Chaos-Based Image Encryption Based on Bit Level Cubic Shuffling

Lazaros Moysis, Ioannis Kafetzis, Aleksandra Tutueva, Denis Butusov, and Christos Volos

Abstract This work studies the problem of chaos-based image encryption. First, a generalization of the 1D chaotic map proposed by (Talhaoui et al. in The Visual Computer, pp 1–11, 2020) [8] is constructed and studied. The generalized map showcases regions of constant chaotic behaviour, similar to the original map. Based on the new map, a statistically secure pseudo-random bit generator is designed, which is utilised in the encryption process. An image encryption technique based on shuffling the bit levels of an image is introduced, by first arranging the bits in a three dimensional matrix, and performing a three level shuffling, on each individual row, column, and bit level of the 3D matrix. The shuffling is then followed by an exclusive OR operation between the shuffled bits and a bitstream from the proposed chaotic bit generator, which results in the encrypted image. The combination of shuffling and XOR yields a ciphertext image that is resistant to a collection of attacks, like histogram, correlation, and entropy analysis, NPCR and UACI measures, cropping attacks, and is also robust to transmission noise. This is verified by testing the encryption process to a collection of plaintext images. Finally, the encryption/decryption process is implemented in a Graphical User Interface for ease of use.

L. Moysis (✉) · I. Kafetzis · C. Volos
Laboratory of Nonlinear Systems - Circuits & Complexity, Physics Department, Aristotle University of Thessaloniki, Thessaloniki, Greece
e-mail: lmousis@physics.auth.gr; moysis.lazaros@hotmail.com

I. Kafetzis
e-mail: kafetzis@physics.auth.gr

C. Volos
e-mail: volos@physics.auth.gr

A. Tutueva · D. Butusov
Youth Research Institute, Saint-Petersburg Electrotechnical University 'LETI', 5, Professora Popova st., 197376 Saint Petersburg, Russia
e-mail: avtutueva@etu.ru

D. Butusov
e-mail: dnbutusov@etu.ru

A. A. Abd El-Latif and C. Volos (eds.), *Cybersecurity*, Studies in Big Data 102,
https://doi.org/10.1007/978-3-030-92166-8_7

1 Introduction

Chaos-based cryptography is a well established field in the discipline of information security, with applications spanning secure communications [1], watermarking [2], hashing [3], random bit generators [4], and different types of data masking, like text, sound and image [5, 6].

Chaotic systems combine determinism and unpredictability, with a low computational cost, which makes them an excellent source of randomness, for use in encryption related applications. This is why there is a constant need to develop novel chaotic systems, and especially low dimensional discrete time maps, that can yield large regions of chaotic behaviour [7–10].

Motivated by this, a novel map is constructed, as a generalization of the single-parameter one-dimensional cosine polynomial (1-DCP) chaotic map proposed in [8]. The original map combines a cubic polynomial with the cosine function, to yield a map with complex chaotic behaviour. The map showcased large regions of chaotic behavior, which makes it suitable for use in encryption designs. On the other hand, the map only had a single parameter, so its key space was not sufficient, to resist brute force attacks. These two properties motivated the consideration of a modified version of this map.

Hence, the map proposed in the current work is a generalization of the (1-DCP) map, with a second parameter added inside the cosine function. The new map has a larger key space, since it consists of two parameters. The analysis of the map is performed through computation of its bifurcation diagrams and Lyapunov exponent, and it is seen that the map has regions of constant chaotic behaviour, which is desired for encryption related applications. Also, the phase diagram for large parameter values has no distinguishable shape, which is an additional desired property for cryptographic applications. Moreover, the computational load of the new map is practically the same to the original, since they differ by a single operation of addition.

Based on the proposed map, a statistically secure pseudo-random bit generator (PRBG) is developed. PRBGs are a very common application for chaotic systems, since they are used as a basis in most encryption schemes. For example, in a recent work [4] a PRBG is proposed based on a delayed Chebysev map, that can generate 8 bits per iteration. In [11], another PRBG that generates 8 bits per iteration is proposed, based on a modified logistic map. In [12], the technique of bit reversal is applied to increase complexity in the PRBG. In [13], adaptive chaotic maps are proposed that can be used to effectively increase the key space of a PRBG. The literature on PRBGs is extensive, so for an in depth presentation, the recent survey [14] considers a large set of research works, and compares their performance. The PRBG constructed here applies commonly used techniques to generate the bits, like the modulo 2 operator. One difference from other methods though is the use of a delay term in the bit generation, which can increase the generator's randomness.

After the PRBG is designed, the problem of image encryption is considered. The application of chaotic systems in image encryption is a well established and expanding research area, with many developed techniques. For a guide, the survey [6] reviews different approaches to chaos-based image encryption. As recent examples of diverse approaches to image encryption, in [15] a technique was proposed based on Barnsleyćs chaos game, and a Graphical User Interface was also developed. In [9] a two step image encryption was proposed based on a novel chaotic map, where the pixels are first shuffled, and then modulated with the values of the chaotic map. In [16], the technique of DNA encoding was applied to medical images. In [17], a continuous chaotic system was used to design an S-box for image encryption. In [18], a continuous hyperchaotic system was applied for image encryption, in combination with a zigzag operation to scramble the image pixels.

From the above works, it can be concluded that the best chaos encryption schemes need to combine the operations of confusion and diffusion. Confusion is the process of hiding the connection between the bits of the original (plaintext) image and the encrypted (ciphertext) image. Confusion can be achieved is through substitution of the image's binary information. The most common operator to achieve this is the exclusive OR operator (XOR) and this is where chaos based PRBGs are utilized, as they can be combined with the plaintext image bits to mask them. Diffusion is the operation of making the encrypted image sensitive to changes in the plaintext image. A way to achieve diffusion is to make the encryption keys plaintext dependent. Moreover the proccess of rearranging the pixels of the plaintext image is beneficial in reducing pixel correlation, and also spreading out the key pixel information throughout the image. This can be achieved by generating shuffling rules chaotically. An even more advanced approach is to perform this operation on the binary level, across all binary levels of the image's pixels. This process not only reduces correlation, but also rearranges the most significant bits of each pixel across all bit levels and pixel positions, which can make the encryption much more secure, and also resistant to loss of information or corruption during the transmission stage.

Motivated by the above key aspects of encryption, a method is developed for chaotic encryption, giving an emphasis to the process of thoroughly permuting the binary information of the plaintext image. In the technique proposed in this work, a rearranging of the bits across all rows, columns, and bit levels of the matrix is performed. Such an approach can yield increased security, as well as resistance to noise and cropping attacks, since the bits that carry most of the image information are redistributed across all bit levels. This is why many recent techniques consider permutation and encryption on the bit levels of a plaintext image. As recent examples, in [19–22] the image pixels are shuffled to reduce correlation in the image. In [23] a technique for colour images was proposed, through a combination of a continuous and a discrete chaotic system, to perform a combined shifting of the pixels and bits of its three RGB channels. In [24], a modified pulsed-coupled spiking neurons circuit map is used to encrypt an image, using a novel technique of breaking each pixel into four bit pairs, and rearranging them to form a new matrix of double size compared to the original image. In [25], colour image encryption is performed by simultaneously shuffling the bit levels of all three colour sub-images, while again both discrete and continuous systems are used for the complete encryption process. In [26], a self-adaptive bit level encryption is proposed, where an image is broken down into parts, and one part is used to encrypt the other. In [27], quadratic and cubic maps are used to shuffle the bits and encrypt a chosen number of bit levels. In [28], bit level permutation is performed based on the popular Rubik's cube game. In [29], shuffling is performed using the Hilbert curve pattern and cyclic shift.

The procedure developed here consists of two main steps. First, the pixels of a given plaintext image are decomposed into 8 bits and arranged in a 3D matrix. Then, each individual row level, column level, and pixel level of the 3D matrix is shuffled. The shuffling rules for each level are generated from the values of three different chaotic maps. The rearrangement of the bits results in the dispersion of the significant bits of the plaintext across all the eight pixel levels and all the rows and columns of the image. This yields a shuffled image with no distinguishable information, as the value of each pixel after the shuffling is changed. Moreover, in addition to showing security against attacks, like correlation and entropy analysis, such a thorough bit permutation makes the image resistant to loss of information and noise, as will be seen in the simulations section. The trade-off for achieving this is an increased cost in computational time.

Moreover, since only rearranging the bits of the plaintext cannot yield security against specific attacks, like known plaintext attacks, a second step of confusion is added in the design. Here, the bits of the shuffled image are combined with a chaotic bitstream of the same length, generated from the proposed PRBG, using the XOR operation. This results in a ciphertext image that is secure against known plaintext attacks as well.

Also, to further improve the resistance to known plaintext attacks, the encryption keys used are plaintext-dependent. Two general approaches for this is to either use a custom rule to generate the keys based on the pixel values of the image [30–35], or to use hash functions [16, 36–40]. Here, the first approach is taken, so the parameter values and initial conditions of the maps used for the shuffling and PRBG are computed based on the plaintext image to be encrypted.

To verify the security of the design, the encryption is performed to a collection of plaintext images, which are then submitted to a series of statistical tests and measures. These include histogram analysis, correlation analysis, global and local information entropy measure, differential attack analysis, cropping and noise attack analysis, senstitivity analysis, as well as resistance to brute force attacks. All of the performed tests verified that the design is indeed secure to all the types of attacks considered.

Finally, the proposed scheme is implemented with a Graphical User Interface (GUI), which is a valuable tool for easily implementing and visualising the procedure. This tool can be picked up by any interested reader that can use the proposed design to encrypt personal data.

Overall, the key aspects of the proposed work can be outlined as follows:

1. A novel chaotic map with increased key space and wide regions of chaotic behavior is proposed.
2. A simple, statistically secure PRBG is designed based on the chaotic map.
3. An image encryption technique is developed, with confusion and diffusion steps. The technique gives emphasis on the permutation of each pixel bits accross all

rows and columns of the image. The resulting method was tested against a collection of statistical tests and attacks, yielding resistance to all of them.
4. The overall design was implemented in a GUI, which facilitates its usage.

The rest of the chapter is structured as follows: Sect. 2 presents the generalized version of the map in [8] and studies its dynamical behaviour. In Sect. 3, a PRBG is constructed using the proposed map, and its statistical randomness is verified. Section 4 presents the complete encryption design. In Sect. 5, a collection of plaintext images are encrypted, and then submitted to a series of tests and measures, to evaluate the design. In Sect. 6, the complete procedure is implemented in a Graphical User Interface. Finally, Sect. 7 concludes the work, with a discussion on future topics of interest.

2 The Proposed Chaotic Map

Recently in [8], Talhaoui, Wang & Midoun proposed the following one-dimensional cosine polynomial chaotic map

$$x_k = \cos(\mu(x_{k-1}^3 + x_{k-1})) \tag{1}$$

where $\mu > 0$ is a parameter. The map exhibits chaotic behaviour for almost all parameter values, especially higher ones. This can be verified by the two bifurcation diagrams of (1) with respect to parameter μ, which are shown in Fig. 1, for low and high values of the parameter, and initial condition $x_0 = 0.1$. For lower values, the map exhibits the common phenomenon of period doubling route to chaos, as well as crisis phenomena, where it abruptly exits chaos, as can be seen around $\mu = 1.5$. For higher parameter values however, wide ranges of chaotic behaviour are observed. Also, since the map utilises a cosine function, the values are mapped on the interval $[-1, 1]$.

Based on this map, a simple generalization is given by:

$$x_k = \cos(\mu(x_{k-1}^3 + x_{k-1}) + a) \tag{2}$$

where $a > 0$ is a second parameter that shifts the argument of the cosine function. Due to the periodicity of the cosine function, this parameter is considered in the interval $a \in [0, 2\pi)$. Clearly, the original map (1) is a special case of (2) for $a = 0$. Two bifurcation diagrams of (2) are shown in Fig. 2 with respect to parameter μ, under the same low and high value ranges, when the value of the second parameter is $a = 4$, and $x_0 = 0.1$. Compared to the original map, it can be seen that the behaviour of the new map showcases similar chaotic phenomena for low values of μ, and remains consistenly chaotic for higher values.

