IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 March 2025, accepted 10 May 2025, date of publication 22 May 2025, date of current version 3 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3572589

RESEARCH ARTICLE

Exploiting Circular Shifts for Efficient Chaotic Image Encryption

LAZAROS MOYSIS[1,2], MARCIN LAWNIK[3], WASSIM ALEXAN[4], (Senior Member, IEEE), SOTIRIOS K. GOUDOS[5], (Senior Member, IEEE), MURILO S. BAPTISTA[6], AND GEORGE F. FRAGULIS[1]

[1]Department of Electrical and Computer Engineering, University of Western Macedonia, 50100 Kozani, Greece
[2]Department of Mechanical Engineering, University of Western Macedonia, 50100 Kozani, Greece
[3]Department of Mathematical Methods in Technics and Informatics, Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland
[4]Communications Department, Faculty of Information Engineering and Technology, German University in Cairo, Cairo 11835, Egypt
[5]ELEDIA@AUTH, School of Physics, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece
[6]Department of Physics, Institute for Complex Systems and Mathematical Biology, SUPA, The University of Aberdeen, AB24 3UX Aberdeen, U.K.

Corresponding authors: Lazaros Moysis (l.moysis@uowm.gr) and Marcin Lawnik (marcin.lawnik@polsl.pl)

ABSTRACT This work presents a chaotic image encryption algorithm that acts on the binary level of the image for effective permutation, and on the byte level for substitution. The permutation step combines actions of bit level rearranging, and bit shuffling. Circular shift operations are used for bit shuffling, to reduce the execution time. Circular shifts are also applied to shuffle the permutation and substitution rules, effectively increasing security at a very low execution cost. The encryption keys are plaintext dependent, and a key mixing step in included, so that each part of the key affects all the subsequent encryption operations. For the encryption, a new Soboleva hyperbolic tangent based map is first designed. The dynamical behavior of the map is studied and shows robust chaos, and an absence of equilibria, meaning that it can generate hidden attractors. This map is used as an entropy source, along with a chaotic pseudo random number generator (PRNG). The encryption process shows a good performance under a collection of tests, and has also a low execution time, 0.0134 sec for a 256×256 image, 0.0530 sec for a 512×512 image, and 0.2050 sec for a 1024×1024 image. So the proposed algorithm is a viable option for fast, efficient, and secure data encryption.

INDEX TERMS Bit level, bit plane, chaos, chaotification, encryption, hidden attractor, pseudo random number generator, Soboleva.

I. INTRODUCTION

A. DATA SECURITY

The rapid proliferation of digital technologies and the widespread exchange of sensitive information in recent years have elevated information security to a critical priority [1]. Protecting the confidentiality and integrity of digital content, such as personal communications, sensitive organizational data, and multimedia data, is essential in diverse fields and IoT ecosystems, including medical and satellite imaging, where security breaches could have severe consequences [2], [3], [4]. Traditional cryptographic algorithms, such as the Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA), have demonstrated their efficacy in securing text and numerical data. However, their direct application to multimedia data, particularly images, introduces significant challenges due to the inherent structural and statistical properties of such data. Consequently, recent years have witnessed the introduction of modified versions of these algorithms to accommodate image encryption applications [5], [6], [7].

Unlike textual data, images are characterized by high dimensionality, strong inter-pixel correlations, and the frequent requirement for real-time processing, making them distinctly different from unstructured data. Conventional encryption algorithms, which are computationally demanding and primarily designed for unstructured or sequential data, often fail to efficiently address these requirements. For instance, applying AES directly to images may result

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masucci.

in substantial computational overhead and suboptimal performance, particularly in real-time scenarios or in cases involving large-scale image transmission. Moreover, the structural regularities of image data can occasionally persist after encryption, exposing encrypted images to pattern-based cryptanalytic vulnerabilities [8]. These limitations necessitate the development of specialized encryption techniques tailored to the unique properties of image data.

B. CHAOTIC ENCRYPTION

Chaos-based cryptography has become a foundational approach in information security, with diverse applications such as secure communications [9], watermarking [10], hashing [11], random number generation [12], and data masking for various types of content, including text, audio, and images [13], [14]. Its adaptability and effectiveness have made it a valuable tool across multiple domains. Building on this foundation, chaos-based encryption has emerged as a highly effective paradigm for addressing the unique challenges of image encryption. By leveraging the intrinsic properties of chaotic systems, such as sensitivity to initial conditions, unpredictability, and ergodicity, these algorithms are exceptionally suited for disrupting the statistical correlations inherent in image pixels [15]. Chaos theory enables the design of efficient permutation and substitution operations, ensuring a high degree of security [16]. As chaotic systems are deterministic in nature, the encryption rules can be replicated and reversed by any party that knows the secret keys of the system, and can correctly simulate its solution. Thus, chaotic systems are especially suited for symmetric encryption. In addition, the lightweight computational frameworks and inherent randomness of chaos-based methods make them particularly advantageous for real-time applications [17]. The integration of chaotic maps and pseudo-random number generators further enhances their security and robustness, ensuring resilience against a wide range of cryptanalytic attacks [18].

C. LITERATURE REVIEW

Recent advancements in image encryption have explored various approaches to address the limitations of traditional algorithms when applied to multimedia data. Among these, chaos-based methods have gained significant attention due to their ability to leverage the inherent properties of chaotic systems for secure and efficient encryption. Numerous studies have focused on combining chaotic maps with innovative techniques, such as bit-level operations and circular shifts, to enhance encryption performance while maintaining low computational complexity [16], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40].

1) CHAOS-BASED IMAGE ENCRYPTION TECHNIQUES

Several works have utilized chaos-based systems for pixel-level rotation, randomization, and bit-level operations. For example, in [19], continuous Chen and Chua systems are employed to carry out pixel rotation and randomization. In contrast, [20] integrates both continuous and discrete chaotic systems to perform cycle shifting at the bit and pixel levels across all color channels. Similarly, [21] presents a modified pulsed-coupled spiking neurons circuit map, introducing a new bit-pair-level rotation technique, while [22] focuses on color image encryption by appending all bit planes for all three channels into a single matrix, followed by row and column shuffling.

2) BIT-LEVEL OPERATIONS IN IMAGE ENCRYPTION

Bit-level manipulation has been a common focus in chaos-based encryption methods. For instance, [23] divides the plain image into two sets, with the first four and last four bit levels being processed separately, allowing the lower part to encrypt the other. In [24], bit-level encryption is extended to include either all, or a selected number of bit levels. A novel bit-level shuffling strategy inspired by Rubik's cube rotations is proposed in [39]. The authors of [25] design a Hilbert curve scanning path for cyclic shifts and shuffling, enhancing encryption robustness. In [26], a time-varying delayed exponentially controlled chaotic system is combined with a shift-coupled pixel-level diffusion algorithm for region-of-interest (ROI) encryption.

3) ADVANCED CHAOTIC MAPS AND HYBRID TECHNIQUES

Innovative chaotic maps and hybrid techniques have also been explored. For instance, [27] introduces a 2D cross-hyperchaotic sine-modulation-logistic map that enables bit-level confusion using zigzag transformations, followed by cross-coupled data diffusions. The work in [28] proposes a bit-level encryption system using chaotic maps and permutations, where bit-plane generation, shuffling with logistic maps, and vertical/horizontal concatenation are employed alongside hash-based keys for enhanced security. Similarly, [29] utilizes a novel 2D sine iterative chaotic map with infinite collapse for triple-stage bit-level encryption, while [30] combines a 1D zero-excluded chaotic map with hash functions to derive encryption keys targeting the first six bits of each pixel.

4) SPECIALIZED CHAOTIC MODELS AND TRANSFORMATIONS

Other studies focus on unique chaotic models and transformations to improve encryption robustness. In [31], a chaotic Bessel map is used for bit-wise permutations and substitutions, while [32] employs a zigzag scrambling technique alongside chain diffusion and discrete wavelet transform (DWT) to enhance robustness. Similarly, [33] introduces a fractal sorting vector (FSV) for global bit-level chaotic permutation, which improves security for images of varying sizes. In [34], a fast bit-level chaotic encryption algorithm adjusts binary distribution to minimize differences between 0s and 1s, ensuring efficiency and security.

5) SPATIOTEMPORAL CHAOS AND COMPLEX SYSTEMS

Spatiotemporal chaos models have also been adopted to improve encryption performance. For instance, [35] proposes a cross-coupled map lattice (CCML) that replaces the logistic map with a tent map to address weak chaotic behavior, using modular operations for secure encryption. A dynamic coupled perturbation map lattice (DCPML) is introduced in [40], where variable functions, dynamic coupling coefficients, and disturbance terms enhance pseudo-randomness in encrypted outputs. Additionally, [37] proposes a new combination chaotic system (NCCS) for bit-level encryption, integrating plaintext-related key streams from SHA-512 with confusion and diffusion operations.

6) HYPERCHAOTIC SYSTEMS AND FPGA IMPLEMENTATIONS

Hyperchaotic systems have been employed to further enhance encryption security. In [38], a hyperchaotic Chen system is combined with bit-level permutations and multi-directional scrambling, utilizing 2D XOR forward and backward diffusion for efficient encryption. Similarly, [16] employs three hyperchaotic systems (memristor, hyperchaotic 7D, and Erbium-doped fiber laser systems) to design a three-stage encryption algorithm with pseudo-random number generators for S-boxes and keys. An FPGA implementation achieves an encryption rate of 0.67 Gbps, demonstrating the practical applicability of their approach. In [41], a hyperchaotic family of systems was designed using Apéry's constant, and applied to image encryption.

D. MOTIVATION

In all of the techniques cited in the previous section, different forms of permutation and substitution are explored, in a collective effort to find the best balance between security and computational efficiency. For permutation, there is an ongoing effort to develop efficient ways to shuffle the plaintext information at the lowest possible computational cost. The permutation can be performed either on the byte level, or the bit level. On the byte level, the pixels of an image are shuffled, either by shuffling the rows and columns, or by shuffling each individual pixel. The final goal is to change the position of all the pixels in the image, as well as their relative positions. When permutation is performed on the bit level, the pixel values are broken into their binary representation, and then permuted. The effect here is much stronger compared to pixel permutation, as the resulting image will have different pixel values compared to the plaintext. Thus, permutation on the binary level has a partial substitution effect on the pixel level, making it better than pixel permutation. The downside is that as the number of bits is 8 times that of the pixels, bit permutation can be computationally heavier.

The action of substitution can also be performed through different techniques, although the use of the exclusive OR operation (XOR) is more universally accepted as the best choice, due to its speed and security. So what is commonly explored in the literature are the rules with which the XOR is applied for substitution. For example, performing XOR between multiple pixels, or changing the order in which the pixels are XORed across the rows and columns of the matrix, are all possible variations of the substitution step.

Of course, each variation on the design of the permutation and substitution steps has an affect on the security and speed of encryption. Motivated by this, the present work develops an encryption technique that performs permutation on the binary level and substitution on the pixel level, in a highly efficient way, without compromising security. The key actions taken to achieve this is to utilize circular shift operations in several parts of the design. First, the bit permutation is performed through circular shifts. But more importantly, the rules of permutation and substitution are themselves circularly shifted, before being applied to the plaintext information. This action increases security at a very low execution cost. Indeed, the encryption achieves a very high speed, and has resistance to all the standard statistical attacks. The novelties of the work are outlined in the next section.

E. CONTRIBUTIONS

The proposed methodology is inspired from the earlier work [42] that arranges the bit planes on a 3D matrix and uses shuffling on the binary level, and [22] that arranges the bit levels in a single 2D matrix. Several modifications are performed, which give novelty to the algorithm, and contribute to its high performance with respect to security and execution speed. Moreover, a new chaotic map is used for encryption, which satisfies the criteria of robust chaos, and a high key space, both desirable in encryption. The contributions can be outlined as follows:

1) A new chaotic map is introduced, following the family of chaotic maps without equilibria (also termed maps with hidden attractors) introduced in [43]. The new map utilizes the Soboleva hyperbolic tangent function [44] in place of the standard tanh. This is among the first chaotic maps that utilize the Soboleva hyperbolic tangent.
2) The new chaotic map shows robust chaos and an absence of equilibria for appropriate parameter values. So it belongs to the family of maps with hidden attractors [43], [45], [46]. It also has a high key space, which is required for encryption.
3) The performance of the map as a seed in a Pseudo Random Number Generator (PRNG) from [47] is evaluated. The generator passes the statistical tests succesfully, and achieves a speed of 16.86 Mbytes/sec.
4) An encryption algorithm is designed with the goal of achieving high security, and effective permutation and substitution at a low cost. This is achieved by a series of actions:
   a) Plaintext dependency through the use of SHA-256, and a key re-initialization technique, that ensures that all parts of the key effect each encryption step.
   b) The random arrangement of the bit planes into a 2D matrix.

c) The row and column shifting of this binary matrix using circular shifts for high execution speed. An additional shifting is performed to the shuffling rules before applying them to the binary matrix, for increased security at a low execution cost.
d) The random rearrangement of the shuffled bit planes into a grayscale image.
e) The substitution through byte-wise XOR, using the chaotic PRNG mentioned earlier. An additional shifting is performed to bytes generated from the PRBG prior to executing the XOR, for increased security at a low execution cost.

5) The encryption shows resistance against a collection of the commonly considered attacks.
6) The analysis of alteration attacks on the ciphertext is performed in a systematic way by considering ciphertext alterations of increasing intensity, and measuring the decryption performance using the Structural Similarity Index, Peak Signal to Noise Ratio, and Spectral Distortion.
7) The analysis of execution time is also performed in a systematic way by considering images of increasing size, from 16×16 up to 2048×2048. The execution speed is indeed low, 0.0134 sec for a 256×256 image, 0.0530 sec for a 512×512 image, and 0.2050 sec for a 1024×1024 image. The execution time curve is shown to follow a power law with respect to the image size increase.

All of the above positive results justify the introduction of the proposed algorithm as an improvement upon the existing literature. Moreover, the algorithm has a great potential for further modification and experimentation, so it can be a starting point for further studies, as discussed in the Conclusions.

The rest of the work is structured as follows: In Section II, the map used during the encryption is presented and studied. In Section III, a PRNG variation from [47] is tested using the proposed map. In Section IV, the encryption algorithm is presented. In Section V, the encryption results are presented and evaluated. Finally, Section VI concludes the work with a discussion on future topics of interest.

Finally, some notational remarks: The symbol ':' in $i : j$ denotes the values $i, i+1, i+2, \ldots, j$, following standard MATLAB notation. Also, the notation $A(j, :)$, $A(:, j)$ denotes the j-th row and column of the matrix A respectively.

