mathematics

MDPI

Article

Design of a Multi-Layer Symmetric Encryption System Using Reversible Cellular Automata

George Cosmin Stănică * and Petre Anghelescu *

Department of Electronics, Computers and Electrical Engineering, National University of Science and Technology POLITEHNICA Bucharest, Pitesti University Centre, 110040 Pitesti, Romania
* Correspondence: george.stanica@upb.ro (G.C.S.); petre.anghelescu@upb.ro (P.A.)

Abstract: The increasing demand for secure and efficient encryption algorithms has intensified the exploration of alternative cryptographic solutions, including biologically inspired systems like cellular automata. This study presents a symmetric block encryption design based on multiple reversible cellular automata (RCAs) that can assure both computational efficiency and reliable restoration of original data. The encryption key, with a length of 224 bits, is composed of specific rules used by the four distinct RCAs: three with radius-2 neighborhoods and one with a radius-3 neighborhood. By dividing plaintext into 128-bit blocks, the algorithm performs iterative transformations over multiple rounds. Each round includes forward or backward evolution steps, along with dynamically computed shift values and reversible transformations to securely encrypt or decrypt data. The encryption process concludes with an additional layer of security by encrypting the final RCA configurations, further protecting against potential attacks on the encrypted data. Additionally, the 224-bit key length provides robust resistance against brute force attacks. Testing and analysis were performed using a custom-developed software (version 1.0) application, which helped demonstrate the algorithm's robustness, encryption accuracy, and ability to maintain data integrity.

Keywords: multi-layer encryption; symmetric cryptography; block encryption; reversible cellular automata; dynamical systems; data security

MSC: 68Q80; 37B15; 94A60; 68P25

check for updates

Received: 25 December 2024
Revised: 15 January 2025
Accepted: 16 January 2025
Published: 18 January 2025

Citation: Stănică, G.C.; Anghelescu, P. Design of a Multi-Layer Symmetric Encryption System Using Reversible Cellular Automata. *Mathematics* **2025**, *13*, 304. https://doi.org/10.3390/math13020304

1. Introduction

In today's increasingly digitalized society, data security has become paramount as critical sectors—such as healthcare, energy, water supply, and infrastructure—are vulnerable to sophisticated cyberattacks. The expansion of connected systems like IoT, smart cities, and online medical databases has brought significant advancements, yet it has also created new opportunities for malicious actors. Attacks on such critical infrastructures can lead to severe disruptions, putting both personal and state security at risk. In recent years, cybersecurity incidents have targeted everything from patient records [1,2] to energy grids [3], highlighting the urgency of securing sensitive information in an era where data underpin both private and public sectors.

In the era of expanding digital communication, the security of sensitive data is paramount, necessitating the continual advancement of encryption algorithms capable of resisting emerging cyber threats. Traditional encryption methods like Advanced Encryption Standard (AES) [4] and Rivest–Shamir–Adleman (RSA) [5], used to secure digital data, are increasingly vulnerable to sophisticated attacks. In response, research has increasingly

explored alternative cryptographic foundations, including biologically inspired and computationally efficient models like cellular automata (CAs). Cellular automata, originally developed to model biological and physical systems, are uniquely suited to encryption due to their simplicity, parallel processing potential, and inherent reversibility in certain configurations. Notably, reversible cellular automata (RCAs) have garnered attention for their ability to facilitate both encryption and decryption processes through rule-based evolution, offering a promising direction for the design of symmetric encryption systems.

Recent advancements in cryptography have introduced innovative encryption methodologies leveraging chaos theory, neuron models, and complex dynamic systems. For instance, the study presented in [6] explores the use of Arnold's Cat Map to shuffle pixel positions and values, achieving high resistance to known-text attacks and robust encryption validated through randomness tests. Other works employ hyperchaotic maps combined with pixel reorganization to enhance encryption efficiency and security, demonstrating superior randomness and resistance to attacks in image encryption [7,8]. Moreover, the memristive tabu learning neuron model introduces multi-wing chaotic attractors for image encryption along with an implementation on FPGA hardware, showcasing scalability and hardware feasibility [9]. Additionally, the study presented in [10] combines tabu search optimization with hyperchaotic systems to enhance encryption performance, providing resistance to brute force and differential attacks while ensuring high randomness and security. Furthermore, the algorithm described in [11] utilizes quadratic polynomial hyperchaos and pixel fusion strategies to produce highly secure encryption systems validated through statistical testing and performance evaluation.

Recent studies have emphasized the potential of RCA-based cryptography. Different investigations have been conducted to prove the potential of integration of cellular automata concepts in designing encryption systems in various domains [12,13]. Research into data security through reversible cellular automata in cryptographic systems has been conducted in order to analyze how different structures influence the encryption outcome [14]. Other studies, presented in [15] and [16], focused on exploring their use as an alternative to traditional ciphers, demonstrating the ability of RCAs to generate complex patterns that can significantly strengthen cryptographic systems. Studies like [17] focused on highlighting RCAs' potential for high-speed, low-power encryption, particularly suited for applications in resource-constrained environments. Specifically, reversible cellular automata have gained attention for their potential to facilitate symmetric encryption by allowing precise reconstruction of original data while maintaining computational efficiency [18]. RCA-based cryptography is considered particularly promising for high-speed, low-power applications in secure communications, making it an adaptable choice for modern cryptographic needs.

The primary purpose of this study is to develop a symmetric block encryption algorithm based specifically on using multiple RCAs with different radius and complementary rules to ensure accurate data restoration. The 224-bit encryption key design allows for intricate data transformations across multiple encryption rounds while maintaining computational efficiency and facilitating reversible operations. An important aspect of this work is to demonstrate that the algorithm offers enhanced security, particularly through the final encryption of RCA configurations to shield against potential reverse-engineering attacks. The strength of the proposed algorithm is further amplified by the key length, which effectively safeguards against brute force attacks. By addressing these needs, this work contributes to the growing field of RCA-based cryptography, proposing a solution that is adaptable to emerging security challenges.

Our research introduces several key advancements in the field of encryption algorithms. These contributions, which highlight the unique aspects and strengths of our proposed system, are outlined below:

- Usage of a bioinspired technique in developing an innovative encryption system utilizing a multi-layer approach based on multiple reversible cellular automata arranged in a cascade architecture, which improves the security of generated ciphertext;
- Our proposed encryption algorithm design is solely based on cellular automata concepts for both the encryption/decryption process itself and also for producing the initialization data required for cryptographic purposes, which provide consistency and were based on analysis carried out in our previous works;
- Enhanced security through a vast keyspace provided by the 224-bit secret key, along with the use of 128-bit encryption which guarantees strong data segmentation and raises algorithm complexity;
- Comprehensive software simulation derived from the custom-developed application used to simulate and validate the encryption algorithm, ensuring practical applicability, along with optimization brought to implementation by operating with low-level functions (bit operations) which lead to reduced computational complexity, guaranteeing efficiency and practical applicability.

This research is divided into five sections, organized as follows. Section 2 presents the working principles of reversible cellular automata and the methods of integrating them into encryption schemes. Section 3 provides an in-depth description of the proposed algorithm, detailing the fundamentals of the multi-layer technique, which combines multiple RCAs with distinct roles across multiple rounds. This section also explains how the encryption key is generated and applied within the algorithm and presents the custom-developed software application created to demonstrate theoretical concepts and enable comprehensive analysis of the algorithm's behavior under varying conditions. Section 4 outlines the testing methods applied to assess the performance and efficiency of the encryption system, along with a discussion and interpretation of the findings. Finally, Section 5 concludes this paper by summarizing the key findings and the future development goals.

2. Materials and Methods

This section explores the integration of reversible cellular automata into encryption systems, emphasizing their potential to enhance security and efficiency through precise, rule-based transformations. It begins by outlining the fundamental principles of RCAs, including their unique reversibility, which ensures accurate reconstruction of original data. Building on this foundation, this section details the proposed symmetric block encryption design leveraging RCAs in a multi-layer framework. This design combines multiple RCAs with distinct roles, creating a robust, adaptable system capable of secure encryption across multiple rounds.

2.1. Cellular Automata Classes Used in the Algorithm

Cellular automata are mathematical models consisting of a grid of cells, each of them evolving over discrete time steps according to a set of predefined rules. The grid can be one-dimensional (1D), two-dimensional (2D), or higher and serves as the framework where cells are arranged. All cells update their value synchronously in discrete time steps. Transition rules dictate how a cell's state evolves over time based on its own current state and the states of its neighbors within a defined radius, forming a "neighborhood" [19]. The neighborhood structure defines which surrounding cells influence a given cell, determined by the radius [20]. In cellular automata, the radius defines the number of neighboring cells on either side of a given cell that influence its next state. This influences the state transitions, making it a crucial factor in the complexity and behavior of the automaton. Some of the lengths that may be considered for the radius including the evolution function f are presented below. In each case, $c_i(x)$ denotes the state of i-th cell at time x.

