Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 4, August 2025, pp. 2899~2911
ISSN: 2302-9285, DOI: 10.11591/eei.v14i4.9500

Image encryption algorithm based on a new one-dimensional chaotic map's generator

Mohamed Htiti[1], Ismail Akharraz[2], Abdelaziz Ahaitouf[1]
[1]Laboratory of Engineering Sciences (LSI), Polydisciplinary Faculty of Taza, Sidi Mohamed Ben Abdellah University, Fez, Morocco
[2]Laboratory of Mathematical and Informatics Engineering, Ibnou Zohr University, Agadir, Morocco

Article Info

Article history:

Received Oct 31, 2024
Revised Mar 29, 2025
Accepted May 27, 2025

Keywords:

Asymmetric encryption
Chaotic maps
Confidential keys
Image encryption
Map's generator

ABSTRACT

Encryption plays a crucial role in protecting sensitive data, including communications, financial transactions, and personal information, from cyber threats. One significant area of encryption is image encryption, which ensures the privacy of visual content, such as in secure image transmission, cloud storage, and medical image processing. Recent advancements in image encryption leverage chaotic maps based on chaos theory, generating unpredictable patterns ideal for securing images. This paper presents a novel chaotic map generator that enhances the dynamics of existing chaotic maps. Based on this generator, we propose a new encryption scheme that operates on the entire input image, obscuring the relationship between the original and encrypted images while spreading pixel changes across the entire encrypted image in one step. The scheme also produces an encrypted image of a different size, making it more efficient and resilient to attacks. While some steps of the proposed system are symmetric, others are asymmetric, ensuring a higher level of security. Based on the obtained results, this approach significantly enhances both security and performance in image encryption.

Corresponding Author:

Mohamed Htiti
Laboratory of Engineering Sciences (LSI), Polydisciplinary Faculty of Taza
Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: mohamed.htiti1@usmba.ac.ma

1. INTRODUCTION

In today's digital world, encryption is essential for protecting sensitive data, securing communications, financial transactions, and personal information against cyber threats [1]. One significant application of encryption is image encryption, where visual data is transformed into an unreadable format to prevent unauthorized access. This is used in secure image transmission, cloud storage, and in medical image processing ensuring that sensitive visual content remains private. A recent advancement in image encryption involves chaotic maps, based on chaos theory, which generate unpredictable patterns ideal for secure encryption [2].

Several studies suggest integrating chaotic systems with cryptographic methods to enhance encryption security, such as using Feistel networks with chaotic maps [3], [4], combining advanced encryption standard (AES) with image band scrambling and chaos [5], and employing chaotic maps with the Hill cipher [6]. Other approaches include utilizing polynomial Chebyshev and fractal Tromino methods [7], [8], Fibonacci sequences with AES [9], and genetic algorithms with chaotic maps [10]. For image encryption, advanced methods combining wavelet transform and chaotic maps [11], chaotic maps with DNA coding [12], and Arnold-Tent map with Walsh-Hadamard transforms [13] have been proposed, alongside hyper-chaotic

systems [14] and improvements to existing algorithms like CAST 128-bit with magic squares [15]. In specialized domains, satellite image encryption with chaotic maps and AES [16], medical image encryption [17], and encryption for satellite images during processing and storage [18] have been explored. Most of existing encryption systems process images in blocks and apply confusion and diffusion techniques separately, while also aiming to maintain the encrypted image's size identical to the original. They also apply the same type of encryption, whether symmetric or asymmetric. These approaches can make the systems vulnerable to cryptanalysis attacks.

In this paper, we introduce a novel chaotic map generator that enhances the dynamics of existing chaotic maps. Based on this generator, we introduce a new encryption scheme that processes the entire image, rather than just in blocks. Our approach simultaneously applies confusion and diffusion. Additionally, the scheme produces an encrypted image of a different size. Some steps of our proposed system are symmetric, while others are asymmetric. Furthermore, the integration of chaotic maps introduces unpredictability, which further enhances the system's resistance to cryptanalysis attacks.

This paper is structured as follows: section 2 details our map generator and the proposed encryption scheme. Section 3 covers the validation tests of the generator's chaotic properties and provides security test results and analysis of the proposed encryption system. Section 4 presents the concluding idea.

2. CHAOTIC GENERATOR MAP AND ENCRYPTION SYSTEM

2.1. Proposed chaos map generator

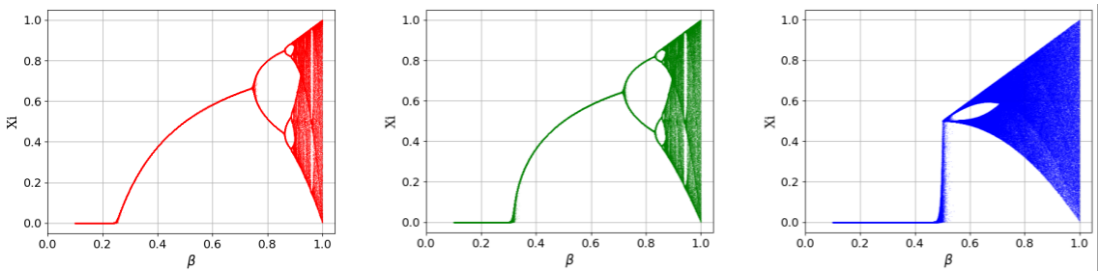
Figures 1 (a) to (c) illustrate two key issues with the logistic, sine, and tent maps. Firstly, chaotic behavior is limited to a small zone. Secondly, the output chaotic sequences have a non-uniform data distribution. To address these issues, this section introduces a new one-dimensional chaotic map generator. The proposed generator leverages previous maps as inputs to construct a new chaotic system. Mathematically, it is defined as (1):

$$x_{n+1} = \left|\cos\left(\alpha\pi g(x_n,\beta)\right)\left(1-\sin\left(\alpha\pi g(x_n,\beta)\right)\right)\right| \quad (1)$$

where $g(x,\beta)$is the input map, β is its control parameter, and α is a hyper-parameter of the generator.

Table 1 displays the mathematical definition of the created maps. With:

- $L(x,\beta)$ is the generated map taking logistic map as input map.
- $S(x,\beta)$ is the generated map taking Sine map as input map.
- $T(x,\beta)$ is the generated map taking Tent map as input map.



