Digital image encryption algorithm through unimodular matrix and logistic map using Python

Cite as: AIP Conference Proceedings **2331**, 020006 (2021); https://doi.org/10.1063/5.0041653
Published Online: 02 April 2021

Indra Bayu Muktyas, Sulistiawati and Samsul Arifin

Digital Image Encryption Algorithm Through Unimodular Matrix and Logistic Map Using Python

Indra Bayu Muktyas[1, a)], Sulistiawati[1, b)], and Samsul Arifin[2, c)]

[1]*Mathematics Education Department, STKIP Surya, Tangerang, Indonesia 15115*
[2]*Department of Mathematics, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480*

[a)]recobayu@gmail.com
[b)]sulistiawati@stkipsurya.ac.id
[c)]Corresponding author: samsul.arifin@binus.ac.id

Abstract. Hill Cipher is one of the methods used in Cryptography. In the Hill Cipher algorithm, the key square matrix must have an inverse modulo. One special matrix that definitely has an inverse is the unimodular matrix. The unimodular matrix can be used as a key in an encryption process. The purpose of this research is to show an alternative in securing digital image data. The type of cryptography used is symmetric cryptography. The algorithm presented by generating a unimodular matrix using the Logistic Map. First, to get a unimodular matrix, we use an identity matrix. The sequence of real number in (0,1) of Logistic Map then converted into integer number from 0 to 255. That number then occupies upper triangular entries of the unimodular matrix. We use the Elementary Row Operation to obtain the complete matrix as a key. Then the multiplication of matrix modulo is used as an encryption process. The unimodular matrix generated by the proposed algorithm guarantees a key in the Hill Cipher for all matrix sizes ($n > 4$) and only need two parameters (password 1 and password 2). We also tested the algorithm in python language on two digital images (grayscale and color) with different sizes. The results show that the encrypted images are very difficult to read by third parties. The time needed is based on password 1, that is the size of the key matrix. So, the unimodular matrix and logistic map work very well with Hill Cipher to encrypt a digital image.

INTRODUCTION

In this decade, digital images are very important. The process of securing digital images is usually done in the process of sending from one person to another. We will try to emphasize the level of security in transmissions that the security level is low. It requires increased security of digital images, in both sending and storing them. One of the ways is to encrypt digital images [12]. One of the classic encryption methods is Hill Cipher. In recent years, there have been several modifications to the Hill Cipher. Among them are Hill Cipher combined with genetic algorithms [20], the hill Cipher with pixel value transformation and image block randomization [13], and the Hill Cipher combined with chaos functions [9, 10]. However, these methods only use a limited key matrix of sizes 3x3 or 4x4. If the size of the matrix key is more than four, it will say that it is difficult to find the inverse key or it takes a long time to find that key matrix which is invertible [10, 20].

This problem can be overcome by utilizing a special matrix, which is a unimodular matrix [4, 7]. We form a unimodular matrix by utilizing the Elementary Row Operations. It is not necessary to use a whole matrix as a key. We will use the Logistic Map to form a unimodular matrix so that fewer parameters are needed. Furthermore, the proposed algorithm will be applied to several grayscale and color standard images with the programming language Python 2.7.14. Python language is very easy to understand [16]. The program is also very easy to get. Python is available on a variety of OS such as Windows, Linux, Mac OS, and Android [17]. Python also has many uses and applications in many groups of expertise and disciplines [5, 18]. Those are the several reasons why we use Python.

Based on the explanation above, we offer an encryption algorithm on digital images through a unimodular matrix and logistical map using Python in this paper. The discussion begins with the research method and a review of the

The 2nd Science and Mathematics International Conference (SMIC 2020)
AIP Conf. Proc. 2331, 020006-1–020006-7; https://doi.org/10.1063/5.0041653
Published by AIP Publishing. 978-0-7354-4075-3/$30.00

theories used, and our Python code in Session 2. The implementation of the algorithm and some analysis will be discussed in Session 3, and Session 4 will close this paper.