The above result can be verified by also considering the bifurcation diagram with respect to a for the proposed map. Figure 3 shows the diagram for a in the range

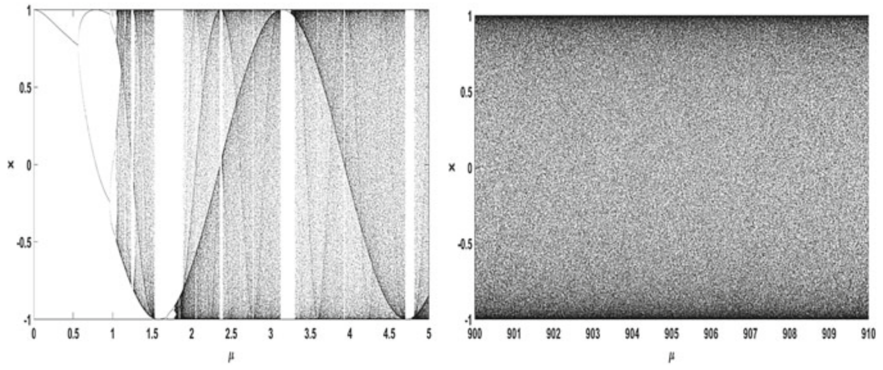

Fig. 1 Bifurcation diagram of the map (1) for two different ranges of the parameter μ and $x_0 = 0.1$

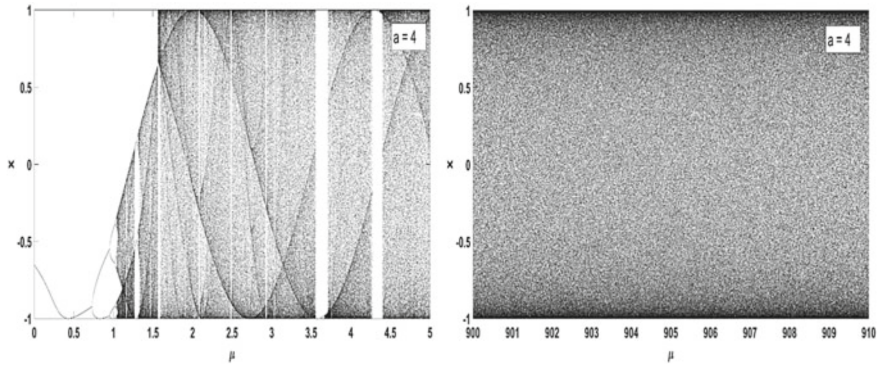

Fig. 2 Bifurcation diagram of the proposed map (2) for two different ranges of the parameter μ and $a = 4$, $x_0 = 0.1$

$[0, 2\pi)$, for two different values of the parameter μ. For $\mu = 1.5$, the map showcases regions of constant chaos, as well as abrupt changes in the shape of the attractor, as can be seen in the range $a \in (3, 2\pi)$. For a higher value of μ though, the system retains its constant chaotic behaviour for all values of a.

For a clearer visualisation of the behaviour of (2) for small values of μ, Fig. 4 shows a graph indicating the chaotic and non-chaotic pairs of parameters (μ, a). Here, interesting patterns emerge, showcasing the complex interchange between chaotic and periodic behaviour.

Also, Fig. 5 shows a comparison between the Lyapunov exponent (LE) of the original map (1) and the proposed map (2) for $a = 4$. Here, it can be seen that the two maps maintain practically the same value of LE in the computed range. Overall, the proposed map can maintain the chaotic behaviour of the original map, and since it has an additional parameter, it has an increased key space, which can make it useful in encryption related applications [41].

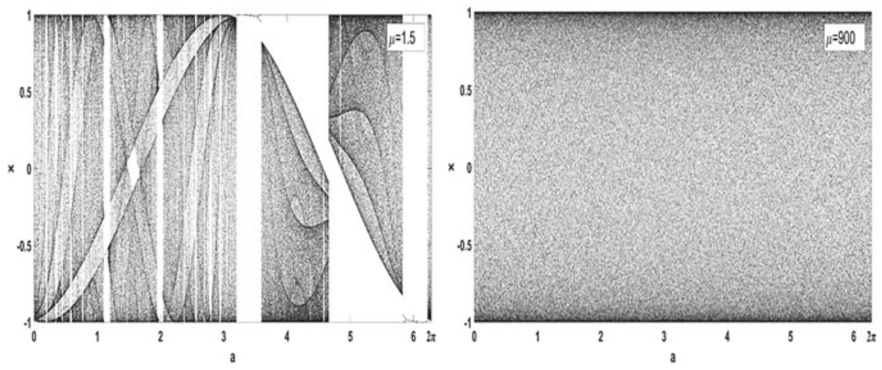

Fig. 3 Bifurcation diagram of the proposed map (2) for two different ranges of the parameter a and $\mu = 1.5$, $\mu = 900$, $x_0 = 0.1$

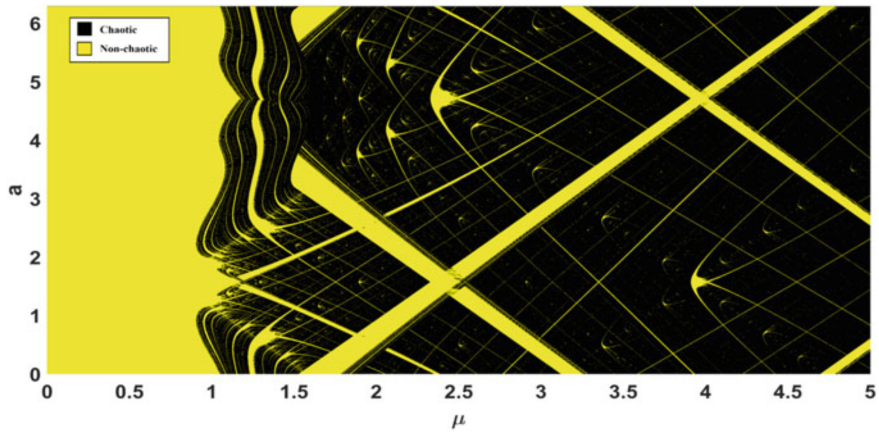

Fig. 4 Pairs of parameter values (μ, a) of the map (2) that yield chaotic (black) and non-chaotic (yellow) behaviour

Finally, Fig. 6 shows different phase diagrams for the original and modified map, under different parameter values. For both maps, it can be seen that for higher values of the parameter μ, the phase diagram does not have a distinguishable shape, a characteristic that is desired for maps used in encryption related applications [4, 12].

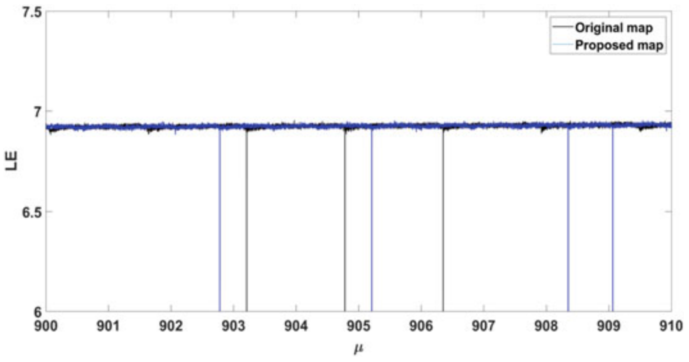

Fig. 5 Diagram of Lyapunov exponent of (1) and (2) for $a = 4$, $x_0 = 0.1$

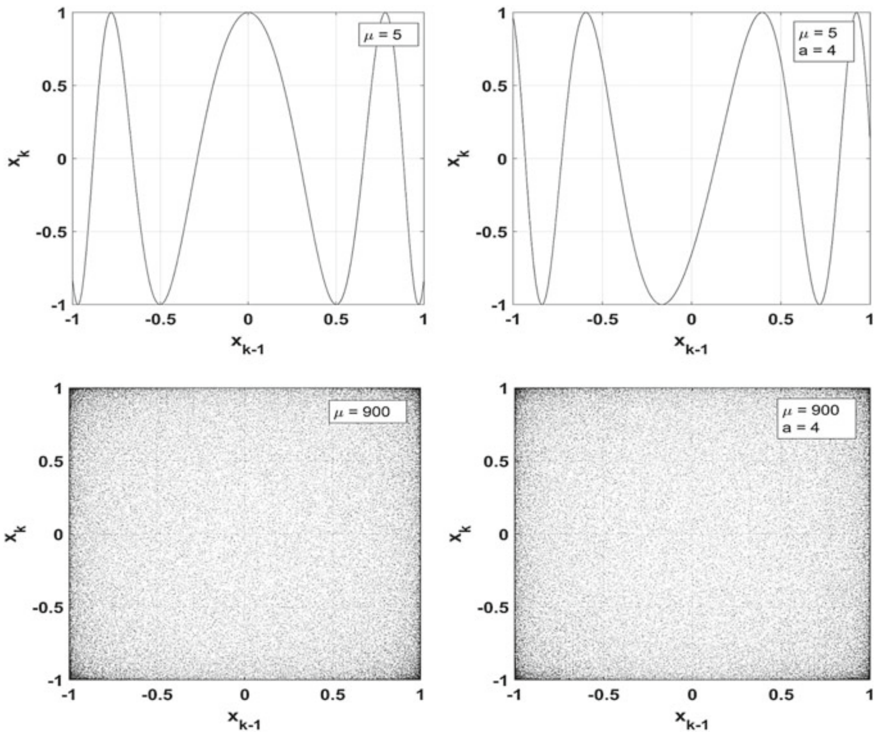

Fig. 6 Phase diagrams of the original (1) (left) and modified (2) (right) maps, for different parameter values, and $x_0 = 0.1$

3 Construction of Pseudo-Random Bit Generator

In this section, a PRBG is designed based on the map (2). Using the values of the map, the bits are generated based on the following rule

$$b_k = \lfloor mod(10^{12}|x_k + x_{k-1}|, 2)\rfloor \tag{3}$$

where $\lfloor\cdot\rfloor$ denotes the floor operation. The resulting bitstream is $\mathcal{B} = \{b_0, b_1, ...\}$. The use of the sum $|x_k + x_{k-1}|$ helps in increasing the complexity of the PRBG. To verify the statistical randomness of the generator, a set of $100 \cdot 10^6$ bitstreams is generated and tested through the National Institute of Standards and Technology (NIST) test suite [42]. The package consists of 15 statistical tests. Each test is used to verify the randomness of a bitstream. For each test, a P-value is returned, and if it is higher than a significance level, chosen here as the default 0.01, the test is considered succesful. For a generator to be considered random, all the tests should be successful. The results are shown in Table 1, where it can be seen that the generator passes all the tests, and can thus be used in an encryption application. Note that for tests that have multiple sub-cases, like the NonOverlappingTemplate, only the P-value of the last case is printed.