II. THE CONSIDERED CHAOTIC MAP

A. THE SOBOLEVA-MODULO MAP

For the encryption, a chaotic map has to be used for generating the rules required for performing the actions of permutation and substitution. The map considered for this is a modified version of the modulo-tanh map proposed in [43]. The original map showcased a collection of desirable characteristics, like robust chaos, controllable domain, and absence of equilibria. The original map is given by:

$$x_i = \mathrm{mod}(x_{i-1} + a + b \tanh\left(\frac{x_{i-1}}{K}\right), K), \tag{1}$$

where $a, b, K > 0$ are the control parameters. In the modification considered in this work, the hyperbolic tangent function in (1) is replaced with the Soboleva modified hyperbolic tangent from [44], [48], [49], and [50]:

$$\mathrm{smht}(x) = \frac{e^{Ax} - e^{-Bx}}{e^{Cx} + e^{-Dx}}, \tag{2}$$

where $x \in \mathbb{R}$, and $A, B, C, D \in \mathbb{R}$ are control parameters, with $A \leq C$, $B \leq D$. This function is a generalization of the standard hyperbolic tangent, and introduces four new parameters in the map. Thus, the new map is termed Soboleva-modulo map, and has the following form:

$$x_i = \mathrm{mod}(x_{i-1} + a + b\ \mathrm{smht}\left(\frac{x_{i-1}}{K}\right), K), \tag{3}$$

with $a, b, K, A, B, C, D > 0$, and $A \leq C$, $B \leq D$. For $A = B = C = D = 1$, the function smht is identical to the hyperbolic tangent tanh, so the original map (1) can be considered a special case of the generalized map (3).

The graphs of the map's function are shown in Figs 1, 2, 3, for fixed values a, b, K, and different values of the Soboleva parameters A, B, C, D. It is evident from these figures that changing any one of the Soboleva parameters can significantly affect the graph of the map (3). So the new map (3) is a significant generalization of the original (1).

In [43], it was shown that the map (1) does not have any equilibria under appropriate parameter values. The term for maps that have no equilibria is maps with hidden attractors. This result can be extended to the map (3) proposed here. The proof follows similar steps to the one provided in [43]. It differentiates though from that of the original map (1), because in contrast to the classic hyperbolic tangent function, the Soboleva hyperbolic tangent (2) is not necessarily increasing for positive values of its argument.

Theorem 1: Consider the map (3), where the Soboleva parameters satisfy $A \leq C$, $B \leq D$. The map (3) will have no equilibria if the following condition holds

$$\frac{-a_0 + jK}{b} \notin [0, \max_{x \in [0,1)} [\mathrm{smht}(x)]), \quad \forall j \in \mathbb{Z}, \tag{4}$$

where $a_0 = \mathrm{mod}(a, K)$.

Proof: The map (3) has fixed a point x^*, when the following condition holds:

$$x^* = \mathrm{mod}(x^* + a + b\ \mathrm{smht}\left(\frac{x^*}{K}\right), K). \tag{5}$$

Considering that $a = a_0 + j_0 K$ for an appropriate $j_0 \in \mathbb{N}$, where $a_0 \in [0, K)$, the condition (5) is rewritten as:

$$x^* = \mathrm{mod}(x^* + a_0 + b\ \mathrm{smht}\left(\frac{x^*}{K}\right), K). \tag{6}$$

By the definition of the modulo function, the above condition is equivalent to the following system:

$$x^* = \begin{cases} \vdots & \vdots \\ x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) + K, & \text{if } -K < x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) \le 0 \\ x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right), & \text{if } 0 < x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) \le K \\ x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) - K, & \text{if } K < x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) \le 2K \\ \vdots & \vdots \end{cases} \tag{7}$$

Solving the above system with respect to the Soboleva function gives the following solutions:

$$\mathrm{smht}\left(\frac{x^*}{K}\right) = \begin{cases} \vdots & \vdots \\ \frac{-a_0 - K}{b}, & \text{if } -K < x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) \le 0 \\ \frac{-a_0}{b}, & \text{if } 0 < x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) \le K \\ \frac{-a_0 + K}{b}, & \text{if } K < x^* + a_0 + b\,\mathrm{smht}\left(\frac{x^*}{K}\right) \le 2K \\ \vdots & \vdots \end{cases} \tag{8}$$

Considering that $x^* \in [0, K)$, and $\mathrm{smht}\left(\frac{x^*}{K}\right) : [0, 1) \to [0, \max_{x\in[0,1)}[\mathrm{smht}(x)])$, the system (8) has a solution when there is at least one integer $j \in \mathbb{Z}$, such that:

$$\frac{-a_0 + jK}{b} \in [0, \max_{x\in[0,1)}[\mathrm{smht}(x)]). \tag{9}$$

So the map (3) will have no fixed points when the above condition is not satisfied, or equivalently:

$$\frac{-a_0 + jK}{b} \notin [0, \max_{x\in[0,1)}[\mathrm{smht}(x)]), \quad \forall j \in \mathbb{Z}. \tag{10}$$

■

The condition (4) for the absence of equilibria is hard to simplify any further, as finding the maximum of the Soboleva hyperbolic tangent analytically is hard, due to the existence of the four free parameters A, B, C, D. It can be computed numerically though for fixed parameter values. As an example, Fig. 4 depicts the graph of $F(x) = \mathrm{mod}(x + a + b\,\mathrm{smht}\left(\frac{x}{K}\right), K)$, and a cobweb diagram for a solution of (3) starting from $x_0 = 0.1$, for $A = 0.2$, $B = 0.1$, $C = 1$, $D = 1$, $a = 5.3$, $b = 5$, $K = 1$. The map is chaotic, and its curve does not intersect the bisector, so there are no equilibria. The maximum value of the Soboleva here is numerically computed as $\max_{x\in[0,1)}[\mathrm{smht}(x)] = 0.1026$, and condition (4) becomes:

$$\frac{-a_0 + jK}{b} = \frac{-0.3 + j}{5} = -0.06 + 0.2j \notin [0, 0.1026), \quad \forall j \in \mathbb{Z} \tag{11}$$

which is satisfied.

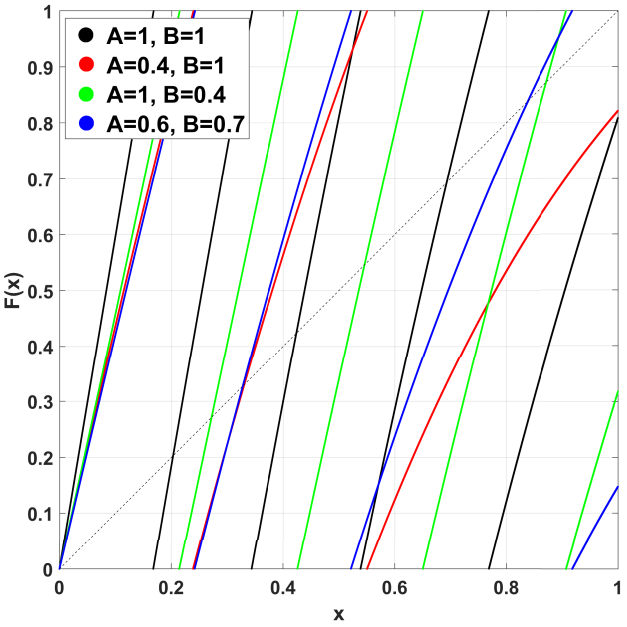

FIGURE 1. Graph of $F(x) = \mathrm{mod}(x + a + b\,\mathrm{smht}\left(\frac{x}{K}\right), K)$, for $a = b = 5$, $K = 1$, $C = D = 1$ and different values of A, B.

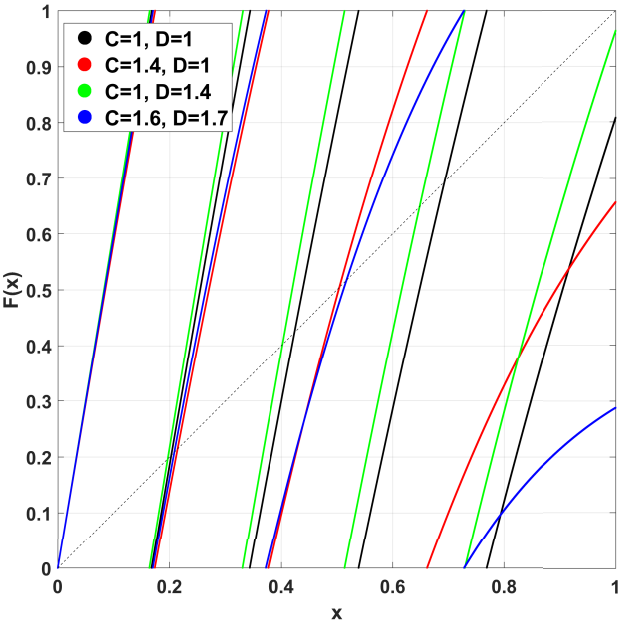

FIGURE 2. Graph of $F(x) = \mathrm{mod}(x + a + b\,\mathrm{smht}\left(\frac{x}{K}\right), K)$, for $a = b = 5$, $K = 1$, $A = B = 1$ and different values of C, D.

B. DYNAMICAL ANALYSIS

In this section, the dynamical behavior of the map is studied. To do so, the density colored bifurcation diagrams and Lyapunov exponent diagrams are computed. The density colored bifurcation diagrams were introduced in [51], and are color-coded versions of the standard bifurcation diagrams, where the color shades represent changes in the statistical distribution of the underlying time series of the map. So the

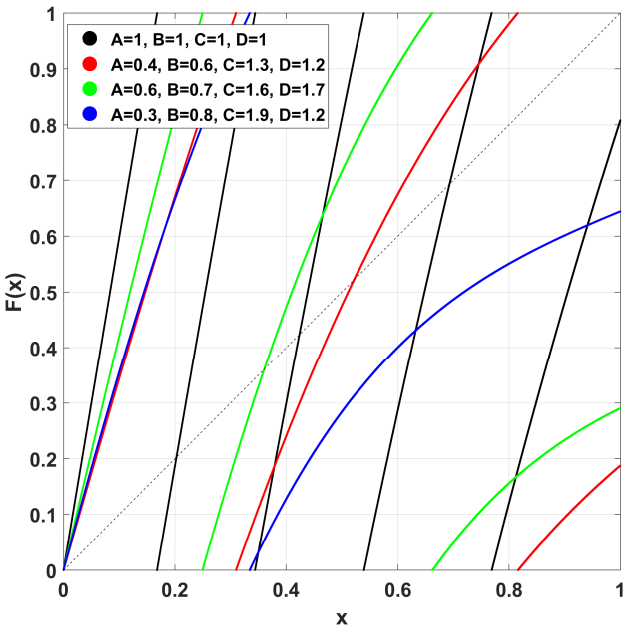

FIGURE 3. Graph of $F(x) = \text{mod}(x + a + b\ \text{smht}\left(\frac{x}{K}\right), K)$, for $a = b = 5$, $K = 1$, and different values of A, B, C, D.

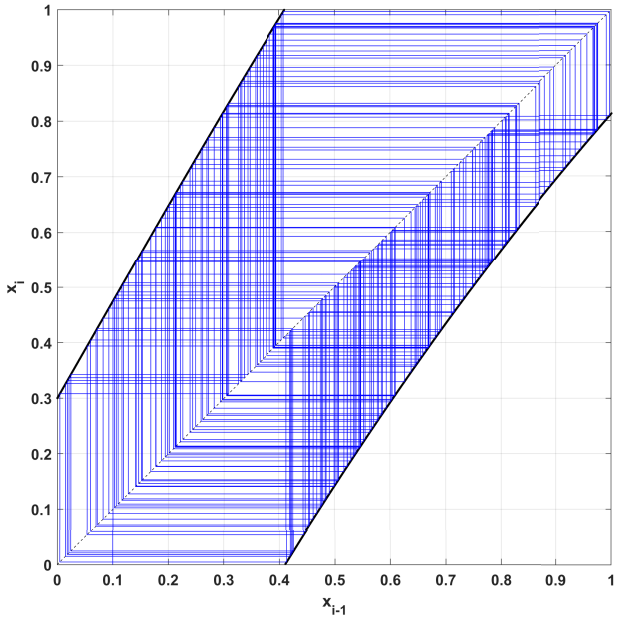

FIGURE 4. Cobweb diagram of (3), for $A = 0.2$, $B = 0.1$, $C = 1$, $D = 1$, $a = 5.3$, $b = 5$, $K = 1$, and $x_0 = 0.1$.

color provides an additional information about the statistical behavior of the map's solution.

In [43], the map (1) showcased robust chaos with respect to changes in both of the parameters a, b, and different values of K. Here, the parameter K, which controls the domain interval, will be fixed to $K = 1$, given that the normalized interval $x_i \in [0, 1)$ is useful when properly computing the encryption operations. Moreover, since the parameters of the Soboleva function are the newly introduced ones, the analysis will focus on the behavior of the map under changes in them. Fig. 5 shows the bifurcation diagram of (3) with respect to A, for $a = b = 5$, $K = 1$, $B = C = D = 1$. It is clear that the map remains chaotic for all the chosen range of A, so it shows robust chaos, which is a desired property for encryption. Observing the color changes, it is also evident that some regions in the state space show higher density, and these regions vary smoothly as A increases. Similarly, robust chaos is observed with respect to changes in parameter B, as can be seen in Fig. 6, where $a = b = 5$, $K = 1$, $A = C = D = 1$. There are no visible periodic regions, and the distribution here appears a bit more uniform compared to the case of varying A.

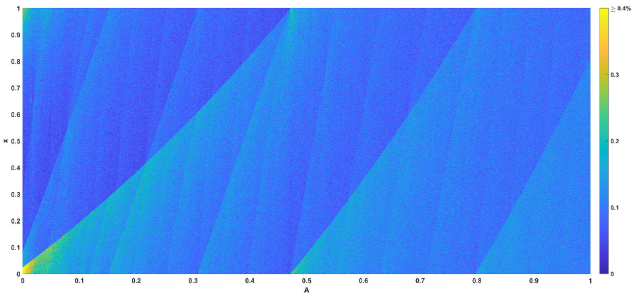

FIGURE 5. Bifurcation diagram of (3) with respect to A, for $a = b = 5$, $K = 1$, $B = C = D = 1$.

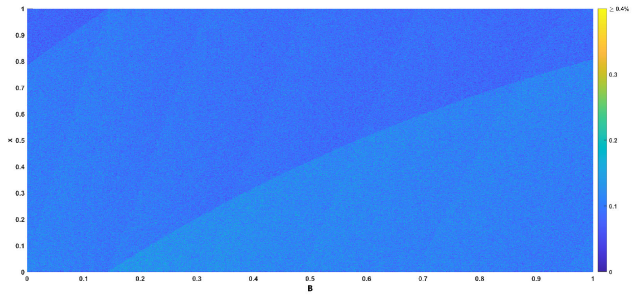

FIGURE 6. Bifurcation diagram of (3) with respect to B, for $a = b = 5$, $K = 1$, $A = C = D = 1$.