- Radius-1 includes the cell itself and one neighbor on each side (3 cells in 1D CA):

$$c_i(t+1) = f\,[c_{i-1}(t), c_i(t), c_{i+1}(t)] \quad (1)$$

- Radius-2 includes the cell itself and two neighbors on each side (5 cells in 1D CA):

$$c_i(t+1) = f\,[c_{i-2}(t), c_{i-1}(t), c_i(t), c_{i+1}(t), c_{i+2}(t)] \quad (2)$$

- Radius-3 includes the cell itself and three neighbors on each side (7 cells in 1D CA):

$$c_i(t+1) = f\,[c_{i-3}(t), c_{i-2}(t), c_{i-1}(t), c_i(t), c_{i+1}(t), c_{i+2}(t), c_{i+3}(t)] \quad (3)$$

Regardless of the considered radius, the finite cellular automata rules encounter the problem of exceeding the grid limits for marginal cells due to lack of left and right neighbors. In this case, boundary conditions define how cells at the edges of the grid interact with their neighbors. Common boundary types include the following:

- Fixed Boundary: edge cells interact with fixed, predefined states outside the grid (e.g., always 0 or 1);
- Periodic Boundary: the grid wraps around, so the leftmost and rightmost cells are considered neighbors;
- Reflective Boundary: edge cells mirror their immediate neighbors.

Figure 1 presents the graphical representation of the previously mentioned boundary condition types, by considering a finite 8-cell, one-dimensional grid and radius-2:

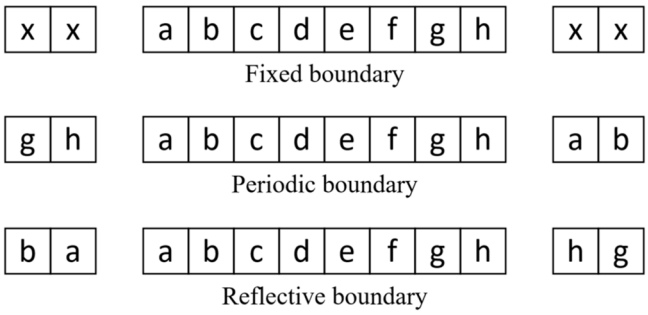


Figure 1. Visual representation of common boundary types.

Elementary cellular automata (ECAs) are the simplest class of cellular automata, consisting of a one-dimensional grid where each cell has two possible states (0 or 1). The evolution of each cell depends on its own state and the states of its two immediate neighbors, forming a 3-cell neighborhood [21]. There are 256 possible rules for ECAs, as each rule represents one of the 256 ways to map a 3-cell neighborhood to a new state. Stephen Wolfram introduced a naming convention in which each rule is assigned a number based on its binary representation as shown in Table 1, which presents two ECA rules (90 and 165).

Table 1. Binary representation of two evolution rules (90 and 165).

Rule [1]	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
90	0	1	0	1	1	0	1	0
165	1	0	1	0	0	1	0	1

[1] Rule in decimal form. The evolution of the central cell is considered in each case.

Extending the radius considered for a cell increases the number of possible rules that can be applied to the formed cellular automata. Therefore, there are 256 radius-1 rules, 2^{32} radius-2 rules, and 2^{128} radius-3 rules.

Reversible cellular automata are a class of cellular automata where each configuration can uniquely determine its predecessor, ensuring the process is invertible. This property is achieved through specially designed transition rules that allow the automaton to evolve forward and backward without losing information. RCAs are particularly useful in applications like encryption and data compression, where reversibility ensures that original data can be reconstructed precisely [22]. They often use complementary rules or rely on specific neighborhood and boundary conditions to maintain their reversible behavior.

In this paper, second-order reversible cellular automata are considered. They extend the concept of traditional RCAs by determining a cell's next state by combining its current state, the states of its neighbors within a defined radius, and its own state from the previous time step ($t - 1$). This combination ensures that the evolution of the system incorporates both spatial and temporal information, maintaining reversibility by allowing the original configuration to be reconstructed uniquely from subsequent states. This design expands the complexity and utility of RCAs for applications requiring robust data transformations. To illustrate this concept, Figure 2 presents the case of a one-dimensional reversible cellular automata cell evolution, considering radius-2 for the evolution function.

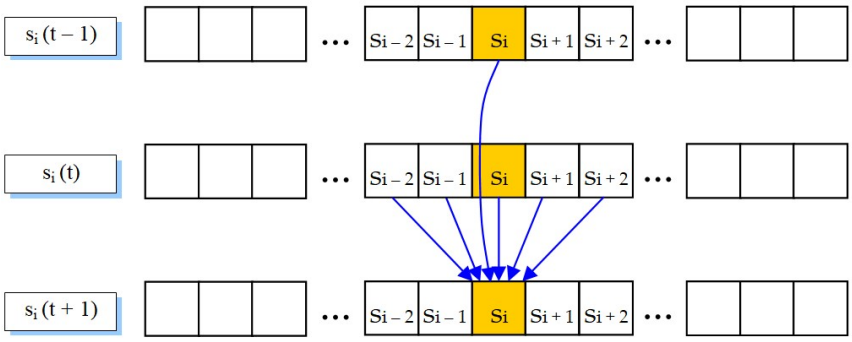


Figure 2. Reversible cellular automata cell state evolution.

In reversible cellular automata, the rule comprises two complementary rules to ensure reversibility. The first rule (denoted CR_1) is applied when the cell's previous state was 1, and the second rule is used when the previous state was 0. The second rule (denoted CR_2) is derived as the complement of the first, ensuring that the states remain uniquely reversible. This is achieved by flipping all bits in the first rule, creating a bitwise negation. Mathematically speaking, the second rule can be calculated starting from the first one according to the following equations, where r is the radius considered for a cell.

$$CR_2 = 2^m - CR_1 - 1 \tag{4}$$

$$m = 2^{2\times r+1} \tag{5}$$

These complementary rules guarantee accurate bidirectional evolution, preserving the ability to reconstruct prior configurations. The reversibility of RCAs constructed in this way is assured by the principle of bijectivity, which requires the function to be both injective (no two inputs map to the same output) and surjective (every possible output is mapped from some input). This ensures that each configuration leads to a unique subsequent state and that any state can be traced back to its original configuration, maintaining the RCAs' reversible nature.

The integration of reversible cellular automata into encryption systems utilizes their forward and backward iterative capabilities to encrypt and decrypt data securely. During encryption, plaintext is encoded as part of the initial RCA state, and the automaton evolves through multiple forward iterations using the predefined complementary rules. Decryption reverses this process by backward evolution, applying the same rules and the same number of iterations, enabling precise reconstruction of the original plaintext from the ciphertext. In each case, the secret key consists in the reversible rule being applied during the encryption and decryption process [23]. Figure 3 illustrates the fundamental concept of a symmetric cryptographic system based on reversible cellular automata.

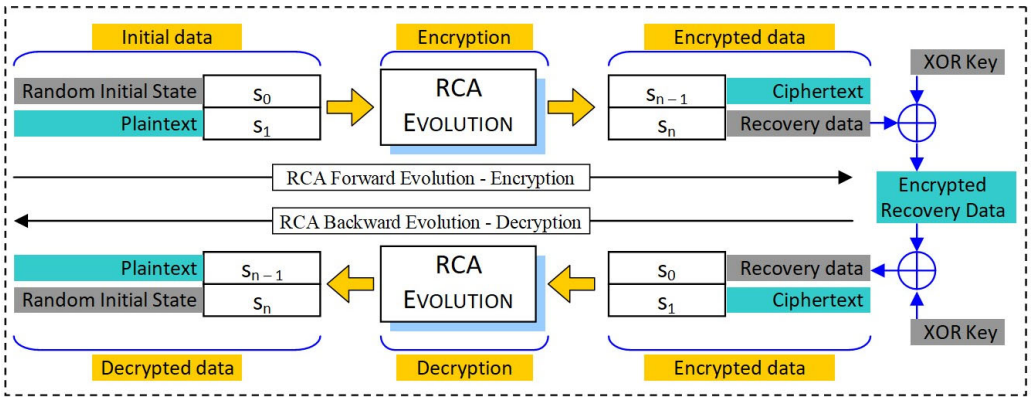


Figure 3. Principles of encryption and decryption using reversible cellular automata.

Encryption and decryption using reversible cellular automata rely on precise configurations of initial states. For encryption, the process begins with two states: s_0, a randomly generated state, and s_1, which represents the plaintext to be secured. After forward iterations for predetermined number of generations, the RCAs produce the ciphertext as state s_{n-1} and an additional state s_n, known as recovery data. To enhance security, this last state s_n is XORed with a defined value in accordance with Vernam algorithm [24] and becomes part of ciphertext, ensuring that the recovery data remain protected, while facilitating accurate plaintext reconstruction in reverse RCA iterations. During decryption, the initial state is composed of $s_0 = s_n$ (recovery data) and $s_1 = s_{n-1}$ (the ciphertext). The original data are produced after backward iteration of the RCAs for the same number of generations performed in the encryption process.

A good random number generator is critical in any encryption system to ensure security and unpredictability [25]. In our RCA-based algorithm, this randomness is essential for generating initial states during the encryption process. In order to maintain an exclusive cellular automata design of the algorithm, the class of linear hybrid cellular automata (LHCAs) based on rules 90 and 150, following Hortensius' construction [26], is used to implement this function. According to this, depending on the number of cells that compose the LHCAs, one of the rules is used for evolving a cell state but in a precise order. Figure 4 below presents the rule construction order for an 8-bit LHCA with a null boundary.