Table 1 NIST test results ($x_0 = 0.1$, $\mu = 900$, $a = 4$)

No.	Test	Chi-square P-value	Rate
1	Frequency	0.779188	99/100
2	BlockFrequency	0.779188	100/100
3	CumulativeSums	0.275709	99/100
4	Runs	0.779188	99/100
5	LongestRun	0.678686	98/100
6	Rank	0.096578	98/100
7	FFT	0.851383	98/100
8	NonOverlappingTemplate	0.017912	100/100
9	OverlappingTemplate	0.122325	100/100
10	Universal	0.419021	99/100
11	ApproximateEntropy	0.514124	100/100
12	RandomExcursions	0.021999	56/57
13	RandomExcursionsVariant	0.759756	57/57
14	Serial	0.419021	99/100
15	LinearComplexity	0.911413	98/100

4 The Proposed Shuffling and Encryption Technique

4.1 Image Bit Levels

In a grayscale image A, every pixel has a grey intensity between black and white, which is represented by an integer between 0 and 255, with 0 corresponding to the black value, 255 to the white, and all integers in between representing the shades of grey. Since each integer in the interval [0, 255] can be represented using 8 bits, every pixel value a_{ij} can be represented in binary format as:

$$a_{ij} = a_{ij}^1 2^0 + a_{ij}^2 2^1 + a_{ij}^3 2^2 + a_{ij}^4 2^3 + a_{ij}^5 2^4 + a_{ij}^6 2^5 + a_{ij}^7 2^6 + a_{ij}^8 2^7 \tag{4}$$

where $a_{ij}^w \in \{0, 1\}$ denotes the value of the w-th bit. By representing each pixel in the above format, an image can be broken down into eight individual binary subimages, where each image has the value a_{ij}^w at the (i, j) position. These binary images are called pixel levels, or pixel planes of an image, as shown in Fig. 7. An example of the different pixel levels of a grayscale image is shown in Fig. 8. By observing the different bit levels, it is clear that lower levels do not contain any distinguishable information, while higher levels carry more information about the image. This is undestandable if one considers (4). A bit value of $a_{ij}^1 = 1$ at the first level will only contribute the value of $2^0 = 1$ on the pixel value a_{ij}, while a bit value of $a_{ij}^8 = 1$ at the eighth level will contribute the value $2^7 = 128$ on the pixel value a_{ij}. Thus, bits at the higher levels have a greater effect on the final pixel value, and thus carry more information about the image [24–26]. The percentage of pixel information for each bit level is given by:

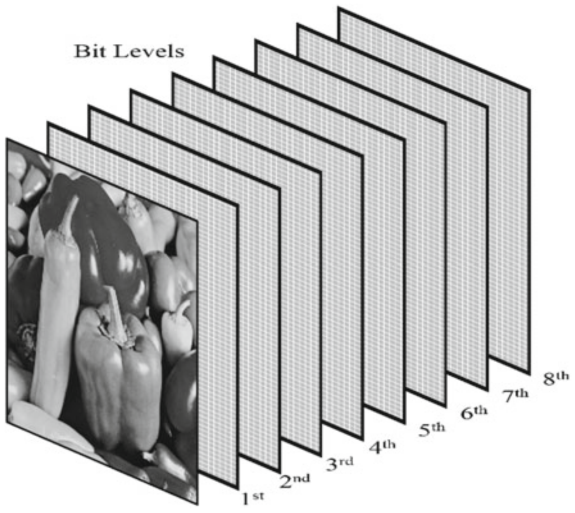

Fig. 7 Bit levels of a grayscale image

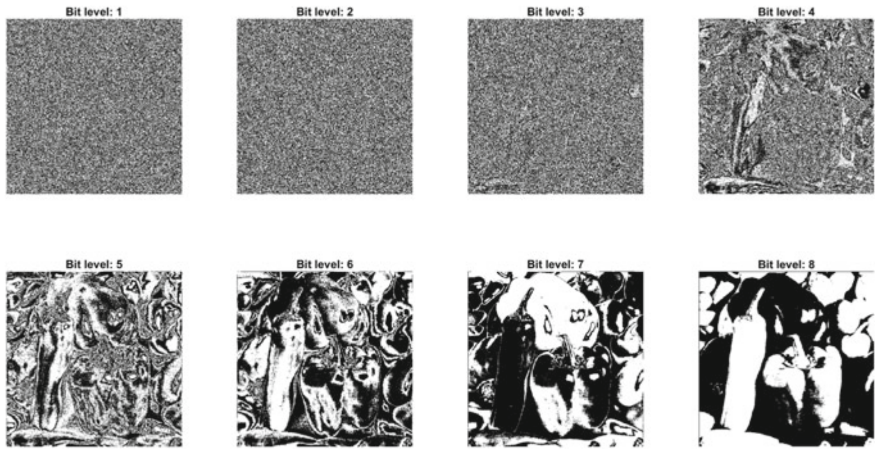

Fig. 8 The individual bit levels of a grayscale image

Table 2 Percentage of pixel information carried in each bit level

Level	Percentage (%)
$p(1)$	0.3921
$p(2)$	0.7843
$p(3)$	1.5686
$p(4)$	3.1373
$p(5)$	6.2745
$p(6)$	12.5490
$p(7)$	25.0980
$p(8)$	50.1961

$$p(i) = \frac{2^{i-1}}{\sum_{j=1}^{8} 2^{j-1}} 100\%, \ \ i = 1, ..., 8 \tag{5}$$

and the results are shown in Table 2. It can be verified that the last four levels carry over 90% of the image information. As will be seen in the following section, a shuffling of the bits among bit levels allows the significant information of a pixel to be distributed along all pixel levels and positions. This not only makes the design more secure, but also allows for reconstruction of the image in case information is lost or corrupted during transmission.

4.2 Encryption Algorithm

The encryption algorithm utilises the proposed chaotic map (2) and the PRBG of Sect. 3, and consists of two main steps, a three level shuffling operation and an exclusive OR operation. Algorithm 1 is a thorough, step-by-step presentation of the proposed method. In the following paragraphs, the outline of each step is explained.

Remark 1 First, a note on notation. When referring to the entries of a matrix, the "**:**" symbol denotes all the entries with respect to the given dimension. For example for a $M \times N$ matrix A, the term $A(1, :)$ refers to the entries of the first row. Similarly, for a $M \times N \times 8$ matrix B, the term $B(1, :, :)$ denotes the $N \times 8$ matrix formed by taking the entries in all the N columns and all 8 levels that correspond to the first row. Note that this syntax is adopted from MATLAB. Such sub-matrices are also depicted in Fig. 9.

Also, in Algorithm 1, the subscript i denotes rows, j denotes columns, w denotes bit levels, and k denotes iterations of the chaotic map used in each Step.

There are four chaotic maps of the form (2) used, three in the shuffling process, and one to generate the PRBG. The first step is to compute their key parameters. The parameters are dependent on the plaintext image used, and are computed using (15). Having the key values being plaintext dependent can make the encryption scheme resistant to plaintext attacks [16, 30–34, 36, 37].

After the key values are chosen, the $M \times N$ input image $\mathcal{A}$ is decomposed into its bit levels, resulting in a 3D binary matrix representation $\mathcal{A}_{bin}$ of dimensions $M \times N \times 8$. So an element a_{ij}^{w} of matrix $\mathcal{A}_{bin}$ represents the w-th bit of the pixel in the position (i, j).

Once the 3D binary matrix $\mathcal{A}_{bin}$ is obtained, the three level shuffling procedure begins. In essence, this procedure consists of performing a shuffling operation individually in each row level, column level, and pixel level of the matrix. The three levels are depicted in Fig. 9. Similar shuffling techniques appear in [19–21], but they are applied to the pixels of the original $M \times N$ image, not its binary representation.

Starting from the row levels, the matrix $\mathcal{A}_{bin}$ is decomposed into M different $N \times 8$ matrices, which correspond to taking all the columns and bit levels for an individual matrix row, that is $\mathcal{A}_{bin}(1, :, :)$, $\mathcal{A}_{bin}(2, :, :)$,..., $\mathcal{A}_{bin}(M, :, :)$. The rows and columns of each individual matrix are then shuffled, via left and right matrix multiplication by invertible matrices, generated using the first chaotic map. This process is analytically described in Steps 3a.-3c. of Algorithm 1. After the shuffling of all individual row levels is complete, the resulting matrix is denoted as $\mathcal{R}$.

Then, the second shuffling is performed, this time on the column levels of $\mathcal{R}$. So the 3D matrix $\mathcal{R}$ is decomposed into N different $M \times 8$ matrices, which correspond to taking all the rows and bit levels for an individual column, that is $\mathcal{R}(:, 1, :)$, $\mathcal{R}(:, 2, :)$,..., $\mathcal{R}(:, N, :)$. Each individual matrix is then shuffled using the same technique described in the previous step, and in Steps 4a.–4c. of Algorithm 1, using the second chaotic map. This process results in matrix C.

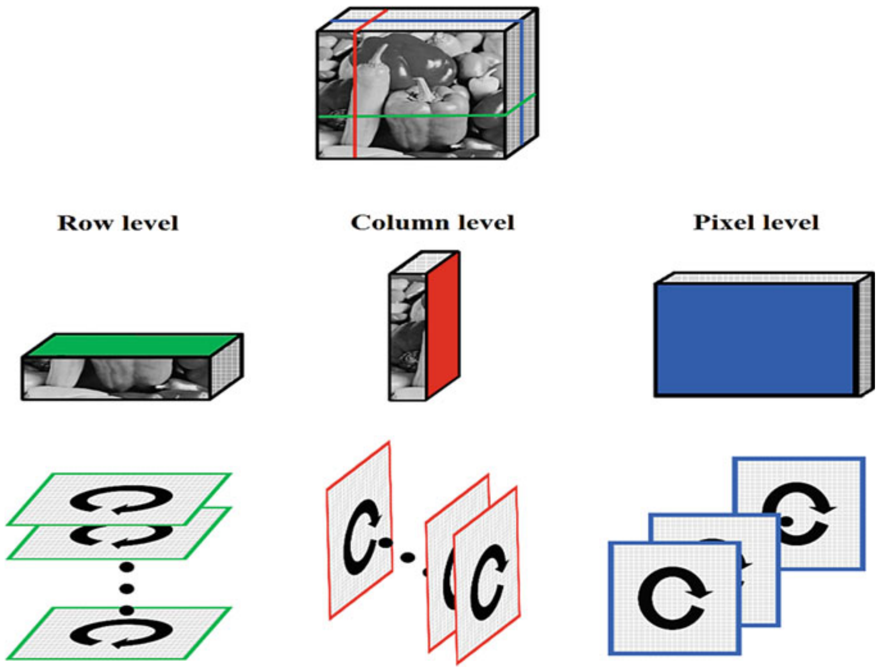

Fig. 9 The three levels of shuffling performed on the $M \times N \times 8$ 3D matrix that represents the image

Similarly, the third shuffling is performed by decomposing matrix C into its eight $M \times N$ pixel levels $C(:, :, 1)$, $C(:, :, 2)$,..., $C(:, :, 8)$. Each individual level is then shuffled using the above technique, as described in Steps 5a.–5c. of Algorithm 1, using the third chaotic map. The resulting 3D matrix is denoted as $\mathcal{P}$.

The three levels of shuffling result in the 3D binary matrix $\mathcal{P}$, which, using (4), can be reversed into a grayscale image. Since the shuffling procedure is performed on all bit levels, the resulting image has different pixel values than the original plaintext image $\mathcal{A}$, so it is actually a ciphertext of the original image. Yet, as will be seen in the next Section, performing only the shuffling procedure will not guarantee security against all types of attacks. For example, an all black image where each pixel value is zero will result in a 3D binary matrix consisting only of zeros, thus the shuffling of the bits will have no effect on the plaintext. This will make the design vulnerable to known plaintext attacks. Hence, the shuffling procedure is followed by a second encryption step, where an exclusive OR operation is performed.

For this second round of encryption, the bits of the $M \times N \times 8$ matrix $\mathcal{P}$ are reshaped into a vector of length $M \cdot N \cdot 8$, and are combined using XOR with a bitstream of the same length, generated using the fourth chaotic map and the PRBG of the previous section. The resulting bitstream denoted as $\mathcal{E}_{stream}$ can then be reshaped into an $M \times N \times 8$ matrix $\mathcal{E}_{bin}$, and transformed into the ciphertext image using (4). This image can then be safely transmitted through a public channel.

To decrypt the cihpertext, the reverse procedure needs to be followed. Once the receiver obtains the ciphertext image and the key values for the maps, each step of Algorithm 1 is followed backwards, starting from the last one. Initially, the ciphertext image is reshaped into a vector and XORed using the same bitstream from the same PRBG used for the encryption process. Then, Step 5 is reversed by computing all the shuffling matrices for each pixel level and performing the reverse shuffling by multiplying by the matrix inverses as $L_w^{-1}(L_w \mathcal{P}_w Q_w) Q_w^{-1} = \mathcal{P}_w$, to obtain the matrix C that was produced at the end of Step 4. Then Step 4 is reversed in the same way to obtain $\mathcal{R}$ at the end of Step 3. Finally, Step 3 is reversed to obtain $\mathcal{A}_{bin}$, which corresponds to the original plaintext image.