When changing parameter C, robust regions also appear, but are interrupted by a periodic region, as can be seen in Fig. 7, for $a = b = 5$, $K = 1$, $A = B = D = 1$. The distribution has some regions with higher density that vary smoothly. Finally, when changing D, robust chaos is observed, as seen in Fig. 8, for $a = b = 5$, $K = 1$, $A = B = C = 1$. The distribution is again more uniform, similar to when changing B.

From those graphs, it is evident that when changing a single Soboleva parameter, the map (3) exhibits robust chaotic behavior. In addition to this observation though, it is important to also evaluate how the map behaves under changes in more parameters. To observe this, two-dimensional Lyapunov exponent diagrams are computed, for

different Soboleva parameter pairs. These diagrams depict the value of the Lyapunov exponent for different parameter values. They are shown in Fig. 9 for (A, B) and $a = b = 5$, $K = 1$, $C = D = 1$, in Fig. 10 for (C, D) and $a = b = 5$, $K = 1$, $A = B = 1$, in Fig. 11 for (A, C) and $a = b = 5$, $K = 1$, $B = D = 1$, and in Fig. 12 for (B, D) and $a = b = 5$, $K = 1$, $A = C = 1$. Observing all four of these diagrams, it is evident that the map (3) exhibits robust chaos for a wide range of the Soboleva parameters. Periodic regions appear for some parameter pairs, which must be avoided when choosing the encryption keys. Another observation that can be made is that the regions between chaotic and non-chaotic regions are clearly defined, so there are no self-similar regions forming, which as a phenomenon appears in other maps [52].

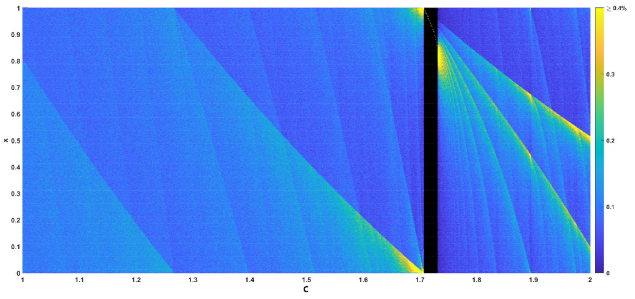

FIGURE 7. Bifurcation diagram of (3) with respect to C, for $a = b = 5$, $K = 1$, $A = B = D = 1$.

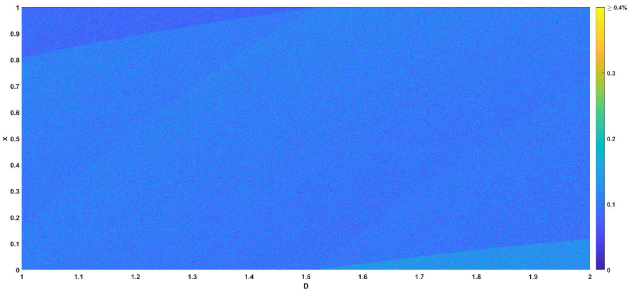

FIGURE 8. Bifurcation diagram of (3) with respect to D, for $a = b = 5$, $K = 1$, $A = B = C = 1$.

Overall, the map exhibits many regions of robust chaos for many different parameter pairs, which is desired for encryption. The statistical distribution observed in the density colored bifurcation diagrams shows small irregularities, which are common, and even more prominent in other maps. But as will be shown in the next section, when the map is used as an input to the PRNG during encryption, the generator's hash function will break this small statistical bias. Moreover, the use of the mod$(\cdot, K)$ operation can control the map's domain, which is helpful in designing the operations of permutation and substitution. The key space of the map is also high, as its keys include an initial condition x_0, and seven control parameters a, b, K, A, B, C, D. So the map (3) is suitable for use as a seed in chaotic encryption.

III. DESIGN OF A PSEUDO-RANDOM NUMBER GENERATOR

Using the chaotic map proposed earlier, a pseudo random number generator (PRNG) must be designed, to be used during the encryption process, at the substitution step. This generator is practically a function that takes as an input the values of the chaotic map in each iteration x_i, and outputs an integer within a specified interval. The PRNG must satisfy several design specifications to be suitable for use. First, it should output a series of numbers that are statistically random. Secondly, it should effectively hide the relation between the input value x_i and the output number. Moreover, these goals must be achieved through a nonlinear function that is as lightweight as possible, so that the overal encryption process will have a sufficiently low execution time.

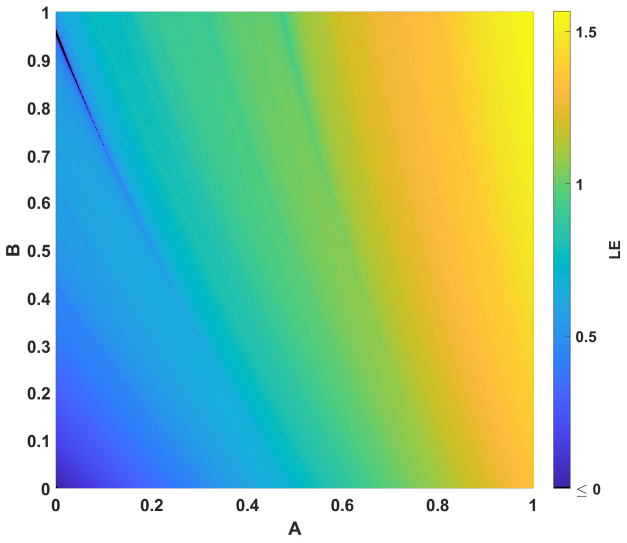

FIGURE 9. Lyapunov exponent values for parameter pairs (A, B), for $a = b = 5$, $K = 1$, $C = D = 1$.

Considering all of the above specifications, the generator used is a modified version of the one from [47]:

$$\mathcal{B}_i = \text{rem}(\lfloor 10^{10} x_i \rfloor, 256). \tag{12}$$

The change made here is that in [47], the result of (12) was then transformed into an 8-bit binary representation, as that work considered a bit generator. This action is discarded here, as the output integer will be used directly for byte-wise XOR in the encryption process. The original generator was tested on ten chaotic maps, and five of them generated statistically random sequences, so it was considered a viable choice for use here.

The statistical randomness of the PRNG (12) using the chaotic map (3) is tested through the NIST statistical test suite [53]. This package consists of 15 statistical tests, that evaluate the randomness of a given time series. To test the generator, a set of $1000 \cdot 125000$ random integers were generated for random initial conditions x_0, and parameter values $a = 5$, $b = 5$, $N = 1$, $A = 0.87$, $B = 0.65$, $C = 1$,

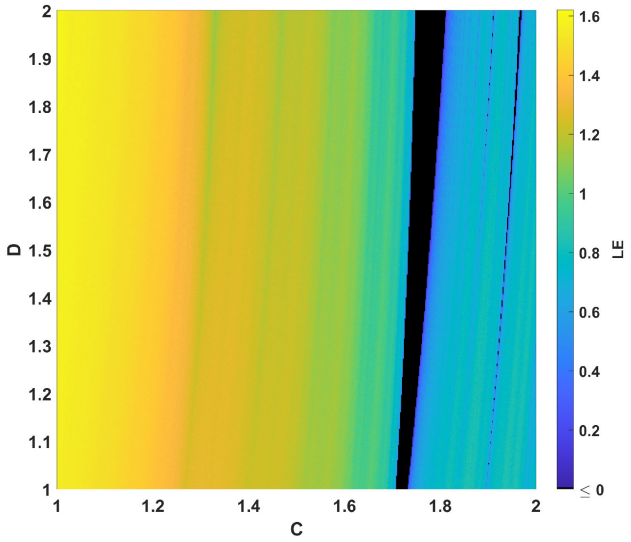

FIGURE 10. Lyapunov exponent values for parameter pairs (C, D), for $a = b = 5$, $K = 1$, $A = B = 1$.

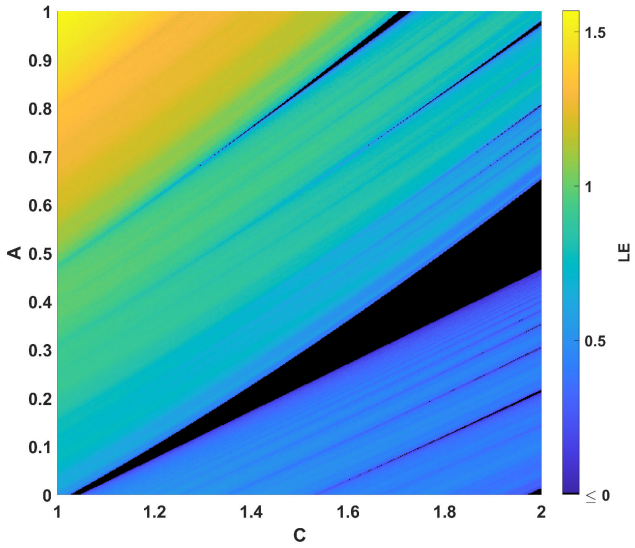

FIGURE 11. Lyapunov exponent values for parameter pairs (A, C), for $a = b = 5$, $K = 1$, $B = D = 1$.

$D = 1$. As the NIST tests require bit values as an input, these integers were transformed into their 8-bit representation, and the output bits were appended into a single file consisting of $1000 \cdot 10^6$ bits, which was used for testing. The results are shown in Table 1, and are all successful, so the generator can be used in the encryption process.

IV. THE ENCRYPTION ALGORITHM

The encryption algorithm is inspired by the previous work [42] that uses shuffling on the binary level, and [22] that arranges the bit levels in a single 2D matrix. The bit levels of an image refer to the individual binary matrices that correspond to each bit position in the 8-bit representation of

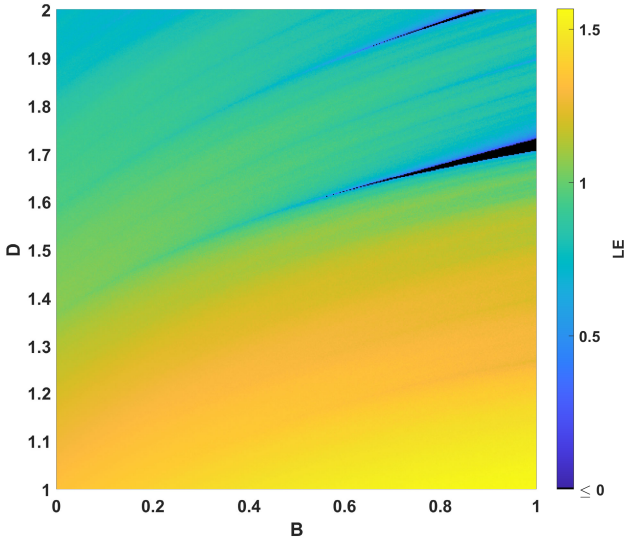

FIGURE 12. Lyapunov exponent values for parameter pairs (B, D), for $a = b = 5$, $K = 1$, $A = C = 1$.

TABLE 1. NIST Test Results.

No.	Test	Chi-square P-Value	Rate	Result
1	Frequency	0.378705	984/1000	✓
2	BlockFrequency	0.655854	991/1000	✓
3	CumulativeSums	0.999230	983/1000	✓
4	Runs	0.670396	988/1000	✓
5	LongestRun	0.262249	985/1000	✓
6	Rank	0.983453	989/1000	✓
7	FFT	0.119508	985/1000	✓
8	NonOverlappingTemplate	0.102526	992/1000	✓
9	OverlappingTemplate	0.686955	991/1000	✓
10	Universal	0.583145	990/1000	✓
11	ApproximateEntropy	0.558502	987/1000	✓
12	RandomExcursions	0.274605	616/625	✓
13	RandomExcursionsVariant	0.421061	623/625	✓
14	Serial	0.967382	991/1000	✓
15	LinearComplexity	0.370262	989/1000	✓

the matrices pixels. The bit representation of the pixels for an $M \times N$ image $\mathcal{A}$ can be described as follows:

$$\mathcal{A} = [a_{i,j}] = 2^0 a^1_{i,j} + 2^1 a^2_{i,j} + 2^2 a^3_{i,j} + 2^3 a^4_{i,j} + 2^4 a^5_{i,j} + 2^5 a^6_{i,j} + 2^6 a^7_{i,j} + 2^7 a^8_{i,j}, i = 1, \ldots, M, j = 1, \ldots, N, \tag{13}$$

where $a_{i,j} \in \{0, \ldots, 255\}$ represent the pixels of the image $\mathcal{A}$, and the superscript $a^n_{i,j} \in \{0, 1\}$, $n = 1, 2, \ldots, 8$ denotes the n-th bit of each pixel's 8-bit binary representation. Appending all of the values $a^n_{i,j}$, $i = 1, \ldots, M$, $j = 1, \ldots, N$ into a single matrix, for each of the bit positions $n = 1, 2, \ldots, 8$, gives the eight individual bit levels $\mathcal{A}_1$, $\mathcal{A}_2$,... $\mathcal{A}_8$ of the image. An example of these eight levels is shown in Fig. 13. It is evident that the higher bit levels carry more significant information about the image. The encryption algorithm performs permutation on the bit levels of an image, which as mentioned in the introduction, is more efficient than pixel permutation.

The encryption algorithm proposed in this work utilizes two chaotic maps of the form (3). The algorithm consists of seven steps, which perform plaintext dependent key generation (step 1), permutation (steps 2-5), and substitution (steps 6-7), which are standard operations in cryptography.

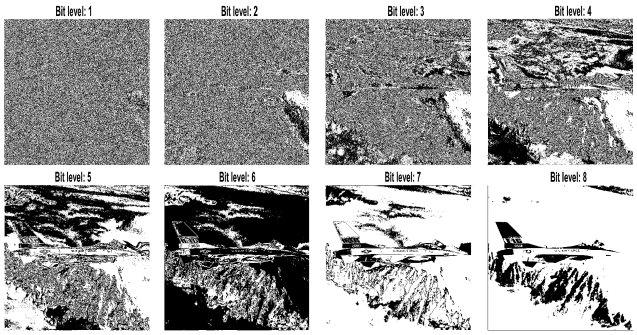

FIGURE 13. The individual bit planes of a grayscale image.

The key generation step is designed in such a way, so that each part of the key affects all of the subsequent steps 2-7.

The permutation steps use the first chaotic map, and are designed with the goal of achieving the most effective dispersion of the plaintext information in the binary level, at a low computational cost. To achieve this, the bit planes of the image are randomly arranged in a 2D matrix. Then, this matrix is shuffled. This is performed using circular shift operations instead of full element shuffling. The downside of circular shifting is that it does not change the relative positions of the elements in a vector. Yet, using an appropriate combination of row and column circular shifts, adjacent elements are successfully shuffled, at a low execution cost. Moreover, before performing the element shifting, the computed shifting rules are circularly shifted once. This action changes the order in which the rules are applied, effectively increasing security at a very low execution cost.