METHOD

Lester Hill discovered how to encrypt plaintext by utilizing a System of Linear Equations (SLE). The encryption process with Hill Cipher is to divide plaintext into several blocks. Suppose each block of plaintext consists of *n* elements, then ciphertext is generated by solving the SLE with *n* equations and *n* variables modulo *m* with *m* are some numbers [1, 2, 9]. The completion of the SLE can be done by matrix multiplication. Because the Hill Cipher is symmetrical cryptography, the key which is generated must have an inverse [14, 15, 19].

Hanson [8] introduced the notion of a unimodular matrix or the Nice Matrix. A matrix *A* with each entry in the form of an integer is called unimodular if det (*A*) = -1 or det (*A*) = 1. Examples of unimodular matrices are identity matrices, upper triangles or lower triangles with diagonals of 1 or -1. It is confirmed by the following theorem.

Theorem 1. Anton (3) *If A is a triangle matrix then,* $\det(A) = a_{11}a_{22}\dots a_{nn}$.

The following lemma explains the steps in generating a unimodular matrix. Note that Lemma 2 will be used as a reference in making a program to generate a unimodular matrix using Python.

Lemma 2. Hanson (8) *A unimodular matrix* A_n *can be constructed in the following ways:*

1. First, make a diagonal matrix with the diagonal entry $a_{ii} = 1$ or $a_{ii} = -1$.
2. Second, fill in any random integers at each entry with i < j. From this, it has formed a top triangular matrix whose determinants are 1 or -1. It is a unimodular matrix.
3. Third, to be a complete matrix, use the Elementary Row Operations (ERO) “add a row with multiply of another row”.

From Lemma 2 above, in the first step, we will use $a_{ii} = 1$ because it will implement positive integer values. As for the second step of the lemma, we will use logistic functions and convert it into integer between 0 and 255 as random numbers. Furthermore, in the third step, we also use the logistic function to get the multiplier factor so that we can apply "add a row with multiply of another row" to the key matrix. The logistic function obtained is in the form of real numbers between 0 to 1. Then the conversion is made from real to an integer by taking the first three digits after the decimal point and then we do modulation with 256.

Logistics map was first introduced by May [11] in 1976. One of the chaotic functions looks simple, but it produces a very complex dynamic sequence [6, 21]. The recursive equation is

$$x_{n+1} = rx_n(1 - x_n) .$$

Logistic maps produce a random sequence of values and spread between (0,1) for the value of *r* in (3.7, 4]. In order to produce a row of integers that valued between 0 to 255, there are several ways that can be done, among them is by taking three the first digit after the decimal point of x generated by the logistic map and then the results modulated 256. Can be developed with 4, 5, 6, ... the first digit of x, but the more digits used will affect the speed of the program. The results of the floor function or roof function from x * 256. In this study, the first three digits taken after the decimal point [22, 23].

In this paper, we propose four algorithms that use a unimodular matrix as the key. The matrix is built using logistic maps. Algorithm 1 and 2 is used in the encryption and decryption process. The Python code of this algorithm can be downloaded here: *https://github.com/muktyas/encryption-unimodular-logistic-map*.

Algorithm 1: Generating Logistic Map Sequence

```
1.  x = x0                        #x0 as initial value
2.  loop to 1000 times first to make sensitive sequence of logistic map
3.  barisan = []
4.  for i in range(n):
5.      x = 3.9 * x * (1 - x)     # 3.9 can be replaced by real number in (3.7, 4]
6.      barisan[i] = x*1000%256   # take 3 first digit after decimal point
7.  return barisan
```

Algorithm 2: Generating Unimodular Matrix (Key)

```
1. K=I_n
2. for i < j:
       K_ij = random numbers generated by logistic map
3. Use ERO add a row with multiply of another row modulo m to complete the matrix K.
```

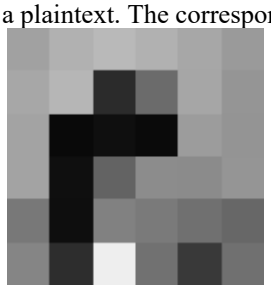
Algorithm 3: Encryption

```
1. Convert an image to a matrix P_(b,k)
2. Generate a key matrix K_n using Algorithm 1, that is n|(bk)
3. Reshape the matrix P_(b,k) to a matrix P_(n, bk/n)
4. C_(n, bk/n) = K_n * P_(n, bk/n)
5. Reshape the matrix C_(n, bk/n) to a matrix C_(b,k).
6. Convert the matrix C_(b,k) to encrypted image.
```