Remark 2 In determining the plaintext dependent key values in (15), in the case where the sum inside the modulo is an integer, then $\lambda = 0$. This could happen in known plaintext attacks, for example in all white and all black images. Choosing $\lambda = 0$ would lead to periodic behaviour in the chaotic maps, so to counter this, in case $\lambda = 0$, the parameter is replaced by $\lambda = mod\left(\log_e \left(M \cdot N + \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij}}{M \cdot N} \right) + \text{Ent}(A), 1 \right)$.

5 Encryption Performance

The proposed technique is applied to seven different grayscale images taken from the USC-SIPI Image Database (http://sipi.usc.edu/database/), namely the peppers, baboon, airplane, earth, gray shades, all black, and all white images. The encryption results are shown in Fig. 10 where the original (plaintext), shuffled, and encrypted (ciphertext) images are shown. In all cases, the encrypted image visually displays no recognisable information on the original plaintext. This is also the case for the shuffled images, except from the case of the all black and all white images, where the shuffling has no effect. In the following, the performance of the encryption scheme is tested with respect to different measures, to verify that the encrypted images are indeed secure against attacks that try to unmask information on the plaintext image.

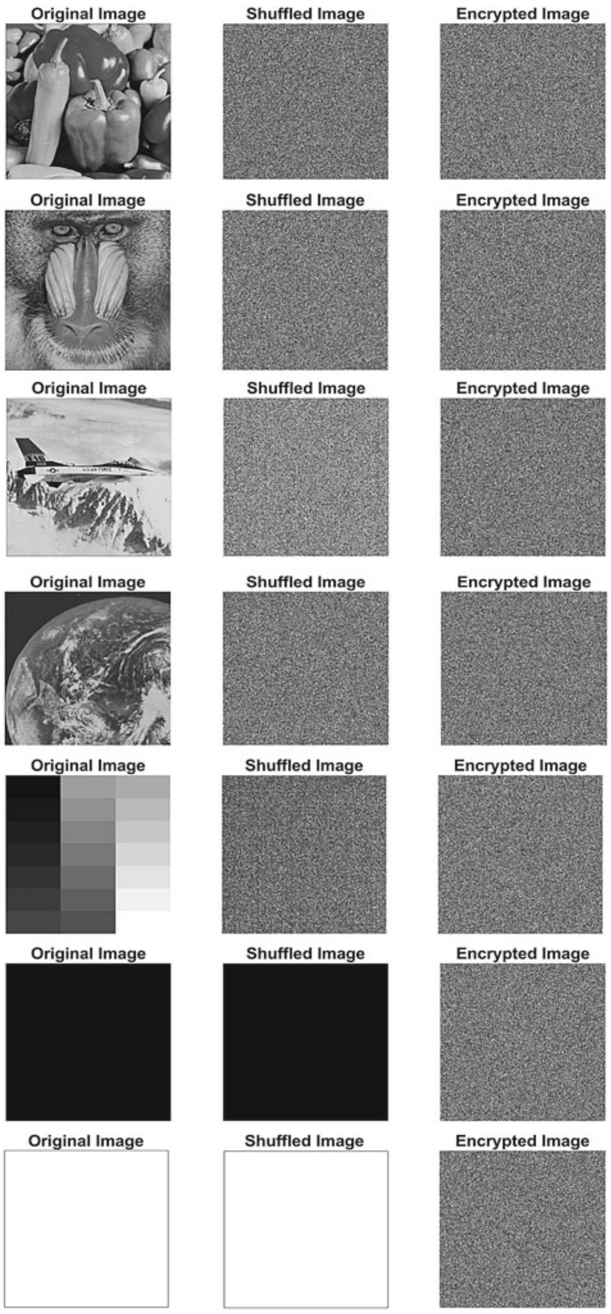

Fig. 10 Plaintext shuffled, and encrypted images

5.1 Histogram

The simplest statistical analysis that can be performed on an image is to measure the distribution of all the 256 possible grey values. The greyscale distribution is plotted in a histogram. An image depicting any sort of information will naturally have an uneven distribution of pixel values. Thus, by looking at the histogram, it is easy to identify images carrying information, despite not being identifiable to the naked eye. So, for a ciphertext to be secure to statistical attacks, it should have a uniform histogram.

The histograms of the plaintext, shuffled and encrypted images are shown in Fig. 11. Looking at the shuffled images first, it can be observed that the histogram is not uniform in all cases, as can be seen in the Airplane and Shades cases. As already pointed out, the shuffling has no effect on the monocoloured images. In the encrypted case though, the histogram is uniform for all image files.

In addition, the variance of each histogram is computed, to obtain a statistical measure of the histogram's uniformity, apart from its visual inspection [43, 44]. The variance is computed as

$$var(\mathcal{H}) = \frac{1}{256^2} \sum_{i=0}^{255} \sum_{i=0}^{255} \frac{(h_i - h_j)^2}{2} \tag{6}$$

where $\mathcal{H} = \{h_0, ..., h_{255}\}$ is the sum of measurements for each grayscale value in the image pixels. A smaller variance indicates a uniform distribution of the pixel values. The results are shown In Table 3. As can be seen, the variance of the encrypted image histograms is lower than the plaintext and shuffled ones. In the same Table, results of recent works are also printed, for the images that are considered in each study. The variance results are comparable to these works, meaning that the method is equally efficient. Notice though that other works reported slightly different variances for the original plaintext images, which could be the result of different source databases or different file types used.

Overall, by studying the histogram of the encrypted images, no information about the pixel value distribution for the plaintext image can be extracted, and the scheme can be considered secure against histogram analysis.

5.2 Correlation

In images depicting information, adjacent pixels will have relatively close values. This means that their correlation coefficient will be high. In a ciphertext image, the correlation coefficient should be close to zero for adjacent pixels, indicating that their values are uncorrelated. The correlation coefficient γ is computed by:

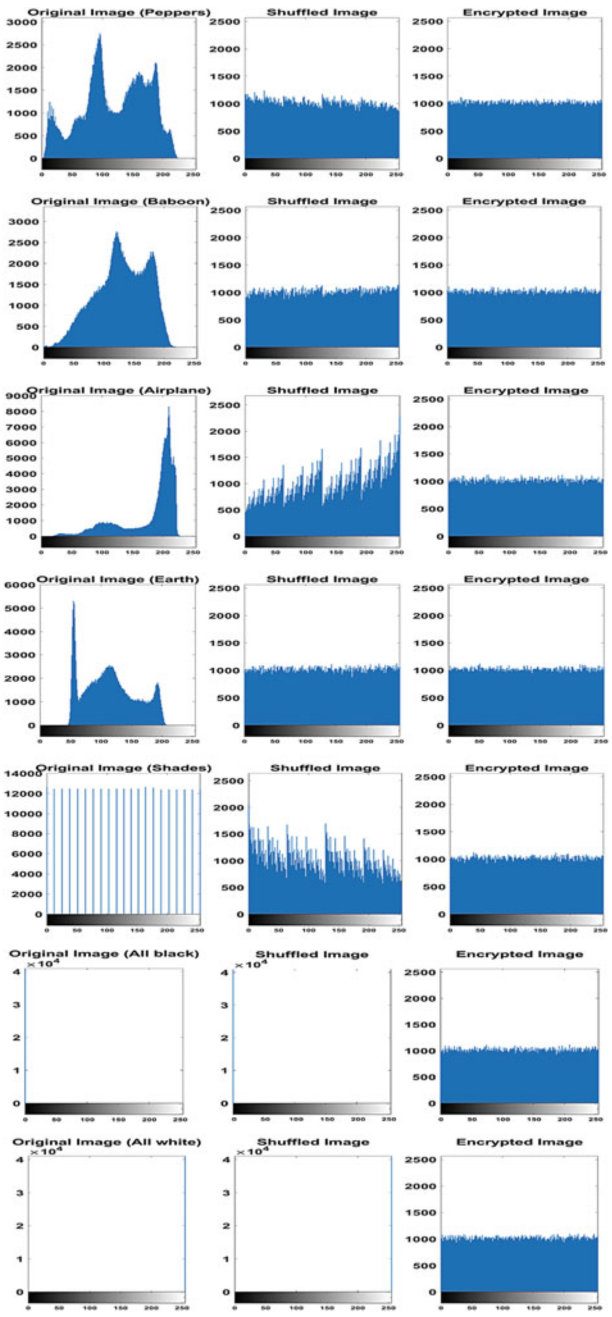

Fig. 11 Histograms of the plaintext, shuffled, and encrypted images

Table 3 Histogram variance for plaintext, shuffled and encrypted images

	Proposed			Safack et al. [43]	Abdelfatah [44]	Nepomuceno et al. [35]
	Original	Shuffled	Encrypted	Encrypted	Encrypted	Encrypted
Peppers	482548.1411	4697.4588	894.7686	–	1076.6	–
Baboon	752365.2156	2846.5411	1095.1058	909.4980	880.2	990.11
Airplane	2882376.6274	97676.7215	1275.6156	–	–	–
Earth	1066350.9254	1670.9098	956.5019	–	–	–
Shades	11780472.3294	63914.9254	1242.5568	–	–	–
All black	268435456	268435456	1091.9215	–	–	–
All white	268435456	268435456	996.2823	–	–	–

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{7}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))^2, \tag{8}$$

$$cov(x, y) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))(y_i - E(y)), \tag{9}$$

$$\gamma(x, y) = \frac{cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{10}$$

where x, y are the gray values of two adjacent pixels, and N is the number of adjacent pairs of pixels (x, y). Table 4 displays the correlation coefficients for all the plaintext, shuffled and encrypted images. While for the original images the correlation between adjacent horizontal, vertical and diagonal pixels is high, for the shuffled and encrypted images, the correlation is consistently close to zero, up to the order of magnitude $10^{-3} - 10^{-4}$. This is the desired result, which is in line with most recent works in image encryption. Thus, both the shuffling and encryption procedures manage to reduce correlation, with the exception again being the all black and all white cases, for which shuffling has no effect.

5.3 Information Entropy

The global information entropy is a measure of randomness and unpredictability for a signal. The information entropy for a grayscale image is computed using the technique given in [5]:

Table 4 Correlation coefficients for the image encryption

	Original			Shuffled			Encrypted		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Peppers	0.9768	0.9792	0.9639	0.0003	0.0022	0.0021	0.0005	0.0002	0.0007
Baboon	0.8665	0.7587	0.7262	0.002	−0.0022	−0.0005	−0.0006	−0.0006	0.0001
Airplane	0.9663	0.9641	0.9370	0.0030	0.0079	0.0032	−0.0002	−0.0007	−0.0020
Earth	0.9709	0.9761	0.9515	0.0017	0.0022	−0.0014	−0.0003	−0.0049	−0.0014
Shades	0.9965	0.9998	0.9964	0.0022	0.0063	0.0003	−0.0020	−0.0002	0.0002
All black	–	–	–	–	–	–	0.0009	0.0037	−0.0024
All white	–	–	–	–	–	–	−0.0018	−0.0010	0.0008

$$H(S) = -\sum_{i=0}^{2^8-1} p(s_i) \log_2 p(s_i), \tag{11}$$

where $p(s_i)$ is the probability of occurrence for the value s_i. The information entropy of an encrypted image should be close to 8, which indicates randomness in the expected pixel values. Table 5 shows the entropy of the original plaintext, shuffled and encrypted images. In all cases, the encrypted images have the highest entropy value, close to the ideal value of 8, which indicates that the resulting ciphertext is secure to entropy checks. This is aligned with recent works on image encryption.

An additional entropy measure that can be used to test the randomness of an image is that of local entropy. In contrast to global entropy described above, which measures entropy over the complete image, local entropy considers individual sub-regions of the image. It is computed by considering a set of randomly chosen non-overlapping blocks of an image, and computing the average entropy across all blocks [45–50]. Here a set of 30 randomly chosen non-overlapping blocks of size 44×44 is considered, so each block consists of $44^2 = 1936$ pixels. The results are shown in Table 6. In all cases, the encrypted images maintain high values of local entropy around 7.9, in contrast to the plaintext images. The results are all comparable to recent works which also considered the local entropy.