The substitution steps use the second chaotic map, and perform an XOR operation between the pixels of the permuted image from the previous step, and a bytestream generated using the PRNG described in Section III. As with the permutation step, before performing the substitution, the random bytestream from the PRNG is circularly shifted once. This action changes the XOR byte pairs, effectively increasing security at a very low execution cost.

The Algorithm is analytically provided in Algorithm 1, and a visual outline is shown in Fig. 14. Moreover, Fig. 15 shows the image output in each individual step of the algorithm, showing the progression of the encryption. Below, the motivation and explanation for each of the steps is provided in detail.

Input: The input to the algorithm is an $M \times N$ plaintext image $\mathcal{A}$, as well as two chaotic maps x_i, y_i of the form (3), with their respective initial conditions and parameter values.

Step 1: Key Initialization: The initial conditions x_0, y_0 of the two chaotic maps are computed through the output of the SHA-256 hash function on the plaintext image. This is performed so that part of the secret key is plaintext dependent, which can increase security against known plaintext attacks.

Afterwards, both maps are iterated 50 times, in order to discard their transient. Then, the initial conditions are re-initialized by combining the final values $x_{49}, x_{50}, y_{49}, y_{50}$ of the two chaotic maps. This action is highly important, as it ensures the avalanche effect will be inherited for all the permutation and substitution actions performed during the encryption. So any changes in any of the secret key values will affect all of the subsequent encryption steps.

Step 2: Bit Plane Reshaping: The bit planes of the image are arranged chaotically into an $M \times 8N$ matrix $\mathcal{A}_{\text{planes}}$, using the first chaotic map. Since there are 8! ways to arrange the bit planes, doing so in a random order instead of sequentially increases security, at a very low execution cost.

Step 3: Row Shifting. Using the first chaotic map, the individual M rows of the matrix $\mathcal{A}_{\text{planes}}$ are circularly shifted. Here, an additional action is added. Before performing the shifting, the shift values for all the rows are first computed and saved in a vector, and this vector is itself circularly shifted. This action changes the order in which the computed circular shift values are applied, which increases security at a very low execution cost.

Step 4: Column Shifting: After the M rows of the matrix $\mathcal{A}_{\text{planes}}$ are circularly shifted, the same action is performed to its $8N$ columns, using the first chaotic map. Similarly to the previous step, before performing the shifting, the shift values for all the rows are first computed and saved in a vector, which is then circularly shifted. This action changes the order in which the shift values are applied. Note that while each individual circular shift operation does not change the relative position of each row/column element, by applying this to all the rows, and all the columns, the relative positions of the elements of $\mathcal{A}_{\text{planes}}$ is succesfully changed.

Step 5: Bit Plane to Grayscale Transformation: After the shuffling is complete, the matrix $\mathcal{A}_{\text{planes}}$ is transformed back into a grayscale $M \times N$ image $\mathcal{A}_{\text{shuffled}}$. The transformation is performed by arranging chaotically the eight parts of the matrix $\mathcal{A}_{\text{planes}}$ into random bit planes, using the second chaotic map. This action mirrors that of Step 2, and increases security at a very low execution cost.

Step 6: Substitution Matrix: Using the second chaotic map, a stream of $M \cdot N$ bytes is generated using the PRNG (12), and is reshaped into an $M \times N$ matrix $\mathcal{B}$. Here, an additional action is added. Before reshaping into a matrix, the stream of bytes is circularly shifted once. This changes the position of the bytes in the final matrix $\mathcal{B}$, and increases security at a very low execution cost.

Step 7: Substitution: Finally, the encrypted image is obtained through byte-wise XOR substitution between $\mathcal{A}_{\text{planes}}$ and $\mathcal{B}$:

$$\mathcal{E} = \mathcal{B} \oplus \mathcal{A}_{\text{shuffled}}. \tag{14}$$

Note that in each consecutive step where the same chaotic map is used, the map's values are generated consecutively, using as x_{i-1}, y_{i-1} the value of the map from the previous iteration.

Each of those steps can be reversed. Thus, once a receiver has obtained the ciphertext image $\mathcal{E}$, assuming they also have knowledge of the secret keys, they can reverse the encryption process. This is done by performing Steps 7 through 1 in a

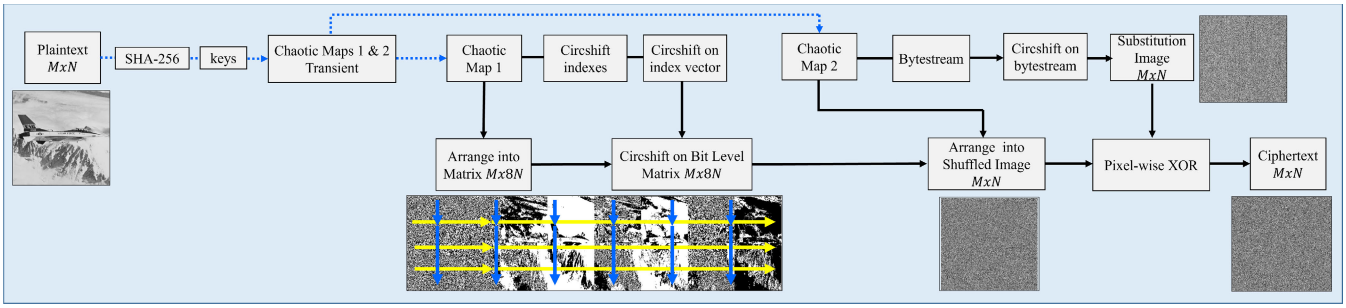

FIGURE 14. Outline of the encryption algorithm, consisting of seven steps.

reverse logical order. This involves first computing the secret keys (Step 1), computing the substitution matrix $\mathcal{B}$ (Step 6), obtaining the shuffled image through XOR $\mathcal{A}_{\text{shuffled}} = \mathcal{B} \oplus \mathcal{E}$ (Step 7), rearranging it back into a 2D matrix $\mathcal{A}_{\text{planes}}$ (Step 5), reversing its column and row shifts (Steps 4 and 3), and finally rearranging the un-shuffled matrix $\mathcal{A}_{\text{planes}}$ into the original plaintext image $\mathcal{A}$ (Step 2).

V. SIMULATION RESULTS

To test the encryption algorithm, several common grayscale images can be considered from the USC-SIPI Image Database (`http://sipi.usc.edu/database/`). Here, the 512 × 512 airplane and peppers images were chosen, and a 512 × 512 all black image, which are representative of an image carrying actual information, and an image that is completely uniform.

The encryption algorithm, as well as all the analysis tests were implemented in MATLAB 2023b. For the encryption, the key values are $a_x = a_y = 5$, $b_x = b_y = 5$, $K_x = K_y = 1$, $A_x = A_y = 0.84$, $B_x = B_y = 0.75$, $C_x = C_y = 1$, $D_x = D_y = 1$, while the initial conditions x_0, y_0 are computed through the use of the SHA-256 function, as described in Step 1 of Algorithm 1. The SHA-256 function is implemented using the code from [54]. Note also that to improve the execution speed, the modulo function is implemented using the rem function, which is identical to mod for positive inputs, but runs slightly faster in the MATLAB environment.

A. VISUAL ASSESSMENT

Figure 15 shows the encryption process for the airplane image at each individual step of the algorithm. The random arrangement of the bit planes, the row and column circular shifting, and the intermediate grayscale images can be seen. Note that in Step 3 the vertical adjacency between the bits is broken, and in Step 4 the horizontal adjacency is also broken. The end result is also displayed in Fig. 16, both for the airplane and the all black images. It is important to note here that for the black image, since all the bits are zero, the shifting actions have no effect on the image. So the shuffled image is still identical to the plaintext. But after the substitution step, the image is successfully encrypted. This fact denotes the importance of having both a permutation and a substitution step. So both the airplane and all black ciphertexts are visually similar to a white noise signal.

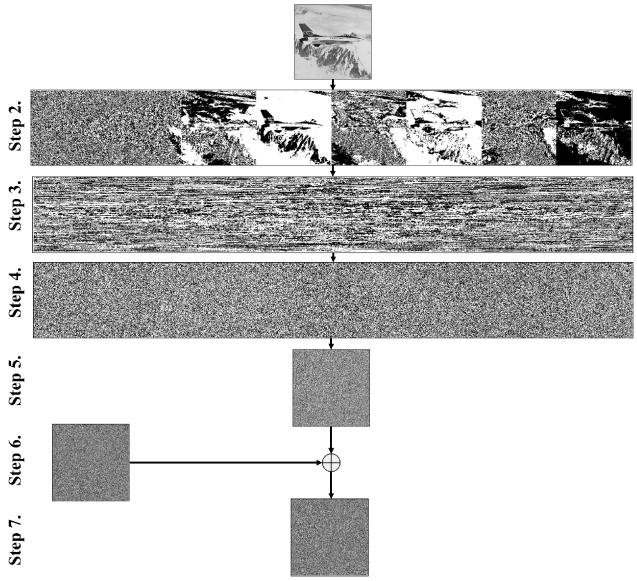

FIGURE 15. Simulation example showing the outcome of the different stages of the encryption algorithm. Step 2 shows image $\mathcal{A}_{\text{planes}}$, Steps 3 and 4 show the image $\mathcal{A}_{\text{planes}}$ after its row and column shifting, Step 5 shows the image $\mathcal{A}_{\text{shuffled}}$, Step 6 shows the image $\mathcal{B}$, and Step 7 shows the encrypted image $\mathcal{E}$.

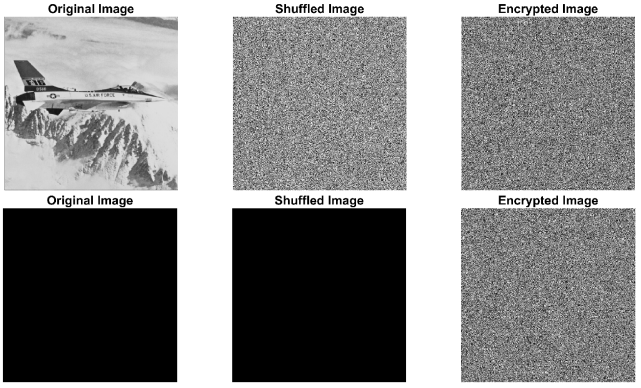

FIGURE 16. Plaintext, shuffled, and encrypted images, for the 512 × 512 airplane and black images.

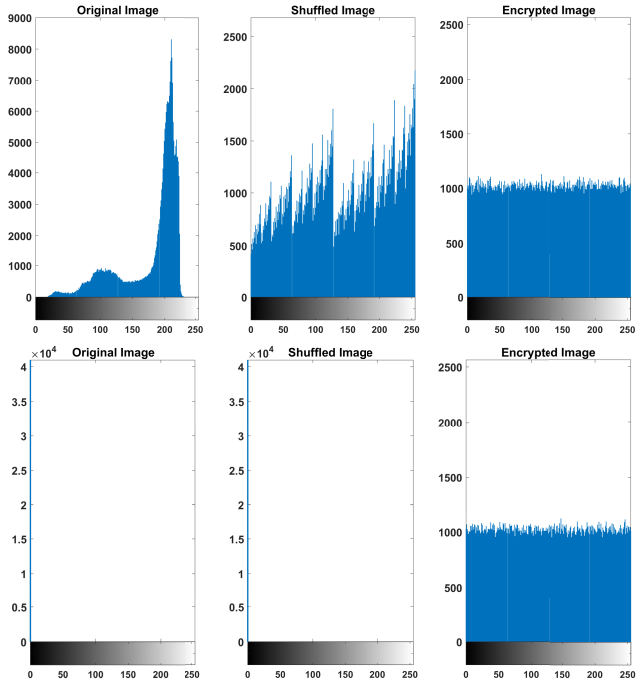

FIGURE 17. Histogram of the plaintext, shuffled, and encrypted images, for the 512 × 512 airplane and black images.

B. HISTOGRAM

The histogram is a fundamental statistical tool to inspect the distribution of the gray shades in an image. Images that carry information will naturally have a non-uniform histogram, while ciphertext images should have a uniform histogram, as they represent noise. The histograms of the airplane and black images are shown in Fig. 17 for the plaintext, shuffled, and ciphertext images. The plaintext histogram is non-uniform, and some gray shades in the range {0, 255} are even absent. The uniformity is achieved for the ciphertext images. Note that since the permutation is performed at the binary level, the resulting shuffled image has different pixel values compared to the plaintext one. Thus, the histograms between the plaintext and shuffled images are different, with the shuffled image having a more uniform histogram. This would not be the case if permutation was performed on the pixel level.

Thus, the permutation on the bit level achieves an effect similar to substitution on the pixel level. Yet, it must be emphasized that although this effect increases the security of the design and is highly desirable, it is not identical to a full-scale substitution. This is because, although the image pixel values have changed, the binary information carried by the shuffled image is identical to that of the plaintext. This can be verified by plotting binary histograms of the three images, shown in Fig. 18. On the binary level, the plaintext and permuted images give the same non-uniform result, while the ciphertext images are uniform.

In addition to the visual inspection of the histogram, an additional objective measure that can be considered is its variance, which is computed as follows:

$$\text{var}(H) = \frac{1}{256^2} \sum_{i=0}^{255} \sum_{j=0}^{255} \frac{(h_i - h_j)^2}{2}, \tag{15}$$

where the values $H = \{h_0, \ldots, h_{255}\}$ denote the number of pixels in the image with the grayscale shade 0,1,…,255. The histogram variance is shown in Table 2 for both the pixel and binary histograms. A lower variance is indicative of higher uniformity, which is observed for each of the ciphertext images. The shuffled images also have a lower variance compared to the plaintext images at the pixel histogram, which indicates the partial substitution achieved through bit level permutation. For the binary histogram though they are identical, as expected.

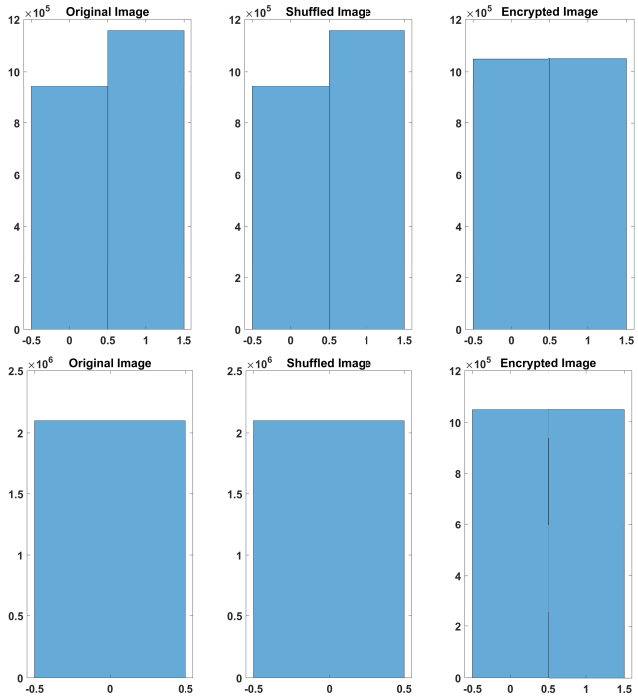

FIGURE 18. Histogram of bits of the plaintext, shuffled, and encrypted images, for the 512 × 512 airplane and black images.

