mathematics

MDPI

Article

Chaos and Cellular Automata-Based Substitution Box and Its Application in Cryptography

Arslan Shafique [1,*], Kashif Hesham Khan [2], Mohammad Mazyad Hazzazi [3], Ismail Bahkali [4], Zaid Bassfar [5] and Mujeeb Ur Rehman [6]

[1] Department of Electrical Engineering, Riphah International University, Islamabad 46000, Pakistan
[2] Department of Computer Science, RMIT University, Melbourne 3001, Australia; kashifhesham.khan@rmit.edu.au
[3] Department of Mathematics, College of Science, King Khalid University, Abha 61421, Saudi Arabia; mmhazzazi@kku.edu.sa
[4] Department of Information Science, King Abdulaziz University, Jeddah 21589, Saudi Arabia; ibahkali@kau.edu.sa
[5] Department of Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia; zbassfar@ut.edu.sa
[6] School of Science, Technology and Health, York St. John University, York YO31 7EX, UK; m.rehman@yorksj.ac.uk
* Correspondence: arslan.shafique@riphah.edu.pk; Tel.: +92-3334983664

Abstract: Substitution boxes are the key factor in symmetric-key cryptosystems that determines their ability to resist various cryptanalytic attacks. Creating strong substitution boxes that have multiple strong cryptographic properties at the same time is a challenging task for cryptographers. A significant amount of research has been conducted on S-boxes in the past few decades, but the resulting S-boxes have been found to be vulnerable to various cyberattacks. This paper proposes a new method for creating robust S-boxes that exhibit superior performance and possess high scores in multiple cryptographic properties. The hybrid S-box method presented in this paper is based on Chua's circuit chaotic map, two-dimensional cellular automata, and an algebraic permutation group structure. The proposed 16×16 S-box has an excellent performance in terms of security parameters, including a minimum nonlinearity of 102, the absence of fixed points, the satisfaction of bit independence and strict avalanche criteria, a low differential uniformity of 5, a low linear approximation probability of 0.0603, and an auto-correlation function of 28. The analysis of the performance comparison indicates that the proposed S-box outperforms other state-of-the-art S-box techniques in several aspects. It possesses better attributes, such as a higher degree of inherent security and resilience, which make it more secure and less vulnerable to potential attacks.

Keywords: chaos theory; cyberattacks; substitution boxes; data security

MSC: 94-04; 94-00; 68-00; 68-04

Citation: Shafique, A.; Khan, K.H.; Hazzazi, M.M.; Bahkali, I.; Bassfar, Z.; Rehman, M.U. Chaos and Cellular Automata-Based Substitution Box and Its Application in Cryptography. *Mathematics* **2023**, *11*, 2322. https://doi.org/10.3390/math11102322

Academic Editor: Manuel Alberto M. Ferreira

Received: 18 April 2023
Revised: 4 May 2023
Accepted: 8 May 2023
Published: 16 May 2023

1. Introduction

As online communication and data transfer continue to advance quickly, the need for security measures is becoming increasingly critical. It is essential to safeguard sensitive data as they travel across networks. Cryptography algorithms are utilized to achieve this objective, as they offer the necessary security over insecure network channels. Essentially, cryptography provides a way to encode and decode information so that it can only be accessed by authorized parties and cannot be intercepted by unauthorized third parties. This is crucial in ensuring the confidentiality and integrity of sensitive data during transmission [1]. Symmetric cryptosystems have been a cornerstone of cryptography for many years, serving as a key tool for implementing security measures. The concept of modern cryptography was first introduced by Shannon in 1949, paving the way for the

development of various types of block cryptosystems. These cryptosystems, such as the Data Encryption Standard (DES), Advanced Encryption Standard (AES), and BLOWFISH, rely on two fundamental concepts, confusion and diffusion, which were also introduced by Shannon in his work [2]. The process of confusion involves scrambling the pixels by shuffling either rows or columns. On the other hand, diffusion involves modifying the pixel values themselves to spread their influence throughout the image. By using both confusion and diffusion together, the correlation between the original image pixels can be broken, resulting in a highly secure encryption system.

These concepts are used to ensure that the encrypted data are secure and cannot be easily deciphered, even if an unauthorized individual obtains the encryption key. Encryption is primarily used to create a secure and confidential communication channel by transforming plain text into an unreadable form known as ciphertext. The main objective of encryption is to ensure that the ciphertext is only intelligible to authorized parties who possess the necessary decryption key. To achieve this, encryption techniques are designed to make it difficult for attackers to steal sensitive information. Additionally, another technique that is commonly employed is "diffusion," which serves to decrease the impact of a single piece of plaintext on encrypted text [3,4]. This is implemented to mask the statistical redundancy of the plaintext, making it harder to decipher. In order to achieve both confusion and diffusion, block ciphers are used. A block cipher repeatedly performs a process to generate multiple effects of these properties, resulting in an encrypted text that is extremely difficult to decode without the correct key.

Block ciphers are encryption methods that divide the plaintext into fixed-size blocks and apply a specific algorithm to each block. Several well-known aforementioned block ciphers use this technique. However, block ciphers are vulnerable to various attacks, including entropy attacks, energy attacks, and other differential attacks [5,6]. Differential cryptanalysis aims to identify patterns in encrypted data. To achieve this, the attacker uses specific input sets to track changes in the output [7–10]. To make encryption algorithms more secure and strengthen them against various cyberattacks, a random association is generated by the confusion component between the ciphertext and the key, which makes it more challenging for potential attackers to identify any patterns or relationships within the encrypted data [11–14]. A substitution box (S-box), a crucial building block of block ciphers, is designed to break the correlation between the input image pixels. Essentially, it replaces a block of plaintext bits with another block of bits (the ciphertext). By applying a nonlinear substitution, the S-box makes it difficult for unauthorized parties to decipher the original input from the output, enhancing the encryption's security [15–18]. It takes in a certain number of input bits and transforms them into a corresponding number of output bits by using a nonlinear transformation. S-boxes play an important role in the security of a cryptographic system, as they are responsible for producing high levels of nonlinearity that make it difficult for attackers to predict the output of the cipher. In order to be effective, S-boxes must be carefully designed to resist various types of attacks and should be able to produce output that is robust and resistant to attempts at decryption. The effectiveness of the substitution box in the AES block cipher has been a significant contributor to its widespread adoption and success [19]. This substitution box, or S-box, has been developed and improved using various techniques, including algebraic methods, optimization, chaotic functions, and structures. Efficient S-boxes of different sizes are difficult to create due to the extremely large search space involved. To address this challenge, meta-heuristic optimization approaches have been developed recently to establish a reliable framework for generating useful S-boxes. These techniques have helped in creating stable and efficient S-boxes for the AES block cipher.

S-boxes are necessary components of various cryptographic algorithms. These algorithms include symmetric block ciphers, symmetric stream ciphers, and hash functions [20–22]. The primary purpose of an S-box is to provide nonlinear substitution, which is a crucial element in achieving robust security in cryptographic systems. By introducing

nonlinearity, S-boxes make it harder for attackers to exploit any linear relationships within the system.

In symmetric block ciphers, S-boxes play a vital role in transforming plaintext data into ciphertext during the encryption process. Symmetric ciphers such as the Advanced Encryption Standard (AES) use S-boxes in multiple rounds to increase the complexity of the cipher and, consequently, its resistance to attacks. Similarly, in symmetric stream ciphers, S-boxes help to substitute the pixel values with other integer values ranging from 0 to 255 to produce the ciphertext. The nonlinear nature of S-boxes ensures that the keystream remains unpredictable and difficult to reverse-engineer. Moreover, hash functions, which are used to generate a fixed-size output (hash) from input data of arbitrary size, also benefit from S-boxes. In hash algorithms such as the Secure Hash Algorithm (SHA), S-boxes contribute to the avalanche effect, where a small change in input data leads to a significant change in the output hash. This characteristic is essential for preventing attackers from deducing any information about the input data based on the hash.

2. Related Work

S-boxes are essential for achieving strong security in encryption methods. An S-box typically takes in a specific number of input bits and transforms them into a different number of output bits; this transformation function is called an $m \times n$ S-box. S-boxes transform the input bits in a nonlinear manner, resulting in a unique output for each input value, and this transformation process involves mapping 2^m possible input values to 2^n possible output values [23–25]. S-boxes can be implemented as a lookup table that has 2^m words, with each word consisting of n bits. It is important to note that m and n can be equal or different. When the input m and output n are equal, the S-box is said to be bijective. In other words, each possible input stream is mapped to a unique output stream, and vice versa. Bijective S-boxes can be represented as a permutation sequence of integers in the range $[0, 2n - 1]$. This means that there are $(2n)!$ possible permutations, which is an extremely large number. For instance, there are more than 10^{50} for 8×8 S-boxes.

In the literature, the study of chaotic systems has been extensively used to create secure S-boxes, which are crucial components of many cryptographic systems. The goal of using chaotic maps is to design S-boxes that are resilient to various cyberattacks, including linear and algebraic attacks. Researchers have demonstrated that by using chaotic systems such as the Lorenz system, it is possible to create secure S-boxes. For instance, Ozkaynak et al. [26] proposed secure S-boxes using the Lorenz chaotic system [27]. Additionally, Wang et al. [28] proposed an S-box scheme that utilizes chaotic maps and a genetic algorithm to ensure robustness against attacks such as entropy attacks [29–31]. Yin et al. [32] used chaotic maps in an iterative process to generate S-boxes, while Lambi'c [33] utilized an advanced one-dimensional chaotic map. Ozkaynak et al. [34] employed a fractional-order Chen chaotic system. In [35], Quiroga et al. proposed dynamic S-boxes in which an improved logistic map [36,37] is employed. Tanyildizi et al. [38] generated S-boxes with robust algebraic and differential properties by utilizing an optimized 1D chaotic map [39]. El-Latif et al. [40] utilized chaos and quantum walk to create a powerful cryptographic S-box. However, relying solely on chaotic systems for generating S-boxes has some drawbacks [41]. Designing S-boxes based on chaos theory can be a computationally demanding task, particularly when generating key-dependent S-boxes [42]. In addition, many of these designs rely on continuous-time systems that may be susceptible to cyberattacks. Even a minor change in the value of a digitized continuous-time system can have a substantial impact on its future behavior. Additionally, these S-box designs require complex hardware implementations. In light of these issues, researchers have explored various techniques to enhance the features of S-boxes. Some of the alternative techniques that have been investigated include algebraic theories such as cellular automata [43] and elliptic curves [44].

One method for creating an S-box involves using a combination of linear fractional transformation (LFT) and a Gaussian distribution [45,46]. To generate the S-box, the process involved several steps, such as the Box–Muller transform and central limit technique.

In [47], Ahmad et al. proposed a different method for generating random S-boxes of size $M \times M$ (where $M \leq 16$). In addition, Basha et al. [48] utilized a DNA technology to develop S-boxes for image encryption, while Farhan et al. [49] employed RNA computing techniques to produce multi-S-boxes. However, a few S-box designs have several weaknesses, such as low nonlinearity [50]. Therefore, several researchers have attempted to use different types of elliptic curves to generate S-boxes that have reduced computational costs. Azam et al. [51] developed efficient S-boxes using Mordell elliptic curves (MEC) over prime fields [52]. In [52], Ullah et al. employed ordered Mordell elliptic curves to create efficient S-boxes with lower time and space complexities, facilitating quicker generation operations [53]. Hayat et al. [54] presented a methodology to create new S-boxes that could be used in image encryption applications. Their approach involved an exhaustive method for generating points on finite elliptic curves, which were then used to create a randomized S-box [55]. In [56], an image encryption technique is proposed in which quasi-resonant triads are used to design a pseudo-random number generator (PRNG) for computing the multiplicative error correction (MEC) needed to construct the substitution box (S-box) [57–60]. By utilizing the resulting PRNG and S-box, the encryption process achieves both diffusion and confusion, which are essential for security. The development of an S-box with robust cryptographic properties is critical for any cryptographic system. Consequently, researchers have explored diverse strategies to achieve specific performance objectives when constructing 8×8 S-boxes.