Algorithm 4: Decryption

```
1. Convert an image to a matrix C_(b,k)
2. Generate a key matrix K_n using Algorithm 1, that is n|(bk)
3. Find the invers matrix K_n^(-1) using ERO mod m
4. Reshape the matrix C_(b,k) to a matrix C_(n, bk/n)
5. P_(n, bk/n) = K_n^(-1) * C_(n, bk/n)
6. Reshape the matrix P_(n, bk/n) to a matrix P_(b,k).
7. Convert the matrix P_(b,k) to decrypted image.
```

The example encryption process of the algorithm is as follows. First, we take cameraman.png which has been resized to 6x6 as a plaintext. The corresponding images and matrices are as follows.



(a)

$$P = \begin{bmatrix} 161 & 177 & 185 & 177 & 167 & 154 \\ 167 & 182 & 43 & 107 & 166 & 149 \\ 163 & 9 & 15 & 10 & 156 & 148 \\ 163 & 15 & 99 & 140 & 139 & 149 \\ 120 & 14 & 129 & 122 & 112 & 103 \\ 133 & 45 & 238 & 113 & 57 & 112 \end{bmatrix}$$

(b)

FIGURE 1. (a) Cameraman.png that resized to 6x6 pixel (b) matrix that represents the cameraman.png

Because the image size is 6x6 = 36, the plaintext matrix can be resized according to a factor of 36, i.e. 2, 3, 4, 6, 9, 12, and 18. For example, the number 4 is chosen as a password 1. Then the size of the plaintext matrix is now 4x9, that is:

$$P = \begin{bmatrix} 161 & 177 & 185 & 177 & 167 & 154 & 167 & 182 & 43 \\ 107 & 166 & 149 & 163 & 9 & 15 & 10 & 156 & 148 \\ 163 & 15 & 99 & 140 & 139 & 149 & 120 & 14 & 129 \\ 122 & 112 & 103 & 133 & 45 & 238 & 113 & 57 & 112 \end{bmatrix}$$

Furthermore, suppose we enter password 2: 19062020. These numbers then converted into real number by adding "0." in front of it and add "1" at the end. Moreover, "0." will turn these numbers into intervals (0,1), while the number "1" at the end will ensure that the values are unique. For example, a password 2: 123 will be different from 1230 or 12300. A 4x4 key size generated, the initial value is 0.190620201, with the following process:

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Next, a sequence is formed based on algorithm 1, which uses a logistic map with an initial value of x0 = 0.190620201. We have sequence 117, 141, 64, 81, 243, 206. These numbers become the entries of an upper triangle matrix as follows:

$$K = \begin{bmatrix} 1 & 117 & 141 & 64 \\ 0 & 1 & 81 & 243 \\ 0 & 0 & 1 & 206 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By Lemma 2 section (3), a key matrix obtained with the following process:

$$\begin{bmatrix} 1 & 117 & 141 & 64 \\ 95 & 108 & 164 & 179 \\ 0 & 0 & 1 & 206 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 117 & 141 & 64 \\ 95 & 108 & 164 & 179 \\ 79 & 27 & 132 & 142 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 117 & 141 & 64 \\ 95 & 108 & 164 & 179 \\ 79 & 27 & 132 & 142 \\ 101 & 41 & 161 & 65 \end{bmatrix}$$

And the consequences are obtained:

$$K = \begin{bmatrix} 1 & 117 & 141 & 64 \\ 95 & 108 & 164 & 179 \\ 79 & 27 & 132 & 142 \\ 101 & 41 & 161 & 65 \end{bmatrix}$$