Table 5 Entropy for plaintext and encrypted images

	Original	Shuffled	Encrypted
Peppers	7.5937	7.9968	7.9994
Baboon	7.3583	7.9980	7.9993
Airplane	6.7025	7.9355	7.9991
Earth	7.1530	7.9989	7.9993
Shades	4.3923	7.9575	7.9991
All black	0	0	7.9993
All white	0	0	7.9993

Table 6 Local entropy for plaintext and encrypted images

	Proposed			Chen et al. [48]	Sambas et al. [49]	Safack et al. [43]	Alanezi et al. [51]
	Original	Shuffled	Encrypted	Encrypted	Encrypted	Encrypted	Encrypted
Peppers	6.2355	7.9077	7.9060	7.9030	7.9027	7.9034	7.9014
Baboon	6.6679	7.9009	7.9092	7.9022	7.9020	7.9030	7.9032
Airplane	5.3389	7.8392	7.9048	–	7.9033	7.9009	7.9022
Earth	6.2019	7.9088	7.9078	–	–	–	–
Shades	0.3694	7.8624	7.9075	–	–	–	–
All black	0	0	7.9044	–	–	–	–
All white	0	0	7.9070	–	–	–	–

5.4 *Differential Attack Analysis*

In an encryption scheme, a small alteration of the original plaintext image should lead to a completely different encrypted image. This makes the design secure to known plaintext attacks, where an attacker chooses to encrypt the same plaintext repeatedly, changing it slightly each time, and observe changes in the ciphertext, in order to unmask the structure of the encryption.

There are two measures to identify the degree of change between almost identical plaintexts, called the number of pixels change rate (NPCR) and the unified average changing intensity (UACI). To compute these measures, two identical images are considered, where the second one differs from the first in only a single pixel. Both images are encrypted, and the above measures are computed by the following equations [5]:

$$\text{NPCR} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} D_{ij}}{MN} 100\% \tag{12}$$

$$\text{UACI} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} |\mathcal{E}_{ij} - \hat{\mathcal{E}}_{ij}|}{MN255} 100\% \tag{13}$$

where $\mathcal{E}$ and $\hat{\mathcal{E}}$ are the two encrypted images, and

$$D_{ij} = \begin{cases} 0 & \mathcal{E}_{ij} = \hat{\mathcal{E}}_{ij} \\ 1 & \mathcal{E}_{ij} \neq \hat{\mathcal{E}}_{ij} \end{cases} \tag{14}$$

The ideal values are 99.61% for the NPCR and 33.46% for the UACI, so the closer the real values are to these, the stronger the encryption design. Since in the proposed methodology, the key values for the maps used for shuffling and XOR are dependent

Table 7 NPCR and UACI measures for different images

	Shuffled		Encrypted	
	NPCR (%)	UACI (%)	NPCR (%)	UACI(%)
Peppers	99.6063	33.4544	99.6239	33.4492
Baboon	99.5949	33.4303	99.6105	33.4467
Airplane	99.5823	33.0876	99.6056	33.4818
Earth	99.6037	33.5053	99.6216	33.4195
Shades	99.5796	33.3158	99.6048	33.4314
All black	0.0004	0.0002	99.6170	33.4088
All white	0.0004	0.00004	99.6048	33.4611

on the plaintext image, changing it even by one pixel will change the key values and thus yield a different shuffled and ciphertext image. Table 7 shows the NPCR and UACI values for all images. It can be seen that both the shuffled and encrypted images perform satisfactorily with respect to both measures, as the results are close to the ideal values, which is also the case for recent works in image encryption. Note that the shuffled all black and all white images yield an NPCR and UACI measure close to, but not identical to zero, as in the case of the edited image with one changed pixel, the keys are computed from (15) and not as in Remark 2, so the shuffling rules are changed.

5.5 *Cropping and Noise Attacks*

During transmission, it is possible that some part of the transmitted message may be lost, or corrupted. Hence, it is essential to evaluate to what extend it is possible to reconstruct an image when the ciphertext is either cropped, tempered with, or affected by noise [10, 15, 22, 33, 34, 52–54]. Figure 12 shows the result of decryption that is performed from a ciphertext that was cropped, and Fig. 13 the result when the ciphertext is tampered with. In all cases, it can be seen that despite the loss of information, the encrypted image can be adequately reconstructed, even in cases where 50% of the ciphertext has been cropped or tampered with. When the distortion reaches 75%, the outlines of the objects are still visible.

The resistance to cropping attacks is attributed to the fact that during the shuffling process, the significant bits on the 5th–8th levels are redistributed across different pixel levels and positions in the resulting shuffled image. Thus, cropping out a specific portion of the ciphertext does not correspond to loss of information for the identical portion of the plaintext.

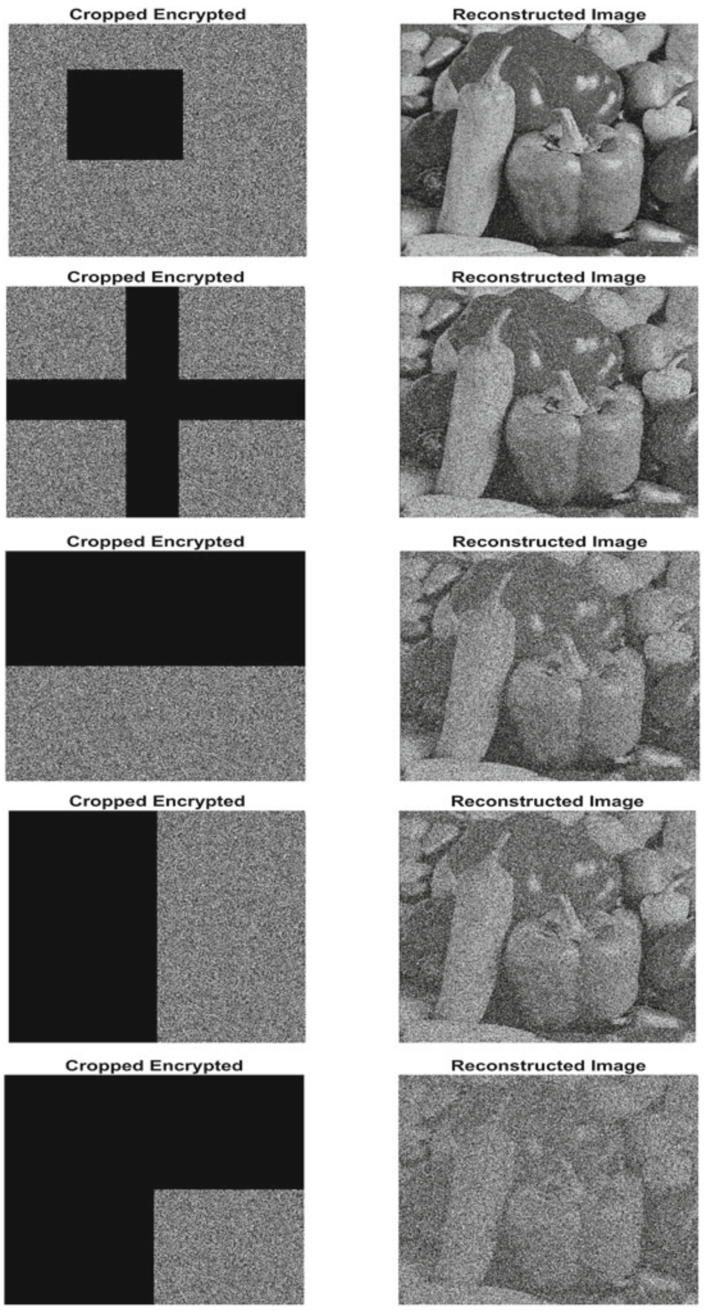

Fig. 12 Decryption from different levels of cropped ciphertext images

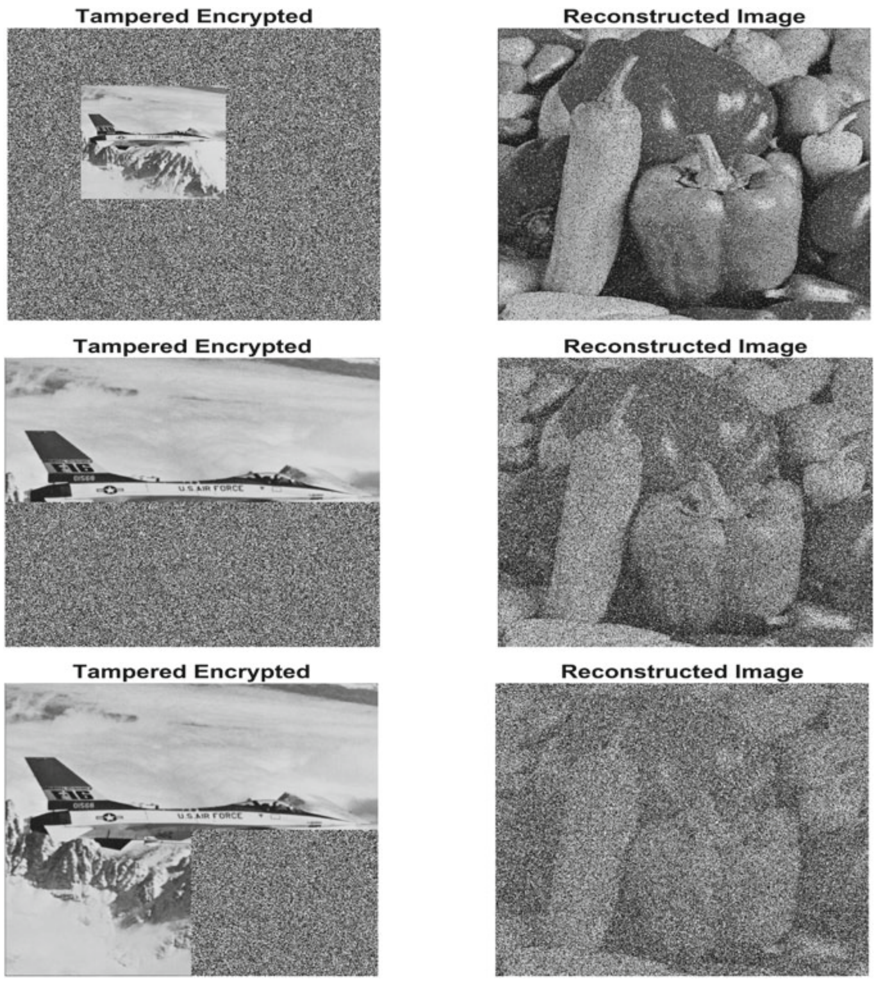

Fig. 13 Decryption from different levels of tampered ciphertext images

Similarly, Fig. 14 shows the decryption performed from ciphertext corrupted by salt and pepper noise of different intensities. Again, the reconstructed image is clear enough to identify the original ciphertext, even when 50% of the pixels are corrupted. At 75% of corruption, most of the information is lost, but still the outlines of the objects are visible, so some information is saved.