In this paper, a novel method for constructing S-boxes is introduced that integrates ideas from 2D cellular automata, Chua's circuit map, and algebraic theory. The approach begins by using 2D cellular automata to generate an initial S-box. To improve the performance and security of the S-box, an algebraic permutation with a customized structure is employed. This results in an S-box that is both robust and secure. The constructed S-box is shown to have excellent security and robustness through a performance comparison analysis. Overall, several existing approaches demonstrate the variety of methods that can be used to generate S-boxes using chaotic systems and highlight the potential benefits, along with the limitations, of using such types of chaotic systems in cryptography.

The rest of the paper is structured as follows: In Section 3, we provide an introduction to the relevant concepts of cellular automata for generating the proposed S-box. In Section 4, the proposed methodology is discussed, which leverages 2D cellular automata, Chua's circuit map, and the algebraic permutation group structure. Section 5 is dedicated to the security assessment and comparative analysis of the constructed S-box, and Section 6 concludes the proposed work.

3. Cellular Automata

John von Neumann and Stanisław Ulam [61] were the first to explore the mathematical model of the cellular automaton (CA). A CA is a simple system with discrete states, space, and time instances. Cellular automata are mathematical models that consist of a regular and structured grid of individual cells. These cells can interact with each other in a way that resembles random phenomena, despite the regularity of their arrangement. The structure of cellular automata can be analyzed as a mesh of cells, where each cell has a binary value of either 0 or 1. Alternatively, it can also be viewed as a grid, network, or array of cells, where each cell stores a single bit of information. At each step in time, each cell in the automaton updates itself independently based on its current state and the states of its neighboring cells. A progress function is a tool used in network analysis to track the development, or "age", of the system. The application of this function occurs across all cells in the network, with consideration given to the input state of each individual cell. The cellular-automata-based rules used to determine the transition to the next state are simultaneously applied across all cells.

To begin, an initial state is assigned to each cell at $T = 0$. Then, as time progresses $(T + 1)$, a predetermined rule (usually in the form of a function) is applied to determine the new state of each cell based on its current state and the states of its neighboring cells. The

rule used to update the cell states is typically the same for every cell and remains constant over time, applied to the entire grid or network. This process of applying the same rule to every cell in the network at each time step can be used to model complex systems and observe their behavior over time.

The cellular automaton (CA) is a mathematical concept that can be described using four different components, or "tuples." These four tuples are D, S, N, and F, and together they define the properties of a specific cellular automaton.

- D represents the set of all possible states that a cell in the automaton can have. This could include any number of different states, depending on the specific problem being studied.
- S is the set of all cells in the automaton. This defines the spatial structure of the automaton and the number of cells that make up the system.
- N defines the set of neighboring cells that each cell in the automaton interacts with. The neighborhood can vary depending on the specific problem being studied, and it can be defined in many different ways.
- Finally, F is the set of rules that determine how each cell in the automaton evolves over time. These rules can be quite simple or quite complex, and they determine the behavior of the entire automaton as it evolves over time.

Together, these four tuples define the formal structure of a cellular automaton, and they provide a framework for understanding the behavior of complex systems that can be modeled using this mathematical concept. Let the number of cells of any dimension be represented by N. Then, the mathematical representation of the CA state is given in Equation (1).

$$Z_T = \{E_0(T), E_1(T), E_2(T), \cdots, E_{N-1}(T)\} \tag{1}$$

where $E_j(T)$ shows the state of the jth cell at time instant T. Similarly, at time instant $(T+1)$, Equation (1) will be:

$$Z(T+1) = \{E_0(T+1), E_1(T+1), E_2(T+1), \cdots, E_{N-1}(T+1)\} \tag{2}$$

where $E_j(T+1) = G\left[E_{j-1}(T), E_j(T), E_{j+1}(T)\right]$, and the quantity of neighbors is contingent upon the radius of the neighborhood. Here, two neighbors are considered, one before the jth cell and the other one after the jth cell, and it is referred to as a neighbor with a radius of 1. The automata states are defined as:

$$\left.\begin{aligned} Z(T) &= \{E_0(T), E_1(T), E_2(T), \cdots, E_{N-1}(T)\} \\ Z(T+1) &= \{E_0(T+1), E_1(T+1), E_2(T+1), \cdots, E_{N-1}(T+1)\} \\ E_j(T+1) &= G[E_{j-1}(T), E_j(T), E_{j+1}(T)] \end{aligned}\right\} \tag{3}$$

where Z_T = the state of automata at time instant T; E_{T+1} = the state of automata at time instant $(T+1)$; $E_j(T)$ = the jth cell at time instant T; and $E_j(T+1)$ = the jth cell at time instant $(T+1)$.

A one-dimensional CA can be illustrated as a linear array of cells, as shown in Figure 1. Let the length of the automaton be denoted as N, and the radius, r, be set to 1. The neighborhood of a given ith cell is N_i, which includes the cell itself plus $2r$ adjacent cells.

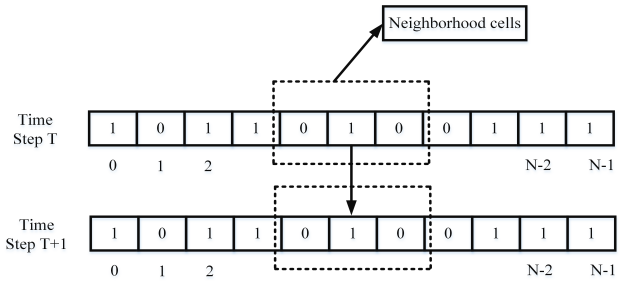


Figure 1. One dimensional CA represented as linear array.

By considering a small-scale CA with tightly encircled borders, the formation of a circular grid occurs. The following statistical measures are applicable to a one-dimensional cellular automaton of length N and with a radius of 1.

- $N = 3$ is for a neighborhood cell;
- $k = 2^N = 2^3 = 8$ is the length of the rule;
- Overall, the number of rules = $2^k = 2^8 = 256$.

Rule numbers 194 and 90 are specific rulesets for a one-dimensional cellular automaton (CA). There are $2^3 = 8$ possible configurations of these three cells (see Table 1), and each configuration can lead to either a 1 or 0 state in the next time step. In a one-dimensional CA, the cell's next state depends on its current state and its two neighbors, resulting in 256 possible rules represented by an integer from 0 to 255.

Rule numbers 194 and 90 are decimal representations of the specific rulesets. To understand how these rules work, the decimal rule number is converted into an 8-bit binary number, where each digit corresponds to the output of a specific 3-cell neighborhood configuration.

For example, Rule 194 in binary is 01000011. The binary digits, read from right to left, correspond to the outputs of the 8 possible configurations, from 111 to 000. Similarly, Rule 90 in binary is 01011010. The outputs for Rules 194 and 94 are displayed in Table 1.

Table 1. Rule generation for 1D CA.

State of Neighborhood	000	001	010	011	100	101	110	111	Rule
State (T)	0	1	0	0	0	0	1	1	194
State ($T + 1$)	0	1	0	1	1	0	1	0	90

Rule 90 generates a fractal pattern called the Sierpinski triangle. It demonstrates interesting properties, such as being reversible and producing intricate yet repetitive structures. Rule 90 is suitable for applications where self-similarity or fractal patterns are needed or for exploring reversible computation. On the other hand, Rule 194 produces more irregular patterns, which can be useful for modeling stochastic or chaotic systems. Therefore, for the proposed work, Rule 194 is more appropriate for image encryption that requires a higher degree of randomness or complexity.

In a 2D CA, the value of a cell in the next time step depends on the values of its neighboring cells in the 2D grid. Such 2D CAs have a wide range of applications, including designing encryption techniques, processing images, analyzing biological data, detecting intrusions, and testing patterns. The two-dimensional cellular automaton consists of a grid of cells, each of which is updated according to a rule that depends on its neighbors in all four directions, as depicted in Figure 2. Figure 2 shows an example of this method, with a red cell and its neighboring cells shaded in gray.

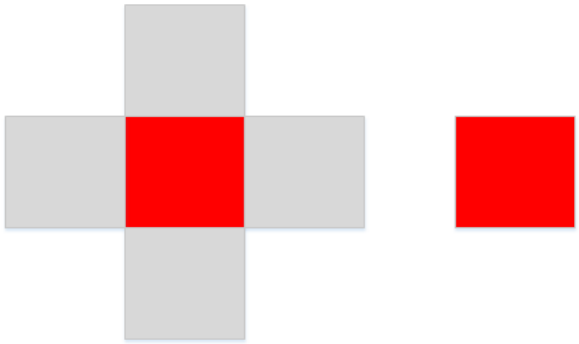
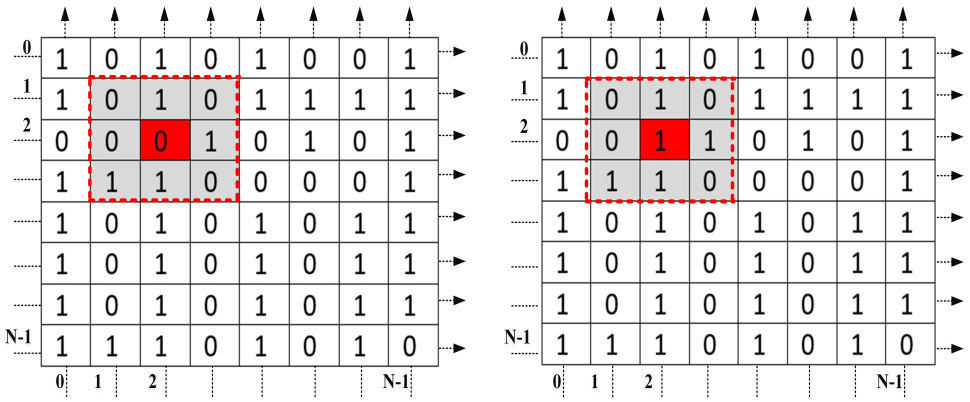


Figure 2. Cell representation: radius (r) = 1.

Von Neumann's method defines neighborhood cells only in the horizontal and vertical directions, and the number of cells included in the neighborhood may depend on the neighborhood radius. However, only cells that are either directly above, below, to the left, or to the right of the updating cell are considered for the update. The neighborhood radius r in a 2D cellular automaton can be any integer, such as 1 or 2. An example of a 2D CA with size $M \times M$ is shown in Figure 3a, where a red cell with a value of 1 at time T is updated according to a rule (e.g., 2361501363) and its neighboring cells (shown in gray shades). All cell values are updated according to the rule, resulting in a new state of the CA at time $(T+1)$. The updated cell values, such as from 11010 to 11110, are shown in Figure 3b. The details of generating the rules are given in Section 4.



(**a**) At time T (**b**) At time $(T+1)$.

Figure 3. Cellular automata with a size of $N \times N$.

4. Proposed Technique for S-Box Generation

The proposed cryptographic scheme uses 2D CA, Chua's circuit map, and an algebraic group structure to design a robust S-box. The 2D CA is first initialized using the PWLCM chaotic map to achieve good randomness throughout the automaton. The randomly initialized cells of the 2D CA are then updated using automata rules for a specified number of iterations. The proposed technique has six major stages: (a) initialization, in which a basic system is employed to define a random function for obtaining the initial values, (b) the definition of neighboring cells, (c) the generation of rules and the updating of cell values, (d) the window selection method, (e) the generation of S-box values from the window, and (f) the proposed group structure and its functioning.

4.1. Initialization

A CA can be initialized by setting the initial state of the cells in the grid. In simple terms, it means defining whether each cell is 1 or 0 at the beginning of the simulation. This initial configuration serves as the starting point for the cellular automata to evolve over

time according to the chosen ruleset. There are several ways to initialize cellular automata, some of which include random initialization, uniform initialization, seed or pattern initialization, and user-defined initialization [62]. In the proposed work, the category of manual initialization of the CA is chosen.