(a) (b) (c)

Figure 1. Bifurcation diagram of the; (a) logistic map, (b) sine map, and (c) tent map; for β ϵ [0,1]

Table 1. Generation of chaotic maps using our map generator

$g(x,\beta)$	Abbreviation	Definition
$L(x,\beta)$	Generator_$Lg(x,\alpha,\beta)$	$x_{n+1} = \lvert\cos(4\alpha\pi\beta x_n(1-x_n))(1-\sin(4\alpha\pi\beta x_n(1-x_n)))\rvert$
$S(x,\beta)$	Generator_$Sn(x,\alpha,\beta)$	$x_{n+1} = \lvert\cos(\alpha\pi\beta\sin(\pi x_n))(1-\sin(\alpha\pi\beta\sin(\pi x_n)))\rvert$
$T(x,\beta)$	Generator_$Tn(x,\alpha,\beta)$	$x_{n+1} = \begin{cases} \lvert\cos(2\alpha\pi\beta x_n)(1-\sin(2\alpha\pi\beta x_n))\rvert, & x_n < 0.5 \\ \lvert\cos(2\alpha\pi\beta(1-x_n))(1-\sin(2\alpha\pi\beta(1-x_n)))\rvert, & x_n \geq 0.5 \end{cases}$

2.2. Proposed image encryption and decryption scheme

2.2.2. Encryption scheme

The proposed scheme is shown in Figure 2. In Step 1, the algorithm generates two sequences of integers under 256 that have the same size as the image's array. These sequences are generated using our map generator. Then, the algorithm applies two XOR operations between the specific array seq1 and seq2.

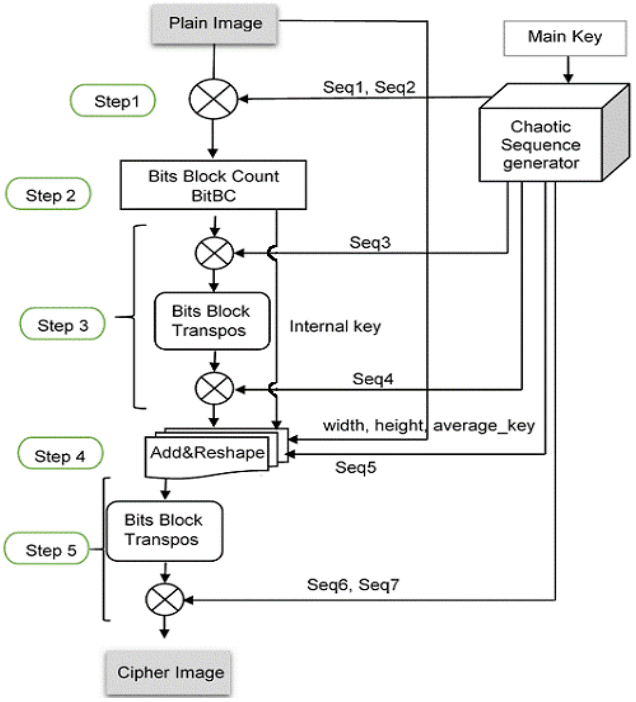


Figure 2. Images encryption process based on our chaotic maps' generator

In Step 2, the array generated in Step 1 is converted into binary form and then split into 9-bit blocks. Without dividing the array into blocks, we might encounter sequences like 23 consecutive bits of the same type (all zeros or all ones). This could confuse the receiver, who might interpret "23" as two ones followed by three zeros, or vice versa. To prevent this, we split the input array into 9-bit blocks. For each block, we count the number of 0s and 1s, saving these counts along with the first bit in a resulting array and an internal key, respectively. This key will be used during the decryption process. The resulting array contains numbers between 1 and 9, which are concatenated to form an array of integers less than 255. This algorithm is called bitBC, which stands for bits block count. Figure 3 provides an example to illustrate this step.

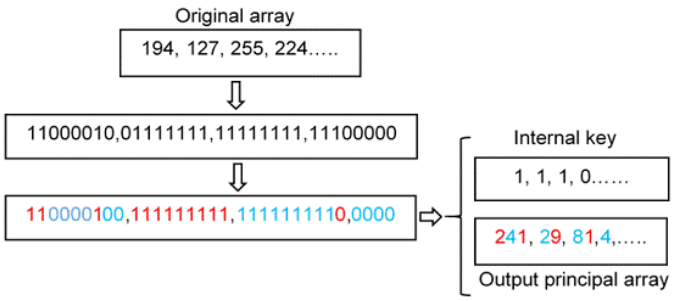


Figure 3. Example of bitBC

Step 3: once the resulting matrix is generated in Step 2, it is XORed with sequence seq3. After performing the XOR operation, the resulting array is converted into binary format and split into blocks of size less than or equal to 64. Each block is then split and resized into an (8,8) shape. Finally, the transpose of each block is taken. We refer to this algorithm as bits block transpose (BBT). The resulting arrays are concatenated and then converted from binary to decimal format. Once the array is in decimal format, it is XORed with a sequence of numbers named seq4.

In Step 4, we add a set of numbers called seq5, which is generated by the key generator. The last array produced in Step 3 has a different size than the original image and, hence, cannot be reshaped into an image shape. To obtain an image shape, we need to add the seq5 to our array. The algorithm hides important information, including the internal key, the original image shape, the size of seq5 (to be deleted in the decryption process), and the average of the main key within seq5.

In Step 5: the matrix obtained from Step 4 is subjected to XOR operations with sequences seq6 and seq7. Before each XOR operation, the BBT algorithm is applied. Subsequently, the array is reshaped into a new dimension. Finally, the resulting output will be the cipher image. The five steps are outlined in Algorithm 1.

