Encrypting ImageNet with Chaotic Logistic Maps And AES in ECB Mode

P. Rivas-Perea[1], **P. Handley**[2], **and R. Aragon Franco**[3]
[1]Department of Computer Science, Marist College, Poughkeepsie, New York, USA
[2]IBM Corporation, Poughkeepsie, New York, USA
[3]Computing Center, Nogales Institute of Technology, Nogales, Mexico

Abstract— *Unlike ordinary data encryption, images posses unique features such as high correlation among pixels, bulky data capacity, and high redundancy; thus, traditional encryption methods such as the Advanced Encryption Standard (AES) in CBC mode may lead to high pixel correlation. In this paper, we review an algorithm that takes advantage of the recent advances in chaos-based encryption methodologies, known as Chaotic Logistic Maps (CLM). Using CLM we encrypted the ImageNet dataset to analyze the algorithm's performance. Results indicate that the algorithm offers a fast processing time and suggest that CLM produces encrypted images with low pixel correlation. CLM presents an alternative way to securely transmit images over an insecure channel.*

Keywords: chaos theory; cyber security; encryption

1. Introduction

Recently, there has been interest in chaos-based image encryption schemes [1], [2], [3], [4]. This is due to the desirable properties Chaotic Maps offer such as mixing, sensitivity to initial conditions/parameters, pseudorandomness, etc. The proposed encryption scheme works by combining both a Chaotic Logistic Map (CLM) [5] and a Rivest Cipher 4 (RC4) [6] encryption method. A CLM comes from a branch of mathematics that deals with nonlinear dynamic systems called Chaos Theory. A CLM gets its name because it maps the population value, x, to a specific growth rate, θ, at some time t, as follows:

$$x_{t+1} = \theta x_t - \theta x_t^2. \tag{1}$$

This chaotic system exhibits a great sensitivity to initial conditions, mixing property and ergodicity, these are properties desired for cryptography and can be considered analogous to properties such as confusion, diffusion, etc. The behavior changes drastically as the system parameter θ, called growth rate, varies from $[0 \ldots 4]$. When θ is in the range $[0 \ldots 1]$, the system converges to population 0, at $\theta = 2$ the system converges at a population level of 0.5. These values that the system settles toward over time are known as attractors. Chaotic behavior is observed when θ is set greater than 3.5, at values beyond 3.5 the system's attractor is no longer at a fixed-point but rather oscillates forever, never repeating itself or converging into a steady state of behavior; when θ reaches 4 the system is capable of landing on any population value in a seemingly random fashion. Although seemingly random, the system is not random at all, it simply follows deterministic rules that are able to produce apparent randomness. This is a key property of chaotic systems: determinism and aperiodicity. And the values of θ within the interval of $[3.6 - 4.0]$ are the most interesting ones for chaos-based image encryption schemes.

The other main component of the proposed scheme is the RC4 Stream Cipher. Designed by Ron Rivest *et.al.*, RC4 has been one of the most popular stream ciphers due to its simplicity and speed. As with any stream cipher, it is used to generate a pseudo-random streams of bits given an input which are invertible with a key. RC4 makes use of a secret internal state to generate the key-stream. This process typically involves two major algorithms: a Key-Scheduling algorithm (KSA), and a Pseudo-random generation algorithm (PRGA) [7]. KSA is used to initialize a permutation of all 256 possible bytes with a variable length key (typically 40 - 2048 bits), and once the initial permutation is complete, a stream of bits is generated using the PRGA. Despite the popularity or RC4, vulnerabilities have been discovered in RC4, rendering it insecure [8]. The main weakness is in the initial state of the algorithm; during that state the output of KSA and PRGA have a probability distribution that indicates the output stream is not random and, therefore, vulnerable to attacks. In this paper, we mitigate such vulnerabilities using CLM to produce pseudo-random numbers for the initial state of RC4. Making this a highly robust encryption system that we review on ImageNet.

The rest of this paper is organized in the following manner. First, in Section 2 we briefly discuss the implementation of the proposed scheme. Then, describe our experiments on ImageNet and results in Section 3. Finally, in Section 4 we present our conclusions.