Then, do the matrix multiplication K_{4x4} with a P_{4x9} mod 256, so we get:

$$K \times P = \begin{bmatrix} 207 & 210 & 25 & 140 & 147 & 6 & 145 & 248 & 220 \\ 157 & 163 & 244 & 34 & 72 & 88 & 20 & 45 & 89 \\ 176 & 253 & 252 & 198 & 30 & 243 & 37 & 116 & 133 \\ 37 & 74 & 68 & 193 & 44 & 76 & 166 & 17 & 60 \end{bmatrix}$$

Finally, change it back to the size of 6x6 as a *C*6x6 and turn it into a ciphertext image.

$$C = \begin{bmatrix} 207 & 210 & 25 & 140 & 147 & 6 \\ 145 & 248 & 220 & 157 & 163 & 244 \\ 34 & 72 & 88 & 20 & 45 & 89 \\ 176 & 253 & 252 & 198 & 30 & 243 \\ 37 & 116 & 133 & 37 & 74 & 68 \\ 193 & 44 & 76 & 166 & 17 & 60 \end{bmatrix}$$

(a)

(b)

FIGURE 2. (a) Matrix ciphertext (b) Ciphertext image

Continuing the example above, we want to carry out the decryption process. The image we have is Figure 2 (b), and the matrix corresponding to that image is Figure 2 (a). Furthermore, Password 1: 4 and Password 2: 19062020, and from Password 1 then, the ciphertext matrix resized to 4x9, namely:

$$C = \begin{bmatrix} 207 & 210 & 25 & 140 & 147 & 6 & 145 & 248 & 220 \\ 157 & 163 & 244 & 34 & 72 & 88 & 20 & 45 & 89 \\ 176 & 253 & 252 & 198 & 30 & 243 & 37 & 116 & 133 \\ 37 & 74 & 68 & 193 & 44 & 76 & 166 & 17 & 60 \end{bmatrix}$$

Based on the previous encryption process, a key matrix K obtained as follows:

$$K = \begin{bmatrix} 1 & 117 & 141 & 64 \\ 95 & 108 & 164 & 179 \\ 79 & 27 & 132 & 142 \\ 101 & 41 & 161 & 65 \end{bmatrix}$$

Furthermore, the inverse of the matrix K is as follows:

$$K^{-1} = \begin{bmatrix} 137 & 139 & 120 & 63 \\ 89 & 1 & 175 & 59 \\ 247 & 0 & 1 & 50 \\ 155 & 0 & 0 & 1 \end{bmatrix}$$

And the plaintext matrix P_{4x9} is as follows:

$$P = K^{-1}C = \begin{bmatrix} 161 & 177 & 185 & 177 & 167 & 154 & 167 & 182 & 43 \\ 107 & 166 & 149 & 163 & 9 & 15 & 10 & 156 & 148 \\ 163 & 15 & 99 & 140 & 139 & 149 & 120 & 14 & 129 \\ 122 & 112 & 103 & 133 & 45 & 238 & 113 & 57 & 112 \end{bmatrix}$$

After that, we reshape the matrix P_{4x9} above becomes a matrix P_{6x6} which corresponds to Figure 1 (b), then we transform the matrix P into the corresponding image in Figure 1 (a). This step ends our decryption process.

RESULT AND DISCUSSION

The computer specifications used in this study are Intel® Core ™ i3-2350M CPU @ 2.30GHz 2.30 GHz, 4GB RAM with Windows 10 64-bit operating system. The proposed algorithm successfully implemented on two digital images, which is cameraman.png (grayscale, 512x512 pixel) and morning.png (color, 4195x2773 pixel) using Python 2.7.14, a programming language that needs a PIL and NumPy package. Moreover, to install PIL and NumPy in Windows OS is very easy (16).