Algorithm 1. Encryption scheme

Input: Plain image, main_key
Output: Cipher image
Begin
Step1:
1. Calculate average of main_Key
2. Generate seq1 and seq2 using calculated average to increase key sensitivity
3. Result1=Plain_Image ⊕ seq1
4. Result2=result1 ⊕ seq2

Step2
5. Convert result2 into binary form
6. Split it into blocks of size 9
7. For each block
   - count the number of 0s and 1s
   - save counted number in a resulting_Array
   - save the first bit in an internal_Key
8. The resulting array contains numbers between 1 and 9, which are concatenated to obtain an array of integers under 256
9. Return resulting_Array, internal_Key

Step3
10. Generate seq3, seq4
11. Result4=resulting_Array ⊕ seq3
12. Convert result4 into binary form
13. Split it into blocks of size 64
14. For each block
    - Reshape it into 2D matrix (8,8)
    - Transpose it
    - Append each block in result5
15. Result6=result5 ⊕ seq4

Step4
16. Generate seq5
17. Result7=concatenation (result6, seq5, internal_Key, width, height, average_Key, size of seq5)

Step5
18. Generate seq6 and seq7
19. Convert result7 into binary form
20. Split it into blocks of size 64
21. For each block
    - Reshape it into 2D matrix (8,8)
    - Transpose it
    - Append each block in result8
22. Result9=result8 ⊕ seq6
23. Result10=result9 ⊕ seq7
24. Reshape the resulting array into a new dimension

End

2.2.1. Key and sequences generation

The main key is given as: $main_{key} = [(x_0, \beta, \alpha)_{i=1\rightarrow 7}, c_1, c_2, c_3, c_4, c_5, c_6, c_7]$. Where $(x_0, \beta, \alpha)_{i=1\rightarrow 7} = [x_{01}, \beta_1, \alpha_1, x_{02}, \beta_2, \alpha_2, x_{03}, \beta_3, \alpha_3, \dots\dots\dots x_{07}, \beta_7, \alpha_7]$ are of type float and used as initial values of the Generator map to create the needed chaotic sequences (seq1, seq2, seq3, seq4, seq5, seq6, and seq7). The seven last key elements are of type integer (take as values; 1: Logistic, 2: Sine, or 3: Tent) to specify which original map will be used as input for our map generator. For example: we can generate sequence seq1 using (2). The initial value of x is x_{01}.

$$x_{n+1} = |\cos(4\alpha_1\pi\beta_1 x_n(1-x_n))(1-\sin(4\alpha_1\pi\beta_1 x_n(1-x_n)))| \quad (2)$$

2.2.3. Decryption scheme

The decryption process aims to reverse the encryption and retrieve the plaintext image. In symmetric encryption, the same operations can be applied in reverse for decryption. However, it is important to note that Steps 2 and 4 have specific versions (not symmetric). Figure 4 illustrates the decryption process.

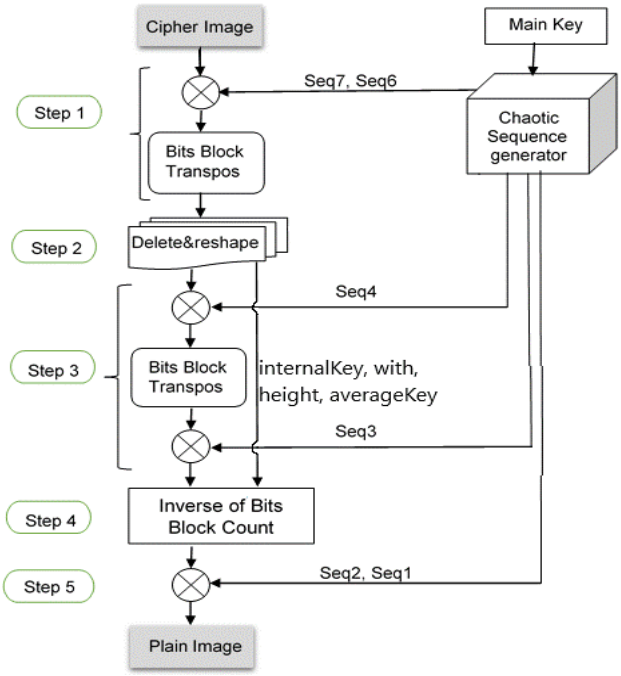


Figure 4. Decryption process

Step 2: in this step, known as "Delete&Reshape", we extract the width and height of the plaintext image along with an internal key that was formed during the encryption process. These hidden elements are retrieved from the encrypted image, and subsequently, the added sequence is removed from the end of the encrypted image, with its size being the difference between the current size and the original size of the image. Afterwards, we reshape the resulting array to match the original size of the plain image. This step provides four essential elements: width, height, the principal array, and the internal key.

Step 4: where we convert each digit in the principal array into a specific number of bits based on the internal key (refer to Figure 5). For example, if the internal key is [1, 1, 1, 0, …] and the principal array is [241, 29, 81, 4, …], we proceed as follows:

- The first item in the internal key is 1. Therefore, we convert:
  Digit "2" to "11"
  Digit "4" to "0000"
  Digit "1" to "1"
  Digit "2" to "00"
- We collect these bits. If the total number of bits is 9, we move to the second item in the internal key, alternating between 0 and 1 since the collected number is 9. Thus:
  Digit "9" converts to "111111111"
  Digit "8" converts to "11111111"
- The third item in the internal key is 1, so we convert:
  Digit "1" to "0"
  Digit "4" to "0000" (since the fourth item in the internal key is 0)

The resulting bit sequence is [1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, …]. When converted to decimal form, it becomes [194, 127, 255, 224, …], which matches the original array shown in Figure 3. The encryption and the decryption process are outlined in Algorithms 1 and 2 respectively.

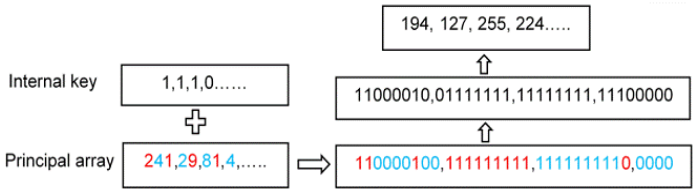


Figure 5. An illustration of step 4 in the decryption process (inverse of biBC)

Algorithm 2. Decryption scheme

Input: Cipher image, main_key
Output: Decrypted image
Begin
Step1:
1. Generate seq6, seq7
2. Result1=Cipher image ⊕ seq7
3. Result2=result1 ⊕ seq6
4. Convert result2 into binary form
5. Split it into blocks of size 64
6. For each block
   – Reshape it into 2D matrix (8,8)
   – Transpose it
   – Append each block in result3

Step2
7. Extract internal_Key, width, height, average_Key and size of seq5
8. Delete internal_Key, width, height, average_Key and seq5
9. Reslult4=result3-(internal_Key, width, height, average_Key and size of seq5)

Step3
10. Generate seq4, seq3
11. Result5=result4 ⊕ seq4
12. Convert result5 into binary form
13. Split it into blocks of size 64
14. For each block
   – Reshape it into 2D matrix (8,8)
   – Transpose it
   – Append each block in result6
15. Result7=result6 ⊕ seq3

Step4
16. Convert result7 to integer form number (Each number has one, two or three digit)
17. we convert each digit into ones or zeros, depending on the content of the internal_Key

Step5
18. Generate seq2, seq1 using average_Key
19. Result9=result8 ⊕ seq3
20. Result10=result9 ⊕ seq3
21. Reshape the resulting array into the dimension (width, height).

