mathematics

MDPI

Article

Cryptographic Algorithm Based on Hybrid One-Dimensional Cellular Automata

George Cosmin Stănică * and Petre Anghelescu *

Department of Electronics, Computers and Electrical Engineering, University of Pitesti, 110040 Pitesti, Romania
* Correspondence: george.stanica@upit.ro (G.C.S.); petre.anghelescu@upit.ro (P.A.)

Abstract: The theory and application of cellular automata (CA) for a stream cipher-based encryption principle are presented in this study. Certain fundamental transformations are developed based on CA theory regarding decentralized computation for modeling different system's behavior. The changes governing state transitions rely on simple evolution rules, which can easily be translated into functions using logic operators. A class of linear hybrid cellular automata (LHCA) based on rules 90 and 150 is used to implement these functions. Symmetric key systems theory is the foundation of the suggested algorithm. The algorithm functions use the proprieties provided by the LHCA evolution in order to convert plain text into cipher text and vice versa, in each case starting from the same initial state of the system and performing the same number of steps for each operation. Cellular automata's parallel information processing property, in addition to their regular and dynamical structure, makes hardware implementation of such schemes best suited for VLSI implementation. Testing of the proposed algorithm was performed by developing both software and hardware solutions. Hardware implementation of the presented cryptosystem was developed using VHDL hardware description language and a FPGA device (XILINX Spartan3E XC3S500E). Design and software simulations have been carried out using the C# programming language.

Keywords: dynamical systems; cellular automata; cryptography; cryptosystems; stream ciphers; biologically inspired systems; reconfigurable computing; symmetric encryption

MSC: 68Q80; 37B15; 68P25; 94A60; 68W35

check for updates

Citation: Stănică, G.C.; Anghelescu, P. Cryptographic Algorithm Based on Hybrid One-Dimensional Cellular Automata. *Mathematics* **2023**, *11*, 1481. https://doi.org/10.3390/math11061481

Academic Editor: Ximeng Liu

Received: 17 February 2023
Revised: 13 March 2023
Accepted: 16 March 2023
Published: 17 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The digitalization of our society is drastically altering the way computer systems are used. A vast proportion of the population is always connected to the Internet using various services. At the same time, we are always vulnerable to attacks: our sensitive data may be stolen, edited, or deleted. In the last two years, the use of online services such as communication, payments and digital data exchange has increased significantly due to the pandemic. In addition, a considerable number of companies have adopted remote working. The spread of blockchain technology, especially in cryptocurrency, is another major aspect that we should consider due to the sensitive data exchanged in this kind of communication. Cybersecurity becomes more and more crucial now, when information is becoming the driving force of society. Therefore, it seems reasonable to assume that secure communication is becoming increasingly important in our everyday lives.

Today's society's requirement for communications is increasingly high and the need for speed and security is also very high. Implicitly, it is vital that these communications are secured and fast, and these attributes can only be provided by the correct and effective implementation and integration of cryptographic algorithms. Currently, the strength of cryptographic algorithms based on bio-inspired systems (e. g., cellular automata) lies in the ease of software simulation and hardware implementation due to their properties: local connectivity, simple components and regular structure. It is essential, therefore, to develop

cryptographic solutions able to cope with the communication requirements of today (and of the near future)—especially concerning wireless communications in cellular networks and in computer networks, where very high-speed data communication is expected (starting at 11 Mbps and rising up to 11 Gbps). Of course, due to the complexity of this field, from a theoretical point of view, there are still many aspects that need to be analyzed, studied and structured.

As the Internet of Things (IoT) continues to grow, the situation is expected to worsen in the coming years. These devices have unique characteristics that pose new security challenges, and traditional security measures are often inadequate. The Mirai botnet, for example, used vulnerable consumer devices such as IP cameras to launch large-scale Distributed Denial of Service (DDoS) attacks.

Moreover, cybersecurity is now a pressing concern for fields beyond computer science that rely on digital devices and services for critical functions. This includes e-health, medicine, robotics, power and water supply plants, smart cities, and other critical infrastructure. The digital transformation of medicine has increased its exposure to cybersecurity risks, as medical devices such as surgical assistants and implants are increasingly connected and vulnerable to attacks.

The goal of this paper is to design and implement a stream cipher algorithm, which is one of the fastest encryption procedures nowadays. The fact that data are encrypted as they stream rather than being stored in the system's memory is the most crucial feature of stream ciphers, which offer the speed of encryption. Stream cipher techniques are widely used in various practical applications, such as the A5 algorithm in GSM communications and the E0 encryption system in Bluetooth specifications. These procedures rely on a secret key, known only to the parties involved, and a public algorithm, typically a sequence generator. By utilizing these elements, stream cipher procedures are able to generate long sequences of seemingly random bits known as keystream sequences, which are in fact pseudorandom sequences used for cryptographic purposes.