2. Methodology

The proposed image encryption process is composed of two main stages. In the first stage we make use of the CLM and take advantage of its chaotic behavior. As shown in Eq. (1), the chaotic logistic map is a second degree polynomial mapping a recurrent relation that exhibits chaotic behavior. During the CLM stage, we first take a 16 character-long (128 bits) external key and map it into its hexadecimal form:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
key:	i	p	c	v	–	1	8	–	v	e	g	a	s	–	b	b
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
k hex:	69	70	63	76	2D	31	38	2D	76	65	67	61	73	2D	62	62

The hexadecimal key, k, is considered as an array of 32 hexadecimal elements then used to calculate the initial condition, x_0, for the logistic map as follows:

$$x_l = (k_0 \times 2^0 + k_1 \times 2^1 + \cdots + k_{15} \times 2^{15})/2^{15} \tag{2}$$

$$x_r = (k_{16} \times 2^{16} + k_{17} \times 2^{17} + \cdots + k_{31} \times 2^{31})/2^{31} \tag{3}$$

$$x_0 = (x_l + x_r) \bmod 1 \tag{4}$$

where k_n is the n-th hexadecimal element of k.

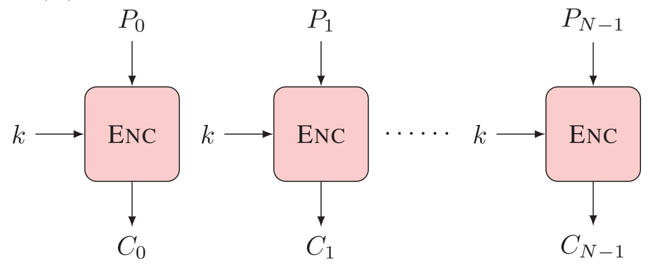
Once the initial condition x_0 is obtained it is fed into the CLM to generate an array of size 256, U, of pseudo random numbers; this process uses x_0 in (1) with $t = 0$ and a value of $\theta = 4$ to recurrently produce values up to $t = 255$.

The array U is further converted to an integer and its values are used as indices to perform permutations in a traditional RC4 stage, in which another array, $S \in \mathcal{Z}^{256}$, whose elements are $[0, 1, \ldots, 255]$ is used along with U to schedule a key in the KSA. The output is a permuted S array, which is then put into the pseudo-random generation stage using traditional PRGA to stream integers in the range $[0, 255]$. The outputs from the PRGA is added to all the existing RGB channel values, then the modulo 256 operation gives us the corresponding encrypted pixel value, thus completing the encryption process.

The decryption process is done by adding 256 to each value and subtracting the value outputted by the PRGA, then taking the modulo by 256. For more details see [2].

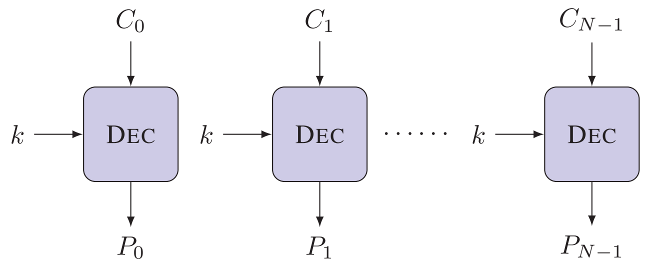
3. Experimental Setup and Results

Using the Python language we implemented the CLM-based encryption protocol; we successfully executed the encryption process by taking an input image, $I(n_1, n_2, n_3)$, and a 16-character encryption key which is used to encrypt I with the scheme described above and outputting an encrypted image $T(n_1, n_2, n_3)$, where n_1, n_2, n_3 are the number of rows, columns, and channels of the input image, respectively. The input image can be processed all at once or in N blocks, $P_0, P_1, \ldots, P_{N-1}$, bounded as $1 \leq N \leq \frac{n_1 \times n_2 \times n_3}{|P|}$. The process is illustrated as follows:



This mode of encryption that processes data in blocks is known as Electronic Codebook (ECB). The reason for using the CLM algorithm on ECB mode is only for comparison purposes with the Advanced Encryption Standard (AES) which is optimized for processing in blocks that, usually, match the key size. The corresponding N blocks of encrypted information, $C_0, C_1, \ldots, C_{N-1}$, are then used to build the encrypted image T.

The decryption process follows the exact same process but in the opposite direction:

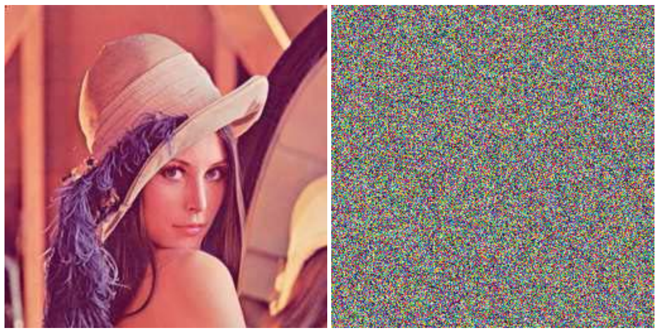


From the decrypted blocks P, the image I is recovered using the same key that was used for the initial encryption process.