End

3. RESULTS AND DISCUSSION

3.1. Analysis of the proposed chaotic map generator

3.1.1. Bifurcation diagram and trajectory analysis

To evaluate the chaotic behavior of our map generator, we use the bifurcation diagram. Figure 6 shows the bifurcation diagrams corresponding to traditional chaotic mappings and their respective improved iterations. Figures 6(a) to (c) show the bifurcation diagrams for the Logistic, Sine, and Tent maps, respectively. With β in the range [0,1] and α=560, these diagrams show the evolution of the values of Xi, thus highlighting areas of stability, periodicity, and chaotic behavior. Figures 6(d) to (f) show the bifurcation diagrams of the improved above-mentioned maps. These improvements were made to enhance dynamic

complexity and the range of chaotic behavior. The resultant diagrams show a more densely packed and uniform distribution of values, which is especially beneficial for applications in fields like cryptography and pseudo-random number generation.

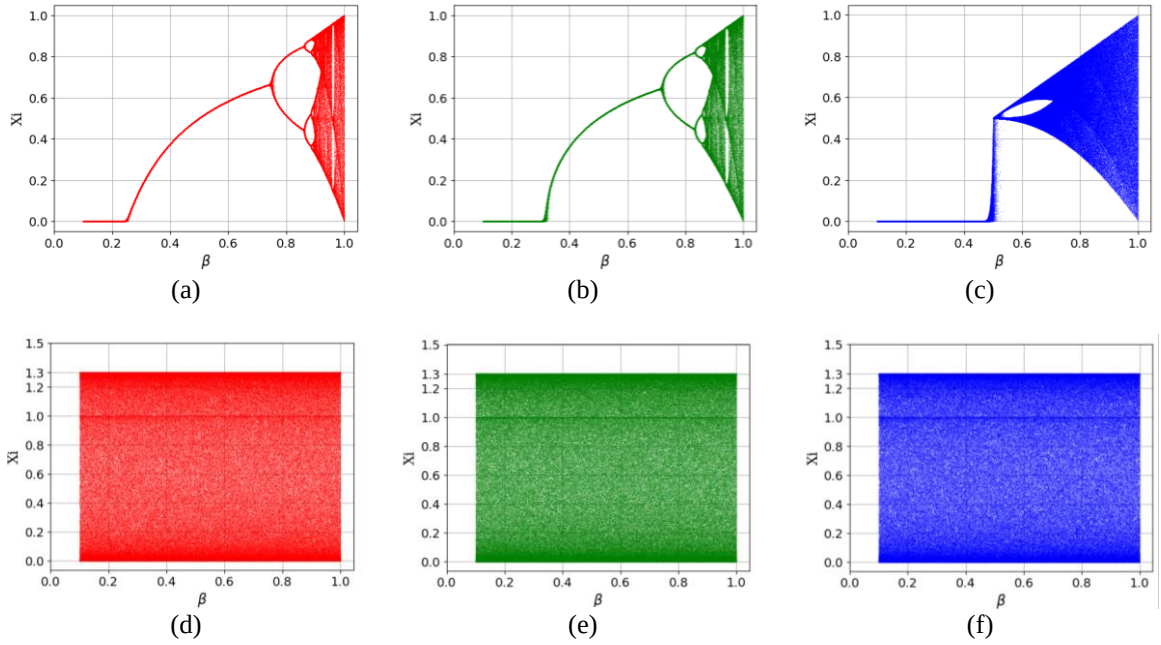


Figure 6. Bifurcation diagram of the; (a) logistic map, (b) sine map, (c) tent map, (d) generator-logistic map, (e) generator-sine map, and (f) generator-tent map for β ϵ [0,1] and α=560

3.1.2. Sensitivity analysis

The "Lyapunov Exponent" concept has been widely used in computer science, especially to study dynamical systems and prove whether they are chaotic or not. If our system can be described by $F(x_i)$, the Lyapunov exponent can be mathematically defined as (3):

$$LE = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln|F'(x_i)| \quad (3)$$

Figure 7 shows that our map generator has a larger chaotic range compared to its input map in all three cases. The Lyapunov exponent of the existing maps is below 1 in the best case and below zero in most cases. However, the improved version ranges from 4 to 8.

3.1.3. Sample entropy

We use sample entropy to measure the randomness of our one-dimensional chaotic map generator. A large value of sample entropy means that the maps have better chaos. For a time series X of size $N(x_1, x_2, x_3, \dots\ x_N)$, we can generate vectors with the size of m and m+1.

$$U_m(i) = (x_i, x_{i+1}, x_{i+2}, \dots \dots x_{i+m-1}) , U_{m+1}(i) = (x_i, x_{i+1}, x_{i+2}, \dots \dots x_{i+m}) \quad (4)$$

$$V_m(j) = (x_j, x_{j+1}, x_{j+2}, \dots \dots x_{j+m-1}) , V_{m+1}(j) = (x_j, x_{j+1}, x_{j+2}, \dots \dots x_{j+m}) \quad (5)$$

where: $d_m(U(i), V(j))$ is the Cheybyshev distance between U_m and V_m.

The sample entropy equation is defined as (6):

$$SE(m, r, N) = -\log \frac{A}{B} \quad (6)$$

where A is the number of vectors U(i) that satisfy the following condition $d_{m+1} < r$, and B is the number of vectors U(i) that satisfy the following condition $d_m < r$, r is the acceptable tolerance. We set m=2 and $r = 0.2 \times std(X)$ to follow the recommendations provided in [19]. Based on the results obtained, it appears

that our map generator has demonstrated an improvement in entropy for the three input maps. The entropy value approaches 2, which is notably higher compared to the maximum value of 0.75 observed in the input maps (logistic, sine, and tent). This means that our map generator exhibits a higher degree of chaos, as shown in Figure 8.