The paper is divided into five sections, which are organized as follows. The next section presents an overview of cellular automata concepts and defines the idea of the ways in which a particular class of hybrid CA can be used for stream encryption. Section 3 presents the proposed algorithm of data encryption through the software and hardware implementations developed and also experimental results of both solutions are discussed here. Testing of the proposed algorithm using statistical and graphical analysis, along with the results discussion, is presented in Sections 4 and 5. Section 6 concludes the study by outlining future research objectives and potential extensions to this work.

2. Basic Concepts on Cellular Automata and Cryptography

2.1. Overview of Cellular Automata Concept

This section presents an overview of cellular automata theory followed by a description of cryptographic techniques based on different variations of CA with references to the literature regarding this concept.

Mathematicians S. Ulam and John von Neumann [1] first proposed the cellular automaton (the two initially calling those cellular spaces) as a model in the 1940s in order to build a self-reproducing machine [2]. However, research in this field has gained momentum, especially in the last 20 years. This is due to the increasing complexity of hardware systems and software applications. The initiator of the field of cellular automata is Von Neumann (the creator of the sequential computing machine model).

Cellular automata refer to discrete dynamical systems that operate on a regular lattice, relying on straightforward local interactions among their components, thereby eliminating the need for partial differential equations [3]. These automata exhibit a series of distinct characteristics, including discreteness in terms of space, time, and values, as well as local interaction, homogeneity, and parallel evolution [4]. The wide potential CA have for modeling complicated systems, despite their simplicity, is the primary driver of their

usage in so many areas. A CA is defined entirely by specifying the neighborhood and the evolution function of the states as presented in Figure 1 below.

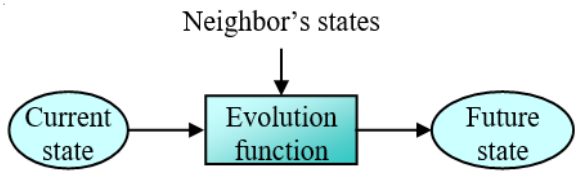


Figure 1. Structure of cellular automata.

Additionally, considering the length, at least theoretically infinite, of the number of cells that make up the automaton, it is also necessary to specify the size and the boundary conditions for its extremities (for the marginal cells). These elements must be specified regardless of the type of cellular automaton used. There are also accepted cases in which not all cells have the same type of neighborhood or the same evolution rule, a situation in which the evolution rules and neighborhoods must be specified for each cell that is part of the automaton [5].

The organizational similarity between cellular automata and natural systems, made up of a large number of simple components with interactions at the local level, allows the modeling of simple natural systems with self-organization. In the case of cellular automata, one way to define self-organization is the property of the system to manifest a global order at the spatial, temporal, or spatio-temporal level. This global order arises only based on a local "order" and the presence of basins of attraction indicates the likelihood of these features appearing. CA is a bio-inspired paradigm with deep involvement in soft computing and hardware for large classes of applications. These methods have been effectively employed in simulating biological and physical systems, and more recently in developing parallel and distributed algorithms, including those used for data encryption [6].

2.2. Related Research about the Usage of Cellular Automata in Cryptography

The use of cryptography techniques is essential for any secure communication. Secret and public key systems are the two primary types of encryptions used today. Reference [7] provides a thorough description of the present and developing encryption methods used in both types of systems. A promising cryptography technique consists of applying cellular automata as a primitive of an encryption system.

For modeling many applications including data encryption, researchers and scientists from various domains have taken advantage of the CA paradigm of local information, decentralized control and universal computing. Reference [8] provides a good summary of all key cryptography techniques. CA have been used in both symmetric-key and public-key cryptography. The first proposal for a stream CA-based encryption algorithm was released by Wolfram [9], and subsequently developed by Tommassini et al. [10] and more recently by Seredynski et al. [11]. Research has also been conducted on the different cryptography applications fields, including image encryption presented in [12,13] and also in one of the increasingly used domains such as blockchain and device networks.

Wolfram was the first to use cellular automata (CA) to encrypt messages, employing a uniform one-dimensional CA and rule 30 to construct pseudorandom number sequences. At each simulation step, the communications are encrypted and decrypted by combining them directly with the state of the CA. Other solutions based on CA used for data encryption are presented in [14,15]. Wolfram has investigated cellular automata using empirical observations and simulations [16]. In the case of two-state, three-neighborhood cellular automata (CA), the evolution of the (i)th cell can be expressed as a function of the current states of the ($i - 1$)th, (i)th, and ($i + 1$)th cells, as demonstrated by Equation (1). Each cell concurrently changes its state based on nearby cells at a specific moment.

$$s_i(t + 1) = f(s_{i-1}(t), s_i(t), s_{i+1}(t)). \quad (1)$$

The equation shown above employs f to denote the combinational logic used in a two-state three-neighborhood cellular automaton. Such a cellular automaton has eight unique neighborhood configurations (as there are $2^3 = 8$ possible combinations of the states of the three neighboring cells) and 256 unique mappings to the subsequent state, each of which corresponds to a distinct cellular automaton rule [17].