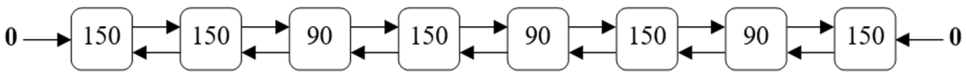


Figure 4. Example of 8-bit LHCA with 90/150 rules.

This approach leverages the properties of LHCAs to produce high-quality random states, ensuring secure and consistent initialization without introducing external randomness sources, thereby preserving the algorithm's integrity. The usage of 90/150 LHCAs

in generating random data for the proposed encryption algorithm is presented in the next subsection.

2.2. RCA-Based Symmetric Block Encryption Design

The proposed encryption algorithm leverages a multi-layer design incorporating three distinct layers formed by four reversible cellular automata in a cascading structure. Each layer has a specific role, ensuring the secure transformation of plaintext into ciphertext while maintaining reversibility for decryption. The design incorporates three layers made up of reversible cellular automata as follows:

- Primary Layer: composed of two RCAs—primary left CA (PLCA) and primary right CA (PRCA)—that takes the plaintext and provides the first step of encryption;
- Shift Layer: composed of one RCA—shift CA (SCA)—which provides shifting functionalities;
- Binding Layer: composed of one RCA—boundary CA (BCA)—which integrates the previous operations and provides the last step of encryption and generates the ciphertext.

According to modern symmetric block cryptographic techniques like Data Encryption Standard or Advanced Encryption Standard, the proposed RCA-based algorithm employs 16 successive rounds, each based on the multi-layer design. This multi-round structure enhances security by repeatedly transforming the input states across multiple layers, increasing diffusion and confusion with every iteration. Each round's output serves as the input for the next, ensuring robust protection. Figure 5 presents the architecture of a single encryption round. Each round is composed of two initial states which differ in function of the round number and generates two final states which form either next round initial data or final data, that is, the encrypted or decrypted information.

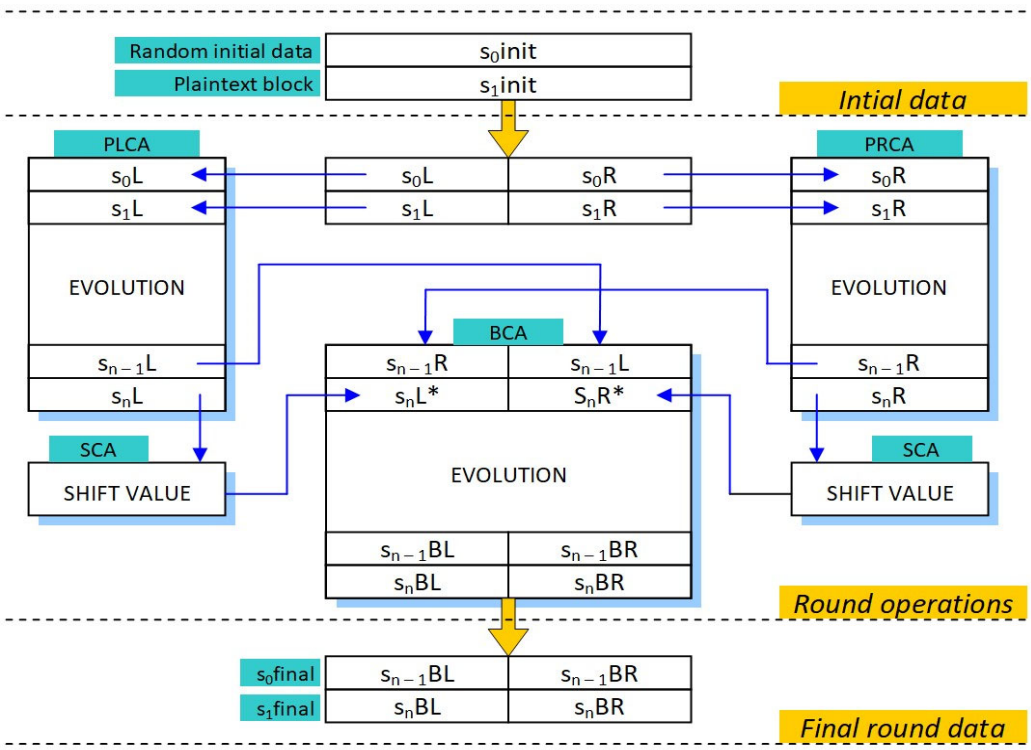


Figure 5. Round operations.

The algorithm follows the principles of symmetric block encryption, and it is specifically designed to process 128-bit data blocks. This means that each RCA has a certain length (number of cells).

The encryption algorithm is composed of four one-dimensional reversible cellular automata defined as PLCA, PRCA, SCA, and BCA. The first two RCAs (PLCA and PRCA) are composed of 64 cells, which is the length required after splitting the initial states. The SCA is composed of 16 cells, and it is used to generate shifting values. The BCA, used to generate the last step of encryption, is composed of 128 cells (equal to a block length). While the reversible cellular automata in primary and shifting layers use a radius-2 neighborhood, the Binding Layer incorporates a radius-3 RCA. Both encryption and decryption involve four functions: evolution of PLCA, PRCA, SCA, and BCA. In each case, periodic boundary is used to perform the iterations.

The plaintext is divided into 128-bit blocks, with encryption of each block beginning using two 128-bit values, $s_0$init and $s_1$init. The plaintext blocks are mapped to $s_1$init, while $s_0$init is randomly generated for the initial block. For subsequent blocks, $s_0$init is derived from the encryption result of the preceding block. Each encryption round produces two outputs, $s_0$final and $s_1$final, which serve as initial values for the next round. Specifically, $s_0$final becomes $s_0$init, and $s_1$final becomes $s_1$init.

The final two configurations produced after the last encryption round of a block, labeled $s_0$final and $s_1$final, have distinct purposes. The first ($s_0$final) becomes the ciphertext block, while the second ($s_1$final) serves as the initial state ($s_0$init) for encrypting the subsequent plaintext block, thus removing the need to generate initial data for each separate block. The process of multiple blocks encryption is presented in Figure 6 below.

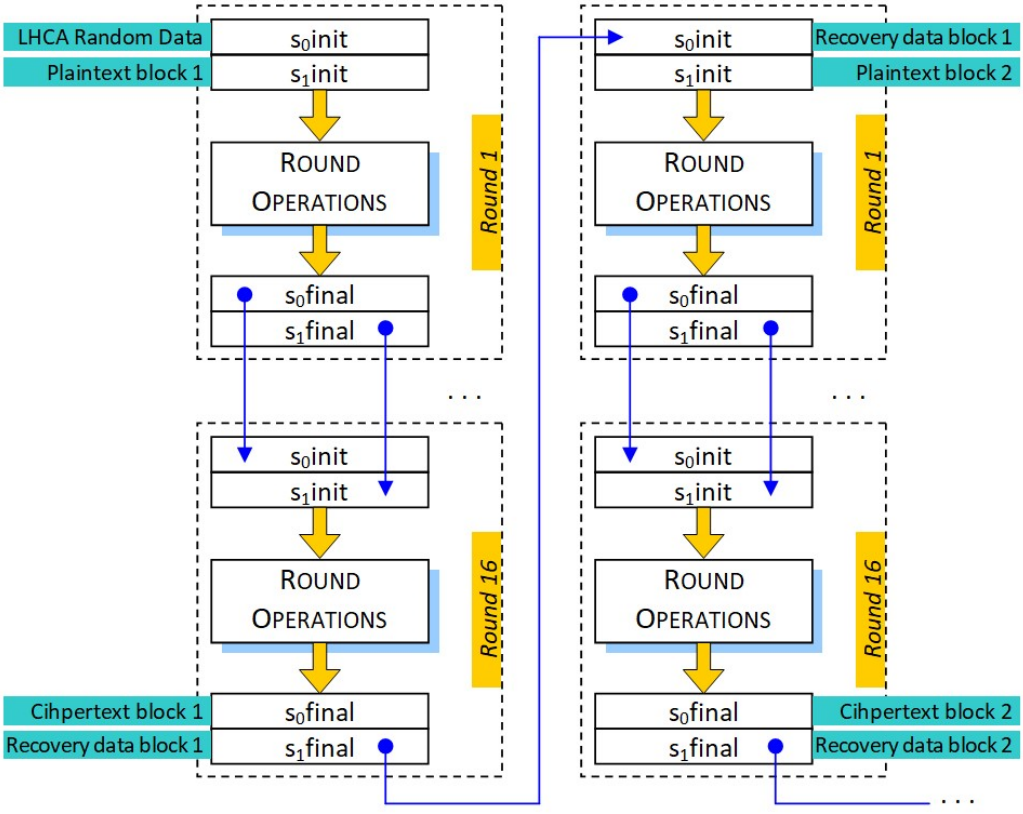


Figure 6. Encryption of multiple blocks of plaintext.