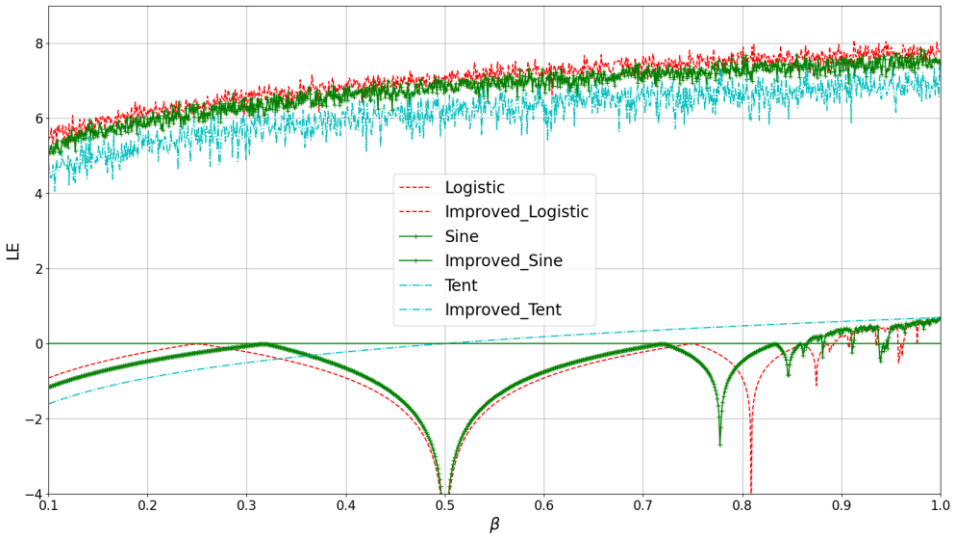


Figure 7. LE evaluation results: improved_logistic; improved _Sine; improved_Tent and input maps for $\alpha = 560$

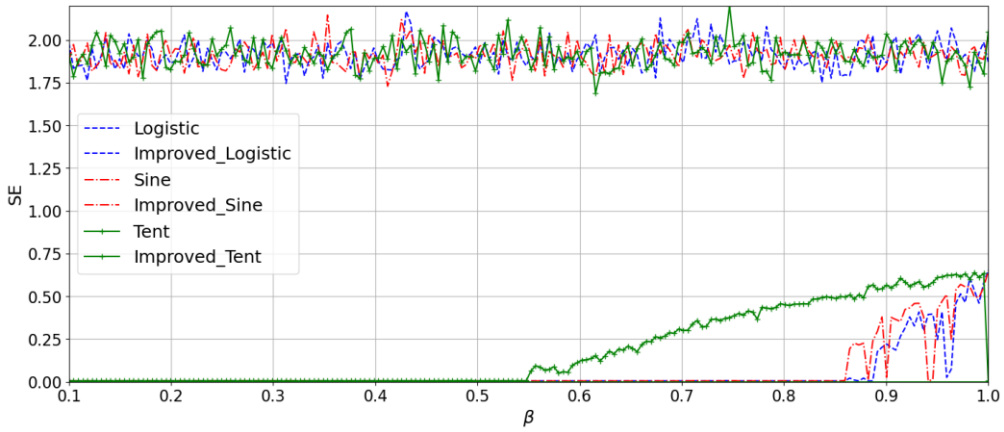


Figure 8. SE evaluation results, generator-logistic; generator-Sine; generator-Tent generated and input maps for $\alpha = 560$

3.2. Encryption system analysis

In this section, we will present the validation tests for our proposed encryption system. The test set includes images from the USC-SIPI miscellaneous dataset 1, as well as other standard images that are commonly used.

3.2.1. Key space analysis

The main key used in our system consists of 21 float numbers and 7 integers; therefore, our algorithm has a key length of 357 bits, resulting in a key space size of 2^{357}, which is much larger than the theoretical limit of 2^{100} for a brute force attack [20].

$$main_{key} = [(x_0, \beta, \alpha)_{i=1\rightarrow 7}, c_1, c_2, c_3, c_4, c_5, c_6, c_7]$$

The brute force attack method involves trying every possible key while the correct one is not found; as a result, the key space needs to be sufficiently large to prevent a brute force attack from being successful in a reasonable amount of time, thereby ensuring the overall security of the encryption system. Table 2 provides a comparative analysis of the key space characteristics among various encryption algorithms and our model.

Table 2. Key space evaluation comparison

Scheme	Ref [21]	Ref [22]	Ref [23]	Our system
Key space	2^{140}	2^{128}	2^{224}	2^{357}

3.2.2. Histogram analysis

Based on the findings in Figure 9, our encryption process successfully produces a cipher image with a uniform histogram from the plain image. Various types of images, including grayscale and RGB, were subjected to experimentation. Plain images, histograms of plain images, cipher images, histograms of cipher images, and deciphered images are presented in the first to fifth columns, respectively. The findings show that the process appropriately encrypts, and decrypts plain_images and prevents statistical attacks.

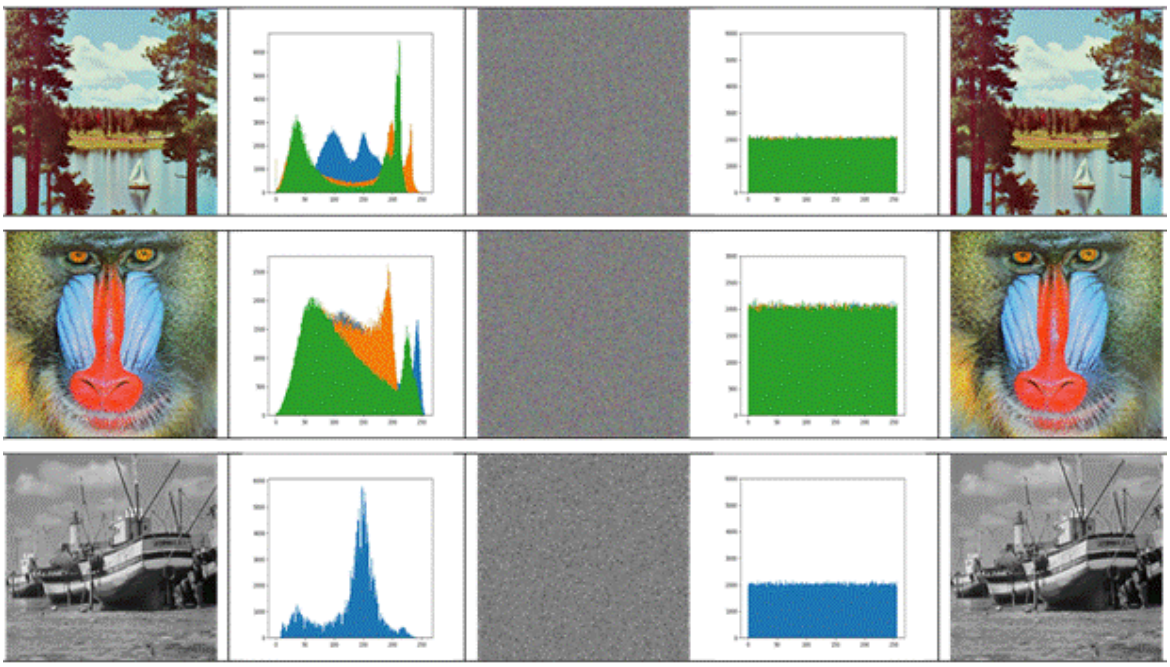


Figure 9. Analysis of the histogram for the proposed image encryption and decryption