In order to create a lookup table that can be modified later, the first step is to initialize the cellular automaton (CA). The proposed method uses a simple system to generate a random function that obtains an initial value for the CA, which leads to a less complex and more time-efficient process. To start, the defined sliding window value is used to calculate new values for all initial values. Afterward, a system-based function is employed to randomly initialize an S-box with dimensions of 16×16 for the proposed method. Starting with a seed automaton is necessary to obtain automata with random values. There are many chaotic maps that can be used to generate pseudo-random numbers with excellent randomness using suitable control parameters and initial conditions. In the proposed method, Chua's circuit is used to initialize the 2D automaton.

4.2. Chua's Circuit

Chua's circuit is a type of nonlinear electronic circuit system that is known for its complex and dynamic behavior, as well as its ability to produce pseudo-random and unpredictable outputs. This makes it useful for a variety of applications, including cryptography and secure communication. The circuit is defined by Equation (4), which describes the behavior of the system.

$$\left.\begin{aligned} \dot{r} &= \boldsymbol{\varphi}\Big(s - r - g(r)\Big) \\ \dot{s} &= r - s + u \\ \dot{u} &= -\boldsymbol{\varpi} s \end{aligned}\right\} \tag{4}$$

where the function $g(r)$ represents the electrical response of a nonlinear resistor. It is described as a piecewise linear function with two segments and is given in Equation (5).

$$g(r) = \sigma r + (\boldsymbol{\sigma} - \boldsymbol{\mu})\left\{\frac{|r+1| - |r-1|}{2}\right\} \tag{5}$$

where the inner and outer segment slopes are denoted by $\boldsymbol{\sigma}$ and $\boldsymbol{\mu}$, respectively. The details on plotting Equation (5) are provided in [63]. The values of $\boldsymbol{\varphi}$ and $\boldsymbol{\varpi}$ in Equation (4) are determined by the specific values of circuit elements and state variables, such as voltage on capacitors and current on inductors. As the value of $\boldsymbol{\varphi}$ increases, the system undergoes a series of asymmetrical bifurcations, which gradually lead to the formation of two asymmetrical attractors. This results in chaotic behavior, where the system exhibits a double vortex chaos attractor.

This system consists of five components, which are two capacitors, one inductor, one resistor, and one Chua's diode, which is a type of nonlinear resistor. These components can be easily built using op-amps. When the system's parameters are set to specific values, the system exhibits a type of chaotic behavior called "double vortex chaos attractor". This behavior occurs when the values of $\boldsymbol{\varphi}$, $\boldsymbol{\varpi}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\mu}$ are set to 10.2, 15.48, -1.1626, and -0.5989, respectively, and the initial conditions for r, s, and u are set to 0.15. The behavior of the system is displayed through the attractors, which are a way of visualizing the system's chaotic behavior, as shown in Figure 4.

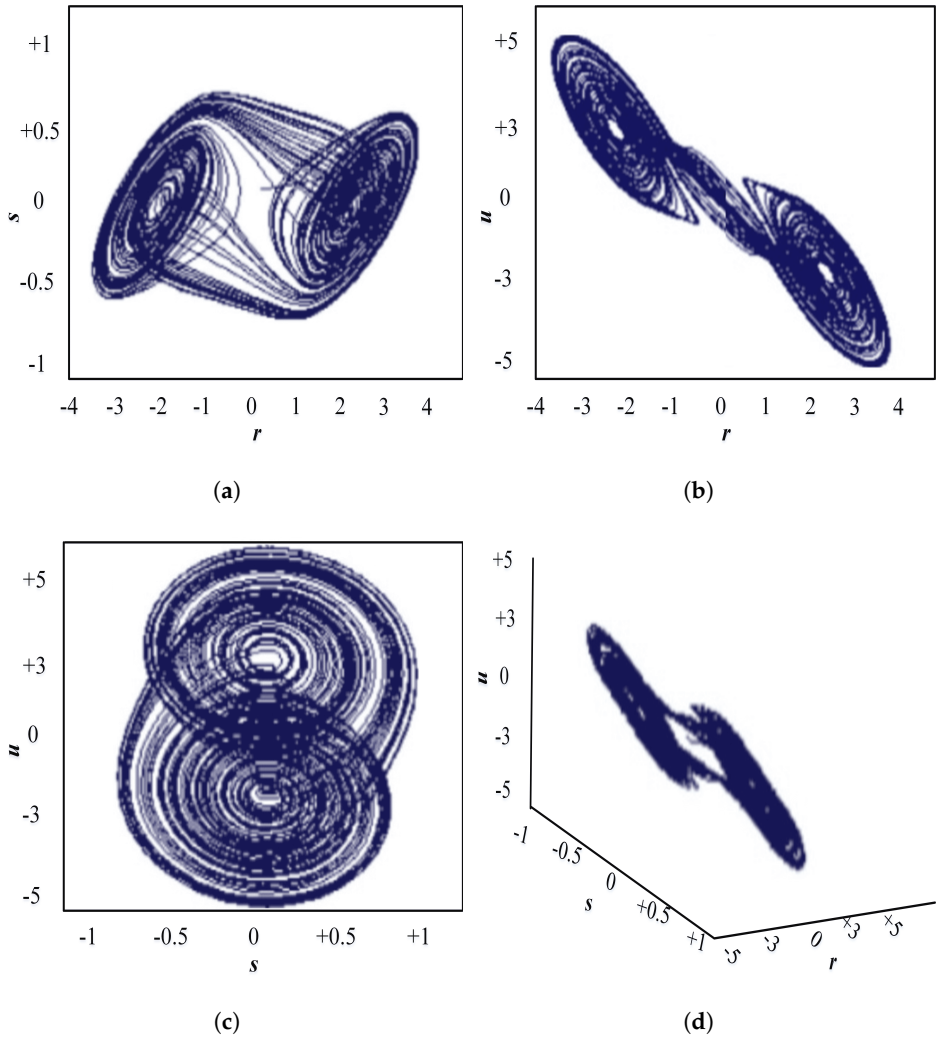


Figure 4. Chua's circuit system attractors. (**a**) r, s plane. (**b**) r, u plane. (**c**) s, u plane. (**d**) r, s and u plane.

4.3. Definition of Neighboring Cells

Von Neumann's method is a technique used to determine the neighboring cells in a 2D cellular automata system. To determine the rule for updating a cell's state at $(T+1)$, a specific sequence and order of indices for each cell must be established. These indices are defined as R and C, where R represents the row number and C represents the column number of the cell located at (R, C).

Figure 5 provides an illustration of the layout of all the neighboring cells of a specific cell located at (R, C), along with the indices of each of these neighboring cells.

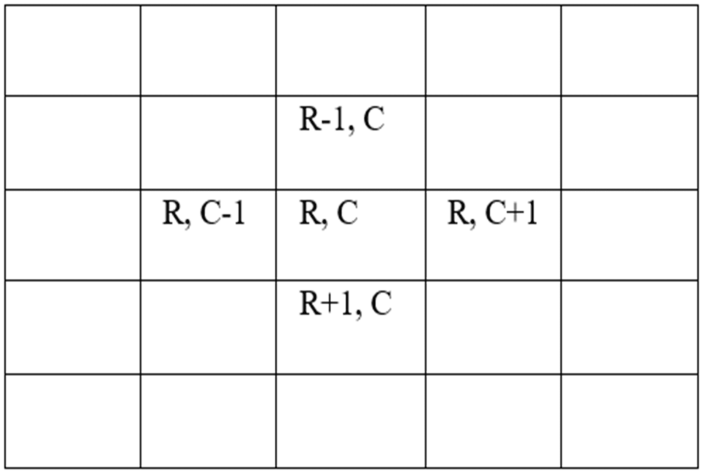


Figure 5. The neighboring cells of the cell located at (R, C) according to von Neumann's method.

4.4. Generating Rules and Updating the Cell Values

In cellular automata theory, the process of generating rules is extremely important. Cellular automata can produce different rules even if they start with the same initial value. These rules define how cellular automata work. For one-dimensional cellular automata, there are at most 256 rules. However, for 2D as well as 1D cellular automata, each cell's state update at time $(T + 1)$ depends not only on its previous state but also on the previous state values of its neighbors and the specific rule being used. These updates can be thought of as Boolean functions, which are mappings that take n inputs of either 0 or 1 and output either 0 or 1.

The local rule that represents these cell state updates can be defined based on these Boolean functions.

$$C^{T+1}_{[R,C-1]} = f\left(C^{T}_{[R+1,C]}, C^{T}_{[R,C]}, C^{T}_{[R,C+1]}, C^{T}_{[R-1,C]}\right) = 0 \text{ or } 1 \text{ (based on the rule)} \quad (6)$$

where T is the time stamp, $C^{T+1}_{[R,C]}$ is the cell state $[R, C]$ at time $(T + 1)$, and $C^{T}_{[R,C]}$ is the cell state $[R, C]$ at time T.

In a 2D CA with defined neighbors for radius r = 1, the local rule involves 5 variables, each of which can have 2 states, resulting in a total of $2^5 = 32$ possible combinations of the variables. This means that there are 2^{32} = 4,294,967,296 different rules that can be used to update the cells. When r is equal to 1, there are a total of five cells that are taken into account to decide the value of the cell being updated. This includes the cell being updated and four other cells. The new value of the cell being updated at time $T + 1$ is determined based on the values of all five cells at time T.

Since there are five cells that need to be arranged, including the cell being updated, there are five factorial (5!) ways to arrange them. The proposed method uses a specific arrangement, which is shown in Figure 6.

Binary: 11100001010101111101011101000011
Rule (decimal equivalent): 3780630339

The rule number for a 2D CA can be determined by arranging the five cells and updating the value of the cell at $(T + 1)$ based on the values of all five cells at T. For each possible arrangement of the five cells, there are 32 different truth tables that can be generated, which correspond to 32 different values for the cell at $T + 1$. By taking the decimal value of each of these 32 arrangements at time $T + 1$, a unique rule number can be generated for CA. Figure 7 shows an example of this process for a specific rule number (decimal equivalent: 7561260679).

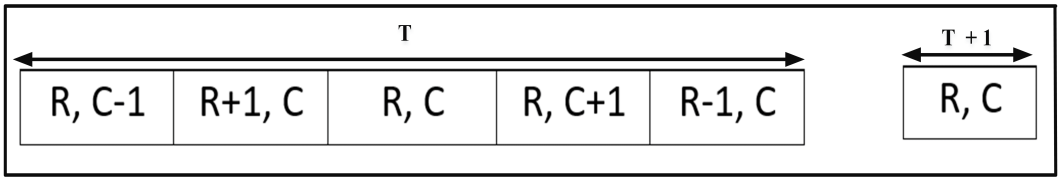


Figure 6. Cell arrangement.

T	00000	00001	00011	00100	00101	00110	00111	01000
T + 1	1	1	0	1	0	1	0	0

T	11000	11001	11010	11011	11100	11101	11110	11111
T + 1	0	0	0	1	1	1	0	0

Figure 7. Example of rule generation.

4.5. Window Selection Method

The process of window selection plays a crucial role in the proposed scheme, as it enables the algorithm to identify the specific element within the anticipated S-box. This is carried out by calculating the value of the S-box cell from the binary values of the 2D automata cells using a sliding window technique. The value of the S-box cell is filled by sliding the window to obtain the next cell value. Since we need 8 bits to represent a value in the range (0, 256), a 2×4 window is used to fulfill this requirement. This ensures that the window can generate all 256 values for the S-box by sliding over the binary cells of the two-dimensional automata.