Typically, a cell's surroundings are assumed to consist of a few nearby cells. This results in the issue of cells that are located on the grid's edge. A CA must define an input for the missing neighbor in order to determine what occurs at the outermost cells. When utilizing a cyclic-boundary cellular automaton, the value from the outermost cell on the opposite end of the array is utilized. Conversely, in the case of a null-boundary cellular automaton, these neighboring cells are assigned a constant value of "0" (or ground, in the context of hardware implementation).

Starting from the point of simplifying the design and modeling of complex systems, many CA improvements have been developed. Evolution rules are referred to as linear if they solely use XOR logic, resulting in linear CA. Another helpful version of these systems, where the cells follow a combination of rules, is called a hybrid cellular automaton. The rule set of this type of CA, applied as a linear variant, can be analyzed through matrix algebra [18].

Although typical applications for cellular automata reside in modeling, game theory [19], pseudorandom signal generation [20] and image processing [21], the main goal of this research is to develop and implement solutions for data encryption. In this sense, cellular automata are used as a mathematical model of complex natural systems composed of simple components [22].

Recent research has focused on the development of encryption techniques using cellular automata, a field of study that has shown great promise in enhancing the security of data transmission. One such study by Li et al. [23] compared three different cellular automata-based encryption schemes and determined that all three were effective in achieving high security with fast encryption and decryption times. However, the study also highlighted the need for further research to optimize these schemes for practical use.

Another study presented in [24] tested the performance of cellular automata-based encryption methods using a special class of reversible rules. The authors suggested that cellular automata encryption techniques show an enhanced throughput in a high-performance system compared with other algorithms.

In addition, several articles have explored the use of cellular automata-based encryption in combination with other technologies, such as chaos theory and neural networks. For example, study [25] proposed a hybrid encryption method that combined cellular automata with neural networks and determined that this approach was highly effective in achieving secure data transmission.

Complex behavior and self-organization phenomena appear in their case through the cooperation of the parts, which follow physical, chemical, and other local laws. It should be noted that cellular automata also have this characteristic: they have simple components that evolve according to local rules, usually simple. However, their global behavior is complex and could present self-organization phenomena.

3. Materials and Methods of the Proposed Cryptographic Algorithm

This part begins with a description of LHCA, the core component of the system, and continues with software where features, development process and acquired results are described. The section ends with the presentation of the entire proposed cryptosystem composed of the software application and the hardware implementation.

3.1. Description of Used LHCA

This study focuses on one-dimensional cellular automata that utilize linear transition rules and have a three-neighborhood structure. Additionally, the cellular automata considered in this work are hybrid, meaning that different cells within the structure follow

distinct transition rules. The automata also have a null-boundary structure, which means that cells adjacent to the outermost cells in the structure have null content.

The transition function of cellular automata is often referred to as the "rule" by most authors, and this naming convention follows the system of numbering introduced by Wolfram. The output bits of each 2^N potential input are taken into account when determining the rule's number, which is subsequently translated into decimal form. An instance of a cellular automaton rule can be represented by a function that takes input strings of 111, 110, . . . , 000 and generates output bits, such as rule 90, which outputs bits 0, 1, 0, 1, 1, 0, 1, 0, respectively, for the corresponding inputs.

The LHCA built using rules 90 and 150 serves as the foundation for the cryptographic method proposed in this work. Only LHCA with states "0, 1" are taken into account. The following is an expression for the equivalent state transition logic of rules:

$$s_i\ (t + 1) = s_{i-1}(t) \oplus s_{i+1}(t); \text{ rule 90;} \tag{2}$$

$$s_i\ (t + 1) = s_{i-1}(t) \oplus s_i(t) \oplus s_{i+1}(t); \text{ rule 150.} \tag{3}$$

Equations (2) and (3) show that at time $t + 1$, the content of the i-th cell depends on the content at time t of either two different cells (rule 90) or three different cells (rule 150), with $i = 1, \ldots, L$, where L is the length of the automaton. Moreover, the state of the automaton at time t is the binary content of the L cells at such an instant. Table 1 displays the subsequent condition of a cell for rules 90 and 150. This condition applies to the middle cell of the trio shown at the top of the table.

Table 1. Evolution rules that determine the LHCA future states.