3.2.3. Information entropy analysis

Information entropy is a useful tool that can be used to quantify the level of randomness that is responsible for determining the images level of security. It can be mathematically represented as (7) [24]:

$$H(m) = -\sum_{i=0}^{L-1} p(m_i) log_2(p(m_i)) \quad (7)$$

where m_i represents the pixel value ranging from 0 to 255, and $p(m_i)$ indicates the probability of the gray value m_i. For an 8-bit grayscale image where L equals 256, the optimal value for information entropy (H) is 8. This means that when the value of H approaches 8, there is an increase in the degree of randomness seen in the image pixel distribution. The local shannon entropy (LSE) is a measure of the average entropy of k randomly selected non-overlapping blocks. Where P_i represents non-overlapping blocks, and T_b represents the number of pixels in each block for a given image.

$$\overline{H_{k,T_b}}(Im) = \sum_{i=0}^{k} H(P_i)/k \quad (8)$$

This test was done by Wu *et al*. [24] on 25 8-bit gray-scale images from the USC-SIPI "Miscellaneous" dataset. The test was performed with k=30, T_b=1936, and $\alpha = 0.001$ significance level. An

image can pass the test if its $H_{k,Tb}$ value falls within the range (7.901515698, 7.903422936), with the ideal value being 7.902469317. Table 3 compares the proposed algorithm to three existing research studies. The global entropy of the proposed method consistently achieves values close to the ideal maximum of 8, with results ranging from 7.998437 to 7.998832 across different images, demonstrating effective randomness in the encrypted images. The local entropy for the proposed method also shows strong performance, with an average pass rate of 4/6, outperforming the three cited references. The standard deviation for local entropy (0.0009907) is the lowest among all methods, reflecting the method's high stability and consistency. The significant improvement in both global and local entropy compared to the references confirms its effectiveness in preventing unauthorized information extraction and resisting cryptanalysis attacks.

Table 3. The local and global information entropy of the "Miscellaneous" image dataset

File name	Global entropy		Local entropy (k, T_b, α)=(30,1936,0.001)				
	Plain image	Proposed	Plain image	Ref [25]	Ref [26]	Ref [27]	Proposed
5.1.09	6.70931	7.998577	6.380955	7.90272	7.90192	7.90093	7.90338251
5.1.10	7.3118	7.998832	7.086833	7.90061	7.90007	7.89976	7.90272544
5.1.11	6.45227	7.9986	5.331843	7.90142	7.90077	7.90105	7.90264014
5.1.12	6.70566	7.998655	5.498883	7.90381	7.902	7.90307	7.90463714
5.1.13	1.54831	7.998437	1.494765	7.90505	7.90104	7.90107	7.90343722
5.1.14	7.34243	7.998594	6.808383	7.903	7.90352	7.90509	7.90282400
Pass rate	--	--	--	2/6	2/6	1/6	4/6
Std	--	--	--	0.00114	0.00106	0.00149	0.0009907

3.2.4. Differential attack analysis

This method encrypts two plain-images with the same key and compares their cipher images' differences. High sensitivity makes the system more resistant to known plain-image attacks. Only one pixel in the original image is modified by adding or subtracting 1, and then the cipher images, C_1 and C_2, are compared. Two widely used metrics for measuring the resistance of cryptographic systems to differential attacks are the number of pixel change rate (NPCR) and the unified average change intensity (UACI). NPCR compares whether the pixel values at the two images corresponding positions are the same, while UACI compares their differences. In (9) calculates these metrics:

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M.N}.100\%, UACI = \frac{1}{M.N}\sum_{i,j}\frac{|C_1(i,j)-C_2(i,j)|}{L}.100\% \quad (9)$$

M and N represent the images height and width, respectively, and L represents the maximum 8-bit pixel value, L=255. The difference between C_1 and C_2 is denoted by (i, j):

$$D(i,j) = \begin{cases} 1 \; if \; C_1(i,j) \neq C_2(i,j) \\ 0 \; if \; C_1(i,j) = C_2(i,j) \end{cases} \quad (10)$$

An excellent encryption system should successfully meet the NPCR and UACI standards [28]. The critical value of the NPCR, at a given level of significance α, is calculated as (11):

$$N_\alpha^* = \frac{L-\phi^{-1}(\alpha)\sqrt{L/MN}}{L+1} \quad (11)$$

The inverse cumulative density function (CDF) of a standard normal distribution N (0, 1) is ϕ^{-1}. For an encrypted image to pass the NPCR test, its NPCR value must exceed N_α^*. In (9) define the critical values for the UACI test (12) and (13):

$$U_\alpha^{*-} = \mu_u - \phi^{-1}\left(\frac{\alpha}{2}\right)\sigma_u, U_\alpha^{*+} = \mu_u + \phi^{-1}(\frac{\alpha}{2})\sigma_u \quad (12)$$

$$\mu_u = \frac{L+2}{3L+3}, \sigma_u = \frac{(L+2)(L^2+2L+3)}{18(L+1)^2LMN} \quad (13)$$

If cipher-image UACI values are within the range $\left(U_\alpha^{*(-)}, U_\alpha^{*(+)}\right)$, it is considered to have passed the UACI test. Table 4 shows the UACI and NPCR results. The average UACI and NPCR values were 33.38% and 99.61%, respectively. According to the findings of Liu and Miao [25], the values presented in this study closely match the expected values for grayscale images, specifically 33.464% for UACI and 99.609% for NPCR.