C. CORRELATION

Correlation between adjacent pixels is another statistical measure for an image. It evaluates the similarity between adjacent pixels. In images that depict information, the similarity of adjacent pixels will be high, while in images that resemble noise, the similarity will be close to zero. The correlation is computed as follows:

$$E(x) = \frac{1}{N} \sum_{i=1}^{N} x_i, \tag{16}$$

$$D(x) = \frac{1}{N} \sum_{i=1}^{N} (x_i - E(x))^2, \tag{17}$$

TABLE 2. Histogram variance for plaintext, shuffled and encrypted images.

Image	Pixel level			Binary level		
	Original	Shuffled	Encrypted	Original	Shuffled	Encrypted
Airplane	2882400	98876	1196.2	$2.3409 \cdot 10^{10}$	$2.3409 \cdot 10^{10}$	6962
Black	268435456	268435456	1054.5	$2.1990 \cdot 10^{12}$	$2.1990 \cdot 10^{12}$	1924722

$$cov(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)), \tag{18}$$

$$\gamma(x, y) = \frac{cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{19}$$

where x, y are the gray values of two adjacent pixels, and N is the number of adjacent pairs of pixels (x, y).

Table 3 depicts the correlation between adjacent horizontal, vertical, and diagonal pixels for the plaintext, shuffled, and encrypted images. As expected, the correlation significantly drops for the shuffled and ciphertext images, with the ciphertext having the lowest. Figure 19 also depicts a scatterplot of a collection of adjacent pixels for the images. In the plaintext images, the pixel pairs are closer to the diagonal, indicating that adjacent pixels have very similar values. The pixel pairs also cover a shorter region in the interval $\{0, 255\}$, due to the fact that in the plaintext image not all the gray shades appear. In the shuffled image on the other hand, the scatterplot becomes closer to uniform, and in the encrypted images the adjacent pixels have unrelated values, so the scatterplot is completely uniform, and covers the complete range of $\{0, 255\}$ shades.

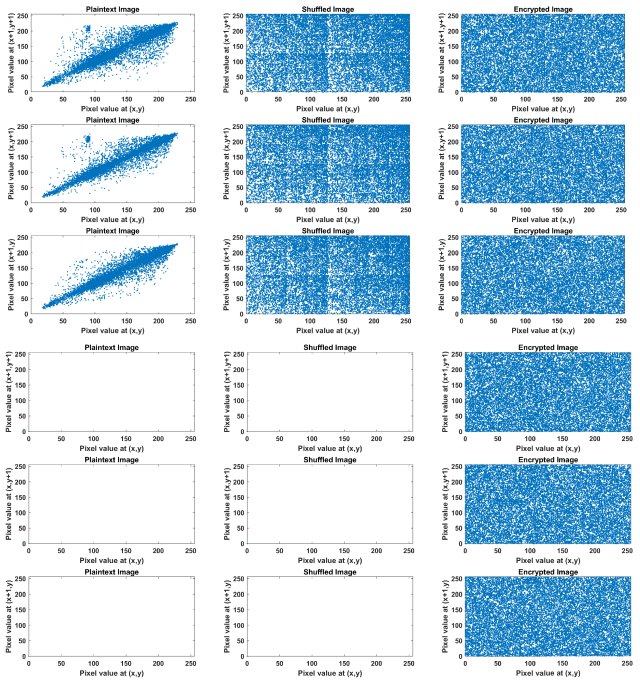

FIGURE 19. Scatter plot of adjacent pixels for the plaintext, shuffled, and encrypted images, for the 512 × 512 airplane and black images.

D. INFORMATION ENTROPY

Entropy is a measure for predictability of a signal. For an image A, it is computed as:

$$\mathrm{En}(A) = -\sum_{i=0}^{2^8-1} p(s_i)\log_2 p(s_i), \tag{20}$$

where $p(s_i)$ is the probability of occurrence for the value s_i. An image that is similar to noise should have a high entropy, close to the maximum value, which is 8. Table 4 shows the entropy values for all the images. The entropy in the shuffled and ciphertext images is increased, with the ciphertext images having values close to the optimal one.

In addition to entropy, which is measured on the complete image, Table 5 shows the values of the local entropy for each image [42], [55]. Local entropy can be used to measure if any information, or lack of it, is uniformly spread across the image. Local entropy is computed as the average entropy of 30 random non-overlapping submatrices of size 44×44. As with entropy, the local entropy is close to the optimal value in the shuffled and encrypted images.

E. DIFFERENTIAL ATTACKS

An encryption algorithm should have all or a part of the secret keys being plaintext dependent, so that changes in the plaintext information will affect the keys, and thus the encryption outcome. This way, the design can be resistant against known plaintext attacks, where an attacker may encrypt a collection of almost identical images, and try to compare their ciphertexts, in order to obtain information about the process. The proposed encryption has part of the secret key being computed from the hash of the input image, so it can resist such attacks.

To measure the performance of the algorithm under almost identical images, there are two measures that can be used. These are the Number of Pixel Changing Rate (NPCR) and the Unified Averaged Changed Intensity (UACI) [56], computed as:

$$\mathrm{NPCR} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} D_{i,j}}{MN}100\%, \tag{21}$$

$$\mathrm{UACI} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} |\mathcal{E}_{i,j} - \hat{\mathcal{E}}_{i,j}|}{255 \cdot MN}100\%, \tag{22}$$

where M and N are the image dimensions, $\mathcal{E}$ and $\hat{\mathcal{E}}$ are two encrypted images, the indexes i, j refer to the pixels, and

$$D_{i,j} = \begin{cases} 0 & \mathcal{E}_{i,j} = \hat{\mathcal{E}}_{i,j} \\ 1 & \mathcal{E}_{i,j} \neq \hat{\mathcal{E}}_{i,j} \end{cases}. \tag{23}$$

TABLE 3. Correlation coefficients.

	Original			Shuffled			Encrypted		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Airplane	0.9663	0.9641	0.9370	0.0038	0.0225	0.0037	0.0005	0.0008	0.0006
Black	-	-	-	-	-	-	-0.0016	-0.0005	0.0035

TABLE 4. Entropy for plaintext and encrypted images.

	Original	Shuffled	Encrypted
Airplane	6.7025	7.9345	7.9992
Black	0	0	7.9993

TABLE 5. Local entropy for plaintext and encrypted images.

	Original	Shuffled	Encrypted
Airplane	5.2225	7.8424	7.9057
Black	0	0	7.9053

TABLE 6. NPCR and UACI measures for different images.

	Shuffled		Encrypted	
	NPCR (%)	UACI (%)	NPCR (%)	UACI(%)
Airplane	99.5735	33.1193	99.6006	33.4709
Black	0.0004	0.0001	99.5846	33.4929

Here, we are comparing the encryption results $\mathcal{E}, \hat{\mathcal{E}}$ from two plaintext images that only differ in a single pixel. The output of the SHA-256 for these two images will be different, and thus the initial conditions x_0, y_0 for the two chaotic maps that are used in the encryption will be different. The other key parameters for the chaotic maps are the same. These differences will result in different substitution and permutation rules, and thus the two ciphertexts will be significantly different from each other. The results are shown in Table 6, and the encrypted images have NPCR and UACI values within the accepted ranges.

F. ALTERATION ATTACKS

During transmission, partial information from the encrypted ciphertext may be lost, either due to tranmission noise, or due to external attacks that aim at corrupting the ciphertext. The encryption technique can counterbalance this information loss, due to the fact that it includes an effective permutation step on the binary information level. Hence, if part of the ciphertext is discarded, corrupted by noise, or tampered with, partial information on the corresponding part of the plaintext can still be retrieved.

To verify this, several simulations have been performed, were the ciphertext has been submitted to different forms of alteration attacks. These are described below:

Cropping Attacks: This refers to the case where part of the ciphertext image is discarded completely. This is simulated by replacing the discarded pixels with zero values.

Bit Level Cropping Attacks: In this variation of cropping, the bits on specific bit levels are discarded, meaning that they are replaced by zero values.

Tampering Attacks: This refers to the case where part of the ciphertext is replaced by another content. This is simulated by replacing part of the ciphertext with another image.

Noise attacks: This refers to the corruption of the ciphertext by noise. The noise added to the ciphertext can be of different types. The most common types include salt and pepper noise of varying density, Gaussian white noise with zero mean and varying variance, speckle noise, which is a multiplicative noise, with varying variance, and Poisson noise, which is generated based on the ciphertext itself.

To measure the performance of decryption under these different alteration attacks, several objective measures are available. These include Structural Similarity Index (SSIM) [40], Peak Signal to Noise Ratio (PSNR), and Spectral Distortion (SD). Let $\mathcal{A}$ denote the plaintext image, and $\mathcal{D}$ denote the decrypted image, which has been obtained after decrypting the corrupted ciphertext image. The SSIM measure is computed as follows:

$$\text{SSIM} = \frac{(2\mu_{\mathcal{A}}\mu_{\mathcal{D}} + S_1)(2\delta_{\mathcal{A},\mathcal{D}} + S_2)}{(\mu_{\mathcal{A}}^2 + \mu_{\mathcal{D}}^2 + S_1)(\delta_{\mathcal{D}}^2 + \delta_{\mathcal{D}}^2 + S_2)}, \tag{24}$$

where $\mu_{\mathcal{A}}, \mu_{\mathcal{D}}$ the mean values of the plaintext and decrypted images respectively, $\delta_{\mathcal{A}}^2, \delta_{\mathcal{D}}^2$ their variances, and $\delta_{\mathcal{A},\mathcal{D}}$ their cross-covariance. The parameters S_1, S_2 have small values, to facilitate the division with small numbers. The SSIM values are between $[-1, 1]$, with 1 for identical signals, and close to 0 for images with no similarity. Thus, the higher the value of SSIM, the better the decryption performance under ciphertext alterations.

The PSNR measure is computed as follows

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(\mathcal{A}_{i,j} - \mathcal{D}_{i,j})^2}, \tag{25}$$

where M and N are the image dimensions, the indexes i, j denote the corresponding pixel values for each image A and D. A higher PSNR value indicates a higher similarity between images, and thus a better decryption performance under ciphertext alterations.

The SD measure is computed as follows:

$$\text{SD} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|\mathcal{F}(\mathcal{A})_{i,j} - \mathcal{F}(\mathcal{D})_{i,j}|, \tag{26}$$

where M and N are the image dimensions, $\mathcal{F}(\cdot)$ denotes the spectrum of the image, and $|\cdot|$ denotes its magnitude. A lower SD value denotes a higher similarity between images in the frequency domain, and thus a better decryption under ciphertext alterations.

Figures 20, 21, 22, 23 show some examples of decryption under an altered ciphertext. Noise was added to the images

TABLE 7. Decryption performance under ciphertext alterations for the 512 × 512 airplane image.

Effect	PSNR	SSIM	SD
Cropping 25%	14.0094	0.1919	19358.3
Cropping 50%	10.9994	0.0678	26753.1
Cropping 75%	9.2330	0.0253	32100.7
Tampering 25%	14.049	0.1935	19274.9
Tampering 50%	11.020	0.0682	26649
Tampering 75%	9.243	0.0252	31993.3
Salt and Pepper 25%	13.8349	0.1018	22468.5
Salt and Pepper 50%	10.8863	0.0502	29826.9
Salt and Pepper 75%	9.2207	0.0208	33791.8
Gaussian 0.00001	13.8311	0.1063	21473.6
Gaussian 0.0001	11.3274	0.0568	17419.3
Gaussian 0.001	9.9218	0.0320	31361.6
Speckle 0.00001	16.6905	0.1695	16219.5
Speckle 0.0001	12.8111	0.0840	24050.2
Speckle 0.001	10.8746	0.0477	29005
Poisson	9.7113	0.0282	31971.5

using MATLAB's built-in 'imnoise' command. In all cases, it can be seen that even if significant information is lost, it is possible to replenish part of the lost information from the corrupted portion of the image. When cropping part of the image, or some bit planes of the image, as seen in Figs. 20, 21, information on the lost regions can be retrieved, and the lost information is dispersed through the decrypted ciphertext. The same holds for tampered images, shown in Fig. 22. In noisy conditions, information can still be retrieved for lower noise intensities, shown in Fig. 23. Of course, when information damage exceeds 70% of the ciphertext, the decryption is poor, but it is worth noting that some basic object shapes can still be perceived.

To test the effect of alteration attacks in detail, Figs. 24 and 25 depict the SSIM, PSNR, and SD measures for the decryption of the airplane image under an increasing alteration intensity. In Fig. 24, it can be seen that the encryption performance is almost identical for cropping and tampering, while for salt and pepper the SSIM and SD are worse. In Fig. 25, where the variance is plotted in logarithmic scale for clarity, the PSNR, SSIM and SD measures are worse for the Gaussian noise. It is important to note that all the curves that show the deterioration under different alterations have a similar shape, which is nonlinear. In addition, some indicative results are provided in Table 7.

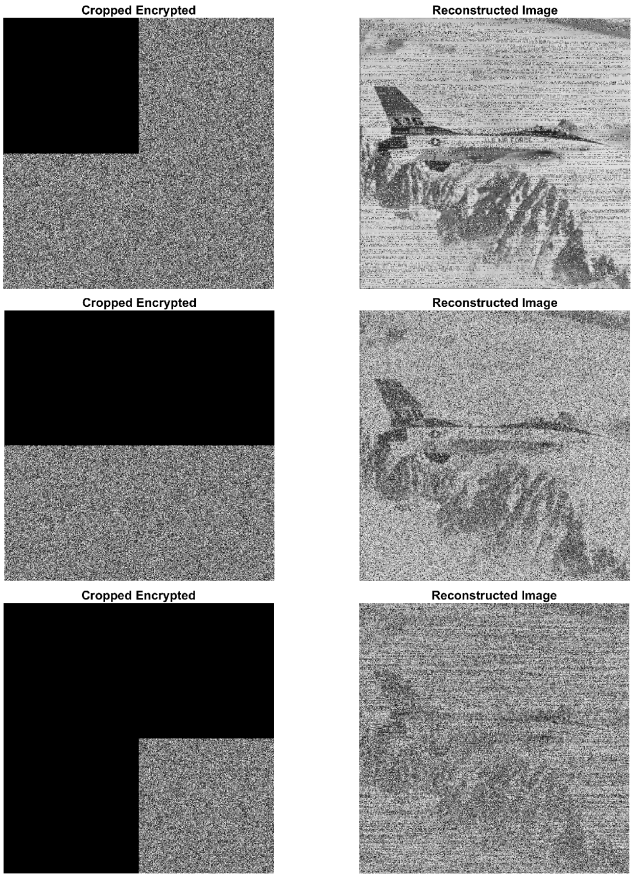

FIGURE 20. Cropped and decrypted images, for the 512 × 512 airplane image.