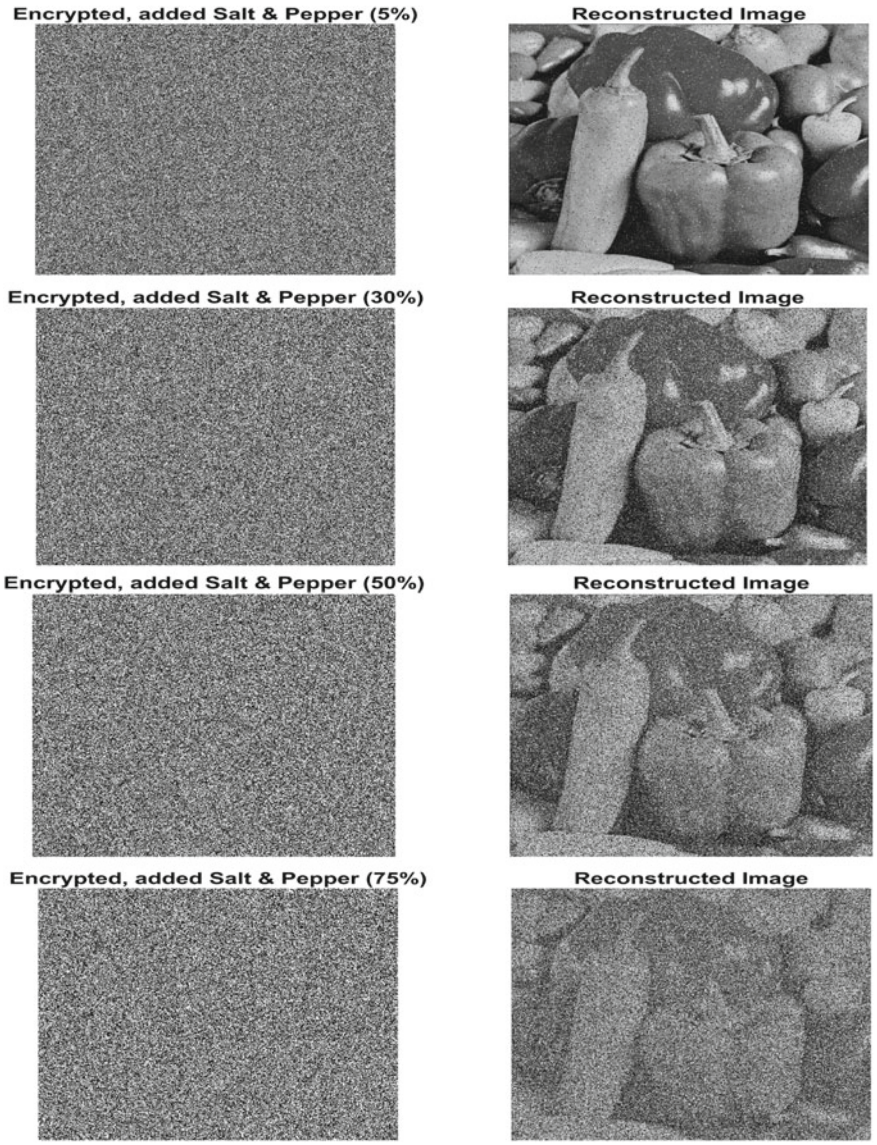
Encrypted, added Salt & Pepper (5%)
Reconstructed Image
Encrypted, added Salt & Pepper (30%)
Reconstructed Image
Encrypted, added Salt & Pepper (50%)
Reconstructed Image
Encrypted, added Salt & Pepper (75%)
Reconstructed Image

Fig. 14 Decryption from different levels of noisy ciphertext images

Table 8 Comparison of key space

Work	Key space
Proposed	2^{640}
Irani et al. [22]	2^{186}
Chen et al. [48]	2^{193}
Sambas et al. [45]	2^{213}
Liu and Miao [9]	2^{277}
Nepomuceno et al. [35]	2^{283}
Safack et al. [43]	2^{478}
Belazi et al. [16]	2^{716}
Abdelfatah [44]	2^{772}

5.6 *Key Space*

Any encryption system should be resistant against brute force attacks. For this, it is required that the key space be higher than 2^{100} [41]. In the proposed system, four chaotic maps (2) are used, with parameters x_0, μ_1, a_1, ,y_0, μ_2, a_2, z_0, μ_3, a_3, w_0, μ_4, a_4. Hence, there are overall 12 parameters. Assuming a 16 digit accuracy, an upper bound for the key space can be computed as $10^{12\cdot16} = 10^{192} = (10^3)^{64} \approx (2^{10})^{64} = 2^{640}$. This is higher than the required bound. Table 8 shows a comparison between the key spaces of recent works. It can be seen that the proposed work has among the highest key spaces.

Moreover, for the case of all black and all white images where the shuffling has no effect, the key space reduces to the key space of the XOR operation. Since 3 parameters are required, the key space upper bound is $10^{3\cdot16} \approx 2^{160}$, which again is higher than the required threshold.

Note that since all 12 key parameters are computed based on the plaintext dependent value λ, if an adversary had complete knowledge on the design and the way the key parameters are computed in (15), they would be able to break the encryption by performing a brute force attack just on parameter λ. This would make the design vulnerable, but this issue can be bypassed by small modifications of the procedure. For example, the plaintext can be broken down into four subfigures, and each subfigure can be used to compute an independent key λ_1, λ_2, λ_3, λ_4, similarly to λ. Then the λ_i can be used to compute the keys for each individual chaotic map as in (15).

5.7 *Key Sensitivity*

Any encryption scheme should be sensitive to changes in the encryption key, meaning that a small change in any of the key values would lead to a failure of the decryption process [45, 47, 51, 55]. This is the case in our proposed design as well. Since chaotic systems are used for encryption, the process inherits the key sensitivity of the chaotic maps that are used in its source. So any change in the parameter values leads to deviating chaotic trajectories, which makes the encrypted image impossible to decrypt, unless exact knowledge of the keys is granted.

To validate this, a plaintext image is first encrypted and decrypted, using the correct key values given in (15) the proposed Algorithm 1. Then, the encrypted image is decrypted using the same keys, where one key value is changed each time.

The simulation results are shown in Fig. 15 for the peppers plaintext image. It is clear than in all cases, the decryption using the wrong key values fails. Thus, the encryption design is sensitive with respect to all the key values used.

5.8 Execution Time

In the present section, the execution time of the proposed method is discussed. The results are shown in Table 9. Along with the total execution time, the time required for the different steps of the encryption process is presented, since decomposing the algorithm into its parts enables the discussion on reducing the execution time. The times presented are obtained using a PC with the following specifications:

Operational system	Windows 10 Home (64bit)
CPU	Intel(R) Core(TM) i5-8250U CPU @ 1.60G Hz 1.80 GHz
RAM	8.00 GB DDR4
MATLAB version	R2018a

Note that the process of displaying the encrypted image is not calculated in the total execution time. Clearly, most of the execution time is spent on shuffling the image bits, which was expected. The execution time is where the limitation of the proposed method is identified, as in most recent works, the encryption time ranges from less than a second for greyscale images, to a couple of seconds for coloured images [18, 29, 49, 51, 53, 55]. Thus, in future works, the problem of reducing the execution time of this or similar methods should be addressed.

One potential approach that could reduce the time cost for the row and column level shuffles is to utilize parallel processing when performing the shuffling of different levels. More explicitly, suppose that the image to be encrypted has M rows. In our approach, the rows or columns are shuffled one at a time, as seen next

$$1 \to 2 \to \cdots \to \frac{M(M+1)}{2} \to \frac{M(M+1)}{2} + 1 \to \cdots \to M.$$

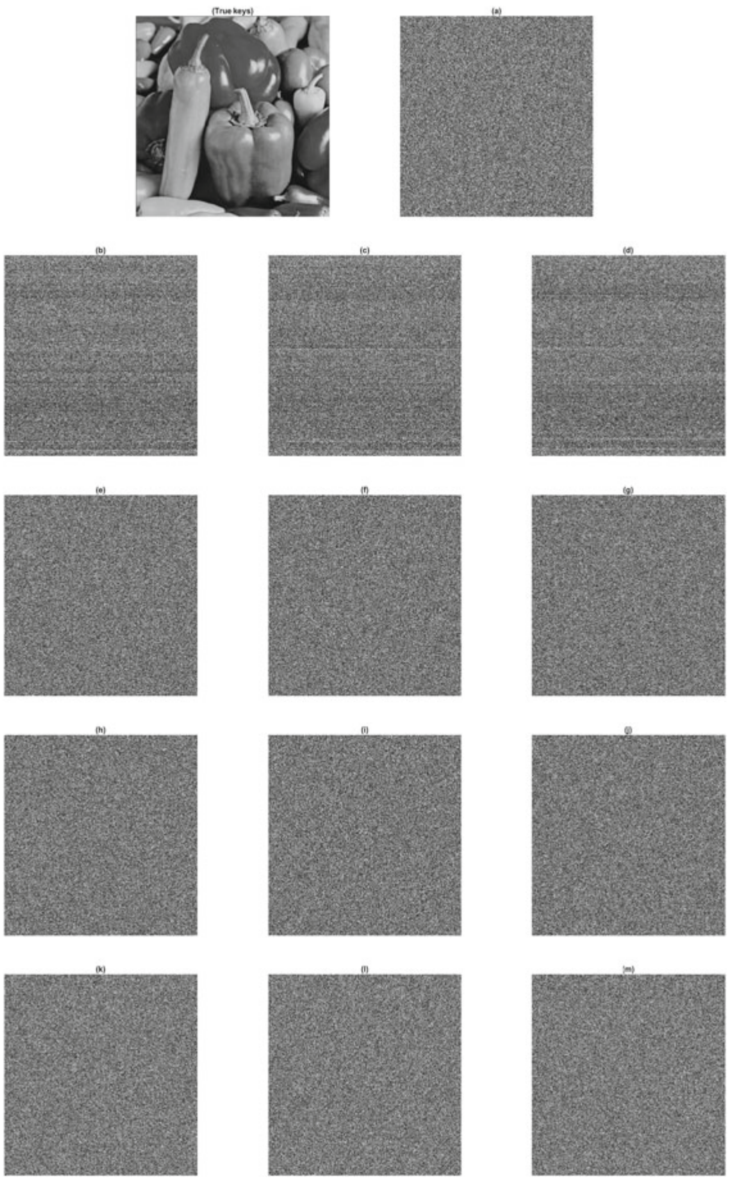

Fig. 15 Decryption using slightly modified keys. **a** Parameter λ is changed to $\lambda + 10^{-16}$, and the rest of the key values are computed based on the new value. **b** Parameter x_0 is changed to $x_0 + 10^{-16}$. **c** Parameter μ_1 is changed to $\mu_1 + 10^{-13}$. **d** Parameter a_1 is changed to $a_1 + 10^{-15}$. **e** Parameter y_0 is changed to $y_0 + 10^{-16}$. **f** Parameter μ_2 is changed to $\mu_2 + 10^{-13}$. **g** Parameter a_2 is changed to $a_2 + 10^{-15}$. **h** Parameter z_0 is changed to $z_0 + 10^{-16}$. **i** Parameter μ_3 is changed to $\mu_3 + 10^{-13}$. **j** Parameter a_3 is changed to $a_3 + 10^{-15}$. **k** Parameter v_0 is changed to $v_0 + 10^{-16}$. **l** Parameter μ_4 is changed to $\mu_4 + 10^{-13}$. **m** Parameter a_4 is changed to $a_4 + 10^{-15}$

Table 9 Time analysis (in seconds) of encryption method for different plaintext images

Image name	Peppers	Airplane	Baboon	Earth	Shades
Dimensions	512 × 512	512 × 512	512 × 512	512 × 512	512 × 512
Arrange bits in 3D matrix	2.4531	2.6563 3	2.5313	2.5938	2.4688
Row shuffle	2.2969	2.2969	2.3281	2.3438	2.3125
Column shuffle	2.25	2.2344	2.1875	2.2813	2.2656
Pixel level shuffle	0.1719	0.1563	0.1719	0.1719	0.1875
XOR and reshape	1.1875	1.2031	1.2813	1.2031	1.2188
Total time	8.35938	8.5469	8.5	8.5938	8.4531

An alternative approach is to perform the shuffling as

$$1 \rightarrow 2 \rightarrow \cdots \rightarrow \frac{M(M+1)}{2} \text{ and } \frac{M(M+1)}{2} + 1 \leftarrow \cdots \leftarrow M-1 \leftarrow M$$

Clearly, the two sides of the shuffling are disjoint and thus the two processes can be performed simultaneously. The drawback is that a different initialization is required for the "backward" shuffling process as well. A similar approach can be utilized for the columns and the bit level shuffling as well. Furthermore, based on the number of threads available, the number of breaks can increase, which would lead to a further decrease in execution time.

Another alternation that could reduce the execution time is to perform the shuffling process only on half of the image bit levels. It has already been discussed that the fifth to eighth bit levels contain the majority of the image information. This way the computational effort required for the row, column and bit level shuffling of the $M \times N \times 4$ matrix is greatly reduced from the original $M \times N \times 8$ case.