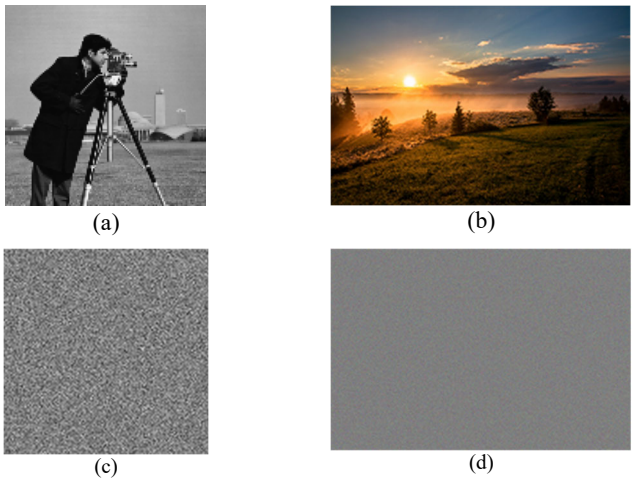


FIGURE 3. (a) Display of cameraman.png (b) Display of morning.png
(c) Ciphertext image of cameraman.png (d) Ciphertext image of morning.png

From Figure 3, the information of ciphertext images (c and d) are hard to read. So, the algorithm successfully encrypts the plaintext image.

Encryption Time analysis

The encryption keys (password 1 and password 2) used on the experiment shown in Table 1.

TABLE 1. Average Encryption Time of Cameraman.png and Morning.png

Data sample	Password 1	Password 2	Time (second)
Cameraman.png	4	0852	0.130
	16	0852	0.132
	128	0852	0.818
	256	0852	4.696
	512	0852	32.340
Morning.png	3	0852	8.686
	47	0852	8.567
	59	0852	9.500
	141	0852	10.382
	177	0852	11.552

From Table 1, we can see that the encryption time is depending on password 1 because it related to the key sizes.

Histogram Analysis

The following is a histogram comparison of the original image (a) and (c) and encrypted image (b) and (d):

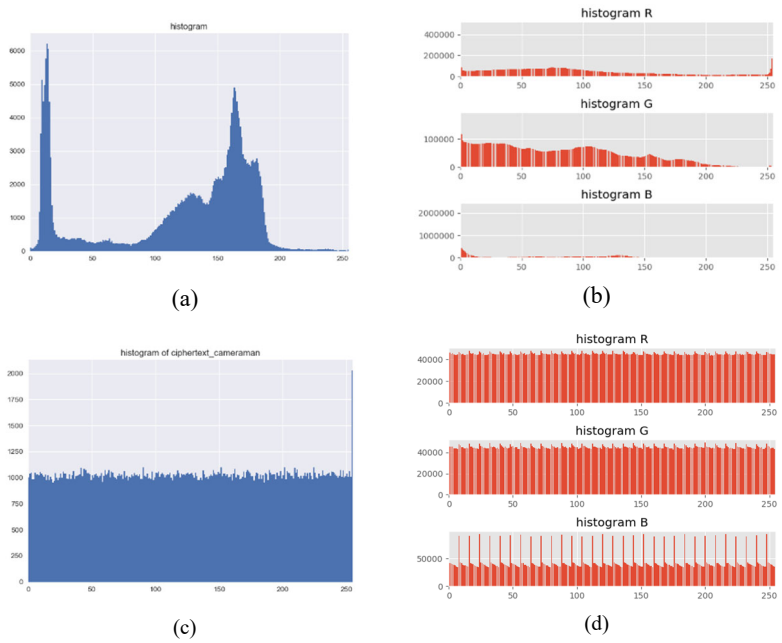


FIGURE 4. (a) Histograms of cameraman.png (b) Histograms of morning.png, (c) Histogram of ciphertext cameraman.png (d) Histograms of ciphertext morning.png

Note that based on Figure 4, it appears that the histogram of the initial images (a) and (b) tends to be fluctuating, while the histogram of the encrypted images (c) and (d) tends to evenly distributed.

Comparison with Standard Hill Cipher Algorithm

The comparison of the proposed algorithm and standard Hill Cipher shown in Table 2 (see 2, 9, 10, 13).