Since one of the main properties of good encryption systems is to produce output that appears to be random, it makes sense to discuss now how an encrypted image looks like. Figure 1 (a) shows input image I as the iconic *Lena* and in (b) its corresponding encrypted image, T. Note that since the output of CLM is in the range $[0, 255]$, the encrypted image requires no adjustment in order to be displayed. Figure 1 (c) displays the histogram of I and (d) shows the histogram of T. Note that the histogram of T is clearly in the same region for all channels and that the distribution of the histogram follows a uniform distribution, indicating high randomness.

Furthermore, the correlation between neighboring pixels is very low for CLM. To demonstrate this, we have compared the CLM algorithm and the AES algorithm on ECB mode using its optimized AES 128-bit version for 16-byte keys [9]. We also used a popular image dataset entitled "ImageNet." We particularly used the Large Scale Visual Recognition Challenge 2017 (ILSVRC2017) test set [10]. This newly released test set contains 40k images that vary in their contents from natural images to digitally produced images of anything. The average size is 482×415 pixels per image.

Figure 2 shows the average correlation, r, between pixels in the encrypted images of ImageNet. The pixel correlations are measured horizontally, vertically, and diagonally. In all cases the correlation between pixels is better (closer to zero) for CLM. Moreover, the box plots indicate also a higher variance in the correlation values for AES and a lower one for CLM. The figure shows that CLM-based encryption is, in most cases, closer to a zero average correlation, i.e., vertically and diagonally; furthermore, it also has the smallest variance when compared to AES.


(a) Input: "Lena" (b) "Lena" encrypted

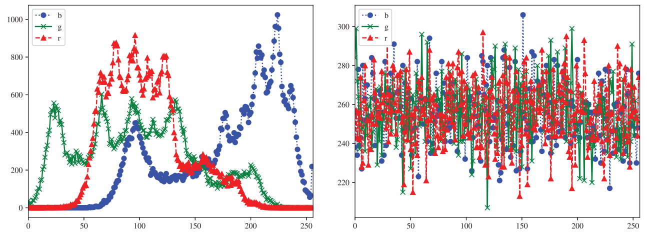

(c) Histogram of input (d) Histogram of encrypted

Fig. 1: Sample encryption using Lena. (a) is the input I, (b) is the output T, (c) is the histogram of I, and (d) is the histogram of T. Note the uniform distribution of T across channels, suggesting randomness.

To appreciate better which images produce the highest and lowest correlation we present Figure 3. The figure shows the top 5 best and worst correlations measured as the sum of the absolute values of the correlations for each channel, indicated as $\Sigma|r|$. The first row is the input image I; the second row corresponds to the AES encrypted image; and the third row corresponds to the CLM encrypted image. Figure 3 (a) shows the top-five lowest, best, correlations for AES and Figure 3 (b) does the same for CLM. Both Figure 3 (a) and (b) suggest that images with random variations in the color patterns of the objects produce the lowest correlation in the encrypted images. However, Figure 3 (c) which shows the images producing the highest correlations for AES, suggest that images with small variations in color tend to produce highly correlated encrypted images. On the contrary, the third row of Figure 3 (c), which corresponds to CLM, indicates that the algorithm does not suffer from this problem. Furthermore, if we observe the worst correlations for CLM shown in Figure 3 (d) they do not seem to exhibit any problems. In fact, it is clear that even the highest aggregated absolute correlation reported for CLM on ImageNet, with a value of $\Sigma|r| = 0.05$, is significantly smaller than for AES, with a value of $\Sigma|r| = 2.84$. The problem with AES can be easily solved by using an initial vector (IV) in Cipher Block Chaining (CBC) mode; however, doing so will not be a fair comparison with CLM since CLM solves the problem