Each round of the encryption algorithm begins with two 128-bit values, $s_0$init and $s_1$init, which are divided into left (s_0L, s_1L) and right (s_0R, s_1R) halves. These halves initialize two RCAs: PLCA processes s_0L and s_1L, while PRCA processes s_0R and s_1R. Both RCAs are iterated for 19 steps, producing intermediate configurations: s_{n-1}L, snL (from PLCA) and s_{n-1}R, snR (from PRCA). Next, a shift operation is applied, where snL shifts left and snR shifts right by Xs positions, with Xs determined by SCA iterations, adding complexity and security.

At the beginning of the encryption process, SCA is initialized only once, using two starting configurations generated by the same random generator used for the initial state of the first plaintext block in round 1. Before every round, SCA evolves through six iterations, with the central cell producing a 6-bit number denoted Xs, as presented in Figure 7. This was chosen so that Xs would take values between 0 and 2^6. This allows for all of 64 cells composing PLCA and PRCA to be shifted partially or completely in a round.

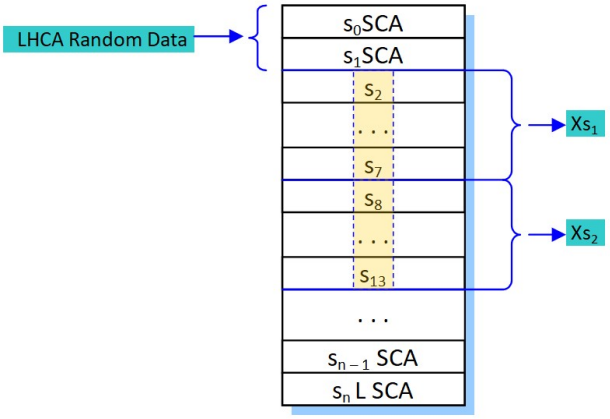


Figure 7. Production of shift values from SCA evolution.

After the shift operations, the configurations snL and snR are transformed into snL* and snR*, respectively. The shifting process concludes with swapping the positions of s_{n-1}L and s_{n-1}R. These swapped values, combined with snL* and snR*, form the initial states for BCA, which is then iterated for 22 steps, completing the round and producing the final configurations required for encryption.

A random number generator based on a linear hybrid cellular automaton (LHCA) using rules 90 and 150 offers a robust method for generating the 128-bit random data required for encryption in this algorithm. Following Hortensius's order construction principles, four LHCA instances of 32 cells each will be employed to produce the necessary bit length. These LHCA modules, operating in parallel, ensure high-quality randomness for initializing the first block in the first round and the two initial states of SCA. This approach leverages the studied behavior of rule 90/150 combinations to maintain consistency and cryptographic strength [27]. The combination order of evolution rules for a 32-bit LHCA is presented in Table 2, where "0" denotes usage of rule 90 and "1" denotes usage of rule 150.

Table 2. Combination of evolution rules.

Number of Cells	Cycle Length	Rule Combination
32	2^{32}	01000110000010011011101111010101

The 90/150 LHCA will be initialized with an arbitrary state, which will then be iterated for a number of generations to produce the final value that will be considered as random data in the encryption algorithm as presented in Figure 8 below.

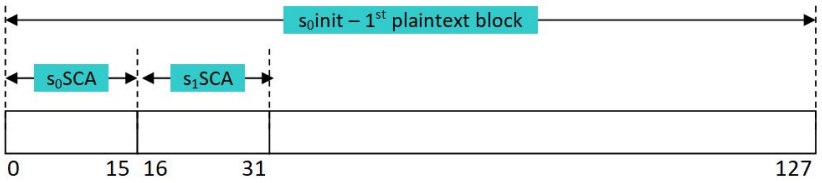


Figure 8. Usage of random generated data in the encryption system.

The last state of the LHCA will be saved as the new initial state for the next data encryption. This is possible due to the design of the encryption system, which does not require the random generated data to be remembered for the opposite process of decryption.

The encryption algorithm utilizes a 224-bit key designed specifically for the multi-layered reversible cellular automata framework. This key is composed of segments allocated to each of the four RCAs based on their respective neighborhood radius. The radius determines the size of the rules needed to evolve each RCA, ensuring that the algorithm maintains consistency and reversibility in its operations.

PLCA and PRCA, operating with a radius-2 neighborhood, are assigned 32-bit rules each. Similarly, SCA, also a radius-2 RCA, is assigned another 32 bits from the key. These segments ensure that PLCA, PRCA, and SCA perform their roles efficiently, with rules designed for localized transformations and bit-shifting processes.

The remaining 128 bits are allocated to BCA, which uses a radius-3 neighborhood. This larger radius requires a more complex rule set, as it incorporates a broader influence of neighboring cells during its evolution. By assigning 128 bits to BCA, the algorithm ensures that the final integration of the intermediate states from the Primary Layer is robust and secure.

The architecture of the 224-bit key is specifically designed based on the radius of the RCAs used in the encryption algorithm, containing the rules that govern their evolution. However, the key structure does not include reversible rules in their entirety according to the formal definition. Instead, the key is constructed to contain only the rules applied when the previous state of an RCA cell is “1”. This design choice simplifies the key structure while maintaining the algorithm’s functionality, as the complementary rule for cells in a previous state of “0” can be easily derived by complementing the specified rule in the key. This approach not only reduces the complexity of the key but also ensures that the reversibility of the RCA system is preserved, as the complementary rules are inherently determined by the algorithm’s design.

The choice of a 224-bit key length is directly related to the structure of the encryption algorithm, which incorporates four reversible cellular automata with specific rules based on their radius. The key is divided into segments corresponding to the evolution rules for each RCA: two 32-bit rules for each RCA in the Primary Layer (PLCA, PRCA) with a radius-2 neighborhood and one 128-bit rule for the BCA with a radius-3 neighborhood used in the Binding Layer. The remaining 32 bits are allocated to the rule used by the Shift Layer RCA, which also uses radius-2 for the evolution of the cells state. This design of the key, in terms of bit length, is solely based on the chosen radiuses for the RCAs in the system without being influenced by the length of the blocks. By selecting these specific radiuses, the algorithm leverages the diverse complexity of neighborhood interactions, ensuring a robust and scalable encryption process. The corresponding rule lengths of 32 bits and 128 bits provide an optimal balance between computational efficiency and cryptographic strength. This structural design allows the key to encapsulate all necessary information for the RCA rules while maintaining the overall 224-bit length.

This key design ensures flexibility, allowing the algorithm to balance complexity and efficiency. The distribution of the 224 bits, presented in Figure 9, ensures that each RCA layer, whether performing transformations, shifts, or final integrations, has access to the rule sets needed for secure and precise operations.

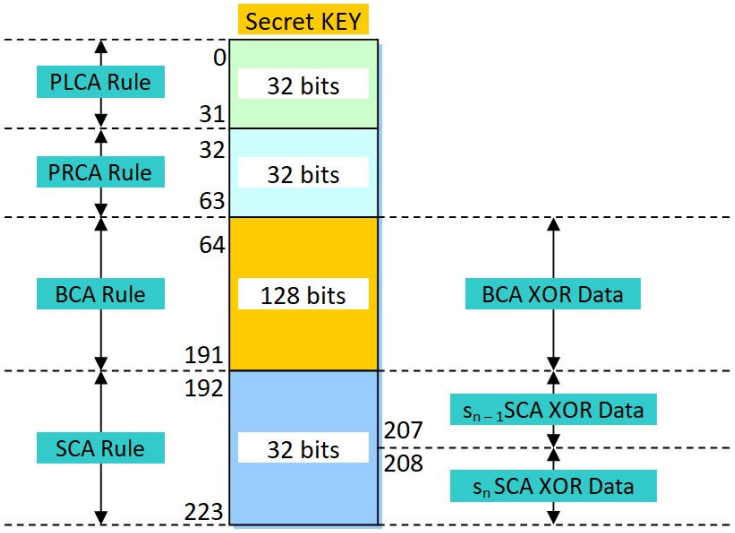


Figure 9. Secret key architecture and role.

The key not only governs the evolution of the RCAs but also acts as a safeguard for the system's outputs, ensuring that the encryption remains secure against potential attacks and fully reversible during decryption. Specifically, it is used to secure the last state of the system through a well-defined XOR operation. Thus, additional data are added to the final ciphertext, with the role of making the process of decryption possible. These are called Encrypted Recovery data (ERd), and they have the crucial role of securing the last state of reversible cellular automata after the encryption process, in order to prevent potentially attacks due to knowing two consecutive configurations of the system. Therefore, the ERd are formed by securing the last state of BCA (recovery data for last plaintext block) along with the last two states of the SCA through an XOR operation with certain bits of the secret key. Thus, bits 64 to 191 are used to encrypt BCA's last configuration, while bits from 192 to 223 are used for encryption of the last two configurations of SCA.

This will result in 160-bit length of the Encrypted Recovery data, which will be added after the last ciphertext block. The corresponding operations that help the formation of ERd are described by Equations (6)–(8) below:

$$ERd[0..127] = s_n\ BCA[0..128]\ XOR\ Key[64..191] \quad (6)$$

$$Erd[128..143] = s_{n-1}\ SCA[0..15]\ XOR\ Key[192..207] \quad (7)$$

$$Erd[144..159] = s_n\ SCA[0..15]\ XOR\ Key[208..223] \quad (8)$$

This process enhances the cryptographic robustness of the algorithm by protecting sensitive data from unauthorized access or manipulation. The XOR operation is a lightweight yet highly effective cryptographic tool, as it introduces an additional layer of security without significantly increasing computational complexity. This dual functionality underscores the key's pivotal role in the algorithm's overall security architecture.