Table 4. Comparison of "Miscellaneous" image dataset NPCR and UACI values

File name	NPCR: N_α^*≥99.5693				UACI: $(U_\alpha^{*(-)}, U_\alpha^{*(+)})$=(33.2824, 33.6447)			
	Proposed	Ref [23]	Ref [27]	Ref [19]	Proposed	Ref [23]	Ref [27]	Ref [19]
5.1.09	99.60897	99.6078	99.6016	99.6064	33.5504	33.4563	33.47	33.4456
5.1.10	99.60943	99.6098	99.6191	99.6154	33.43374	33.451	33.4826	33.4946
5.1.11	99.59903	99.6077	99.6042	99.6244	33.47071	33.4832	33.5648	33.5541
5.1.12	99.61019	99.6033	99.5956	99.5703	33.50509	33.4538	33.4725	33.4302
5.1.13	99.61586	99.6066	99.6109	99.6109	33.54058	33.4363	33.4813	33.4438
5.1.14	99.59715	99.6129	99.6199	99.6364	33.45738	33.4848	33.4725	33.4655
Std	0.007525	0.00376	0.00527	0.01364	0.033581	0.02298	0.02421	0.0342

The proposed method demonstrates excellent performance in terms of NPCR and UACI, key metrics for evaluating image encryption robustness. The NPCR values for the proposed method consistently exceed or match those of reference methods [19], [23], [27] across all test images, with a notably high value of 99.61586 for image "5.1.13." The low standard deviation (0.007525) highlights its stability and reliability. Compared to the researcher [23] shows slightly lower variability but generally underperforms in NPCR, while [19] exhibits the highest variability. These results confirm the proposed method's superior ability to achieve significant pixel changes during encryption.

In UACI analysis, the proposed method achieves competitive intensity change rates, ranging from 33.43374 to 33.5504, with moderate variability (standard deviation=0.033581). Wang *et al*. [27] occasionally outperforms the proposed method in specific cases (e.g., image "5.1.11"), the proposed method maintains consistent performance and outmatches Hua *et al*. [19] in stability. Overall, the proposed method's strong NPCR and UACI values affirm its capability to ensure secure and reliable image encryption, proving it to be a robust choice for enhancing image security.

4. CONCLUSION

In this paper, we presented a novel encryption scheme based on a proposed chaotic map generator, which offers significant improvements in both security and efficiency for image encryption. Our findings demonstrate that by integrating both symmetric and asymmetric steps, the proposed system successfully obfuscates the relationship between the original and encrypted images, while ensuring that even small changes in a single pixel influence the entire encrypted image. This design enhances resistance to cryptanalysis attacks, making the system more robust than traditional encryption methods.

The implications of this work extend beyond its theoretical contribution, as our approach provides a practical solution for secure image transmission and storage, particularly in fields such as medical imaging, cloud storage, and satellite imagery. The ability to generate encrypted images of a different size further enhances the system's efficiency, increasing its complexity. Future work could focus on optimizing the system for real-time processing and exploring its potential in multi-layer encryption schemes. Additionally, the proposed chaotic map generator could be applied to other maps, specifically two-dimensional maps.

ACKNOWLEDGEMENTS

Special appreciation is owed to the Laboratory of Engineer Sciences for providing the necessary facilities and resources.

FUNDING INFORMATION

Authors state there is no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Mohamed Htiti	✓	✓		✓	✓				✓	✓				
Ismail Akharraz		✓								✓	✓	✓	✓	
Abdelaziz Ahaitouf		✓								✓	✓	✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available on request from the corresponding author, [M. H.].

REFERENCES

[1] M. Kaur and V. Kumar, "A comprehensive review on image encryption techniques," *Archives of Computational Methods in Engineering*, vol. 27, no. 1, pp. 15-43, 2020, doi: 10.1007/s11831-018-9298-8.

[2] U. Zia *et al.*, "Survey on image encryption techniques using chaotic maps in spatial, transform and spatiotemporal domains," *International Journal of Information Security*, vol. 21, no. 4, pp. 917-935, 2022, doi: 10.1007/s10207-022-00588-5.

[3] X. Liu, Y. Song, and G.-P. Jiang, "Hierarchical Bit-Level Image Encryption Based on Chaotic Map and Feistel Network," *International Journal of Bifurcation and Chaos*, vol. 29, no. 02, 2019, doi: 10.1142/s0218127419500160.

[4] E. Winarno, W. Hadikurniawati, K. Nugroho, and V. Lusiana, "Integrating Quadratic Polynomial and Symbolic Chaotic Map-based Feistel Network to Improve Image Encryption Performance," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3436558.

[5] S. Inam, S. Kanwal, R. Firdous, K. Zakria, and F. Hajjej, "A new method of image encryption using advanced encryption Standard (AES) for network security," *Physica Scripta*, vol. 98, no. 12, p. 126005, 2023, doi: 10.1088/1402-4896/ad0944.

[6] Y. Zheng, Q. Huang, S. Cai, X. Xiong, and L. Huang, "Image encryption based on novel Hill Cipher variant and 2D-IGSCM hyper-chaotic map," *Nonlinear Dynamics*, vol. 113, no. 3, pp. 2811-2829, 2025, doi: 10.1007/s11071-024-10324-4.

[7] M. Khan, A. S. Alanazi, L. S. Khan, and I. Hussain, "An efficient image encryption scheme based on fractal Tromino and Chebyshev polynomial," *Complex & Intelligent Systems*, vol. 7, no. 5, pp. 2751-2764, 2021, doi: 10.1007/s40747-021-00460-4.

[8] N. Louzzani, A. Boukabou, H. Bahi, and A. Boussayoud, "A novel chaos based generating function of the Chebyshev polynomials and its applications in image encryption," *Chaos, Solitons & Fractals*, vol. 151, 2021, doi: 10.1016/j.chaos.2021.111315.

[9] C. Maiti, B. C. Dhara, S. Umer, and V. Asari, "An efficient and secure method of plaintext-based image encryption using Fibonacci and tribonacci transformations," *IEEE Access*, vol. 11, pp. 48421-48440, 2023, doi: 10.1109/ACCESS.2023.3276723.

[10] M. Ghazvini, M. Mirzadi, and N. Parvar, "A modified method for image encryption based on chaotic map and genetic algorithm," *Multimedia Tools and Applications*, vol. 79, no. 37, pp. 26927-26950, 2020, doi: 10.1007/s11042-020-09058-3.