5.9 *Overall Evaluation*

Considering the results from all of the tests performed above, it can be concluded that the encrypted image is secure with respect to all types of analysis, like histogram analysis, correlation analysis, entropy measure, differential attacks, cropping attacks and noise interference, as well as brute force attacks. Performing only the shuffling procedure on the other hand may yield secure performance with respect to some measures like entropy, correlation, and brute force, but is vulnerable to chosen plaintext attacks, and also histogram analysis. Thus, both steps need to be implemented to have a secure design.

6 Graphical User Interface

In this section, the proposed encryption and decryption method is implemented with a Graphical User Interface (GUI). The reason behind this construction is twofold. Initially, when proposing an encryption and decryption scheme, it is of utmost importance to provide a simple and clear framework, in which anyone can utilize the proposed method, without requiring any familiarity with the theoretical part of the method, or do the implementation method from scratch. Furthermore, considering the fact that such encryption methods can potentially have a hardware implementation, the GUI provides a clear view on what the requirements and functions of such an implementation are.

Moving on to the presentation of the GUI, the interface can be seen in Fig. 16(left). Pressing the "Load Image" button opens a window to the file manager where the user can select the image file to be encrypted or decrypted. A preview of the selected image can be seen on the upper right part, under "Original Image".

To encrypt the loaded image, the "Encrypt and Save Key" button can be pressed. This initializes the encryption process. Once this process is finished, the user is asked to select the destination where the keys required for the decryption are saved in .txt format. Once this is chosen, a preview of the encrypted image is shown on the bottom right side, under the "Resulting Image". Once the encryption is complete, the user can press the "Save Result" button, which allows the user to save the encrypted image. One important notice here is that the encrypted image should not be saved in ".jpg" form, since this leads to round off errors that do not allow for the image to be decrypted.

For the case of decryption, the encrypted image is loaded, and pressing the "Load Decrypt Key" button opens a window where the user selects the .txt file containing the decryption keys. Again, loading the image will cause its preview to appear under "Original Image". Having selected the image file and the keys, pressing the "Decrypt" button will initiate the decryption process. Once finished, a preview of the decrypted image is shown under "Resulting Image" and again the user can save the decrypted image through the "Save Image" button.

In the example shown in Fig. 16, the encryption and decryption of the "peppers" image is considered.

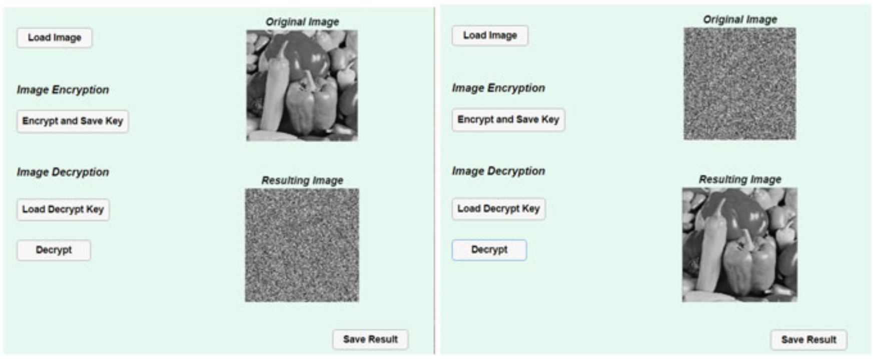

Fig. 16 Utilizing the GUI for the encryption (left) and decryption (right) of the "peppers" image

7 Conclusions

This work presented an image encryption procedure based on bit level shuffling. First, a generalized version of the chaotic map proposed in [8] was developed and its dynamical behaviour was studied through computation of its bifurcation diagrams and Lyapunov exponent diagrams. The map showcased large regions of chaotic behaviour for high parameter values, which makes it suitable for encryption.

Based on the proposed map a secure pseudo-random bit generator was developed. Then, an image encryption system was designed. The system is based on first decomposing an image into its pixel levels, and then performing a shuffling of the bits with respect to each row, column, and bit level. This is followed by an XOR operation, that yields the encrypted image. The resulting image was shown to be random and secure with respect to a collection of different tests and measures. Finally, the design was implemented in a GUI, for ease of use by any interested individual.

Extensions of this work will consider the application to colour images and the combination of this technique with DNA coding. Moreover, steps will be taken to improve the limitations of this work, which is mainly its execution time, by considering permutation only among the most significant bit levels, and also the use of compressed sensing techniques, to reduce file size before encryption. Also, problems like watermarking are of interest.

Acknowledgements The authors would like to thank the anonymous reviewers for their comments that contributed to the improvement of the manuscript's quality.

Appendix

Algorithm 1 Image Encryption

Input: A $M \times N$ grayscale image $\mathcal{A} = [a_{ij}]$, $i = 1, ..., M$, $j = 1, ..., N$.
Four chaotic maps of the form (2) with parameters $x_0, \mu_1, a_1, ,y_0, \mu_2, a_2, z_0, \mu_3, a_3, w_0, \mu_4, a_4$, computed in Step 1.

Output: An encrypted $M \times N$ grayscale image $\mathcal{E}$.

Step 1. Compute the parameters of the chaotic maps as

$$\lambda = mod\left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} a_{ij}}{M \cdot N} + \text{Ent}(A), 1\right) \tag{15a}$$

$$x_0 = \lambda, \qquad \mu_1 = 900 + \lambda, \qquad a_1 = 2\pi\lambda \tag{15b}$$

$$y_0 = mod(10^6\lambda, 1), \qquad \mu_2 = 901 + y_0, \qquad a_2 = 2\pi y_0 \tag{15c}$$

$$z_0 = mod(10^9\lambda, 1), \qquad \mu_3 = 902 + z_0, \qquad a_3 = 2\pi z_0 \tag{15d}$$

$$v_0 = \cos(\lambda), \qquad \mu_4 = 905 + v_0, \qquad a_4 = 2\pi v_0 \tag{15e}$$

where λ is dependent on the average pixel value of image A, as well as its entropy $\text{Ent}(A)$.

Step 2. Transform the $M \times N$ image $\mathcal{A}$ into its binary 3D matrix representation $\mathcal{A}_{bin}$ of dimension $M \times N \times 8$.

Step 3. Perform row-level shuffling of the 3D binary matrix $\mathcal{A}_{bin}$. For each individual row $i = 1, ..., M$, consider the $N \times 8$ binary matrix $\mathcal{R}_i = A_{bin}(i, :, :)$, that is extracted from A_{bin} by taking all the columns set at row i, in all pixel levels. Then, for each $\mathcal{R}_i$ perform the following:

Step 3.a. Starting from x_0 for the first row matrix $i = 1$, iterate the chaotic map x_k and compute $p_{\mathfrak{r}}^i = \lfloor N|x_k| \rfloor + 1$, until N distinct integers are generated in the interval $[1, N]$, that is $\{p_{\mathfrak{r},1}^i, ..., p_{\mathfrak{r},N}^i\}$. These indexes denote the row permutations for the 2D matrix $\mathcal{R}_i$. So the p_1^i row of the matrix $\mathcal{R}_i$ is moved to the first row, the p_2^i row is moved to the second row and so are the rest. This procedure is equivalent to the left multiplication $L_i\mathcal{R}_i$, where L_i is an invertible matrix where each row n has zero entries, and 1 in the position $p_{\mathfrak{r},n}^i$.

Step 3.b. Similarly, setting as x_0 the value of the last iteration of the map in the previous step, compute $q_{\mathfrak{r}}^i = \lfloor 8|x_k| \rfloor + 1$, until 8 distinct integers are generated in the interval $[1, 8]$, that is $\{q_{\mathfrak{r},1}^i, ..., q_{\mathfrak{r},8}^i\}$. These indexes denote the column permutations for the 2D matrix $\mathcal{R}_i$. So the q_1^i column of the matrix $\mathcal{R}_i$ is moved to the first column and so on. This procedure is equivalent to the right multiplication $\mathcal{R}_i Q_i$, where Q_i is an invertible matrix where each column n has zero entries, and 1 in the position $q_{\mathfrak{r},n}^i$.

Step 3.c. Repeat this procedure for all matrices $\mathcal{R}_i$, $i = 2, ..., M$. For each individual matrix, the initial value of the chaotic map x_0 is taken as the value of the final iteration from the previous step.
The resulting shuffled $M \times N \times 8$ matrix is the concatenation of the shuffled matrices $L_i\mathcal{R}_i Q_i$, denoted as $\mathcal{R}$.

Step 4. Perform column-wise shuffling of the 3D binary matrix $\mathcal{R}$. For each individual column $j = 1, ..., N$, consider the $M \times 8$ binary matrix $C_j = \mathcal{R}(:, j, :)$, that is extracted from $\mathcal{R}$ by taking all the rows set at column j, in all pixel levels. Then for each C_j perform the following:

Algorithm Image Encryption (Continued.)

Step 4. (Cont.)

Step 4.a. Starting from y_0 for the first column $j = 1$, iterate the chaotic map y_k and compute $p_{\mathtt{c}}^j = \lfloor M|y_k|\rfloor + 1$, until M distinct integers are generated in the interval $[1, M]$, that is $\{p_{\mathtt{c},1}^j, ..., p_{\mathtt{c},M}^j\}$. These indexes denote the row permutations for the 2D matrix C_j, which are performed similarly to Step 3a.

Step 4.b. Similarly, setting as y_0 the value of the last iteration of the map in the previous step, compute $q_{\mathtt{c}}^i = \lfloor 8|y_k|\rfloor + 1$, until 8 distinct integers are generated in the interval $[1, 8]$, that is $\{q_{\mathtt{c},1}^j, ..., q_{\mathtt{c},8}^j\}$. These indexes denote the column permutations for the 2D matrix C_j, which are performed similarly to Step 3b.

Step 4.c. Repeat this procedure for all matrices C_j, $j = 2, ..., M$. For each individual matrix, the initial value of the chaotic map y_0 is taken as the value of the final iteration from the previous step.

The resulting column shuffled $M \times N \times 8$ matrix is the concatenation of the shuffled matrices $L_j C_j Q_j$, denoted as C.

Step 5. Perform bit level shuffling of the 3D binary matrix C. For each individual bit level $w = 1, ..., 8$, consider the $M \times N$ binary matrix $\mathcal{P}_w = C(:, :, w)$, that is extracted from C by taking all the rows and columns set at bit level w. Then for each $\mathcal{P}_w$ perform the following:

Step 5.a. Starting from z_0 for the first bit level $w = 1$, iterate the chaotic map z_k and compute $p_{\mathtt{p}}^w = \lfloor M|z_k|\rfloor + 1$, until M distinct integers are generated in the interval $[1, M]$, that is $\{p_{\mathtt{p},1}^w, ..., p_{\mathtt{p},M}^w\}$. These indexes denote the row permutations for the 2D matrix $\mathcal{P}_w$, which are performed similarly to Steps 3a. and 4a.

Step 5.b. Similarly, setting as z_0 the value of the last iteration of the map in the previous step, compute $q_{\mathtt{p}}^w = \lfloor N|z_k|\rfloor + 1$, until N distinct integers are generated in the interval $[1, N]$, that is $\{q_{\mathtt{p},1}^w, ..., q_{\mathtt{p},N}^w\}$. These indexes denote the column permutations for the 2D matrix $\mathcal{P}_w$, which are performed similarly to Steps 3b and 4b.

Step 5.c. Repeat this procedure for all matrices $\mathcal{P}_w$, $w = 2, ..., 8$. For each individual matrix, the initial value of the chaotic map z_0 is taken as the value of the final iteration from the previous step.

The resulting pixel shuffled $M \times N \times 8$ matrix is the concatenation of the shuffled matrices $L_w \mathcal{P}_w Q_w$, denoted as $\mathcal{P}$.

Step 6. After all three shuffling levels are completed, the resulting matrix $\mathcal{P}$ is reshaped into a vector of length $M \cdot N \cdot 8$ and is XORed using a bitstream of the same length, generated from the fourth chaotic map with key values v_0, μ_4, a_4. The resulting bitstream is denoted $\mathcal{E}_{\text{stream}}$.