In order to summarize the operation of securing data, a detailed step-by-step explanation of actions performed for encryption is presented below:

- **Step 1:** *System initialization*

  Secret key is set along with the random initial data produced by the evolution of the LHCA. The plaintext is divided into 128-bit blocks to ensure consistent processing. If the final block is shorter than the required 128-bit length, padding is applied by adding SPACE characters until the block reaches the necessary size.

- **Step 2:** *Input initialization*

  The plaintext block (128 bits) is split into two 64-bit sections: s_1L and s_1R. The random initial state is split into two 64-bit sections: s_0L and s_0R.

- **Step 3:** *Primary Layer transformations*

  The two composing RCAs in this layer are initialized: PLCA is set with initial states s_0L and s_1L, while PRCA is set with initial states s_0R and s_1R. After their evolution, each produced the outputs, consisting of the last two states as follows: PLCA outputs $s_{n-1}L$ and s_nL, while PRCA outputs $s_{n-1}R$ and s_nR.

- **Step 4:** *Shift Layer transformations*

  In this step, the shift value Xs is calculated. SCA is initialized with two random 64-bit states derived from the same LHCA used in step 1. SCA evolves for 6 steps, and the shift value Xs is produced as a 6-bit number formed by the concatenated values of the central cell derived from the 6 states of the SCA. The second function subsequent to this layer is the actual operation of shifting the previous obtained states, s_nL and s_nR, by Xs positions left or right accordingly.

$$s_nL^* = \text{LeftShift}\ (s_nL, Xs) \tag{9}$$

$$s_nR^* = \text{RightShift}\ (s_nR, Xs) \tag{10}$$

- **Step 5:** *Binding Layer transformations*

  The RCAs in this layer, operating on 128 bits, are initialized by combining 64-bit states obtained in previous steps as follows:

$$s_0BCA = s_{n-1}R + s_{n-1}L \tag{11}$$

$$s_1BCA = s_nL^* + s_nR^* \tag{12}$$

  The evolution of BCA produces two outputs denoted s_{n-1}final and $s_n$final.

- **Step 6:** *Final operations*

  The ciphertext blocks are formed using s_{n-1}final. The recovery data ($s_n$final) are secured through XOR operations as previously described, and they are saved for decryption process.

The process of decryption, which has to reconstruct the original data, starts from the Encrypted Recovery data, incorporated in the ciphertext. The first step of decryption involves applying the same XOR operation as in the encryption process, using the same values derived from the secret key. This operation ensures that the original recovery states of BAC and SCA are securely retrieved, allowing the decryption process to proceed.

The ciphertext is processed in fixed blocks of 128 bits, maintaining consistency with the block size used during encryption. However, a critical distinction in this RCA-based algorithm is that the decryption process works from the end to the start, beginning with the last ciphertext block. The importance of starting decryption from the last ciphertext block lies in the RCAs' design, which inherently relies on the final states of the system

to reconstruct earlier configurations. This means that the last two states of the system, derived from the encryption of the last plaintext block, are essential for decrypting all previous blocks.

Decryption proceeds in reverse by reconstructing each plaintext block sequentially, starting with the last block. The outputs from each RCA layer in the final round are processed in reverse order, iterating backward to retrieve the initial states. These initial states are then used to decrypt the preceding ciphertext block, continuing the backward sequence until the first plaintext block is recovered. This approach leverages the bijective nature of RCAs, which ensures that each step of the decryption accurately reverses the transformations applied during encryption.

Basically, decryption is performed by executing the same operations as in the encryption process but in reverse order. While encryption traverses the layers in the sequence of the Primary Layer, Shift Layer, and Binding Layer, decryption reverses this order, starting with the Binding Layer, followed by the Shift Layer, and concluding with the Primary Layer. The Binding Layer first combines the decrypted states to provide the inputs for the Shift Layer. In this reversed process, the Shift Layer reconstructs the original states by shifting the PLCA data to the right and the PRCA data to the left, effectively undoing the shifts applied during encryption. Finally, the Primary Layer processes the reconstructed states to produce the plaintext block, completing the decryption round. This reversed traversal ensures that the original plaintext is accurately restored.

3. Encryption Algorithm Implementation

This section presents the methods and techniques used for the software-level implementation of the multi-layer encryption algorithm. The implementation was developed in order to test, verify, and validate the concepts of encryption using multiple RCAs, as outlined in Section 2. The software application, developed in this stage, allows for thorough testing of the correct evolution of each RCA, the specific functions performed by each layer, and, most importantly, whether data can be successfully encrypted and decrypted.

The implementation process closely adhered to the proposed system design. For every operation performed within a layer, a corresponding function (method) was carefully developed to ensure the functionality and correctness of that layer, taking into consideration all possible scenarios like incomplete blocks of data. Since the encryption algorithm operates on fixed blocks of 128 bits (equivalent to 16 characters at 8 bits per character), any incomplete block must be properly handled to maintain the required block size. To address this, padding is applied to the last block of data using SPACE characters until the block reaches the necessary 16-character length. This ensures that the RCA in the Primary Layer receives input with the correct bit length, allowing for seamless processing and maintaining the algorithm's consistency and functionality.

Although the software implementation's main goal is to demonstrate the theory behind the encryption method, computational improvements were made in order to optimize RCA evolution, thus increasing efficiency regarding parameters like execution time and resources. Among the functions implemented in the algorithm, a key component is the *ApplyRCARule* function. This method is critical, as it governs the evolution of the reversible cellular automata utilized in the system, operating with a radius-2 neighborhood and periodic boundary conditions (wrap-around). Optimizations were made in this function so that it could determine the next state of a cell using bitwise operations which are highly efficient high-level operations. The C# source code for function *ApplyRCARule* is as follows:

```
public static string ApplyRCARule (string previousRow, string currentRow, BigInteger
rule)
{
    char[] nextRow = new char[currentRow.Length];
    for (int i = 0; i < currentRow.Length; i++)
    {
    // Extract 2 neighbors to the left and 2 neighbors to the right (radius 2)
    int leftNeighbor1 = (i < 2) ? currentRow[currentRow.Length - 2 + i] - '0' :
currentRow[i - 2] - '0';
    int leftNeighbor2 = (i < 1) ? currentRow[currentRow.Length - 1] - '0' : currentRow[i -
1] - '0';
    int rightNeighbor1 = (i >= currentRow.Length - 1) ? currentRow[0] - '0' :
currentRow[i + 1] - '0';
    int rightNeighbor2 = (i >= currentRow.Length - 2) ? currentRow[i -
currentRow.Length + 2] - '0' : currentRow[i + 2] - '0';
    // Form a 5-bit neighborhood (radius 2)
    int pattern = (leftNeighbor1 << 4) | (leftNeighbor2 << 3) | (currentRow[i] << 2) - '0'
| (rightNeighbor1 << 1) | rightNeighbor2;
    // Choose rule based on the previous state
    if (previousRow[i] == '1')
    // Rule 1 (when previous state is 1)
    nextRow[i] = ((rule & (BigInteger.One << pattern)) >> pattern) == 1 ? '1' : '0';
    else
    // Complementary Rule 2 (when previous state is 0)
    nextRow[i] = (((~rule) & (BigInteger.One << pattern)) >> pattern) == 1 ? '1' : '0';
    }
    return new string(nextRow);
}
```

Both the Primary Layer and Shift Layer composing RCAs utilize this function in the evolution of their states. The Binding Layer RCA, which evolves based on a radius-3 neighborhood, employs a similar version of this function (*ApplyRCARule3R*) where two more additional cells, one left and one right, are considered in producing the future state of the current cell.

To better illustrate the process of securing data, Algorithm 1 below presents the description of the encryption procedure, using pseudocode:

Algorithm 1: Encryption Using Multi-Layer RCA-Based System

Input: Plaintext P (128-bit blocks) and Key K (224 bits)
rand_data = iterate(LHCA);
$s_0$init = rand_data[0..127];
s_0SCA = rand_data[0..15];
s_1SCA = rand_data[16..31];
plaintext_blocks = split(P, 128);
for i = 1 to plaintext_blocks **do**
$s_1$init = plaintext_blocks[i];
for j = 1 to 16 **do**
s_{0L}(PLCA) = $s_0$init[0..63];
// Split current plaintext block and random initial data into two 64-bit sections
var (s_0L, s_0R) = split(s_0Init, 64);

Algorithm 1: *Cont.*

var (s_1L, s_1R) = split(s_1Init, 64);
// Evolve left (PLCA) and right (PRCA) parts of Primary Layer and save two last states
var (s_{n1}L, s_nL) = EvolveRCA(s_0L, s_1L, plcaRule);
var (s_{n1}R, s_nR) = EvolveRCA(s_1L, s_1R, prcaRule);
// Perform bitshift calculation—value Xs
Xs = CalculateXs(s_0SCA, s_1SCA, scaRule);
// Apply shift operations
s_nL* = BitShift(s_nL, "left", Xs);
s_nR* = BitShift(s_nR, "right", ns);
// Form BCA input states
s_0BCA = q_{n1}R + q_{n1}L;
s_1BCA = s_nL* + s_nR*;
// Evolve BCA and save two last states
var ($s_0$final, $s_1$final) = EvolveBCA(s_0BCA, s_1BCA, bcaRule);
$s_0$init = $s_0$final;
$s_1$init = $s_1$final;
end for
Ciphertext_blocks[i] = $s_0$final;
$s_0$init = $s_1$final;
end for
// Securing of final states of BCA and SCA
ERd[0..127] = $s_1$final[0..128] ^ K[64..191];
ERd[128..143] = s_{n1}SCA[0..15] ^ K[192..207];
ERd[144..159] = s_nSCA[0..15] ^ K[208..223];
Output: Ciphertext (Ciphertext_blocks) and Encrypted Recovery data (Erd).