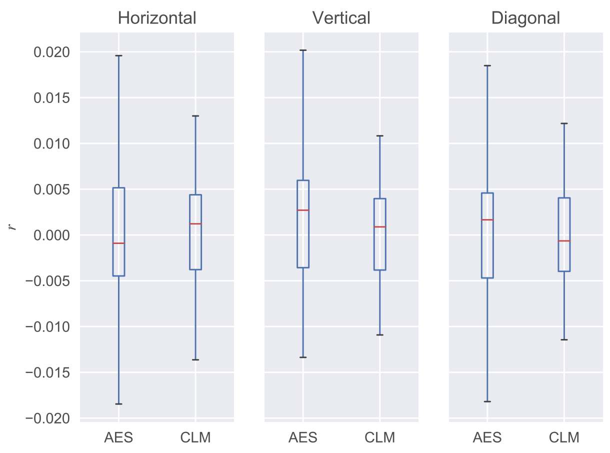


Fig. 2: Comparison of correlation, r, between AES and CLM over ImageNet test set. Left: horizontal correlation between pixels. Middle: vertical correlation. Right: Diagonal correlation. In all cases the box plots indicate that CLM has a correlation closer to zero, and with a significantly smaller variance.

without the need of an IV nor CBC mode.

Finally, it is worth mention that the CLM implementation is comparable in speed to AES, as shown in Figure 4. The figure depicts the summarized trend of processing time on the ImageNet dataset as a function of the number of pixels, where each pixel is considered a three-dimensional vector for color images. Figure 4 suggests that CLM is comparable to AES using Python on a single core machine.

4. Conclusion

In this paper we analyzed a CLM algorithm for encryption that exhibits the characteristic randomness of a good cypher. We further compared the encrypted output with AES over the ImageNet dataset showing that CLM produces low correlation and a comparable speed to AES. We believe that CLM is a good alternative to ciphers that do not take advantage of image structures or that need additional mechanisms to reduce correlation in encrypted images.

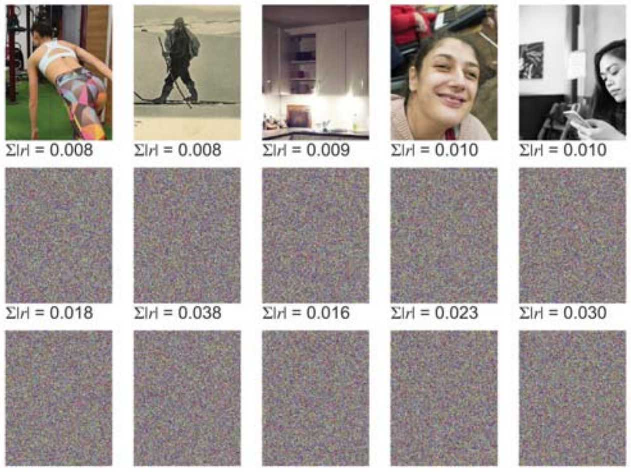
References

[1] J.-x. Chen, Z.-l. Zhu, C. Fu, H. Yu, and L.-b. Zhang, "A fast chaos-based image encryption scheme with a dynamic state variables selection mechanism," *Communications in Nonlinear Science and Numerical Simulation*, vol. 20, no. 3, pp. 846–860, 2015.

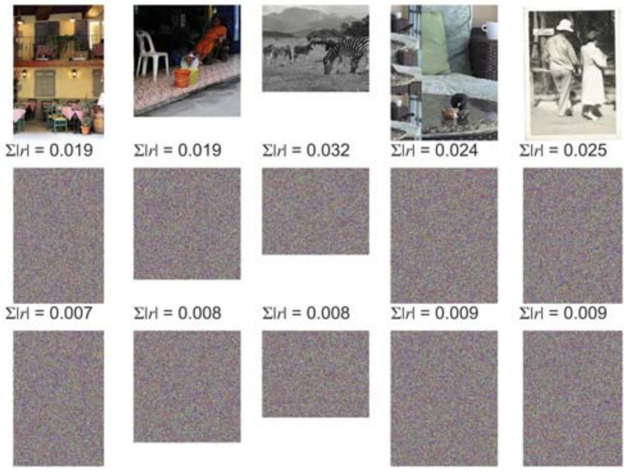
[2] M. T. Gaata and F. F. Hantoosh, "An efficient image encryption technique using chaotic logistic map and rc4 stream cipher," *International Journal of Modern Trends in Engineering and Research*, vol. 3, no. 9, pp. 213–218, 2016.

[3] R. Enayatifar, A. H. Abdullah, and I. F. Isnin, "Chaos-based image encryption using a hybrid genetic algorithm and a dna sequence," *Optics and Lasers in Engineering*, vol. 56, pp. 83–93, 2014.