[11] H. Gao and W. Zeng, "Image compression and encryption based on wavelet transform and chaos," *Computer Optics*, vol. 43, no. 2, pp. 258-263, 2019, doi: 10.18287/2412-6179-2019-43-2-258-263.

[12] H. Wen and Y. Lin, "Cryptanalysis of an image encryption algorithm using quantum chaotic map and DNA coding," *Expert Systems with Applications*, vol. 237, p. 121514, 2024, 10.1016/j.eswa.2023.121514.

[13] P. Sneha, S. Sankar, and A. S. Kumar, "A chaotic colour image encryption scheme combining Walsh–Hadamard transform and Arnold–Tent maps," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 1289-1308, 2020, doi: 10.1007/s12652-019-01385-0.

[14] H. Chen, E. Bai, X. Jiang, and Y. Wu, "A fast image encryption algorithm based on improved 6-D hyper-chaotic system," *IEEE Access*, vol. 10, pp. 116031-116044, doi: 10.1109/ACCESS.2022.3218668, 2022.

[15] S. M. Kareem, A. Al-Adhami, and A. M. S. Rahma, "An improvement for CAST-128 encryption based on magic square and matrix inversion," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 13, no. 1, pp. 377-387, 2024, doi: 10.11591/eei.v13i1.5340.

[16] E. Bensikaddour and Y. Bentoutou, "Satellite image encryption based on AES and discretised chaotic maps," *Automatic Control and Computer Sciences*, vol. 54, no. 5, pp. 446-455, 2020, 10.3103/S014641162005003X.

[17] V. Kumar, V. Pathak, N. Badal, P. S. Pandey, R. Mishra, and S. K. Gupta, "Complex entropy based encryption and decryption technique for securing medical images," *Multimedia Tools and Applications*, vol. 81, no. 26, pp. 37441-37459, 2022, doi: 10.1007/s11042-022-13546-z.

[18] M. A. S. Al-Khasawneh, M. Faheem, E. A. Aldhahri, A. Alzahrani, and A. A. Alarood, "A MapReduce based approach for secure batch satellite image encryption," *IEEE Access*, vol. 11, pp. 62865-62878, 2023, doi: 10.1109/ACCESS.2023.3279719.

[19] Z. Hua, Y. Zhou, and H. Huang, "Cosine-transform-based chaotic system for image encryption," *Information Sciences*, vol. 480, pp. 403-419, 2019, doi: 10.1016/j.ins.2018.12.048.

[20] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos*, vol. 16, no. 08, pp. 2129-2151, 2006, doi: 10.1142/S0218127406015970.

[21] W. Zhang, H. Yu, and Z.-l. Zhu, "An image encryption scheme using self-adaptive selective permutation and inter-intra-block feedback diffusion," *Signal Processing*, vol. 151, pp. 130-143, 2018, doi: 10.1016/j.sigpro.2018.05.008.

[22] C. Pak and L. Huang, "A new color image encryption using combination of the 1D chaotic map," *Signal Processing*, vol. 138, pp. 129-137, 2017, doi: 10.1016/j.sigpro.2017.03.011.

[23] X. Wang and M. Zhao, "An image encryption algorithm based on hyperchaotic system and DNA coding," *Optics & Laser Technology*, vol. 143, p. 107316, 2021, doi: 10.1016/j.optlastec.2021.107316.

[24] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Information Sciences*, vol. 222, pp. 323-342, 2013, doi: 10.1016/j.ins.2012.07.049.
[25] L. Liu and S. Miao, "A new simple one-dimensional chaotic map and its application for image encryption," *Multimedia Tools and Applications*, vol. 77, pp. 21445-21462, 2018, doi: 10.1007/s11042-017-5594-9.
[26] H. Diab, "An efficient chaotic image cryptosystem based on simultaneous permutation and diffusion operations," *IEEE access*, vol. 6, pp. 42227-42244, 2018, doi: 10.1109/ACCESS.2018.2858839.
[27] W. Wang *et al.*, "An encryption algorithm based on combined chaos in body area networks," *Computers & Electrical Engineering*, vol. 65, pp. 282-291, 2018, doi: 10.1016/j.compeleceng.2017.07.026.
[28] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, vol. 1, no. 2, pp. 31-38, 2011.

BIOGRAPHIES OF AUTHORS

Mohamed Htiti received the B.Sc. degree in nuclear physics from Sidi Mohamed Ben Abdallah University, in 2002. Received the M.Sc. degree in physics and nuclear techniques from Cadi Ayyad University, in 2005. Received the B.Sc. degree in IT engineering from Poly disciplinary Faculty of Taza, Sidi Mohamed Ben Abdallah University, in 2020. Received the M.Sc. degree in intelligent and mobile systems from the poly disciplinary Faculty of Taza, Morocco, in 2023. Now he is a Ph.D. student registered in the LSI Laboratory of Engineer Sciences of the Poly disciplinary Faculty of Taza, Sidi Mohamed Ben Abdallah University, Fez, Morocco. His research interests include cryptography, steganography, deep learning, and image processing. He can be contacted at email: mohamed.htiti1@usmba.ac.ma.

Ismail Akharraz received his Master and Ph.D. degrees in number theory from the University of Sidi Mohmed Ben Abdellah Fez, Morocco, in 2000. From 2003 to 2020, he was at the University of Sidi Mohamed Ben Abdellah Fez, as permanent professor and permanent member of the laboratory of engineering sciences. From 2021, he joined Ibn Zohr University, in Agadir Morocco, as a permanent professor and permanent member of the Laboratory of Mathematical and Informatic Engineering. His current areas of research are error-correcting codes and cryptography, intelligent systems, and recommendation systems. He can be contacted at email: ismail.akharraz@usmba.ac.ma.

Abdelaziz Ahaitouf received his physics diploma at the University of Moulay Ismail Meknes. From 1995 to 1999, he received his MD and Ph.D. degrees in electronics from the University of Metz, France. In 2000, he worked in a postdoctoral position on the development of a SOI fully and partially depleted process at the Swiss Federal Institute of Technology (EPFL), Switzerland. From 2003, he joined the University of Sidi Mohamed Ben Abdellah Fez, Morocco where he is teaching in the field of electronic and IC manufacturing. He is currently working in the field of microelectronics, electrical device characterization, intelligent systems, and LDPC encoding/decoding. He can be contacted at email: abdelaziz.ahaitouf@usmba.ac.ma.