To calculate the value of the S-box cell, the initialization of the window is at location (0, 0) of the 2D CA, and the calculated value is then appended to the S-box table. The process of obtaining the next value of the S-box involves shifting a window of two positions to the right and checking the calculated value in the S-box table. Whenever the value is absent from the table, it is included, and the window is shifted again. This process continues until the window reaches the bottom-right corner of the automaton. After this, the automaton is updated by invoking the update function, and the rest of the computations are performed using a similar approach. This sliding or shifting of the window from left to right and top to bottom allows the proposed scheme to compute all 256 unique values of the 16×16 S-box table. This approach ensures that each value in the S-box table is unique and that the entire table is generated efficiently and accurately. Figure 8 illustrates the process of sliding windows used in the proposed scheme.

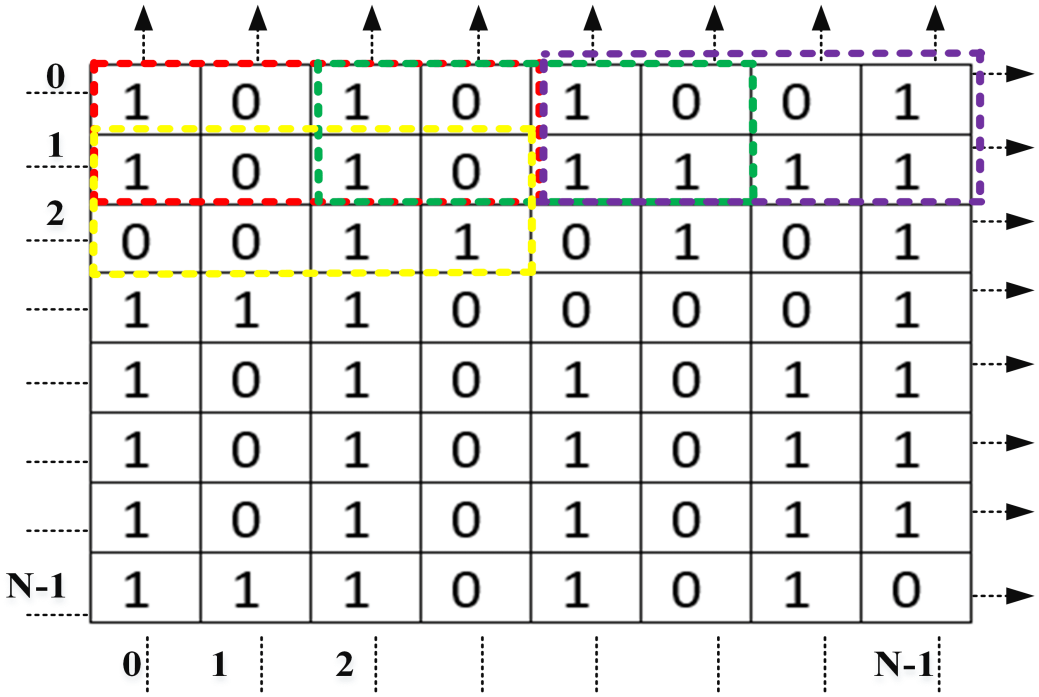


Figure 8. Process of sliding windows.

4.6. Generating S-Box Values from Window

In this section, the process of calculating decimal values from a 2×4 window is explained. A linear order is taken for the current window rows, with the first row being appended after the second row. The decimal equivalent number of the resulting 8-bit binary number is then calculated. Figure 9 illustrates this process, which uses the first window from Figure 7 as an example.

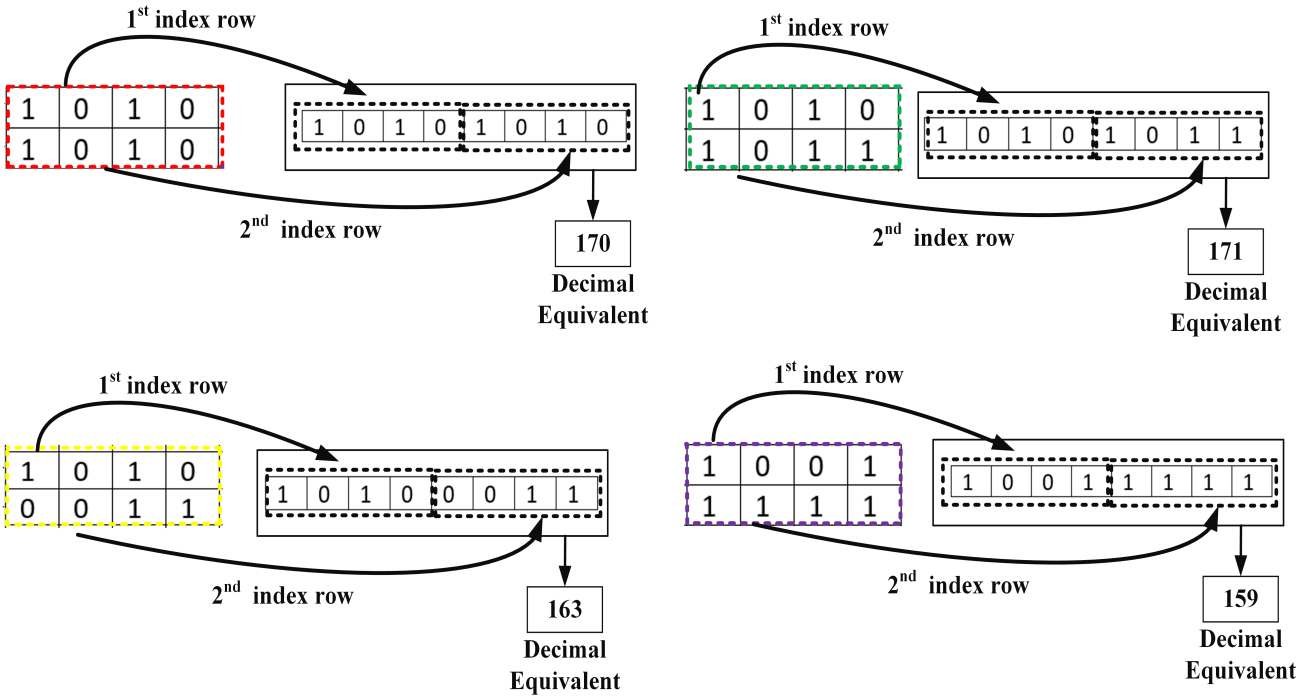


Figure 9. Calculating value based on the current sliding window.

The proposed S-box scheme consists of several components. Firstly, an empty S-box is created. Secondly, a 2D cellular automaton is generated. The automata are further iterated a pre-defined number of times to enhance randomness. The third part of the algorithm involves using a sliding window technique to compute the S-box values from the 2D automaton. This involves comparing the calculated value with the value already present in the automaton. If the calculated value matches the existing S-box value, the window is slid to calculate a new unmatched value. On the other hand, if the calculated value does not match, it is appended to the S-box.

The proposed method works as follows:

- Initially, the 8×8 matrix is empty.
- The 8×8 S-box is first filled with random values, which will be replaced later with values computed using 2D cellular automata.
- The initialization of the 2D CA lookup table is created using a chaotic tent map as follows:
  - Set the initial conditions.
  - Iterate the chaotic map 100 times. This will generate 100 different values, ranging from 0 to 1.
  - Multiply each value with any large integer number to amplify the created values.
  - Truncate the numbers that are placed right after the decimal point.
  - Take the modulo of the values generated in the previous step as mod(value, 16), and add +1 to each value to restrict the values in the range 1 to 16.
  - Choose the first 16 unique values to generate a permutation vector.
  - Similarly, a 2D lookup table can be created by selecting the first 256 unique values obtained after performing a modulus operation on the values with 256.
- Two functions are defined for generating an output bit based on the defined rule and updating the initial automata for a specified number of iterations.
- The S-box (S_1) is updated with the calculated values obtained from the automata using a sliding window protocol.
- Using the permutation group, S_1 is again updated. The permutation group has 16 different permutation vectors.
- Finally, the resulting substitution box exhibits strong nonlinearity and other cryptographic properties.

4.7. Proposed Group Structure and Its Functioning

Algebraic theories have been found to be useful in designing S-boxes since the successful implementation of the AES S-box [64]. Various proposals have been made for S-boxes that rely on algebraic structures. However, S-boxes that are solely based on algebraic techniques are vulnerable to algebraic attacks. To make algebraic attacks more difficult, there is a need to integrate the concepts of chaos, cellular automata, and algebraic techniques. Limited research has focused on improving S-boxes using effective algebraic structures. By combining these different approaches, the design of S-boxes can become more secure and less susceptible to attacks. To improve the security of S-boxes, it is important to develop a strong algebraic structure that can be used to enhance the S-box's security. To achieve this, a hybrid method is proposed that involves designing and implementing an algebraic permutation group. After conducting extensive experimentation, a suitable algebraic structure was designed, and its action on the S-box, obtained by using 2D CA, was found to enhance the S-box's security strength. The resulting S-box (S_1) that is obtained after the action of the suggested algebraic group is presented in Table 2.

Table 2. Proposed S-box (S_1).

91	109	125	165	190	25	250	161	242	149	198	162	207	241	213	201
180	132	40	108	9	205	131	140	160	172	88	138	249	233	57	0
116	85	45	133	105	44	82	225	113	211	43	181	216	251	252	230
112	193	119	86	141	3	159	27	221	127	4	67	156	202	196	186
68	121	103	191	92	78	143	95	76	55	224	238	179	195	46	7
236	97	59	120	176	150	94	168	29	227	173	209	234	20	239	5
130	53	1	56	114	158	171	110	69	26	215	37	232	35	199	237
71	16	72	10	118	145	208	200	107	147	220	51	183	204	18	226
23	128	24	129	73	167	38	124	137	163	135	240	245	42	212	192
81	79	101	75	223	84	50	90	184	136	188	218	144	229	8	14
99	32	153	28	175	197	117	33	64	63	2	194	169	170	185	6
157	62	152	70	74	123	65	30	247	248	155	52	203	54	246	11
104	111	100	106	222	93	243	89	48	142	164	19	60	13	187	244
146	126	154	139	178	49	22	36	41	174	134	34	228	177	255	12
96	80	102	235	115	39	83	122	58	217	189	219	182	210	254	253
166	206	151	87	231	47	98	66	77	31	148	17	61	214	21	15

To improve the proposed S_1, a set of sixteen permutation vectors is introduced in a permutation group, as shown below:

$P_1 = \{11,4,16,12,2,7,1,3,13,8,6,5,9,15,14,10\}$ $P_2 = \{6,14,3,16,4,1,10,12,8,2,11,9,13,7,5,15\}$

$P_3 = \{14,10,13,11,8,12,15,1,4,3,9,6,5,7,2,16\}$ $P_4 = \{11,15,5,7,3,2,16,13,14,4,8,9,12,6,10,1\}$

$P_5 = \{2,14,9,10,8,15,12,4,6,1,13,5,3,11,7,16\}$ $P_6 = \{12,7,11,3,13,65,14,10,15,2,4,8,9,1,16\}$

$P_7 = \{14,1,7,16,3,15,8,6,4,11,2,12,5,10,9,13\}$ $P_8 = \{1,11,2,13,9,14,7,4,12,5,8,16,10,15,6,3\}$

$P_9 = \{3,4,13,12,16,7,10,5,15,9,14,2,6,8,11,1\}$ $P_{10} = \{6,14,8,12,15,7,11,4,5,3,1,9,10,13,2,16\}$

$P_{11} = \{8,2,12,16,7,11,10,14,15,6,9,1,3,4,13,5\}$ $P_{12} = \{10,1,11,14,13,9,15,3,4,12,7,2,16,5,6,8\}$

$P_{13} = \{13,11,4,16,9,15,2,8,3,1,7,6,10,5,14,12\}$ $P_{14} = \{15,4,13,1,9,10,11,16,8,2,5,12,14,6,3,7\}$

$P_{15} = \{6,16,3,2,14,9,7,8,4,15,5,1,12,10,13,11\}$ $P_{16} = \{9,10,6,16,3,1,2,11,12,13,7,8,14,4,15,5\}$

The permutation vectors ($P_1, P_2, \cdots, P_{16}$) are generated according to Example 1.