TABLE 2. Comparison of Proposed Algorithm and Standard Hill Cipher

Properties	**Standard Hill Cipher**	**Unimodular Hill Cipher (proposed algorithm)**
Size of key matrix K_n	usually $n \leq 4$	any $n > 0$
Storage of key matrix K_n	one whole matrix K_n	only 2 parameters

CONCLUSION

The standard Hill Cipher usually use a small size of the key matrix $K_n, n \leq 4$ because of the difficult to search that invertible matrix. Furthermore, if $n > 4$ using standard Hill Cipher, it takes the whole matrix K_n as key. On this paper, we generate a unimodular matrix using the logistic map as the key that can solve that problem. That is $n > 4$ but only takes two parameters (password 1 and password 2). The experimental result obtained shows that the encrypted image is hard to read by normal eyes. From time analysis, we conclude that the greater the password 1, the longer the encryption time needed, not based on grayscale or color images. The histogram frequency of encrypted images tends to evenly distributed.

ACKNOWLEDGMENT

The authors wish to thank Nerru Pranuta Murnaka. This work was supported with the help of facilities, funding, and infrastructure of Bina Nusantara University. Thank you deeply for the support during this time.

REFERENCES

1. B. Acharya, S.K. Panigrahy, S.K. Patra and G. Panda, International Journal of Recent Trends in Engineering, **1** (1), pp. 663-667 (2009).
2. G. Agrawal, S. Sing and M. Chaudhary, IJARCS, **1** (4), (2010).
3. H. Anton and C. Rorres, Aljabar Linear Elementer Versi Aplikasi Edisi Kedelapan, (Erlangga, Jakarta, 2004).
4. S. Arifin and I.B. Muktyas, Jurnal Derivat, **5** (2), (2018).
5. S. Arifin and H. Garminia, International Journal of Scientific & Technology Research, **8** (7), pp. 194-199 (2019).
6. J. Bao and Q. Yang, Nonlinear Dyn, **70**, pp. 1365–1375 (2012). Available from: https://doi.org/10.1007/s11071-012-0539-3.
7. R.K. Guy, A Determinant of Value One. §F28 in Unsolved Problems in Number Theory, 2nd ed., (Springer-Verlag, New York, 1994), pp. 265-266.
8. R. Hanson, The Two-Year College Mathematics Journal, **13** (1), pp. 18-21 (1982).
9. S. Hraoui, F. Gmira, M.F. Abbou, A.J. Oulidi and A.A. Jarjar, Procedia Computer Science, **148**, pp. 399-408 (2019).
10. M. Jarjar, S. Najah, K. Zenkouar and S. Hraoui, in 2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), Meknes, (Morocco, 2020), pp. 1-6. DOI: 10.1109/IRASET48871.2020.9092046.
11. R. May, Nature, **261** (5560), pp. 459-467 (1976).
12. I.B. Muktyas, in Prosiding Seminar Nasional Matematika 2014 (Departemen Matematika FMIPA UI, 2014).
13. S. K. Muttoo, D. Aggarwal and B.A. Ahuja, Bulletin of Electrical Engineering and Informatics, **1** (1), pp. 51-60 (2011).
14. D. Nofriansyah, et.al., J. Phys.: Conf. Ser., **954** (1), p. 012003 (2018).
15. J. Overbey, W. Traves and J. Wojdylo, Cryptologia, **29** (1), pp. 59-72 (2005). DOI: 10.1080/0161-110591893771.
16. Python, Available at https://www.python.org/ [Accessed June 15, 2020].
17. QPython - Python for Android, Available at https://play.google.com/store/apps/details?id=org.qpython.qpy&hl=en [Accessed June 15, 2020].
18. B. Rahman, S. Arifin and I.B. Muktyas, International Journal of Engineering & Technology, **8** (9), pp. 2282-2285 (2019).
19. R. Ranjan, R.K. Sharma and M. Hanmandlu, Applications and Applied Mathematics: An International Journal (AAM), **11** (1), pp. 45 – 60 (2016).
20. A.P.U. Siahaan, AIJRSTEM, **15** (1), 84-89 (2016).
21. T. Li, B. Du and X. Liang, *IEEE Access*, **8**, pp. 13792-13805 (2020).
22. K.W. Wong, *Studies in Computational Intelligence vol. 184* (Springer, Heidelberg. 2009).
23. L, Zhang, X. Liao and Wang X, *Solitons & Fractals*, **24** (3), 759-765 (2005).