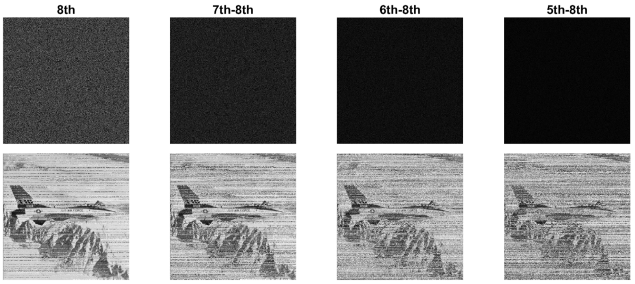

FIGURE 21. Cropped and decrypted images, for the 512 × 512 airplane image, when cropping is performed on the 8th, 7th-8th, 6th-8th, and 5th-8th most significant bit levels of the ciphertext.

G. KEY SPACE

The encryption keys refer to the number of the secret values that are required to encrypt and decrypt an image. These keys are the initial conditions and parameters of the two chaotic maps used during the encryption process. The key space refers to the set of all possible key combinations that can be used. Each of the two maps has an initial condition and seven control parameters a, b, K, A, B, C, D. This would bring the key set to the size of 16 control parameters:

$$\mathcal{K} = \{x_0, a_x, b_x, A_x, B_x, C_x, D_x, K_x, y_0, a_y, b_y, A_y, B_y, C_y, D_y, K_y\}. \quad (27)$$

Assuming a 16-digit accuracy, this brings the upper bound for the key space size to $10^{16\cdot16} = (10^3)^{85.3} \approx (2^{10})^{85.3} = 2^{853}$, which is significantly higher than the required bound of 2^{100} [57].

Table 8 provides a key space comparison with the literature. It is clear that proposed algorithm showcases a superior key space, that is much higher than that of its counterparts.

Note that the consideration of all 16 parameters on one hand leads to very high key space, but may also make key management difficult, due to the overall key being too large.

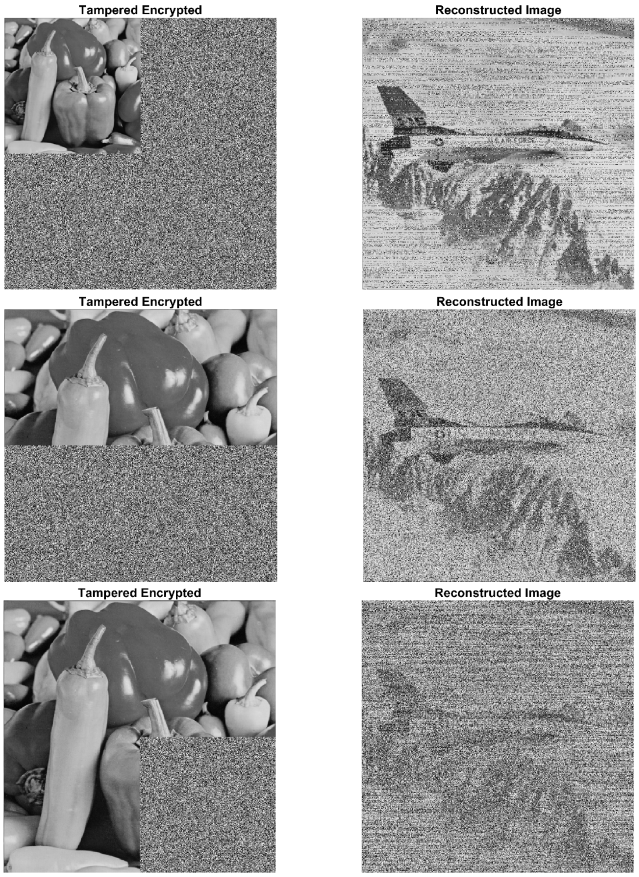

FIGURE 22. Tampered and decrypted images, for the 512 × 512 airplane image.

TABLE 8. Key space comparison with the literature.

Algorithm	Key space
Proposed	$2^{853} \approx 10^{256}$
[58]	10^{105}
[59]	1.16×10^{182}
[60]	2^{647}
[61]	2^{747}
[62]	2^{200}
[63]	2^{360}
[64]	2^{512}
[65]	2^{215}
[66]	2^{419}

Thus, it is preferable to keep some parameters fixed, to reduce the size of the key to a manageable size. Thus, it can be assumed that the parameters $K_x = K_y = C_x = C_y = D_x = D_y = 1$ are fixed. This reduces the key to the following ten parameters:

$$\mathcal{K} = \{x_0, a_x, b_x, A_x, B_x, y_0, a_y, b_y, A_y, B_y\}. \quad (28)$$

With ten parameters, the key space upper bound is $10^{10 \cdot 16} = (10^3)^{53.3} \approx (2^{10})^{53.3} = 2^{530}$, which is again high.

Note that for the case of known plaintext attacks, where an adversary is using images of their choice to test the outcome, the initial conditions x_0, y_0 can be considered known, since

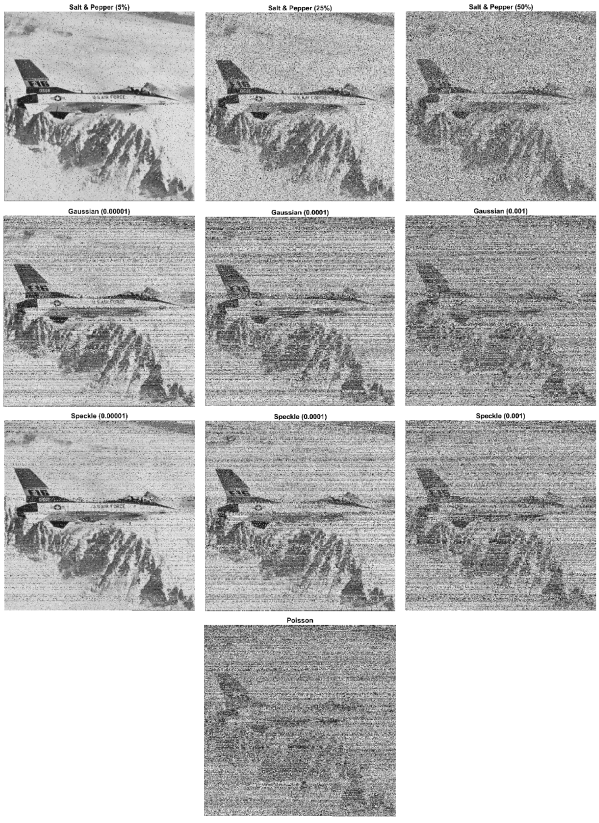

FIGURE 23. Decrypted images under salt and pepper, Gaussian, speckle, and Poisson noise, for the 512 × 512 airplane image.

they are computed from the SHA-256 result of the image. So in this case, the number of key values is reduced by 2, giving a key space size of $10^{8 \cdot 16} = (10^3)^{42.6} = 2^{426}$, which is again above the required threshold.

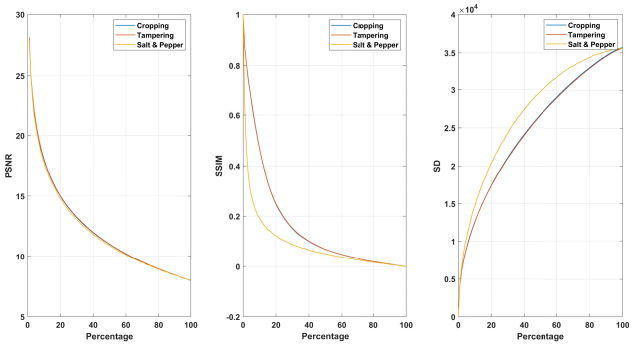

FIGURE 24. Cropping, tampering, and Salt & Pepper noise decryption performance, for the 512 × 512 airplane image.

H. KEY SENSITIVITY

Given the nature of the chaotic maps used during the encryption process, any change at their key parameters will have an avalanche effect on the encryption process. To test this, the encrypted black image is decrypted using slightly altered key values. The results are shown in Fig. 26 for the

TABLE 9. Encryption execution time (seconds) of grayscale images for specific sizes.

Image Size	Proposed	[58]	[59]	[60]	[61]	[62]	[63]	[67]	[34]	[64]	[65]	[66]
64×64	0.0012	-	-	-	-	-	-	-	-	-	-	-
128×128	0.0037	-	-	-	-	-	-	-	-	-	-	-
256×256	0.0134	0.0108	0.1830	2.5469	0.5317	0.0081	0.20	0.0020	0.1081	0.9196	0.0347	1.8298
512×512	0.0530	-	-	-	1.6349	0.0358	-	0.0040	0.4154	3.6573	0.1439	7.5767
1024×1024	0.2050	-	-	-	-	0.1524	-	-	-	12.7136	-	30.2415
2048×2048	0.8071	-	-	-	-	-	-	-	-	-	-	-

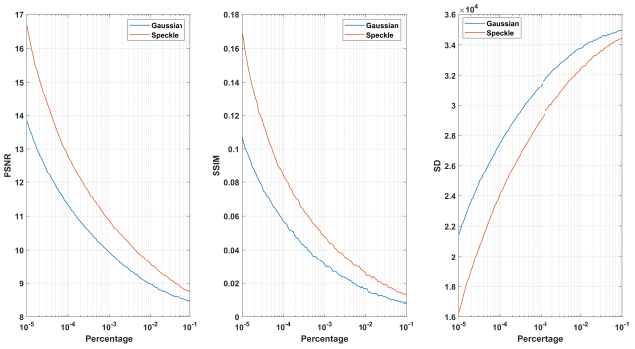

FIGURE 25. Gaussian and Speckle noise decryption performance, for the 512 × 512 airplane image. X-axis is logarithmic.

all black image. In each case, the decryption fails when the wrong keys are used.

Note that due to the action of re-initializing the initial conditions of the two chaotic maps (35) in Step 1b, any change in the keys of both maps will affect all of the steps performed in the algorithm, ensuring a complete avalanche effect of the secret key. Without this step, the algorithm would not have inherited this property for all parts of the key. For example, without this step, the all black image, which is unaffected by the permutation, would still be successfully decrypted using the wrong key values for the first map, as long as the key values of the second map were chosen correctly. So decryption would be successful despite half the secret key being wrong. This potential flaw is succesfully addressed through the initialization performed in Step 1b.

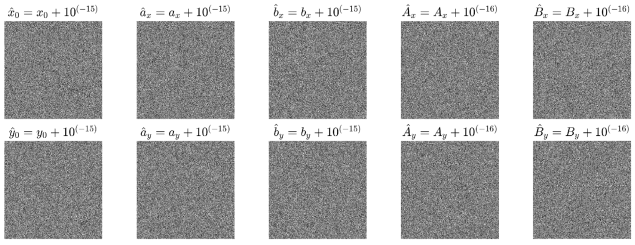

FIGURE 26. Sensitivity of decryption under altered key values, for the 512 × 512 all black image.

I. EXECUTION TIME

Finally, in addition to all the security tests performed, an essential aspect of any encryption algorithm is its

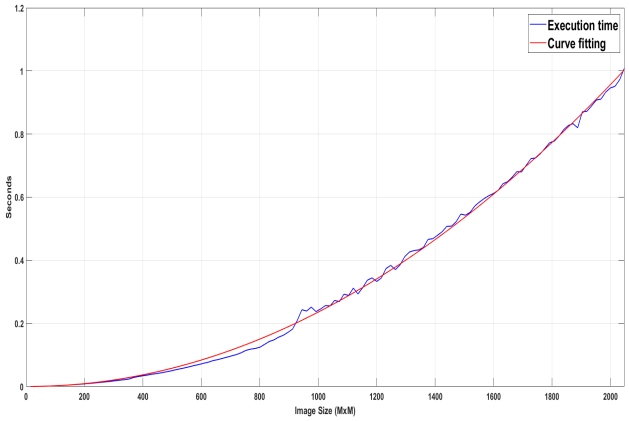

FIGURE 27. Average execution time for encryption, for different $M \times M$ grayscale image sizes (airplane), and its fitted curve.

execution speed. Overly complex designs may improve upon the encryption security, but if this comes at the cost of speed, then any potential implementability in a real life setting is diminished. The proposed algorithm is aimed at balancing security and speed through the use of circular shifting operations, which are relatively lightweight.

Regarding the PRNG, the execution speed is tested on an ASUS Vivobook X1504 laptop with a Windows 11 operating system, a 13th Gen Intel(R) Core(TM) i5-1335U, 1.30 GHz processor, and 16 GB of RAM, using MATLAB 2023b, and MATLAB's built-in *timeit* function. The generator can output 10^6 bytes in 0.0593 seconds, which is equivalent to a speed of 16.86 Mbytes/sec.

For the encryption process, the execution speed is shown in Fig. 27, for square images of increasing size. This was generated by considering the grayscale airplane image, resizing it appropriately, and computing the average encryption time from ten calls to MATLAB's built-in function *timeit*. The execution is indeed fast, which underlines the potential implementability of the technique in the future. Moreover, using MATLAB's curve fitting toolbox, it was shown that the execution time curve fits to a power function of the form:

$$t = a \cdot M^b, \tag{29}$$

where M is the dimension of the square $M \times M$ image, with parameters $a = 2.0698 \cdot 10^{-7}$, $b = 2.019$, with an R-square of 0.9985. Table 9 also displays results for some standard image sizes. Note that these are times reported from different

hardware specifications. The execution time is significantly faster than most recent works, and is comparable to other state of the art works, with a few exceptions that achieve a superior time. The execution time can be further reduced with algorithm modifications, as discussed in the Conclusions. It is also worth noting here that many recent works do not report on the execution time, which is a big omission, and all works only report the execution time for only a couple image sizes. In this work, the execution time is holistically studied by considering increasing image sizes, as shown in Fig. 27, which the authors hope will be the standardized approach for future works.

VI. CONCLUSION

In this work, a chaotic encryption algorithm was proposed, utilizing a robust chaotic map, that performed permutation on the binary level using circular shifts, and substitution on the byte level using XOR. Several actions were included to increase the algorithm's security at a low execution cost, like the chaotic arrangement of the bit planes into a matrix, and the circular shifting of the vectors that contained the rules for permutation and substitution. It also included an initialization step that ensured the avalanche effect from all the key values across all the encryption steps. The encryption result is secure against a collection of attacks, has a high key space, and shows resistance to information loss and alteration. Moreover, it has a low execution speed, requiring 0.0134 sec to encrypt a 256×256 image, 0.0530 sec for a 512×512 image, and 0.2050 sec for a 1024×1024 image. So overall, it is suitable for future implementations.