Step 7. The bitstream $\mathcal{E}_{\text{stream}}$ is reshaped into an $M \times N \times 8$ matrix $\mathcal{E}_{bin}$, which generates the encrypted image $\mathcal{E}$.

References

1. Tang, X., Mandal, S.: Encrypted physical layer communications using synchronized hyperchaotic maps. IEEE Access **9**, 13286–13303 (2021)
2. Ali, Z., Imran, M., Alsulaiman, M., Shoaib, M., Ullah, S.: Chaos-based robust method of zero-watermarking for medical signals. Fut. Gener. Comput. Syst. **88**, 400–412 (2018)

3. Teh, J.S., Tan, K., Alawida, M.: A chaos-based keyed hash function based on fixed point representation. Cluster Comput. **22**(2), 649–660 (2019)
4. Liu, L., Miao, S., Cheng, M., Gao, X.: A pseudorandom bit generator based on new multi-delayed chebyshev map. Inf. Process. Lett. **116**(11), 674–681 (2016)
5. Abdelfatah, R.I., Nasr, M.E., Alsharqawy, M.A.: Encryption for multimedia based on chaotic map: several scenarios. Multimed. Tools Appl. **79**(27), 19717–19738 (2020)
6. Kumar, M., Saxena, A., Vuppala, S.S.: A survey on chaos based image encryption techniques. In: Multimedia Security Using Chaotic Maps: principles and Methodologies, pp. 1–26. Springer (2020)
7. Hua, Z., Zhou, B., Zhou, Y.: Sine chaotification model for enhancing chaos and its hardware implementation. IEEE Trans. Ind. Electron. **66**(2), 1273–1284 (2018)
8. Talhaoui, M.Z., Wang, X., Midoun, M.A.: A new one-dimensional cosine polynomial chaotic map and its use in image encryption. In: The Visual Computer, pp. 1–11 (2020)
9. Liu, L., Miao, S.: A new simple one-dimensional chaotic map and its application for image encryption. Multimed. Tools Appl. **77**(16), 21445–21462 (2018)
10. Zhou, Y., Bao, L., Chen, C.P.: A new 1d chaotic system for image encryption. Signal Process. **97**, 172–182 (2014)
11. Murillo-Escobar, M.A., Cruz-Hernández, C., Cardoza-Avendaño, L., Méndez-Ramírez, R.: A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. Nonlinear Dyn. **87**(1), 407–425 (2017)
12. Alawida, M., Samsudin, A., Teh, J.S.: Enhanced digital chaotic maps based on bit reversal with applications in random bit generators. Inf. Sci. **512**, 1155–1169 (2020)
13. Tutueva, A.V., Nepomuceno, E.G., Karimov, A.I., Andreev, V.S., Butusov, D.N.: Adaptive chaotic maps and their application to pseudo-random numbers generation. Chaos Solitons Fractals **133**, 109615 (2020)
14. Sathya, K., Premalatha, J., Rajasekar, J.: Investigation of strength and security of pseudo random number generators. In: IOP Conference Series: materials Science and Engineering, vol. 1055, pp. 012076. IOP Publishing (2021)
15. Ayubi, P., Setayeshi, S., Rahmani, A.M.: Deterministic chaos game: a new fractal based pseudo-random number generator and its cryptographic application. J. Inf. Secur. Appl. **52**, 102472 (2020)
16. Belazi, A., Talha, M., Kharbech, S., Xiang, W.: Novel medical image encryption scheme based on chaos and DNA encoding. IEEE Access **7**, 36667–36681 (2019)
17. Zhu, C., Wang, G., Sun, K.: Cryptanalysis and improvement on an image encryption algorithm design using a novel chaos based S-box. Symmetry **10**(9), 399 (2018)
18. Zhang, D., Chen, L., Li, T.: Hyper-chaotic color image encryption based on transformed zigzag diffusion and RNA operation. Entropy **23**(3), 361 (2021)
19. Moysis, L., Tutueva, A., Christos, K., Butusov, D.: A chaos based pseudo-random bit generator using multiple digits comparison. Chaos Theory Appl. **2**(2), 58–68 (2020)
20. Moysis, L., Kafetzis, I., Volos, C., Tutueva, A.V., Butusov, D.: Application of a hyperbolic tangent chaotic map to random bit generation and image encryption. In: 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), pp. 559–565. IEEE (2021)
21. Gao, T., Chen, Z.: A new image encryption algorithm based on hyper-chaos. Phys. Lett. A **372**(4), 394–400 (2008)
22. Irani, B.Y., Ayubi, P., Jabalkandi, F.A., Valandar, M.Y., Barani, M.J.: Digital image scrambling based on a new one-dimensional coupled sine map. Nonlinear Dyn. **97**(4), 2693–2721 (2019)
23. Li, Z., Peng, C., Tan, W., Li, L.: A novel chaos-based color image encryption scheme using bit-level permutation. Symmetry **12**(9), 1497 (2020)
24. Ge, R., Yang, G., Jiasong, W., Chen, Y., Coatrieux, G., Luo, L.: A novel chaos-based symmetric image encryption using bit-pair level process. IEEE Access **7**, 99470–99480 (2019)
25. Liu, H., Wang, X.: Color image encryption using spatial bit-level permutation and high-dimension chaotic system. Opt. Commun. **284**(16–17), 3895–3903 (2011)

26. Teng, L., Wang, X.: A bit-level image encryption algorithm based on spatiotemporal chaotic system and self-adaptive. Opt. Commun. **285**(20), 4048–4054 (2012)
27. Kar, M., Mandal, M.K., Nandi, D., Kumar, A., Banik, S.: Bit-plane encrypted image cryptosystem using chaotic, quadratic, and cubic maps. IETE Tech. Rev. **33**(6), 651–661 (2016)
28. Raza, S.F., Satpute, V.: A novel bit permutation-based image encryption algorithm. Nonlinear Dyn. **95**(2), 859–873 (2019)
29. Shahna, K.U., Mohamed, A.: A novel image encryption scheme using both pixel level and bit level permutation with chaotic map. Appl. Soft Comput. **90**, 106162 (2020)
30. Murillo-Escobar, M.A., Meranza-Castillón, M.O., López-Gutiérrez, R.M., Cruz-Hernández, C.: A chaotic encryption algorithm for image privacy based on two pseudorandomly enhanced logistic maps. In: Multimedia Security Using Chaotic Maps: principles and Methodologies, pp. 111–136. Springer (2020)
31. Zhou, Y., Bao, L., Chen, C.L.P.: Image encryption using a new parametric switching chaotic system. Signal Process. **93**(11), 3039–3052 (2013)
32. Wang, X., Liu, L., Zhang, Y.: A novel chaotic block image encryption algorithm based on dynamic random growth technique. Opt. Lasers Eng. **66**, 10–18 (2015)
33. Hsiao, H.-I., Lee, J.: Fingerprint image cryptography based on multiple chaotic systems. Signal Process. **113**, 169–181 (2015)
34. Kari, A.P., Navin, A.H., Bidgoli, A.M., Mirnia, M.: A new image encryption scheme based on hybrid chaotic maps. Multimed. Tools Appl. **80**(2), 2753–2772 (2021)
35. Nepomuceno, E.G., Nardo, L.G., Arias-Garcia, J., Butusov, D.N., Tutueva, A.: Image encryption based on the pseudo-orbits from 1D chaotic map. Chaos: Interdiscip. J. Nonlinear Sci. **29**(6), 061101 (2019)
36. Niu, Y., Zhang, X.: A novel plaintext-related image encryption scheme based on chaotic system and pixel permutation. IEEE Access **8**, 22082–22093 (2020)
37. Zhou, S., He, P., Kasabov, N.: A dynamic DNA color image encryption method based on SHA-512. Entropy **22**(10), 1091 (2020)
38. Gopalakrishnan, T., Ramakrishnan, S.: Chaotic image encryption with hash keying as key generator. IETE J. Res. **63**(2), 172–187 (2017)
39. Zhu, S., Zhu, C., Wang, W.: A new image encryption algorithm based on chaos and secure hash SHA-256. Entropy **20**(9), 716 (2018)
40. Sun, C., Wang, E., Zhao, B.: Image encryption scheme with compressed sensing based on a new six-dimensional non-degenerate discrete hyperchaotic system and plaintext-related scrambling. Entropy **23**(3), 291 (2021)
41. Alvarez, G., Li, S.: Some basic cryptographic requirements for chaos-based cryptosystems. Int. J. Bifurc. Chaos **16**(08), 2129–2151 (2006)
42. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-Allen and Hamilton Inc Mclean Va (2001)
43. Safack, N., Iliyasu, A.M., De Dieu, N.J., Zeric, N.T., Kengne, J., Abd-El-Atty, B., Belazi, A., Abd EL-Latif, A.A.: A memristive RLC oscillator dynamics applied to image encryption. J. Inf. Secur. Appl. **61**, 102944 (2021)
44. Abdelfatah, R.I.: Secure image transmission using chaotic-enhanced elliptic curve cryptography. IEEE Access **8**, 3875–3890 (2019)
45. El-Latif, A.A.A., Abd-El-Atty, B., Belazi, A., Iliyasu, A.M.: Efficient chaos-based substitution-box and its application to image encryption. Electronics **10**(12), 1392 (2021)
46. Wu, Y., Zhou, Y., Saveriades, G., Agaian, S., Noonan, J.P., Natarajan, P.: Local shannon entropy measure with statistical tests for image randomness. Inf. Sci. **222**, 323–342 (2013)
47. Vaidyanathan, S., Sambas, A., Abd-El-Atty, B., Abd El-Latif, A.A., Tlelo-Cuautle, E., Guillén-Fernández, O., Mamat, M., Mohamed, M.A., Alçin, M., Tuna, M., et al.: A 5-D multi-stable hyperchaotic two-disk dynamo system with no equilibrium point: circuit design. FPGA realization and applications to trngs and image encryption. IEEE Access (2021)
48. Chen, J.-X., Zhu, Z.-L., Chong, F., Zhang, L.-B., Zhang, Y.: An efficient image encryption scheme using lookup table-based confusion and diffusion. Nonlinear Dyn. **81**(3), 1151–1166 (2015)

49. Sambas, A., Vaidyanathan, S., Tlelo-Cuautle, E., Abd-El-Atty, B., Abd El-Latif, A.A., Guillén-Fernández, O., Hidayat, Y., Gundara, G., et al.: A 3-D multi-stable system with a peanut-shaped equilibrium curve: circuit design, FPGA realization, and an application to image encryption. IEEE Access **8**, 137116–137132 (2020)
50. Tsafack, N., Sankar, S., Abd-El-Atty, B., Kengne, J., Jithin, K.C., Belazi, A., Mehmood, I., Bashir, A.K., Song, O.Y., Abd El-Latif, A.A.: A new chaotic map with dynamic analysis and encryption application in internet of health things. IEEE Access **8**, 137731–137744 (2020)
51. Alanezi, A., Abd-El-Atty, B., Kolivand, H., El-Latif, A., Ahmed, A., El-Rahiem, A., Sankar, S., Khalifa, H., et al.: Securing digital images through simple permutation-substitution mechanism in cloud-based smart city environment. Secur. Commun. Netw. **2021** (2021)
52. Yao, L., Yuan, C., Qiang, J., Feng, S., Nie, S.: Asymmetric color image encryption based on singular value decomposition. Opt. Lasers Eng. **89**, 80–87 (2017)
53. Hua, Z., Zhou, Y., Pun, C.M., Chen, C.P.: 2D sine logistic modulation map for image encryption. Inf. Sci. **297**, 80–94 (2015)
54. Zhu, C., Wang, G., Sun, K.: Improved cryptanalysis and enhancements of an image encryption scheme using combined 1D chaotic maps. Entropy **20**(11), 843 (2018)
55. Nestor, T., De Dieu, N.J., Jacques, K., Yves, E.J., Iliyasu, A.M., El-Latif, A.: A multidimensional hyperjerk oscillator: dynamics analysis, analogue and embedded systems implementation, and its application as a cryptosystem. Sensors **20**(1), 83 (2020)