The software interface offers a comprehensive environment for simulating and testing the cryptographic process. The application supports both encryption and decryption processes, allowing data to be inputted directly through the interface or select files for processing in case of large data. It includes features that enable testing and validation of data throughout the encryption and decryption process on data fed to the system. The main sections include the following:

- Algorithm parameters: allow setting initial data, including the secret key and initialization data, which are shown, in the interface, in hexadecimal format;
- Original data: section for setting original data to be encrypted (directly or from file);
- Encrypted data: show the result of encryption of original data with the option of exporting it for further analysis and also the Encrypted Recovery data in hexadecimal format;
- Decrypted data: show the result of decryption of encrypted data;
- Data analysis: provides information about the first and last two states of each RCA after each round, both for encryption and decryption processes, through which intermediate states can be analyzed.

The Data Analysis section extends, at the bottom of the interface, by presenting the key architecture, offering a detailed breakdown of each segment's composition in binary format. Each segment corresponds to specific RCA evolution rules, as outlined in the previous section, highlighting how the key integrates into the algorithm's operations. The binary data for each segment are clearly displayed, allowing for analyzing the relationship between the key structure and the algorithm's functionality.

4. Results and Discussion

This section analyzes and discusses the results of the proposed encryption algorithm to verify its functionality and performance. Additionally, this section examines the algorithm's performance through various tests, including statistical and graphical analysis of the encrypted data. It also presents an evaluation of the secret key's effectiveness in securing encrypted data, highlighting its role in maintaining the system's overall security and reliability. Testing was conducted using a desktop computer with an Intel i7-7500U processor (2.7 GHz frequency) and 16 GB of RAM.

Using the software application, developed during this study, initial sample data were provided to the system, along with the required algorithm parameters consisting of the secret key and random initial data. After these configurations, the original data were encrypted, thus producing the ciphertext and the associated additional data in the form of Encrypted Recovery data (ERd). The next phase consisted in decryption of the previously obtained results. In this case, the ciphertext was considered part of the initial configuration of the system along with the ERd information. The obtained data after this process matched the initial sample data provided, thus demonstrating the correct evolution and functionality of the multi-layer encryption system.

The positive results regarding the validation of the system's functionality marked the first step in confirming the effectiveness of the encryption algorithm. These initial tests demonstrated the system's ability to accurately encrypt and decrypt data. Building on this foundation, the testing process continued with a series of statistical analyses on the encrypted data provided by the application.

Statistical testing represents a critical step in evaluating an encryption system, as it assesses the randomness of the encrypted data produced by the algorithm. According to cryptographic theory, a secure encryption system must generate output that appears statistically random. This randomness is vital for preventing patterns that could be exploited by attackers.

Statistical tests developed by the National Institute of Standards and Technology (NIST) were employed to evaluate whether the generated sequences met the standards for pseudo-random number generators [28]. This suite includes 14 tests and can identify patterns of non-randomness in binary sequences generated by both hardware and software cryptographic systems.

Each test produces a statistic used to calculate a p-value, representing the probability of obtaining a sequence less random than the tested one. A significance level ($\alpha = 0.01$) was chosen, allowing 99% confidence in randomness while permitting up to 1 in 100 sequences to fail. To ensure accurate results, approximately 1,000,000 bits of encrypted data were analyzed, divided into ten streams of 100,000 bits each. In each test, the p-value and pass rate (the percentage of passing sequences) were calculated. These results provided a comprehensive evaluation of the randomness of the encrypted data, with the findings summarized in Table 3.

The NIST statistical test suite also provides additional analyses on encrypted data in respect to the distribution of values of zeroes and ones in the generated ciphertext for each of the 10 sequences. An important characteristic of an encryption system is ensuring a balanced distribution of zeroes and ones in the encrypted data. This balance enhances randomness, making it more resistant to statistical attacks, as no discernible patterns can be exploited by an attacker to deduce the original plaintext or the encryption key. The results are presented in Table 4.

Table 3. Results of *p*-values and the proportion of passing sequences (NIST test suite).

NIST Statistical Test	*p*-Value	Proportion (Pass Rate)
Frequency (monobits)	0.911413	10/10 (PASS)
Block frequency	0.911413	10/10 (PASS)
Runs	0.534146	10/10 (PASS)
Longest run of one in a block	0.035174	10/10 (PASS)
Cumulative sums (forward)	0.739918	10/10 (PASS)
Cumulative sums (reverse)	0.534146	10/10 (PASS)
Rank	0.739918	10/10 (PASS)
FFT	0.739918	10/10 (PASS)
Non-overlapping template	0.911413	10/10 (PASS)
Overlapping template	0.350485	9/10 (PASS)
Approximate Entropy	0.213309	10/10 (PASS)
Serial 1	0.534146	10/10 (PASS)
Serial 2	0.534146	10/10 (PASS)
Linear complexity	0.911413	10/10 (PASS)

Table 4. Frequency analysis on encrypted data (NIST).

	Seq1	Seq2	Seq3	Seq4	Seq5	Seq6	Seq7	Seq8	Seq9	Seq10
No. of zeroes	49,996	49,823	50,209	49,965	49,947	50,093	50,178	50,083	50,435	49,859
No. of ones	50,004	50,177	49,791	50,035	50,053	49,907	49,822	49,917	49,565	50,141

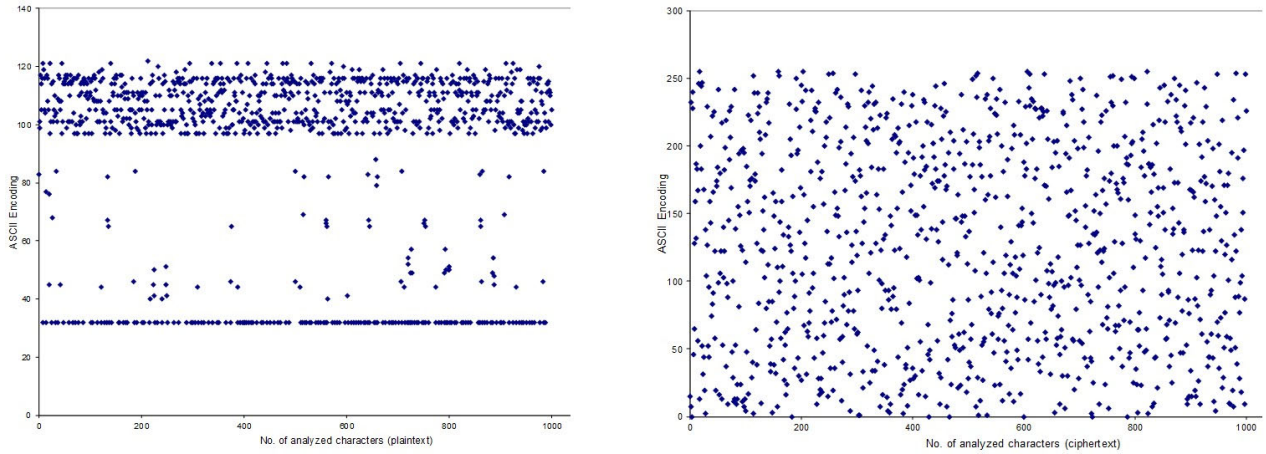
The National Institute of Standards and Technology's tests evaluate critical properties like uniformity, scalability, and consistency, which are key indicators of a sequence's randomness. These tests rigorously assess whether a sequence demonstrates the characteristics necessary for being classified as potentially random. According to NIST's guidelines, the minimum requirement for passing each statistical test is around 8 for a sample size of 10 binary sequences. Based on the results obtained, the sequence exhibits properties that suggest it has the potential for randomness, aligning with the expectations for cryptographic applications.

The same NIST statistical test suite was employed in a previous study [27] to evaluate and validate the capability of the LHCA based on rules 90 and 150 in generating high-quality random sequences. This reinforced the decision of using it for generation of necessary random data in the proposed encryption algorithm.

The testing on encrypted data also included graphical analysis, conducted to examine the diffusion rate of the original text over the ASCII interval. This analysis aimed to verify how effectively the encryption algorithm dispersed the plaintext's characters across the ASCII range. By evaluating the graphical representation of character distribution in the encrypted output, this study provided additional insights into the encryption system's ability to produce random-like and uniformly dispersed data. Figure 10a,b below shows the dispersion of the ciphertext in comparison to plaintext. The graphics depict the approximately 500 characters used for this test on the X-axis, while the Y-axis contains the ASCII values (encoding of alpha-numerical characters).