The present work can be a starting point for a collection of new studies. First of all, it is possible to develop a standalone Graphical User Interface (GUI) that implements the encryption algorithm, similar to [42]. GUIs in MATLAB can be created to run on any computer, without requiring a MATLAB installation, which is a very attractive feature. Moreover, an FPGA implementation similar to [16] is also of interest, to test the speed, memory, and power consumption requirements for the encryption process.

Moreover, although the XOR substitution step is very standardized, there are several variations that can be considered on the permutation steps to further reduce the execution time, without significantly influencing its effectiveness. For example, as the most significant bit planes carry most of the image's essential information, permutation can be performed just on them. So the binary matrix $\mathcal{A}_{\text{planes}}$ would be $M \times zN$, where z is the number of significant bit planes chosen for permutation. Another variation is to execute the permutation algorithm on $z \times z$ blocks (4×4, 8×8, 16×16, etc) of rows and columns in the $M \times 8N$ binary matrix $\mathcal{A}_{\text{planes}}$. Both of the above variations would significantly reduce the amount of operations required to complete the permutation, which would help in implementations with limited computational resources. In addition, other fast operations can be performed on the bit planes before arranging them in the $\mathcal{A}_{\text{planes}}$ matrix, like rotation, negative-positive transformation, and color channel rearranging, similar to [68]. So overall, there are many more research topics to be studied in the future.

Algorithm 1 Image Encryption

Input: -An $M \times N$ grayscale image $\mathcal{A} = [a_{ij}]$, $i = 1, \ldots, M$, $j = 1, \ldots, N$.
-Two chaotic maps x_i, y_i of the form (3).
-Secret key values $x_0, a_x, b_x, A_x, B_x, y_0, a_y, b_y, A_y, B_y$, with the initial conditions x_0, y_0 specified below.

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

Step 1: Key Initialization. The plaintext is used to generate the secret keys.

Step 1a. Compute the following hash value from the plaintext image:

$$H = \text{SHA-256}(\mathcal{A}), \tag{30}$$

where H is a hexademical string of 64 characters. Using H, generate the initial conditions for the two chaotic maps as follows:

$$h_1 = \text{hex2dec}(H_{1:13}),\ h_2 = \text{hex2dec}(H_{14:26}), \tag{31}$$

$$h_3 = \text{hex2dec}(H_{27:39}),\ h_4 = \text{hex2dec}(H_{40:52}), \tag{32}$$

$$h_5 = \text{hex2dec}(H_{53:64}), \tag{33}$$

$$x_0 = \frac{h_1 + h_2 + h_5}{10^{16}},\ y_0 = \frac{h_3 + h_4 + h_5}{10^{16}}, \tag{34}$$

where hex2dec denotes the hexademical to decimal transformation.

Step 1b. Both chaotic maps x_i, y_i are iterated 50 times. Afterwards, the keys are re-initialized as:

$$x_{0,\text{new}} = \text{rem}(x_{50} + y_{49}, 1),\ y_{0,\text{new}} = \text{rem}(x_{49} + y_{50}, 1). \tag{35}$$

Step 2: Bit Plane Reshaping. The plaintext bit planes are arranged chaotically in a matrix.

Step 2a. The first chaotic map x_i is iterated 8 times. These eight values are sorted in ascending order, resulting in an index vector ind, that describes the arrangement order of the time series.

Step 2b. The bit planes $\mathcal{A}_i$, $i = 1, \ldots, 8$ of the plaintext image are arranged into a matrix $\mathcal{A}_{\text{planes}}$ of size $M \times 8N$. The ordering of the bitplanes in this matrix follows the order indicated by ind, as follows:

$$\mathcal{A}_{\text{planes}} = \left(\mathcal{A}_{\text{ind}_1}\ \mathcal{A}_{\text{ind}_2}\ \cdots\ \mathcal{A}_{\text{ind}_8}\right). \tag{36}$$

Step 3: Row Shifting. The rows of the matrix $\mathcal{A}_{\text{planes}}$ are shifted.

Step 3a. The first map x_i is iterated M times. In each iteration, an integer value is computed following the rule:

$$r_i = \lfloor 8N x_i \rfloor,\ i = 1, \ldots, M. \tag{37}$$

and the result is appended in a vector r.

Step 3b. The map x_i is iterated one more time, and an integer is computed using the rule:

$$r_{\text{shift}} = \lfloor M x_i \rfloor. \tag{38}$$

Then, the vector r is shifted once:

$$r = \text{circshift}(r, r_{\text{shift}}). \tag{39}$$

Step 3c. The rows of the matrix $\mathcal{A}_{\text{planes}}$ are shifted following the rule

$$\mathcal{A}_{\text{planes}}(i, :) = \text{circshift}(\mathcal{A}_{\text{planes}}(i, :), r_i),\ i = 1, \ldots, M. \tag{40}$$

Algorithm 1 Image Encryption (Continued.)

Step 4: Column Shifting. The columns of the matrix $\mathcal{A}_{\text{planes}}$ are shifted.

Step 4a. The first map x_i is iterated $8N$ times. In each iteration, an integer value is computed following the rule:

$$c_i = \lfloor Mx_i \rfloor, i = 1, \ldots, 8N. \tag{41}$$

and the result is appended in a vector c.

Step 4b. The map x_i is iterated one more time, and an integer is computed using the rule:

$$c_{\text{shift}} = \lfloor 8Nx_i \rfloor. \tag{42}$$

Then, the vector c is shifted once:

$$c = \text{circshift}(c, c_{\text{shift}}). \tag{43}$$

Step 4c. The columns of the matrix $\mathcal{A}_{\text{planes}}$ are shifted following the rule

$$\mathcal{A}_{\text{planes}}(:, i) = \text{circshift}(\mathcal{A}_{\text{planes}}(:, i), c_i), i = 1, \ldots, 8N. \tag{44}$$

Step 5: Bit Plane to Grayscale Transformation. The shuffled bit planes are transformed into a grayscale image in a chaotic order.

Step 5a. The second chaotic map y_i is iterated 8 times. These eight values are sorted in ascending order, resulting in an index vector pln, that describes the arrangement order of the time series.

Step 5b. The shuffled matrix is $\mathcal{A}_{\text{planes}}$, is broken into eight $M \times N$ submatrices:

$$\mathcal{A}_{\text{planes}} = \begin{pmatrix} P_1 & P_2 & \cdots & P_8 \end{pmatrix}. \tag{45}$$

These submatrices are then transformed into a plaintext image $\mathcal{A}_{\text{shuffled}}$ of size $M \times N$. The ordering of the bit planes in this matrix follows the order indicated by pln, as follows:

$$\mathcal{A}_{\text{shuffled}} = 2^0 P_{\text{pln}_1} + 2^1 P_{\text{pln}_2} + 2^2 P_{\text{pln}_3} + 2^3 P_{\text{pln}_4} + 2^4 P_{\text{pln}_5} + 2^5 P_{\text{pln}_6} + 2^6 P_{\text{pln}_7} + 2^7 P_{\text{pln}_8}. \tag{46}$$

Step 6: Substitution Matrix. A matrix is generated for byte-wise substitution.

Step 6a. The second map y_i is iterated $M \cdot N$ times. In each iteration, an integer value is computed following the rule of the PRNG (12):

$$\text{stream}_i = \text{mod}(\lfloor 10^{10} y_i \rfloor, 256). \tag{47}$$

and the result is appended in a vector stream.

Step 6b. The map y_i is iterated one more time, and an integer is computed using the rule:

$$s_{\text{shift}} = \lfloor M \cdot Ny_i \rfloor. \tag{48}$$

Then, the vector stream is shifted once:

$$\text{stream} = \text{circshift}(\text{stream}, s_{\text{shift}}). \tag{49}$$

Step 6c. The vector stream is reshaped into an $M \times N$ matrix $\mathcal{B}$.

Step 7: Substitution. The plaintext substitution is performed. The output encrypted image $\mathcal{E}$ is obtained through the following byte-wise XOR operation:

$$\mathcal{E} = \mathcal{B} \oplus \mathcal{A}_{\text{shuffled}}. \tag{50}$$

REFERENCES

[1] D. Sargiotis, "Data security and privacy: Protecting sensitive information," in *Data Governance: A Guide*. Cham, Switzerland: Springer, 2024, pp. 217–245.

[2] Q. Chen, D. Li, and L. Wang, "Network security in the Internet of Things (IoT) era," *J. Ind. Eng. Appl. Sci.*, vol. 2, no. 4, pp. 36–41, 2024.

[3] D. El-Damak, W. Alexan, E. Mamdouh, M. El-Aasser, A. Fathy, and M. Gabr, "Fibonacci Q-matrix, hyperchaos, and Galois field (2^8) for augmented medical image encryption," *IEEE Access*, vol. 12, pp. 102718–102744, 2024.

[4] M. Youssef, M. Gabr, W. Alexan, M. B. M. Mansour, K. Kamal, H. Hosny, and D. El-Damak, "Enhancing satellite image security through multiple image encryption via hyperchaos, SVD, RC5, and dynamic S-box generation," *IEEE Access*, vol. 12, pp. 123921–123945, 2024.

[5] M. Naim and A. Ali Pacha, "New chaotic satellite image encryption by using some or all the rounds of the AES algorithm," *Inf. Secur. J., Global Perspective*, vol. 32, no. 3, pp. 187–211, May 2023.

[6] A. Hafsa, A. Sghaier, J. Malek, and M. Machhout, "Image encryption method based on improved ECC and modified AES algorithm," *Multimedia Tools Appl.*, vol. 80, no. 13, pp. 19769–19801, May 2021.

[7] J.-Y. Sun, H. Cai, and H. Zhang, "A novel image encryption algorithm combined complex order chaotic system and modified AES," *Multimedia Tools Appl.*, vol. 83, no. 14, pp. 40361–40376, Oct. 2023.

[8] Y. Naseer, T. Shah, Attaullah, and A. Javeed, "Advance image encryption technique utilizing compression, dynamical system and S-boxes," *Math. Comput. Simul.*, vol. 178, pp. 207–217, Dec. 2020.

[9] N. F. Karagiorgos, S. G. Stavrinides, C. D. Benito, S. Nikolaidis, and R. Picos, "Unconventional security for IoT: Hardware and software implementation of a digital chaotic encrypted communication scheme," *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19914–19925, Jun. 2024.

[10] W. Alexan, E. Mamdouh, A. Aboshousha, Y. S. Alsahafi, M. Gabr, and K. M. Hosny, "Stegocrypt: A robust tri-stage spatial steganography algorithm using TLM encryption and DNA coding for securing digital images," *IET Image Process.*, vol. 18, no. 13, pp. 4189–4206, Nov. 2024.

[11] A. V. Tutueva, A. I. Karimov, L. Moysis, C. Volos, and D. N. Butusov, "Construction of one-way hash functions with increased key space using adaptive chaotic maps," *Chaos, Solitons Fractals*, vol. 141, Dec. 2020, Art. no. 110344.

[12] S. Kosmidis, C. Volos, L. Moysis, and E. Meletlidou, "A novel chaotic system with hyperbolic tangent terms and its application to random bit generator and image encryption," *J. Vibrat. Test. Syst. Dyn.*, vol. 8, no. 3, pp. 341–353, Sep. 2024.

[13] L. Ding and P. Wang, "Analysis of the dynamics of a cubic nonlinear five-dimensional memristive chaotic system and the study of reduced-dimensional synchronous masking," *Adv. Math. Phys.*, vol. 2024, pp. 1–22, Apr. 2024.

[14] M. Lawnik, L. Moysis, and C. Volos, "Chaos-based cryptography: Text encryption using image algorithms," *Electronics*, vol. 11, no. 19, p. 3156, Oct. 2022.

[15] K. M. Hosny, M. A. Zaki, N. A. Lashin, M. M. Fouda, and H. M. Hamza, "Multimedia security using encryption: A survey," *IEEE Access*, vol. 11, pp. 63027–63056, 2023.

[16] W. Alexan, N. Elabyad, M. Khaled, R. Osama, D. El-Damak, M. A. A. El Ghany, Y. Korayem, E. Mamdouh, and M. Gabr, "Triple layer RGB image encryption algorithm utilizing three hyperchaotic systems and its FPGA implementation," *IEEE Access*, vol. 12, pp. 118339–118361, 2024.

[17] D. Clemente-Lopez, J. D. J. Rangel-Magdaleno, and J. M. Muñoz-Pacheco, "A lightweight chaos-based encryption scheme for IoT healthcare systems," *Internet Things*, vol. 25, Apr. 2024, Art. no. 101032.

[18] L. Laskaridis, C. Volos, A. Giakoumis, H. Nistazakis, I. P. Antoniades, and I. Stouboulos, "A microcontroller implementation of a pseudo-random bit generator based on a 2D discrete memristive hyperchaotic map," in *Proc. Panhellenic Conf. Electron. Telecommun. (PACET)*, Mar. 2024, pp. 1–4.

[19] M. Gabr, Y. Korayem, Y.-L. Chen, P. L. Yee, C. S. Ku, and W. Alexan, "R^3—Rescale, rotate, and randomize: A novel image cryptosystem utilizing chaotic and hyper-chaotic systems," *IEEE Access*, vol. 11, pp. 119284–119312, 2023.

[20] Z. Li, C. Peng, W. Tan, and L. Li, "A novel chaos-based color image encryption scheme using bit-level permutation," *Symmetry*, vol. 12, no. 9, p. 1497, Sep. 2020.

[21] R. Ge, G. Yang, J. Wu, Y. Chen, G. Coatrieux, and L. Luo, "A novel chaos-based symmetric image encryption using bit-pair level process," *IEEE Access*, vol. 7, pp. 99470–99480, 2019.

[22] H. Liu and X. Wang, "Color image encryption using spatial bit-level permutation and high-dimension chaotic system," *Opt. Commun.*, vol. 284, nos. 16–17, pp. 3895–3903, Aug. 2011.

[23] L. Teng and X. Wang, "A bit-level image encryption algorithm based on spatiotemporal chaotic system and self-adaptive," *Opt. Commun.*, vol. 285, no. 20, pp. 4048–4054, Sep. 2012.

[24] M. Kar, M. K. Mandal, D. Nandi, A. Kumar, and S. Banik, "Bit-plane encrypted image cryptosystem using chaotic, quadratic, and cubic maps," *IETE Tech. Rev.*, vol. 33, no. 6, pp. 651–661, Nov. 2016.

[25] K. U. Shahna and A. Mohamed, "A novel image encryption scheme using both pixel level and bit level permutation with chaotic map," *Appl. Soft Comput.*, vol. 90, May 2020, Art. no. 106162.

[26] J. Wang, R. Zhang, and J. Liu, "Partial-privacy image encryption algorithm based on time-varying delayed exponentially controlled chaotic system," *Nonlinear Dyn.*, vol. 112, no. 12, pp. 10633–10659, Jun. 2024.