Example 1. *Pseudo-code for generating the permutation vectors*

- *Start*
- *Let a portion of the original generated chaotic sequence be:*

$$X = 0.145, 0.134, 0.279, 0.347, 0.134, 0.154$$

- *Now, to generate the desired values, follow the following algorithm:*
  1. *Amplify the values stored in X by multiplying with any large integer number; let us say: 989. The resultant values will be:*

$$A = 143.405, 132.526, 275.931, 343.183, 132.526, 152.306$$

  2. *Truncate the integers placed after the decimal points using the floor function (the floor is a MATLAB built-in function) and store the result in Z:*

$$Z = 143, 132, 275, 343, 132, 152$$

  3. *Take the module of Z as follows:*

$$M = mod\ (Z, 16) + 1.$$

  4. *This gives M = 16, 5, 4, 8, 5, 9.*

The generated vectors ($P_1, P_2, P_3, \cdot, P_{16}$) are used to scramble the values of S_1 that are located in each row and column of the S-box. A permutation is a process used in the

proposed work for rearranging the existing integer values in the S-box. In this process, each value is scrambled according to the integer value given in the permutation groups, as mentioned above. The entire permutation process is explained below:

1. **Read the existing S-box values:** The first step is to read the existing S-box values into a digital format that can be manipulated.
2. **Create a permutation key:** The next step is to create a permutation key, which will be used to determine the new positions of the pixels. In this case, sixteen permutation groups ($P_1, P_2, P_3, \cdots, P_{16}$) are used as permutation keys.
3. **Map the original values to new positions:** Using the permutation groups, each value is mapped to a new position. This process involves swapping the original values with new values based on the permutation groups
4. **Store the updated permuted S-box:** After all the values have been rearranged, the permuted S-box is stored in a digital format.

In order to provide a clear explanation of the permutation process, the first row of S_1 is taken as a sample.

The first row of S_1 consists of the values [91 109 125 165 25 250 161 242 149 198 162 207 241 213 201]. To permute these values, a permutation group or vector (P_1) is used. The first element of P_1 is 11, indicating that the value located at position (1, 11) in S_1 will be moved to the first position (1, 1). Correspondingly, the second element of P_1 is 4, signifying that the value situated at position (1, 4) in S_1 will be transferred to the second position (1, 2), and so on. The final value in P_1 is 10, meaning that the value located at position (1, 10) in S_1 will be shifted to position (1, 16). Once the permutation process is performed on the first row of S_1 utilizing P_1, the resulting updated row is [198 165 201 162 109 250 91 125 207 161 25 190 242 213 241 149]. Similarly, for each row that needs to be permuted, a distinct permutation group will be chosen, as shown below:

$$\begin{gathered}\text{For Row 1: } P_1 \text{ is chosen.}\\ \text{For Row 2: } P_2 \text{ is chosen.}\\ \text{For Row 3: } P_3 \text{ is chosen.}\\ \vdots\\ \text{For Row 16: } P_{16} \text{ is chosen.}\end{gathered}$$

This process further enhances the security and complexity of the final proposed S-box (S_F). The updated S-box (S_F) that is created after the employment of a permutation group on S_1 is given in Table 3. The entire process of the proposed method is illustrated in Figure 10.

Table 3. Updated S-box (S_F).

129	190	6	246	216	3	154	226	53	233	132	123	28	107	109	182
91	18	219	171	124	248	12	130	85	1	92	32	247	117	47	7
82	119	131	215	101	145	225	187	71	67	194	69	251	116	42	201
221	106	94	122	24	78	205	158	58	4	161	99	79	108	254	184
169	157	189	103	245	232	250	255	27	128	125	16	180	65	60	83
222	142	159	146	178	217	48	202	59	97	186	214	242	45	88	111
135	50	252	37	35	126	156	61	64	183	0	199	38	210	206	209
89	29	195	229	137	100	8	19	11	239	163	36	49	31	56	90
203	95	9	150	51	175	244	227	136	120	55	139	196	63	14	43
164	127	138	72	44	153	179	152	204	105	241	151	243	15	121	81
236	87	86	147	253	17	134	70	235	144	174	84	41	54	104	173
165	208	211	34	198	200	74	26	162	20	39	177	10	93	191	188
52	166	167	5	76	2	62	172	57	115	13	140	224	193	212	113
40	231	228	77	238	68	168	25	30	110	21	46	230	22	181	112
160	155	149	234	148	218	143	176	237	133	213	118	207	114	223	240
249	66	73	170	33	98	80	141	197	220	75	102	23	96	192	185

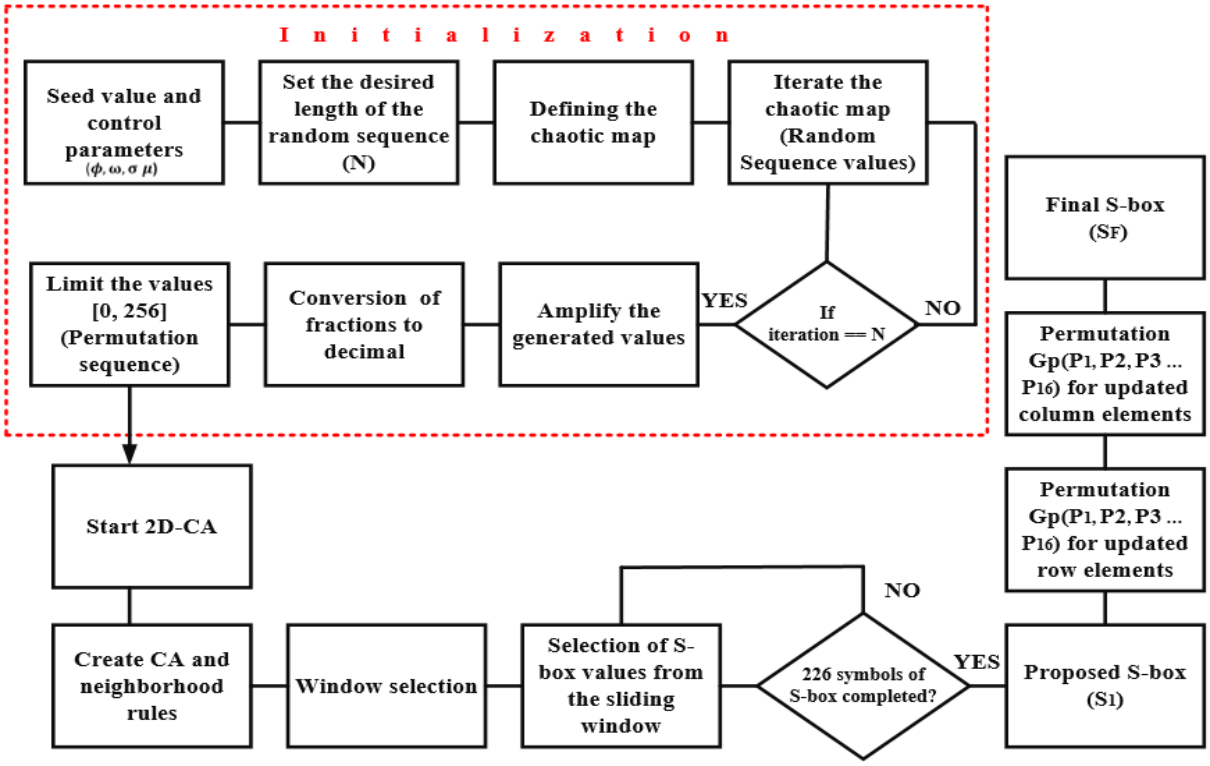


Figure 10. Steps to propose a new S-box.

4.8. Application of S-Box

To evaluate how well the S-box performs, it was tested on common plaintext images such as Lena, a camera, and a baboon, each of size 256 × 256. Figure 11 presents the histograms for the plaintext images and their corresponding pixel distributions. The histograms indicate that the distribution of pixels is not uniform, which suggests a significant level of correlation between the pixels.

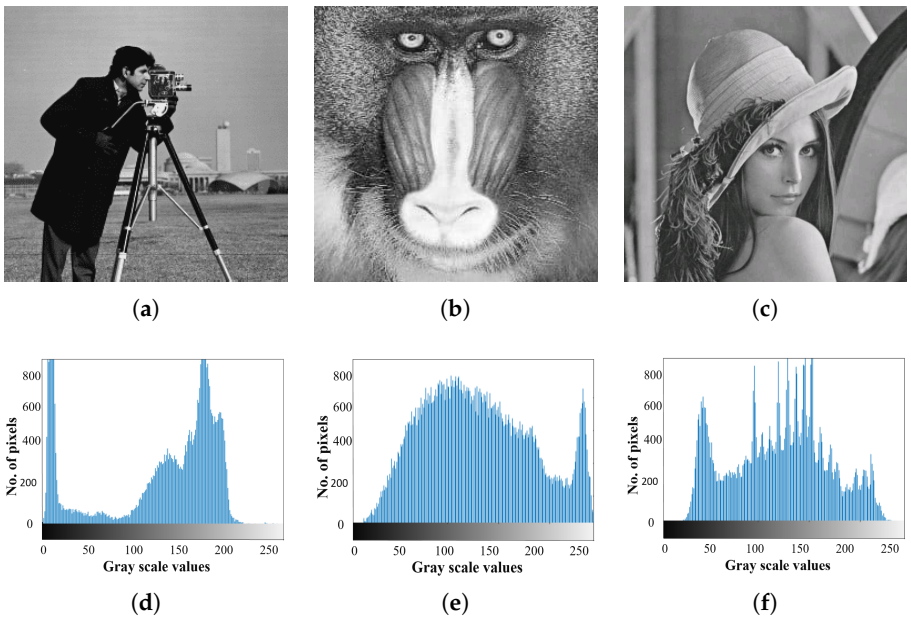


Figure 11. (**a**–**c**) Plaintext images; (**d**–**f**) histogram of the plaintext images.

The resulting ciphertext images after applying the proposed S-box s_F are shown in Figure 12. The ciphertext images appear to be completely encrypted, and the original

information cannot be seen. Additionally, the histograms of the ciphertext images show an even distribution of pixel values, indicating that there is very little correlation between the pixels.

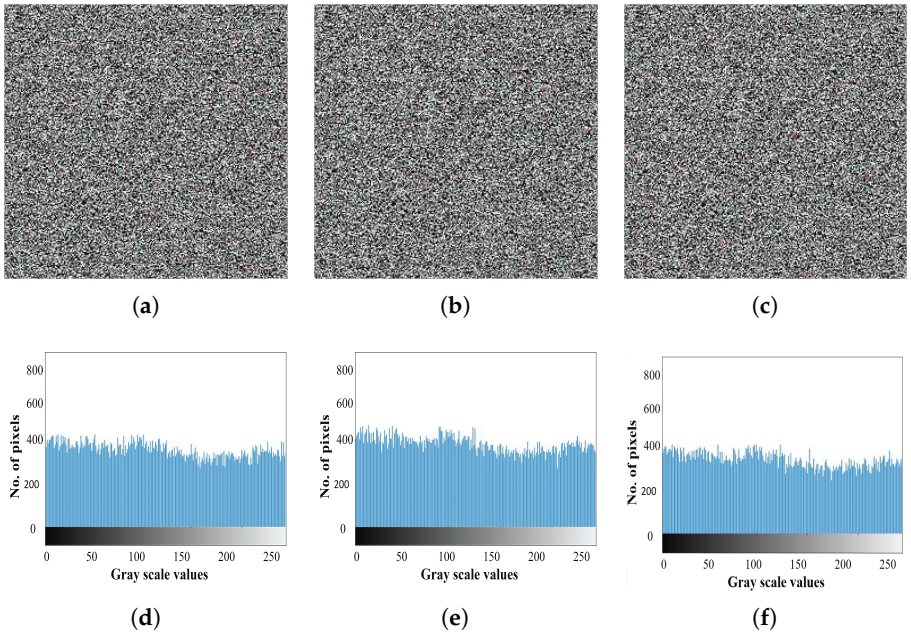


Figure 12. (**a**–**c**) Ciphertext images; (**d**–**f**) histogram of the ciphertext images.