An important aspect of the proposed encryption algorithm lies in the choice of 16 rounds for encryption and decryption of data. This decision was made firstly by reporting to the currently used algorithm, which also relies on multiple rounds, but secondly, and most importantly, this was based on additional analyses meant to determine the optimal number of rounds for achieving the maximum degree based on the avalanche effect, determined by analyzing how a small change in plaintext or key affected the generated ciphertext. Changing the value of one randomly chosen bit in either of the first two should produce a

change in nearly half of the values of the ciphertext. Figure 11 presents the result of this operation performed throughout a different number of rounds, showing that the desired propriety is achieved around the chosen 16 rounds, which provides an optimal balance between efficiency and performance of the algorithm.



(**a**) (**b**)

Figure 10. Data diffusion across ASCII range: (**a**) Plaintext; (**b**) Ciphertext.

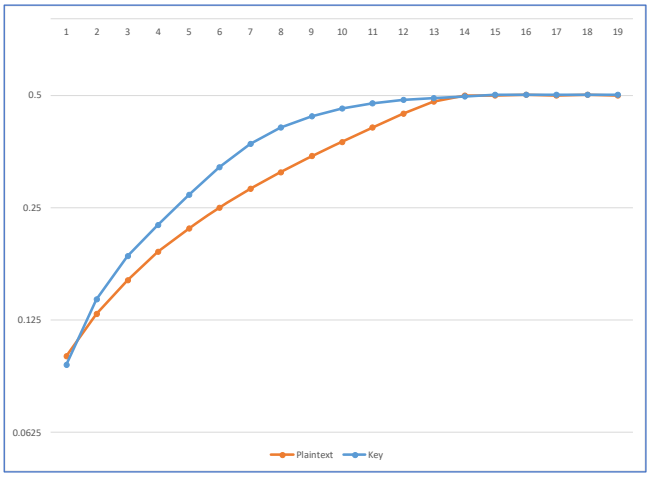


Figure 11. Avalanche effect across a different number of rounds.

Following the statistical and graphical analysis of the system, it is essential to examine another critical factor in ensuring data security, specifically a cryptanalysis based on the secret key. The 224-bit key not only provides the rules necessary for RCA evolution but also acts as a critical security measure. Its length makes brute-force attacks practically infeasible, as the vast keyspace of 2^{224} potential keys ensures a high level of cryptographic strength. Additionally, the structured allocation of bits based on neighborhood radius guarantees that each RCA performs optimally within its designated role, contributing to the overall security and efficiency of the encryption algorithm.

Moreover, the key contributes to preventing an attack based on reverse engineering which can be attempted by starting from the final configurations of the system. This assumes finding the last state of BCA and the last states of SCA, which in turn are secured based on specific sections of the secret key, along with the rules for BCA (2^{128} possible configurations), SCA (2^{16} possible configurations), and PLCA and PRCA (each having 2^{64} possible configurations). The large keyspace for each RCA rule's possible configuration is

assured by the length on 128 bits of blocks used in the algorithm, thus making this type of attack computationally impractical.

Performance analyses were conducted to evaluate potential vulnerabilities of the proposed encryption algorithm, particularly concerning its resistance to linear and differential cryptanalysis. These analyses focused on assessing the algorithm's avalanche effect across multiple test sequences in order to measure the sensitivity to changes in the key and plaintext. Figure 12 shows the results of testing how small modifications in the key or plaintext influenced the resulting ciphertext, demonstrating the algorithm's robustness in ensuring that minor input changes produce significant, seemingly random alterations in the encrypted output, thereby reinforcing its security.

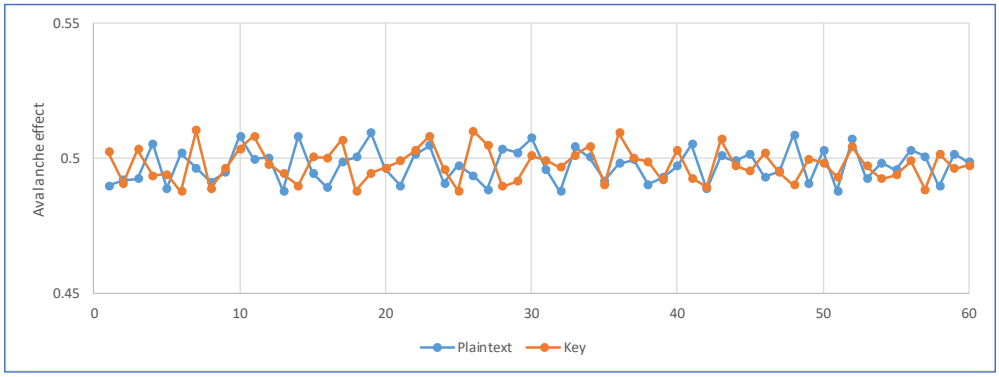


Figure 12. Avalanche effect (multiple sample data).

Another key aspect of the encryption algorithm is its ability to ensure that even if the same plaintext block appears multiple times, it is encrypted into a different ciphertext each time. This is achieved because the encryption of each block begins with initial data derived from the encryption of the previous block. This chaining process introduces variability into the initial conditions for each block, ensuring that identical plaintexts are transformed into unique ciphertexts, further enhancing the security of the system by preventing linear cryptanalysis attacks.

An efficiency analysis was performed to evaluate the throughput of the proposed encryption algorithm in Mbps, comparing its performance with other state-of-the-art algorithms (Table 5). The results indicate that the algorithm's performance is influenced by its software-level implementation, as this study represents the initial phase of research. The next phase involves transitioning the algorithm to a hardware implementation on platforms such as FPGA, where significantly improved performance is anticipated. Dedicated hardware often provides a performance gain factor in the range of 5 to 10 compared to software implementations, leveraging the parallel nature of the algorithm and the inherent efficiency of hardware acceleration. This transition is expected to further demonstrate the potential of the proposed system in real-world applications.

Table 5. Comparative analysis results.

Algorithm	[29]	[30]	[31]	Proposed Algorithm
Mbps (avg.)	6.3558	10.3303	21.7254	19.4357

While our proposed encryption algorithm shares some structural similarities with DES and AES, such as the use of fixed-size blocks and multiple rounds, it diverges significantly in terms of design principles and underlying mechanisms.

The primary distinction lies in the use of reversible cellular automata as the foundation of the encryption process. Unlike DES or AES, which rely on complex mathematical

operations like substitution–permutation networks or modular arithmetic, our scheme utilizes the inherent properties of RCAs, such as locality of interactions and reversibility. These features enable efficient and bijective transformations, ensuring that the encryption and decryption processes are symmetric and computationally lightweight.

Additionally, the multi-layer design of the proposed algorithm incorporates multiple RCAs with distinct roles, each governed by rules determined by their radius. This approach introduces dynamic complexity into the encryption process, with operations such as the shifting layer driven by shift cellular automaton (SCA) evolution. Such features are not present in DES or AES, making our biologically inspired encryption system highly adaptable and very versatile.

Moreover, the key structure in our algorithm is uniquely tailored to the RCA-based architecture, incorporating specific rules for RCA evolution. This contrasts with the fixed-length keys in DES and AES, offering a more flexible and scalable framework while maintaining a secure 224-bit keyspace, addressing potential vulnerabilities such as identical ciphertext for repeated plaintext blocks. These distinctions collectively highlight the contributions and potential advantages of our algorithm over traditional encryption methods like DES and AES. The proposed algorithm, leveraging reversible cellular automata, has unique characteristics that make it particularly effective in certain specific applications and aspects compared to AES, such as the following:

Hardware Implementation: The RCA-based encryption algorithm is inherently parallel due to the local interaction rules of cellular automata. This makes it highly suitable for implementation on FPGA or VLSI architectures, where its parallel nature can be fully exploited. Such implementations could outperform AES in scenarios requiring high-speed, energy-efficient encryption, such as real-time IoT communications and embedded systems.

Dynamic Key Behavior: The algorithm's ability to produce different ciphertexts for identical plaintext blocks due to its RCA-based architecture ensures enhanced security against linear, pattern-based cryptanalysis. This makes it particularly effective in securing communications where repeated data patterns are common, such as sensor networks and multimedia transmissions.

Scalability and Customization: The flexibility in configuring the RCA rules and layers allows the algorithm to be adapted to different security levels and computational environments. This adaptability makes it suitable for applications requiring tailored encryption, such as lightweight cryptography for resource-constrained devices or high-security cryptographic needs for critical infrastructure.

The proposed algorithm introduces unique advantages, such as leveraging reversible cellular automata to achieve encryption and decryption through simple, local, and highly parallelizable operations. Performance-wise, while the computational throughput of the algorithm may be slightly lower in purely software implementations compared to AES, the algorithm's parallel-friendly nature makes it highly suitable for hardware implementation using FPGA or VLSI, ensuring that the encryption and decryption processes are symmetric and computationally lightweight offering potential gains in speed and energy efficiency.