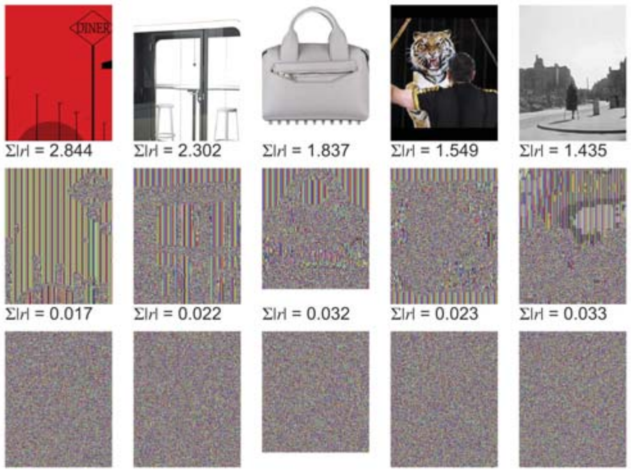
[4] M. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, R. López-Gutiérrez, and O. A. Del Campo, "A rgb image encryption algorithm based on total plain image characteristics and chaos," *Signal Processing*, vol. 109, pp. 119–131, 2015.



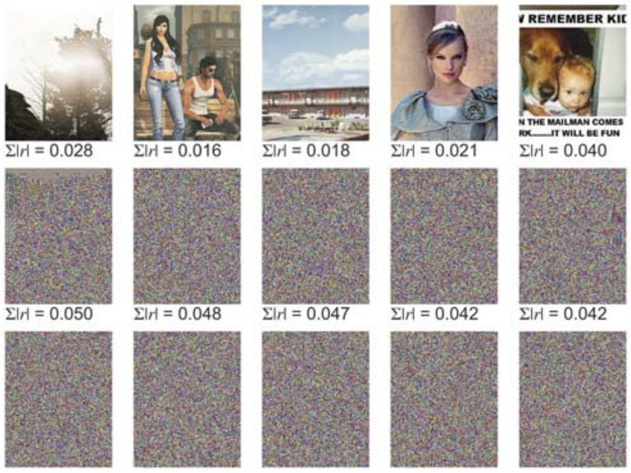
(a) AES top-five best $\Sigma|r|$



(b) CLM top-five best $\Sigma|r|$



(c) AES top-five worst $\Sigma|r|$



(d) CLM top-five worst $\Sigma|r|$

Fig. 3: ImageNet top-five best and worst $\Sigma|r|$ for AES and CLM. Top row is I, middle is T with AES, and bottow row is T with CLM.

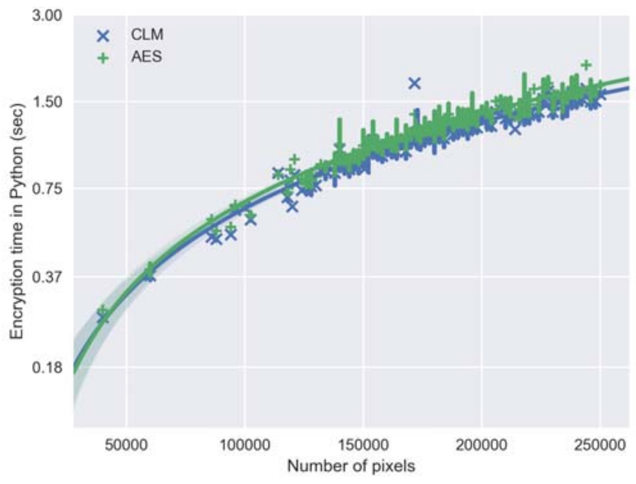


Fig. 4: ImageNet processing time for AES and CLM.

[5] I. Hussain, T. Shah, M. A. Gondal, and H. Mahmood, "An efficient approach for the construction of lft s-boxes using chaotic logistic map," *Nonlinear Dynamics*, vol. 71, no. 1-2, pp. 133–140, 2013.

[6] A. Fenyi, J. G. Davis, and K. Riverson, "Comparative analysis of advanced encryption standard, blowfish and rivest cipher 4 algorithms," *International Journal of Innovative Research and Development*, vol. 3, no. 11, 2014.

[7] S. S. Gupta, S. Maitra, G. Paul, and S. Sarkar, "(non-) random sequences from (non-) random permutationsâĂŤanalysis of rc4 stream cipher," *Journal of Cryptology*, vol. 27, no. 1, pp. 67–108, 2014.

[8] C. Garman, K. G. Paterson, and T. Van der Merwe, "Attacks only get better: Password recovery attacks against rc4 in tls." in *USENIX Security Symposium*, 2015, pp. 113–128.

[9] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.