Moreover, a statistical security analysis was also carried out for the ciphertext images that are generated by applying the proposed S-box. Statistical security parameters include entropy, correlation, contrast, energy, and homogeneity. The details of the mathematical calculation of such parameters are given in [65–68]. Table 4 displays the statistical values of the security parameters for plaintext and ciphertext images generated using S_F and the existing S-boxes. Based on the comparison analysis, it is evident that S_F outperforms the existing ones in terms of encrypting the plaintext image with higher security.

Table 4. A comparison and security analysis of ciphertext images generated using S_F and existing S-boxes.

Images and References	Correlation	Entropy	Contrast	Energy	Homogeneity
Plaintext image (Lena)	0.5778	7.6345	6.6347	0.1578	0.8764
Ciphertext image	0.0014	7.9986	9.8567	0.0159	0.1334
Zahid et al. [69]	0.0465	7.8986	9.4678	0.0164	0.3798
Ahmad et al. [47]	0.0223	7.9764	9.3647	0.0159	0.3472
Malik et al. [70]	0.01345	7.9697	9.1678	0.0161	0.4671
Shakiba et al. [71]	0.0149	7.9463	9.5374	0.0160	0.5316
Alhadawi et al. [72]	0.0155	7.9726	9.3360	0.0163	0.4445

5. Statistical Analysis of the Proposed S-Box

The proposed scheme for generating a hybrid S-box was carried out utilizing MATLAB 2022a, and the computer specifications used are as follows:

Read-Only Memory (RAM): 8.00 GB;
Solid-State Drive (SSD): 512 GB;
Windows: 11;
Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40 GHz 2.42 GHz.

This section reports the evaluation of the performance of the proposed S-boxes using well-known standard security parameters. Additionally, the obtained results are compared with several existing S-box methods that have been recently investigated in the literature.

5.1. Nonlinearity

In block ciphers, an S-box is usually used to create a nonlinear transformation from plaintext to ciphertext, and the nonlinearity offered by the cipher is a critical factor in ensuring security [73]. To reduce the vulnerability to linear cryptanalysis, block ciphers are designed to have strong resistance and nonlinearity. The nonlinearity of an $M \times M$ S-Box is an important metric for measuring the strength of a block cipher. One way to measure the nonlinearity of an S-Box is by calculating its degree of nonlinearity, denoted as N_l. One way to accomplish this is by finding the N-variable affine function that has the smallest Hamming distance with the N-variable Boolean function of the S-Box among all possible N-variable affine functions. A higher value of (N_l) indicates that the corresponding S-box is more nonlinear and offers better resistance against attacks, making it more secure.

$$N_l(f) = min\Big[R \in F_2^N | f(R) \neq g(R)|\Big] \tag{7}$$

$$N_l(f) = 2^{N-1}\Big[1 - 2^{-N} \underset{\Omega \in GF(2^N)}{max} |WF_f(\Omega)|\Big] \tag{8}$$

To evaluate the nonlinearity of an 8-bit Boolean function, the Walsh spectrum is commonly utilized, as shown in [34].

$$WF_f(\Omega) = \sum_{R \in F_2^N} (-1)^{f \oplus a \cdot b} \tag{9}$$

It should be noted that $a \cdot b$ represents the bitwise dot product. The proposed S-box is found to have excellent nonlinearity performance statistics, with nonlinearities of 102, 106, 104, 106, 102, 102, 106, and 106. The N_{Ls} for S_F have minimum, maximum, and mean values of 102, 106, and 104.25, respectively, indicating high scores.

Keyspace Analysis

Keyspace analysis in the generation of an S-box involves the evaluation of the total number of possible S-boxes that can be generated given a particular key size. The key size determines the number of bits that can be used to generate the S-box. For example, a key size of 8 bits allows for 256 possible S-boxes, while a key size of 16 bits allows for 65,536 possible S-boxes.

To generate an S-box, a permutation of the numbers 0 to 255 (or 0 to 65,535 for a 16-bit key) is created based on the key value. The permutation is used to determine the mapping of input values to output values. The larger the key size, the more possible permutations there are, resulting in a larger keyspace and a higher level of security.

The proposed method employs two chaotic keys denoted as $\boldsymbol{\varphi}$ = 10.200000000000000 and $\boldsymbol{\varpi}$ = 15.480000000000000, each having a sensitivity of 10^{-15}. This indicates that each key has a keyspace of 10^{+15}. Thus, the total keyspace is $10^{15*2} = 10^{30}$, which is $\approx 2^{100}$. According to Alvarez's criteria of keyspace, to resist a brute-force attack, the keyspace should be $\gtrapprox 2^{100}$ [74]. The proposed method satisfies Alvarez's criteria for keyspace, which means that the proposed S-box is capable of withstanding brute-force attacks.

5.2. Differential Uniformity

Differential cryptanalysis is a method of analyzing the security of a cryptographic system by examining the maximum difference in output for a given range of input differences [75]. The differential uniformity (DU) value is a measure of the uniformity of this function for all input and output differences. A secure substitution box should have a uniform differential, which means that the maximum difference in the output should be

evenly distributed across all possible input differences. The mathematical formula used to calculate the DU value for an S-box is a key tool in assessing the strength of a cryptographic system.

$$DU = max\left[j \in B : T(j) \oplus T(j + \delta j) = \delta k\right]^{\delta j \neq 0, \delta k} \quad (10)$$

The terms δ_j and δ_k refer to the differences between input and output values. Table 5 reveals that the proposed S-box (S_F) has a maximum differential uniformity score of 5 that occurs only three times. This indicates that these S-boxes have demonstrated a strong ability to resist attacks using differential cryptanalysis.

Table 5. Differential uniformity for the proposed S-box.

3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	5	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	5	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	5	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

5.3. Strict Avalanche Test

Webster and Tavares established the strict avalanche criterion (SAC) as a crucial factor for evaluating the strength of S-boxes [73]. In order to meet the SAC requirement, changing a single input bit should result in a 50% change in the output vector. A half-avalanche is necessary to prevent any correlation between input and output mixes and to prevent the leakage of sensitive data. A SAC value close to 0.5 is considered desirable.

For the proposed S-box (S_F) shown in Table 6, the dependency matrix was calculated, and all the entries in the table were found to be quite close to 0.5. The average score of the dependency matrix is 0.4993, which indicates that the proposed S-box satisfies the strict avalanche criterion well and is very close to the ideal score of 0.5. Therefore, the proposed S-box exhibits a good avalanche effect and satisfies the requirements for strong S-boxes.

Table 6. Strict avalanche test for the proposed S-box.

0.5625	0.5312	0.5625	0.4844	0.5469	0.5469	0.5312	0.5000
0.5000	0.5000	0.3906	0.3906	0.5312	0.4375	0.4844	0.5469
0.4375	0.4062	0.4688	0.5156	0.5000	0.5156	0.5781	0.5156
0.4844	0.4844	0.5469	0.4219	0.4375	0.4375	0.5312	0.5469
0.5469	0.4688	0.5312	0.4844	0.5156	0.4375	0.5156	0.4219
0.5469	0.5469	0.5000	0.4688	0.5625	0.5000	0.5469	0.5469
0.5312	0.5000	0.5469	0.5312	0.4375	0.4844	0.5312	0.4844
0.4844	0.4531	0.4531	0.4844	0.5000	0.5000	0.5000	0.5156

5.4. Linear Approximation Probability

Linear cryptanalysis was first introduced by Mitsuru in 1994 [76], and it has become an important field of study in cryptology, alongside differential cryptanalysis. While differ-

ential cryptanalysis focuses on the analysis of individual bit positions, linear cryptanalysis is a method that deals with multiple bit positions. For an S-Box F with a size of $N \times N$, the linear cryptanalysis problem is typically defined as follows:

$$L_{\{LPA\}} = \underset{\Theta y, \Theta z \neq 0}{max} L_{\{LPA\}}(\Theta y, \Theta z) \tag{11}$$

$L_{\{LPA\}}(\Theta y, \Theta z)$ can be calculated as follows:

$$L_{\{LPA\}}(\Theta y, \Theta z) = \frac{\left[y \in Y | f(y \oplus \Theta y) = f(y) \oplus \Theta z|\right] - 2^{N-1}}{2^N} \tag{12}$$

The variables Θy and Θz refer to the parity of input and output bits in their respective masks. Like in DP, $L_{\{LPA\}}(\Theta y, \Theta z)$ follows the same rule:

$$\sum_{\Theta y, \Theta z \in F_2^N} LP_F(\Theta y, \Theta z) = 1 \tag{13}$$

The effectiveness of an S-box in resisting linear cryptanalysis attacks is determined by its LAP value. Reducing the LPF (linear probability function) is an effective way to increase the complexity of a linear cryptanalysis attack [40]. By decreasing the LPF, it becomes more difficult for an attacker to identify linear relations between input and output bits, making the system more resistant to linear cryptanalysis.

In the case of the proposed S-box (S_F), the LAP score is 0.0601. This demonstrates that the S-box is much more capable of resisting linear cryptanalysis than the S-box prior to the group action.

5.5. Bit Independence Criterion

The bit independence criterion (BIC) is an equally important factor in ensuring the strength of S-boxes. In order to test for the BIC, a method was proposed by Adams and Tavares [76]. Essentially, to satisfy the BIC, the Boolean functions XOR(B_{i_1}, B_{j_1}) (where i_1 and j_1 are distinct integers between 0 and 7) should be highly nonlinear and demonstrate a strong avalanche effect. To confirm that a given 8×8 S-box meets the BIC, one must calculate the SAC and nonlinearity of each of the 56 possible XOR(B_{i_1}, B_{j_1}) Boolean functions. The researchers evaluated the potential nonlinearity values of all 56 XOR(B_{i_1}, B_{j_1}) functions for the intended S-box, and their findings are summarized in Table 7. They found that the average nonlinearity score for the BIC is 111.12. This indicates that the proposed S-box performs exceptionally well in meeting the bit independence criterion, as verified by its high BIC score.

Table 7. Bit independence criterion for the proposed S-box.

-	111	112	111	111	111	111	112
111	-	111	111	112	112	110	111
112	112	-	110	111	112	112	111
111	111	111	-	112	111	110	111
112	112	111	111	-	110	111	110
111	111	111	111	112	-	112	111
110	111	112	112	111	111	-	110
112	111	111	112	110	110	110	-

5.6. Auto-Correlation Function

If we have Boolean mappings $D(r)$ and $E(r)$, their correlation can be represented by the symbol $C_{DE}(y)$, mathematically defined as:

$$C_{DE}(y) = \sum_{r} \frac{1}{2^N} \left[(-1)^{D(r) \oplus E(r \oplus y)} \right] \tag{14}$$

However, $C_{DD}(y)$ and $C_{EE}(y)$ represent the auto-correlation functions (ACFs) for mapping $D(r)$ and $E(r)$, respectively. Mathematically, they can be calculated using Equations (15) and (16), respectively.

$$C_{ACF} = W_D(y) = \sum \frac{1}{2^N} \left[(-1)^{D(r) \oplus D(r \oplus y)} \right] \tag{15}$$

$$C_{ACF} = W_E(y) = \sum \frac{1}{2^N} \left[(-1)^{E(r) \oplus E(r \oplus y)} \right] \tag{16}$$

In order for an S-box to exhibit a strong diffusion effect, it is desirable to have a lower score for its ACF.

