springer-Verlag)
[12] Hemanta Kr Sarmah et al 2016 Characterising chaos in gaussian map International Journal of Advanced Scientific and Technical Research 6 160-172
[13] Andrew R and Juan S et al 2010 A statistical test suite for random and pseudorandom number generators for cryptographic applications NIST Special Publication 800-22 Revision 1a