[27] M. Wang, L. Teng, W. Zhou, X. Yan, Z. Xia, and S. Zhou, "A new 2D cross hyperchaotic sine-modulation-logistic map and its application in bit-level image encryption," *Expert Syst. Appl.*, vol. 261, Feb. 2025, Art. no. 125328.

[28] K. Konathalapalli, N. H. Bhat, and K. A. K. Patro, "An effective, permutation-only, bit-level image encryption system using chaotic maps," in *Proc. Int. Conf. Quantum Technol., Commun., Comput., Hardw. Embedded Syst. Secur. (iQ-CCHESS)*, Sep. 2023, pp. 1–7.

[29] J. Wen, X. Xu, K. Sun, Z. Jiang, and X. Wang, "Triple-image bit-level encryption algorithm based on double cross 2D hyperchaotic map," *Nonlinear Dyn.*, vol. 111, no. 7, pp. 6813–6838, Apr. 2023.

[30] X. Feng, G. Han, F. Yan, D. Shen, Z. Pang, and Q. Li, "Local bit-level image encryption algorithm based on one dimensional zero excluded chaotic map," *Phys. Scripta*, vol. 99, no. 6, Jun. 2024, Art. no. 065214.

[31] A. Toktas, U. Erkan, S. Gao, and C. Pak, "A robust bit-level image encryption based on Bessel map," *Appl. Math. Comput.*, vol. 462, Feb. 2024, Art. no. 128340.

[32] J. Zhang and H. Wen, "Dynamic feedback bit-level image privacy protection based on chaos and information hiding," *Sci. Rep.*, vol. 14, no. 1, p. 5742, Mar. 2024.

[33] Y. Xian, X. Wang, Y. Zhang, X. Yan, and Z. Leng, "A novel chaotic image encryption with FSV based global bit-level chaotic permutation," *Multimedia Tools Appl.*, vol. 82, no. 1, pp. 407–426, Jan. 2023.

[34] Y. Su, X. Wang, and H. Gao, "Chaotic image encryption algorithm based on bit-level feedback adjustment," *Inf. Sci.*, vol. 679, Sep. 2024, Art. no. 121088.

[35] M. Wang, X. Wang, T. Zhao, C. Zhang, Z. Xia, and N. Yao, "Spatiotemporal chaos in improved cross coupled map lattice and its application in a bit-level image encryption scheme," *Inf. Sci.*, vol. 544, pp. 1–24, Jan. 2021.

[36] X. Wang and M. Zhao, "A new spatiotemporal chaos model and its application in bit-level image encryption," *Multimedia Tools Appl.*, vol. 83, no. 4, pp. 10481–10502, Jan. 2024.

[37] W. Zhou, X. Wang, M. Wang, and D. Li, "A new combination chaotic system and its application in a new bit-level image encryption scheme," *Opt. Lasers Eng.*, vol. 149, Feb. 2022, Art. no. 106782.

[38] D. Wei, M. Jiang, and Y. Deng, "A secure image encryption algorithm based on hyper-chaotic and bit-level permutation," *Expert Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119074.

[39] S. F. Raza and V. Satpute, "A novel bit permutation-based image encryption algorithm," *Nonlinear Dyn.*, vol. 95, no. 2, pp. 859–873, Jan. 2019.

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[41] O. Kocak, U. Erkan, and I. Babaoglu, "Design and practical implementation of a novel hyperchaotic system generator based on Apéry's constant," *Integration*, vol. 103, Jul. 2025, Art. no. 102399.

[42] L. Moysis, I. Kafetzis, A. Tutueva, D. Butusov, and C. Volos, "Chaos-based image encryption based on bit level cubic shuffling," in *Cybersecurity: A New Approach Using Chaotic Systems*. Cham, Switzerland: Springer, 2022, pp. 157–191.

[43] L. Moysis, M. Lawnik, M. S. Baptista, C. Volos, and G. F. Fragulis, "A family of 1D modulo-based maps without equilibria and robust chaos: Application to a PRBG," *Nonlinear Dyn.*, vol. 112, no. 14, pp. 12597–12621, Jul. 2024.

[44] A. Golev, A. Iliev, and N. Kyurkchiev, "A note on the Soboleva'modified hyperbolic tangent activation function," *Int. J. Innov. Sci. Eng. Technol.*, vol. 4, no. 6, pp. 177–182, 2017.

[45] C. García-Grimaldo and E. Campos-Cantón, "Comparative analysis of chaotic features of maps without fixed points," in *Proc. 2nd Int. Conf. Complex Syst. Their Appl. (EDIESCA)*. Cham, Switzerland: Springer, Jan. 2022, pp. 151–176.

[46] O. A. Almatroud and V.-T. Pham, "Building fixed point-free maps with memristor," *Mathematics*, vol. 11, no. 6, p. 1319, Mar. 2023.

[47] L. Moysis, M. Lawnik, C. Volos, M. S. Baptista, G. F. Fragulis, and S. K. Goudos, "Validating a chaos based PRBG under different chaotic maps," in *Proc. Panhellenic Conf. Electron. Telecommun. (PACET)*, Mar. 2024, pp. 1–4.

[48] E. V Soboleva and V. V Beskorovainyi, *The Utility Function in Problems of Structural Optimization of Distributed Objects*. Kharkiv, Ukraine: Kharkiv Univ. Aair Force, 2008, p. 121.

[49] E. V. Soboleva, "The S-shaped utility function of individual criteria for multi-objective decision–making in design," *Kharkiv Nat. Univ. Radioelectronics*, vol. 247, Jan. 2009.

[50] EV Soboleva and VV Beskorovainyi, "Identification of utility functions in multi–objective choice modelling by using S–shaped functions," *Kharkiv Nat. Univ. Radioelectronics*, pp. 50–54, 2010.

[51] L. Moysis, M. Lawnik, and C. Volos, "Density-colored bifurcation diagrams—A complementary tool for chaotic map analysis," *Int. J. Bifurcation Chaos*, vol. 33, no. 15, Dec. 2023, Art. no. 2330036.

[52] J. A. C. Gallas, "Dissecting shrimps: Results for some one-dimensional physical models," *Phys. A, Stat. Mech. Appl.*, vol. 202, nos. 1–2, pp. 196–223, Jan. 1994.

[53] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep., 2001.

[54] Jan. (2019). *Datahash*. [Online]. Available: mathworks.com/matlabcentral/fileexchange/31272-datahash

[55] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Inf. Sci.*, vol. 222, pp. 323–342, Feb. 2013.

[56] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. (JSAT)*, vol. 1, no. 2, pp. 31–38, 2011.

[57] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation Chaos*, vol. 16, no. 8, pp. 2129–2151, Aug. 2006.

[58] M. Hussain, N. Iqbal, and Z. Bashir, "Image pixels swapping encryption based on the TetraVex game and a publicly hash-sharing algorithm," *Cluster Comput.*, vol. 27, no. 4, pp. 5355–5376, Jul. 2024.

[59] M. Hanif, N. Iqbal, F. Ur Rahman, M. A. Khan, T. M. Ghazal, S. Abbas, M. Ahmad, H. Al Hamadi, and C. Y. Yeun, "A novel grayscale image encryption scheme based on the block-level swapping of pixels and the chaotic system," *Sensors*, vol. 22, no. 16, p. 6243, Aug. 2022.

[60] N. Iqbal, M. Hanif, Z. U. Rehman, and M. Zohaib, "On the novel image encryption based on chaotic system and DNA computing," *Multimedia Tools Appl.*, vol. 81, no. 6, pp. 8107–8137, Mar. 2022.

[61] S. F. Yousif, A. J. Abboud, and R. S. Alhumaima, "A new image encryption based on bit replacing, chaos and DNA coding techniques," *Multimedia Tools Appl.*, vol. 81, no. 19, pp. 27453–27493, Aug. 2022.

[62] W. Feng, J. Zhang, Y. Chen, Z. Qin, Y. Zhang, M. Ahmad, and M. Woźniak, "Exploiting robust quadratic polynomial hyperchaotic map and pixel fusion strategy for efficient image encryption," *Expert Syst. Appl.*, vol. 246, Jul. 2024, Art. no. 123190.

[63] H. Zhao, S. Wang, and Z. Fu, "A new image encryption algorithm based on cubic fractal matrix and L-LCCML system," *Chaos, Solitons Fractals*, vol. 185, Aug. 2024, Art. no. 115076.

[64] L. Li, "A self-reversible image encryption algorithm utilizing a novel chaotic map," *Nonlinear Dyn.*, vol. 113, no. 7, pp. 7351–7383, Apr. 2025.

[65] 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 Dyn.*, vol. 113, no. 3, pp. 2811–2829, Feb. 2025.

[66] S. Zhou, Y. Wei, Y. Zhang, H. H.-C. Iu, and H. Zhang, "Image encryption algorithm based on the dynamic RNA computing and a new chaotic map," *Integration*, vol. 101, Mar. 2025, Art. no. 102336.

[67] M. M. Hazzazi, M. U. Rehman, A. Shafique, A. Aljaedi, Z. Bassfar, and A. B. Usman, "Enhancing image security via chaotic maps, fibonacci, tribonacci transformations, and DWT diffusion: A robust data encryption approach," *Sci. Rep.*, vol. 14, no. 1, p. 12277, May 2024.

[68] H. Wen, Y. Lin, Z. Xie, and T. Liu, "Chaos-based block permutation and dynamic sequence multiplexing for video encryption," *Sci. Rep.*, vol. 13, no. 1, p. 14721, Sep. 2023.

LAZAROS MOYSIS received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Mathematics, Aristotle University of Thessaloniki, Greece, in 2011, 2013, and 2017, respectively. He is currently a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Western Macedonia, Greece. His research interests include the theory of descriptor systems, chaotic systems, and their applications, including observer design, chaotification, chaos-based encryption, and chaotic path planning.

MARCIN LAWNIK received the M.Sc. degree in mathematics from the Faculty of Mathematics and Physics, Silesian University of Technology, Poland, in 2010, and the Ph.D. degree in computer science from the Faculty of Mechanical Engineering and Computer Science, Czestochowa University of Technology, Poland, in 2015. He is currently an Assistant Professor with the Faculty of Applied Mathematics, Silesian University of Technology. His research interests include the modeling of chaotic systems and chaos-based cryptography.

WASSIM ALEXAN (Senior Member, IEEE) was born in Alexandria, Egypt, in 1987. He received the B.Sc., M.Sc., and Ph.D. degrees in communications engineering and the M.B.A. degree from the German University in Cairo (GUC), Egypt, in 2010, 2012, 2017, and 2019, respectively, and the M.A. degree in educational leadership from The American University in Cairo (AUC), Egypt, in 2024.

He was with the Mathematics Department, from 2010 to 2017. Since 2017, he has been a member with the Faculty of Information Engineering and Technology, GUC, teaching various courses in relation to wireless communications, modulation and coding, information theory, digital logic design, circuit theory, and mathematics. In 2023, he was promoted to an Associate Professor of electrical engineering and information technology. He has also been an Associate Professor with the Mathematics Department, German International University (GIU), New Administrative Capital, Egypt, since 2019. He is the author or co-author of more than 100 journal articles and conference papers. He is listed among the top 2% of world scientists according to the Stanford/Elsevier ranking published, in October 2024. His research interests include wireless communications, information security, image and signal processing, mathematical modeling, and engineering education.

Dr. Alexan is also a Professional Member of ACM and has been granted the Best Paper Award at the 19th and 26th IEEE Conferences on Signal Processing Algorithms, Architectures, Arrangements, and Applications, (SPA'2015 and SPA'2023), Poznan, Poland, respectively; the AEG Writer of the Year Award from AUC, in 2019; and the Best Poster Award at the 37th IEEE National Radio Science Conference, Cairo, Egypt, in 2020. He serves as the Editor-in-Chief for IECE and *International Journal of Intelligent Signal Processing*, an Academic Editor for *IET Computers and Digital Techniques* journal, and a Guest Editor to the Springer Link journal *Discover Imaging*. Moreover, he is a Reviewer and a Technical Program Committee Member of various IEEE, IET, Springer, and Elsevier journals and international conferences.

SOTIRIOS K. GOUDOS (Senior Member, IEEE) received the B.Sc. degree in physics, the M.Sc. degree in electronics, and the Ph.D. degree in physics from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1991, 1994, and 2001, respectively, the M.Sc. degree in information systems from the University of Macedonia, Greece, in 2005, and the Diploma degree in electrical and computer engineering from the Aristotle University of Thessaloniki, in 2011.

He is currently a Professor with the Department of Physics, Aristotle University of Thessaloniki. He is the Director of the ELEDIA@AUTH Laboratory and a member of the ELEDIA Research Center Network. He is the author of the book *Emerging Evolutionary Algorithms for Antennas and Wireless Communications*, Institution of Engineering and Technology, in 2021. His research interests include antenna and microwave structure design, evolutionary algorithms, wireless communications, machine learning, and semantic web technologies. He is the founding Editor-in-Chief of *Telecom* open-access journal (MDPI Publishing). He is serving as Associate Editor for IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, IEEE ACCESS, and IEEE OPEN JOURNAL OF THE COMMUNICATION SOCIETY.

MURILO S. BAPTISTA received the B.Sc. and Ph.D. degrees in physics from the University of São Paulo, Brazil. He is currently a Reader with The University of Aberdeen, U.K., where he is the Head of the Department of Physics. Before joining UoA, he was a Postdoctoral Researcher with the University of Maryland, USA, from 1997 to 1999, the University of São Paulo, from 1999 to 2003, the University of Potsdam, Germany, from 2004 to 2006, as a Guest Scientist with the Max-Planck-Institute for the Physics of Complex Systems, Germany, from 2007 to 2008, and as a Guest Assistant Professor with the Centre for Applied Mathematics, University of Porto, Portugal, from 2008 to 2009. His research contributes to fields, such as communications, neuroscience, smart systems, and environmental sustainability. His research interests include nonlinear dynamics, complex systems, information theory, causality, communication and cryptography, and data science.

GEORGE F. FRAGULIS is currently a Full Professor with the Department of Electrical and Computer Engineering, University of Western Macedonia. He published more than 100 papers in peer-reviewed journals and international conferences (more than 20 IEEE journal and conference contributions. He participated in 15 (three as a PI) European (five Horizon 2020) programs and national research and development projects and serves as a reviewer to more than 100 international scientific journals and conferences in his area of expertise. His research interests include mathematical systems and control theory, analysis and synthesis of singular/descriptor systems, polynomial descriptions of linear multivariable systems, machine learning, fuzzy systems, robotics and autonomous systems, robotic vision, web programming, and applications. He served as the Program Chair in 14 and as a member of the Program Committee in more than 30 international conferences. He was a recipient of the Best Paper Award at the ACM International Conference on Educational Technology, Language, and Technical Communication, Fukushima, Japan (ETLTC-2020). He serves as the Editor-in-Chief for *International Journal of Entertainment Technology and Management* (Inderscience).