5.7. Comparison Analysis and Discussion

In the last several decades, researchers have been exploring various methods to create S-boxes that are resilient against cryptographic attacks [77]. To evaluate the effectiveness of any new S-box design, it is important to compare its security features against existing S-boxes [35,47,69–72,78,79]. In Table 8, the values of several statistical analyses for existing schemes are displayed. These schemes utilize a variety of design concepts, including chaos, the elliptic curve, and the algebraic method. By comparing the suggested S-box design with the methods shown in Table 8, researchers can determine how it performs in relation to these state-of-the-art S-box designs. This comparison can help to determine whether the suggested S-box is a significant improvement over existing designs and whether it is a suitable candidate for use in cryptographic applications.

In the design of S-boxes, nonlinearity is typically the primary focus among all performance parameters considered [80]. Other performance parameters must also be taken into account to ensure that the S-box is robust against attacks other than linear ones. Therefore, it is important to generate S-box designs that score well on all relevant parameters, in addition to nonlinearity, to enhance the overall security of the S-box. In Table 8, it is revealed that the suggested S-box has excellent nonlinearity compared to almost all of the S-box designs listed. The proposed S-box (S_F) has a minimum nonlinearity score of 102, with an average score of 104.25. In block cryptosystems, the suggested S-box can transform plaintext to ciphertext in a highly nonlinear way during the substitution phase. This means that the output ciphertext is significantly different from the input plaintext and cannot be easily predicted or reversed without knowledge of the encryption key. Along with its high nonlinearity feature, the suggested S-box design also demonstrates good performance in terms of XOR distribution analysis, with a decent DU score of 5. This DU score is the lowest among those presented in Table 8, which suggests that the proposed S-box offers better resistance to differential cryptanalysis compared to other recently investigated S-box designs.

The strict avalanche criterion (SAC) value of the proposed S_F is 0.4993, which is comparable to the SAC values of other S-box designs listed in Table 8. This indicates that the suggested S-box meets the SAC criterion and is suitable for use in cryptographic applications. Overall, the performance of the suggested S-box design appears to be quite robust and secure based on its high nonlinearity, low DU score, and satisfactory SAC value.

The BIC analysis of the proposed S_F shows that the BIC-NL (bit independence criterion—nonlinearity) value is 111.12, which is higher than existing S-box designs. To resist Mitsui's linear cryptanalysis method, it is crucial for the S-box to exhibit a lower LAP. The suggested S-box design has a LAP score of 0.0603, which is the lowest among all other

S-box designs listed in Table 8. This indicates that the suggested S-box has an improved ability to withstand linear attacks.

Another important performance parameter used to assess the strength of the S-box is the auto-correlation function (ACF). The suggested S-box design is found to have an excellent ACF score of 28, which is the best among all other S-box designs listed in Table 8. In summary, the suggested S-box design demonstrates high scores in various performance parameters, such as nonlinearity and BIC, and low scores for DU, LAP, and ACF, indicating its robustness and suitability for use in cryptographic applications.

Table 8. Comparison analysis of the proposed S-box with existing ones.

S-Box	$\mathbf{Min}_{N_l}$	$\mathbf{Max}_{N_l}$	$\mathbf{Avg}_{N_l}$	DU	SAC	LAP	BIC	ACF
Proposed (S_1)	92	106	101.65	6	0.5010	0.0903	110.65	30
Proposed (S_F)	102	106	104.25	5	0.4993	0.0601	111.12	28
Quiroga et al. [35]	96	112	102.25	12	0.5059	-	103.42	-
Ahmad et al. [47]	111	112	111.5	-	0.4978	0.125	103.86	
Zahid et al. [69]	110	112	111.3	10	0.5030	0.1258	103.8	-
Malik et al. [70]	-	-	112	-	0.5009	0.0625	112	-
Shakiba et al. [71]	100	104.5	108	-	0.5017	0.1328	104.29	-
Alhadawi et al. [72]	106	110	108.5	10	0.4995	0.1094	103.85	-
Lambi et al. [78]	106	-	-	10	-	0.0705	104.07	-
Özkaynak et al. [79]	106	108	106.7	-	0.4063	-	103.2	-

6. Conclusions

This paper discusses the importance of a strong S-box, which is a mathematical function used in encryption. A strong S-box needs to have excellent performance in multiple areas, but existing methods often focus only on one area, such as nonlinearity, while neglecting other important parameters. This paper proposes a new hybrid method for creating an S-box that uses chaos, 2D cellular automata, and algebra. The proposed method first creates an S-box using chaos and 2D cellular automata, which is then utilized to secure the digital image to show that the proposed S-box is capable of properly encrypting the image pixels. The goal is to create an S-box that has excellent scores in most performance parameters, such as an average N_l of 104.25, a SAC of 0.4993, a DU of 5, a BIC of 111.12, a LAP of 0.0603, and an ACF of 28. The comparison analysis shows that the S-box created using the proposed method is more secure and robust than many recent S-boxes, making it a suitable candidate for use in secure block cipher implementations.

Author Contributions: Conceptualization, methodology, and writing—original draft preparation, A.S.; data curation, K.H.K.; validation, editing, and visualization, M.M.H.; formal analysis and project administration, I.B.; investigation and funding acquisition, Z.B.; resources, supervision, writing—review, and software, M.U.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Deanship of Scientific Research at King Khalid University under grant number R. G. P. 2/109/43.

Data Availability Statement: The data utilized in this research are confidential and can be made available upon reasonable request.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through a research group program under grant number R. G. P. 2/109/43.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, T.; Bhuiyan, M.Z.A.; Wang, G.; Qi, L.; Wu, J.; Hayajneh, T. Preserving balance between 653 privacy and data integrity in edge-assisted Internet of Things. *IEEE Internet Things J.* **2019**, *7*, 2679–2689. [CrossRef]
2. Shannon, C.E. *Claude Elwood Shannon: Collected Papers*; IEEE Press: New York, NY, USA, 1993.
3. Shafique, A.; Shahid, J. Novel image encryption cryptosystem based on binary bit planes extraction and multiple chaotic maps. *Eur. Phys. J. Plus* **2018**, *133*, 331. [CrossRef]
4. Anees, A.; Siddiqui, A.M.; Ahmed, F. Chaotic substitution for highly autocorrelated data in encryption algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2014**, *19*, 3106–3118. [CrossRef]
5. Fan, T.; Li, L.; Wei, Y.; Pasalic, E. Differential cryptanalysis of full-round ANU-II ultra-lightweight block cipher. *Int. J. Distrib. Sens. Netw.* **2022**, *663*, 15501329221119398. [CrossRef]
6. Chan, Y.Y.; Khor, C.Y.; Teh, J.S.; Teng, W.J.; Jamil, N. Differential Cryptanalysis of Lightweight Block Ciphers SLIM and LCB. In Proceedings of the Emerging Information Security and Applications: Third International Conference, EISA 2022, Wuhan, China, 29–30 October 2022; pp. 55–67.
7. Dwivedi, A.D.; Dhar, S.; Srivastava, G.; Singh, R. Cryptanalysis of round-reduced fantomas, robin and iSCREAM. *Cryptography* **2019**, *3*, 4. [CrossRef]
8. Biham, E.; Dunkelman, O.; Keller, N. Enhancing differential-linear cryptanalysis. In Proceedings of the Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, 1–5 December 2002; pp. 254–266.
9. Gao, S.; Wu, R.; Wang, X.; Liu, J.; Li, Q.; Wang, C.; Tang, X. Asynchronous updating Boolean network encryption algorithm. *IEEE Trans. Circuits Syst. Video Technol.* **2023**. [CrossRef]
10. Gao, S.; Wu, R.; Wang, X.; Wang, J.; Li, Q.; Wang, C.; Tang, X. A 3D model encryption scheme based on a cascaded chaotic system. *Signal Process.* **2023**, *202*, 108745. [CrossRef]
11. Gupta, R.; Tanwar, S.; Tyagi, S.; Kumar, N. Machine learning models for secure data analytics: A taxonomy and threat model. *Comput. Commun.* **2020**, *153*, 406–440. [CrossRef]
12. Shafique, A.; Ahmed, F. Image encryption using dynamic S-box substitution in the wavelet domain. *Wirel. Pers. Commun.* **2020**, *115*, 2243–2268. [CrossRef]
13. Anees, A.; Siddiqui, A.M.; Ahmed, J.; Hussain, I. A technique for digital steganography using chaotic maps. *Nonlinear Dyn.* **2014**, *75*, 807–816. [CrossRef]
14. Shafique, A.; Ahmed, J.; Boulila, W.; Ghandorh, H.; Ahmad, J.; Rehman, M.U. Detecting the security level of various cryptosystems using machine learning models. *IEEE Access* **2020**, *9*, 9383–9393. [CrossRef]
15. Hussain, I.; Anees, A.; Al-Maadeed, T.A. A novel encryption algorithm using multiple semifield S-boxes based on permutation of symmetric group. *Comput. Appl. Math.* **2023**, *42*, 80. [CrossRef]
16. Agarwal, P.; Singh, A.; Kilicman, A. Development of key-dependent dynamic S-boxes with dynamic irreducible polynomial and affine constant. *Adv. Mech. Eng.* **2018**, *10*, 1687814018781638. [CrossRef]
17. Anees, A.; Ahmed, Z. A technique for designing substitution box based on van der pol oscillator. *Wirel. Pers. Commun.* **2015**, *82*, 1497–1503. [CrossRef]
18. Shafique, A. A new algorithm for the construction of substitution box by using chaotic map. *Eur. Phys. J. Plus* **2020**, *135*, 194. [CrossRef]
19. Sanchez-Avila, C.; Sanchez-Reillol, R. The Rijndael block cipher (AES proposal): A comparison with DES. In Proceedings of the IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No. 01CH37186), London, UK, 16–19 October 2001; pp. 229–234.
20. Razaq, A.; Alolaiyan, H.; Ahmad, M.; Yousaf, M.A.; Shuaib, U.; Aslam, W.; Alawida, M. A novel method for generation of strong substitution-boxes based on coset graphs and symmetric groups. *IEEE Access* **2020**, *8*, 75473–75490. [CrossRef]
21. Siddiqui, N.; Yousaf, F.; Murtaza, F.; Ehatisham-ul Haq, M.; Ashraf, M.U.; Alghamdi, A.M.; Alfakeeh, A.S. A highly nonlinear substitution-box (S-box) design using action of modular group on a projective line over a finite field. *PLoS ONE* **2020**, *15*, e0241890. [CrossRef]
22. Ahmad, M.; Al-Solami, E. Evolving dynamic S-boxes using fractional-order hopfield neural network based scheme. *Entropy* **2020**, *22*, 717. [CrossRef]
23. Zahid, A.H.; Arshad, M.J.; Ahmad, M. A novel construction of efficient substitution-boxes using cubic fractional transformation. *Entropy* **2019**, *21*, 245. [CrossRef]
24. Rehman, M.U.; Shafique, A.; Khalid, S.; Hussain, I. Dynamic substitution and confusion- diffusion-based noise-resistive image encryption using multiple chaotic maps. *IEEE Access* **2021**, *9*, 52277–52291. [CrossRef]
25. Hussain, I.; Anees, A.; Al-Maadeed, T.A.; Mustafa, M.T. Construction of s-box based on chaotic map and algebraic structures. *Symmetry* **2019**, *11*, 351. [CrossRef]
26. Özkaynak, F.; Özer, A.B. A method for designing strong S-Boxes based on chaotic Lorenz system. *Phys. Lett. A* **2010**, *374*, 3733–3738. [CrossRef]
27. Anees, A.; Hussain, I.; Algarni, A.; Aslam, M. A robust watermarking scheme for online multimedia copyright protection using new chaotic map. *Secur. Commun. Netw.* **2018**, *2018*, 1840207. [CrossRef]