5. Conclusions and Future Developments

Cryptosystems employing novel techniques and methodologies represent a highly dynamic field of research, driven by the rapid advancements in technology and emerging threats to classical encryption algorithms. The development of quantum computing, in particular, poses a significant challenge to traditional cryptographic methods, as many of these rely on mathematical problems that quantum algorithms are capable of solving efficiently. In this context, finding alternatives to classical encryption algorithms has become a critical area of study. The proposed encryption system aligns with this trend by leveraging

reversible cellular automata in a multi-layer framework, introducing a biologically inspired, computation-efficient approach. This work contributes to the exploration of innovative solutions that address the evolving landscape of cryptographic security.

The multi-layer structure ensures that data undergo transformations at multiple levels, with each RCA layer contributing a unique form of modification. The combination of localized (radius-2) and extended (radius-3) dependencies prevents straightforward analysis of the data, making the algorithm highly resistant to cryptographic attacks. Additionally, the use of a 224-bit key along with a 128-bit block size and the secure handling of recovery data further enhance the encryption system's robustness.

The positive results obtained in this study pave the way for advancing system development in order to overcome the limitations of the current implementation, such as its reliance on software simulations. Future research will focus on hardware-level implementation, particularly on FPGA platforms [32], to exploit the parallel nature of cellular automata and further improve performance. Additional investigations into scalability and energy efficiency are also planned. These advancements should further enhance the system's scalability and efficiency, making it suitable for real-time cryptographic applications in various fields.

Author Contributions: Conceptualization, G.C.S. and P.A.; Methodology, G.C.S. and P.A.; Software, G.C.S. and P.A.; Validation, G.C.S. and P.A.; Writing—original draft, G.C.S. and P.A.; Supervision, P.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no funding.

Data Availability Statement: The original contributions presented in this study are included in the article.

Acknowledgments: The authors acknowledge the use of the National Institute of Standards and Technology (NIST) test suite for statistical testing and analysis of the encrypted data to evaluate the system's randomness.

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

References

1. Seh, A.H.; Zarour, M.; Alenezi, M.; Sarkar, A.K.; Agrawal, A.; Kumar, R.; Ahmad Khan, R. Healthcare Data Breaches: Insights and Implications. *Healthcare* **2020**, *8*, 133. [CrossRef] [PubMed]
2. Techtarget. Available online: https://www.techtarget.com/healthtechsecurity/feature/Largest-healthcare-data-breaches (accessed on 11 November 2024).
3. IEA (International Energy Agency). Available online: https://www.iea.org/reports/ukraines-energy-security-and-the-coming-winter/ukraines-energy-system-under-attack (accessed on 11 November 2024).
4. Scripcariu, L.; Diaconu, F.; Mătăsaru, P.D.; Gafencu, L. AES Vulnerabilities Study. In Proceedings of the 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 28–30 June 2018. [CrossRef]
5. Rahmani, M.; Nitaj, A.; Ziane, M. Partial Exposure Attacks on a New RSA Variant. *Cryptography* **2024**, *8*, 44. [CrossRef]
6. Gokcay, E.; Tora, H. A novel data encryption method using an interlaced chaotic transform. *Expert Syst. App.* **2024**, *237*, 121494. [CrossRef]
7. Feng, W.; Yang, J.; Zhao, X.; Qin, Z.; Zhang, J.; Zhu, Z.; Wen, H.; Qian, K. A Novel Multi-Channel Image Encryption Algorithm Leveraging Pixel Reorganization and Hyperchaotic Maps. *Mathematics* **2024**, *12*, 3917. [CrossRef]
8. Feng, W.; Wang, Q.; Liu, H.; Ren, Y.; Zhang, J.; Zhang, S.; Qian, K.; Wen, H. Exploiting Newly Designed Fractional-Order 3D Lorenz Chaotic System and 2D Discrete Polynomial Hyper-Chaotic Map for High-Performance Multi-Image Encryption. *Fractal Fract.* **2023**, *7*, 887. [CrossRef]
9. Deng, Q.; Wang, C.; Sun, Y.; Deng, Z.; Yang, G. Memristive Tabu Learning Neuron Generated Multi-Wing Attractor With FPGA Implementation and Application in Encryption. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2024**, *72*, 300–311. [CrossRef]
10. Ma, X.; Wang, Z.; Wang, C. An Image Encryption Algorithm Based on Tabu Search and Hyperchaos. *Int. J. Bifurc. Chaos* **2024**, *34*, 2450170. [CrossRef]

11. Feng, W.; Zhang, J.; Chen, Y.; Qin, Z.; Zhang, Y.; Ahmad, M.; Woźniak, M. Exploiting robust quadratic polynomial hyperchaotic map and pixel fusion strategy for efficient image encryption. *Expert Syst. Appl.* **2024**, *246*, 123190. [CrossRef]
12. 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. [CrossRef]
13. Corona-Bermúdez, E.; Chimal-Eguía, J.C.; Téllez-Castillo, G. Cryptographic Services Based on Elementary and Chaotic Cellular Automata. *Electronics* **2022**, *11*, 613. [CrossRef]
14. Degala, D.P.; Athithan, S.; Chinnasamy, P. A survey on data security using reversible cellular automata. In Proceedings of the International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 23–25 January 2023; pp. 1–6.
15. Martín del Rey, A.; Casado Vara, R.; Hernández Serrano, D. Reversibility of Symmetric Linear Cellular Automata with Radius r = 3. *Mathematics* **2019**, *7*, 816. [CrossRef]
16. Nanda, S.K.; Mohanty, S.; Pattnaik, P.K.; Sain, M. Throughput Optimized Reversible Cellular Automata Based Security Algorithm. *Electronics* **2022**, *11*, 3190. [CrossRef]
17. Vijaya Bhaskara Rao, B.; Rawat, U.; Roy, S.; Lal, C. SORCHIC: A Hybrid Image Cipher for IoT Applications Using Second Order Reversible Cellular Automata. *IEEE Access* **2024**, *12*, 146147–146159. [CrossRef]
18. Kumar, S.; Gupta, A.; Walia, G.S. Reversible data hiding: A contemporary survey of state-of-the-art, opportunities and challenges. *Appl. Intell.* **2022**, *52*, 7373–7406. [CrossRef]
19. Wolfram, S. *A New Kind of Science*; Wolfram Media: Champaign, IL, USA, 2002.
20. Anghelescu, P. Evolution of Hybrid Cellular Automata for Density Classification Problem. *Symmetry* **2024**, *16*, 599. [CrossRef]
21. Kozlov, V.; Tatashev, A.; Yashina, M. Elementary Cellular Automata as Invariant under Conjugation Transformation or Combination of Conjugation and Reflection Transformations, and Applications to Traffic Modeling. *Mathematics* **2022**, *10*, 3541. [CrossRef]
22. Stănică, G.C.; Anghelescu, P. Theory and application of reversible cellular automata in cryptography. In Proceedings of the 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Bucharest, Romania, 29–30 June 2023; pp. 1–4. [CrossRef]
23. Stănică, G.C.; Anghelescu, P. Reversible Cellular Automata Based Cryptosystem. *Electronics* **2024**, *13*, 2515. [CrossRef]
24. Hatkar, S.S.; Pawar, B.K. Symmetric key algorithm using Vernam cipher: VSA. In Proceedings of the International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–27 August 2016.
25. Wolfram, S. Random Sequence Generation by Cellular Automata. *Adv. Appl. Math.* **1986**, *7*, 123–169. [CrossRef]
26. Hortensius, P.; McLeod, R.; Card, H. Parallel random number generation for VLSI systems using cellular automata. *IEEE Trans. Comput.* **1989**, *38*, 1466–1473. [CrossRef]
27. Stănică, G.C.; Anghelescu, P. Cryptographic Algorithm Based on Hybrid One-Dimensional Cellular Automata. *Mathematics* **2023**, *11*, 1481. [CrossRef]
28. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, N.; et al. A Statistical Test Suite for Random and PseudoRandom Number Generators for Cryptographic Applications. *Natl. Inst. Stand. Technol. Spec. Publ.* **2005**, *2005*, 800–822.
29. Li, H.; Li, T.; Feng, W.; Zhang, J.; Zhang, J.; Gan, L.; Li, C. A novel image encryption scheme based on non-adjacent parallelable permutation and dynamic DNA-level two-way diffusion. *J. Inf. Secur. Appl.* **2021**, *61*, 102844. [CrossRef]
30. Hua, Z.; Zhu, Z.; Yi, S.; Zhang, Z.; Huang, H. Cross-plane colour image encryption using a two-dimensional logistic tent modular map. *Inf. Sci.* **2021**, *546*, 1063–1083. [CrossRef]
31. Li, H.; Yu, S.; Feng, W.; Chen, Y.; Zhang, J.; Qin, Z.; Zhu, Z.; Wozniak, M. Exploiting Dynamic Vector-Level Operations and a 2D-Enhanced Logistic Modular Map for Efficient Chaotic Image Encryption. *Entropy* **2023**, *25*, 1147. [CrossRef] [PubMed]
32. Anghelescu, P. Parallel Optimization of Program Instructions Using Genetic Algorithms. *Comput. Mater. Contin.* **2021**, *67*, 3293–3310. [CrossRef]