28. Wang, Y.; Wong, K.W.; Li, C.; Li, Y. A novel method to design S-box based on chaotic map and genetic algorithm. *Phys. Lett. A* **2012**, *376*, 827–833. [CrossRef]
29. Shafique, A.; Ahmed, J.; Rehman, M.U.; Hazzazi, M.M. Noise-resistant image encryption scheme for medical images in the chaos and wavelet domain. *IEEE Access* **2021**, *9*, 59108–59130. [CrossRef]
30. Gao, S.; Wu, R.; Wang, X.; Liu, J.; Li, Q.; Tang, X. EFR-CSTP: Encryption for face recognition based on the chaos and semi-tensor product theory. *Inf. Sci.* **2023**, *621*, 766–781. [CrossRef]
31. Wu, R.; Gao, S.; Wang, X.; Liu, S.; Li, Q.; Erkan, U.; Tang, X. AEA-NCS: An audio encryption algorithm based on a nested chaotic system. *Chaos Solitons Fractals* **2022**, *165*, 112770. [CrossRef]
32. Yin, R.; Yuan, J.; Wang, J.; Shan, X.; Wang, X. Designing key-dependent chaotic S-box with larger key space. *Chaos Solitons Fractals* **2009**, *42*, 2582–2589. 722 [CrossRef]
33. Lambić, D. S-box design method based on improved one-dimensional discrete chaotic map. *J. Inf. Telecommun.* **2018**, *2*, 181–191. [CrossRef]
34. Özkaynak, F.; Çelik, V.; Özer, A.B. A new S-box construction method based on the fractional-order chaotic Chen system. *Signal Image Video Process.* **2017**, *11*, 659–664. [CrossRef]
35. Cassal-Quiroga, B.B.; Campos-Cantón, E. Generation of dynamical S-boxes for block ciphers via extended logistic map. *Math. Probl. Eng.* **2020**, *2020*, 2702653. [CrossRef]
36. Shafique, A.; Hazzazi, M.M.; Alharbi, A.R.; Hussain, I. Integration of spatial and frequency domain encryption for digital images. *IEEE Access* **2021**, *9*, 149943–149954. 730 [CrossRef]
37. Anees, A. An image encryption scheme based on lorenz system for low profile applications. *3D Res.* **2015**, *6*, 1–10. [CrossRef]
38. Tanyildizi, E.; Özkaynak, F. A new chaotic S-box generation method using parameter optimization of one dimensional chaotic maps. *IEEE Access* **2019**, *7*, 117829–117838. [CrossRef]
39. Shafique, A.; Mehmood, A.; Elhadef, M.; Khan, K.H. A lightweight noise-tolerant encryption scheme for secure communication: An unmanned aerial vehicle application. *PLoS ONE* **2022**, *17*, e0273661. [CrossRef] [PubMed]
40. Abd El-Latif, A.A.; Abd-El-Atty, B.; Mazurczyk, W.; Fung, C.; Venegas-Andraca, S.E. Secure data encryption based on quantum walks for 5G Internet of Things scenario. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 118–131. [CrossRef]
41. Anees, A.; Chen, Y.P.P. Designing secure substitution boxes based on permutation of symmetric group. *Neural Comput. Appl.* **2020**, *32*, 7045–7056. [CrossRef]
42. Shafique, A.; Ahmed, J. A Color Image Encryption Algorithm Based on Chaotic Map and Discrete Wavelet Transform. In Proceedings of the 2022 Global Conference on Wireless and Optical Technologies (GCWOT), Malaga, Spain, 14–17 February 2022; pp. 1–5.
43. Wolfram, S. Computation theory of cellular automata. *Commun. Math. Phys.* **1984**, *96*, 15–57. [CrossRef]
44. Vahedi, V.; Jafarpour, M.; Aghabozorgi, H.; Cristea, I. Extension of elliptic curves on Krasner hyperfields. *Commun. Algebra* **2019**, *47*, 4806–4823. [CrossRef]
45. Khompysh, A.; Kapalova, N.; Algazy, K.; Dyusenbayev, D.; Sakan, K. Design of substitution nodes (S-Boxes) of a block cipher intended for preliminary encryption of confidential information. *Cogent Eng.* **2022**, *9*, 2080623. [CrossRef]
46. Hussain, I.; Anees, A.; Alkhaldi, A.H.; Aslam, M.; Siddiqui, N.; Ahmed, R. Image encryption based on Chebyshev chaotic map and S8 S-boxes. *Opt. Appl.* **2019**, *49*, 317–330.
47. Ahmad, M.; Al-Solami, E.; Alghamdi, A.M.; Yousaf, M.A. Bijective S-boxes method using improved chaotic map-based heuristic search and algebraic group structures. *IEEE Access* **2020**, *8*, 110397–110411. [CrossRef]
48. Basha, H.A.M.A.; Mohra, A.S.S.; Diab, T.O.M.; El Sobky, W.I. Efficient image encryption based on new substitution box using DNA coding and bent function. *IEEE Access* **2022**, *10*, 66409–66429. [CrossRef]
49. Farhan, A.K.; Ali, R.S.; Yassein, H.R.; Al-Saidi, N.M.G.; Abdul-Majeed, G.H. A new approach to generate multi S-boxes based on RNA computing. *Int. J. Innov. Comput. Inf. Control* **2020**, *16*, 331–348.
50. Ahmed, F.; Anees, A. Hash-based authentication of digital images in noisy channels. In *Robust Image Authentication in the Presence of Noise*; Springer: Cham, Switzerland, 2015; pp. 1–42.
51. Azam, N.A.; Hayat, U.; Ullah, I. Efficient construction of a substitution box based on a Mordell elliptic curve over a finite field. *Front. Inf. Technol. Electron. Eng.* **2019**, *20*, 1378–1389. [CrossRef]
52. Ullah, I.; Azam, N.A.; Hayat, U. Efficient and secure substitution box and random number generators over Mordell elliptic curves. *J. Inf. Secur. Appl.* **2021**, *56*, 102619. [CrossRef]
53. Mahlake, N.; Mathonsi, T.E.; Du Plessis, D.; Muchenje, T. A Lightweight Encryption Algorithm to Enhance Wireless Sensor Network Security on the Internet of Things. *J. Commun.* **2023**, *18*. [CrossRef]
54. Hayat, U.; Azam, N.A. A novel image encryption scheme based on an elliptic curve. *Signal Process.* **2019**, *155*, 391–402. [CrossRef]
55. Toughi, S.; Fathi, M.H.; Sekhavat, Y.A. An image encryption scheme based on elliptic curve pseudo random and advanced encryption system. *Signal Process.* **2017**, *141*, 217–227. [CrossRef]
56. Ullah, I.; Hayat, U.; Bustamante, M.D. Image encryption using elliptic curves and Rossby/drift wave triads. *Entropy* **2020**, *22*, 454. [CrossRef]
57. Anees, A.; Gondal, M.A. Construction of nonlinear component for block cipher based on one-dimensional chaotic map. *3D Res.* **2015**, *6*, 1–5. [CrossRef]

58. Hussain, I.; Ahmed, F.; Khokhar, U.M.; Anees, A. Applied cryptography and noise resistant data security. *Secur. Commun. Netw.* **2018**, *2018*, 962821. [CrossRef]
59. Shafique, A.; Mehmood, A.; Alawida, M.; Khan, A.N.; Khan, A.U.R. A novel machine learning technique for selecting suitable image encryption algorithms for IoT applications. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 5108331. [CrossRef]
60. Khan, M.A.M.; Azam, N.A.; Hayat, U.; Kamarulhaili, H. A novel deterministic substitution box generator over elliptic curves for real-time applications. *J. King Saud-Univ.-Comput. Inf. Sci.* **2023**, *35*, 219–236. [CrossRef]
61. Wolfram, S. *A New Kind of Science*; Wolfram Media Champaign: Champaign, IL, USA, 2002; Volume 5.
62. Tomassini, M.; Perrenoud, M. Cryptography with cellular automata. *Appl. Soft Comput.* **2001**, *1*, 151–160. [CrossRef]
63. Luo, Y.; Lin, J.; Liu, J.; Wei, D.; Cao, L.; Zhou, R.; Cao, Y.; Ding, X. A robust image encryption algorithm based on Chua's circuit and compressive sensing. *Signal Process.* **2019**, *161*, 227–247. [CrossRef]
64. Fadhil, M.S.; Farhan, A.K.; Fadhil, M.N.; Al-Saidi, N.M. A new lightweight AES using a combination of chaotic systems. In Proceedings of the 2020 1st Information Technology To 796 Enhance E-Learning and Other Application (IT-ELA), Baghdad, Iraq, 12–13 July 2020; pp. 82–88.
65. Zheng, J.; Zeng, Q. An image encryption algorithm using a dynamic S-box and chaotic maps. *Appl. Intell.* **2022**, *52*, 15703–15717. [CrossRef]
66. Sha, Y.; Sun, B.; Cheng, X.; Mou, J.; Wang, L. Cross-plane colour image encryption scheme based on BST model and chaotic map. *Eur. Phys. J. Spec. Top.* **2022**, *231*, 3249–3263. [CrossRef]
67. Rani, N.; Mishra, V.; Sharma, S.R. Image encryption model based on novel magic square with differential encoding and chaotic map. *Nonlinear Dyn.* **2022**, *111*, 2869–2893. [CrossRef]
68. Anees, A.; Hussain, I.; Khokhar, U.M.; Ahmed, F.; Shaukat, S. Machine learning and appliedcryptography. *Secur. Commun. Netw.* **2022**, *2022*, 9797604. [CrossRef]
69. Zahid, A.H.; Ahmad, M.; Alkhayyat, A.; Arshad, M.J.; Shaban, M.M.U.; Soliman, N.F.; Algarni, A.D. Construction of optimized dynamic S-boxes based on a cubic modular transform and the sine function. *IEEE Access* **2021**, *9*, 131273–131285. [CrossRef]
70. Malik, M.S.M.; Ali, M.A.; Khan, M.A.; Ehatisham-Ul-Haq, M.; Shah, S.N.M.; Rehman, M.; Ahmad, W. Generation of highly nonlinear and dynamic AES substitution-boxes (S-boxes) using chaos-based rotational matrices. *IEEE Access* **2020**, *8*, 35682–35695. [CrossRef]
71. Shakiba, A. Generating dynamical S-boxes using 1D Chebyshev chaotic maps. *J. Comput. Secur.* **2020**, *7*, 1–17.
72. Alhadawi, H.S.; Majid, M.A.; Lambić, D.; Ahmad, M. A novel method of S-box design based on discrete chaotic maps and cuckoo search algorithm. *Multimed. Tools Appl.* **2021**, *80*, 7333–7350. [CrossRef]
73. Anees, A.; Khan, W.A.; Gondal, M.A.; Hussain, I. Application of mean of absolute deviation method for the selection of best nonlinear component based on video encryption. *Z. Nat. A* **2013**, *68*, 479–482. [CrossRef]
74. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
75. Gondal, M.A.; Anees, A. Analysis of optimized signal processing algorithms for smart antenna system. *Neural Comput. Appl.* **2013**, *23*, 1083–1087. [CrossRef]
76. Adams, C.; Tavares, S. The structured design of cryptographically good S-boxes. *J. Cryptol.* **1990**, *3*, 27–41. [CrossRef]
77. Rehman, M.U.; Shafique, A.; Khan, K.H.; Khalid, S.; Alotaibi, A.A.; Althobaiti, T.; Ramzan, N.; Ahmad, J.; Shah, S.A.; Abbasi, Q.H. Novel privacy preserving non-invasive sensing-based diagnoses of pneumonia disease leveraging deep network model. *Sensors* **2022**, *22*, 461. [CrossRef]
78. Lambić, D. A new discrete-space chaotic map based on the multiplication of integer numbers and its application in S-box design. *Nonlinear Dyn.* **2020**, *100*, 699–711. [CrossRef]
79. Özkaynak, F. Construction of robust substitution boxes based on chaotic systems. *Neural Comput. Appl.* **2019**, *31*, 3317–3326. [CrossRef]
80. Artuğer, F.; Özkaynak, F. SBOX-CGA: Substitution box generator based on chaos and genetic algorithm. *Neural Comput. Appl.* **2022**, *34*, 20203–20211. [CrossRef]