Rules	111	110	101	100	011	010	001	000
90	0	1	0	1	1	0	1	0
150	1	0	0	1	0	1	1	0

Additionally, CA may be defined using a transition matrix (S-matrix). Equation (4) displays the CA general transition matrix, where S_i stands for the rule that is taken into consideration for the i-th cell. Stage i follows rule 90 if $S_i = 0$ and rule 150 in all other cases. The unpredictability of the patterns produced can be quantified by the matrix analysis. This matrix is capable of producing pseudorandom sequences.

$$T_{CA} = \begin{bmatrix} s_1 & 1 & 0 & \ldots & 0 & 0 \\ 1 & s_2 & 1 & \ldots & 0 & 0 \\ 0 & 1 & s_3 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & s_{n-1} & 1 \\ 0 & 0 & 0 & \ldots & 1 & s_n \end{bmatrix}. \tag{4}$$

This hybrid cellular automaton's combination of rules 90 and 150 is a crucial component of its operating concept. When the cellular automaton operates independently, it can achieve a maximal cycle length with no zero state. This cycle can be calculated using the formula $2^n - 1$, where n is the number of bytes. Table 2 displays the hybrid design required to produce a CA with the maximum cycle length ($3 \leq n \leq 14$).

In the table above, we assumed that the rule was presented in hexadecimal format, with "0" standing for a rule 90 cell and "1" for a rule 150 cell.

The following figure (Figure 2) provides an example of an 8-stage LHCA based on the building guidelines stated in Table 2. The null-boundary condition helps the cellular automaton increase cycle lengths. Additionally, CA have drawn a lot of interest since each cell is only connected to its neighbors. As a result, a CA is easily expandable by adding more cells to the end of it.

Table 2. Combination of evolution rules.

Number of Cells	Rule	Cycle Length
3	001 = 1	7
4	0101 = 5	15
5	11001 = 19	31
6	010101 = 15	63
7	1101010 = 6A	127
8	11010101 = D5	255
9	110010101 = 195	511
10	0101010101 = 155	1023
11	11010101010 = 6AA	2047
12	010101010101 = 555	4095
13	1100101010100 = 1954	8191
14	01111101111110 = 1F7E	16,383
...	...	...

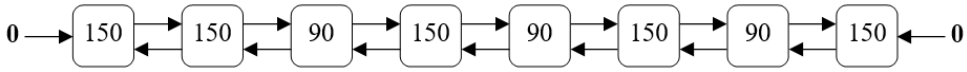


Figure 2. Example of 8-bit LHCA with 90/150 rules.

The arrangement of the rules causes the composed cellular automata to exhibit chaotic behavior, meaning that even a small change in the initial state of the system can lead to vastly different outcomes, though taken separately it seems that their behavior is simple. Below is an example of the chaotic behavior that can result from this combination of rules in which we consider starting with an initial state of all cells in 0, except for one cell, which is a 1.

Initial state:	0 0 0 0 0 1 0 0
After 20 generations:	0 0 1 0 1 0 1 1
After 40 generations:	1 1 1 0 1 1 0 1
After 60 generations:	1 0 1 1 1 0 1 1
After 80 generations:	0 1 1 0 1 1 1 0
After 100 generations:	0 1 1 1 0 1 0 1

As the system evolves according to the rules, patterns emerge that can be very different depending on which cell was chosen to be the initial one. The pattern changes dramatically after just a few generations, even though the initial state only differs by a single cell. This is a result of the complex and unpredictable interactions between the two rules. This demonstrates the sensitivity of the system to its initial conditions, and that even small differences can result in vastly different outcomes.

Furthermore, the choice of the rule 90/150 was based on our previous work presented in [17], in which we investigated their behavior as pseudorandom number generator by testing the produced sequences using NIST Statistical Test. The positive results obtained after the test represent a factor that encouraged us in choosing these rules.

The stream cipher (decipher) procedure can be defined as follows:

- Set up encryption/decryption key (initial state of LHCA);
- Perform XOR operation between character and LHCA state;
- Evolve LHCA state;
- Repeat steps 2 and 3 for N times (N = number of iterations through LHCA).

After the first stage, the XOR operation will be performed between the new state of the LHCA and the result of the previous operation. The result of the final XOR operation constitutes the processed message (encrypted/decrypted data) depending on the operation performed.

The algorithm is designed to perform identical number of steps for both encryption and decryption, provided that the private key (initial state of the system) is the same. The

same operations are performed in both processes, the only difference being the initial data provided to the system (plain text or cipher text). The system works by taking the initial data and performing the XOR operations with the cellular automata states. The only requirements to decrypt data are to use the same initial state of the CA as well as to perform the same number of iterations through the CA as was previously performed on encryption process.

The plain text is encrypted and decrypted at the byte level, so the characters are instantaneously changed to cipher text. The LHCA data encryption and decryption mechanism is shown in Figure 3.

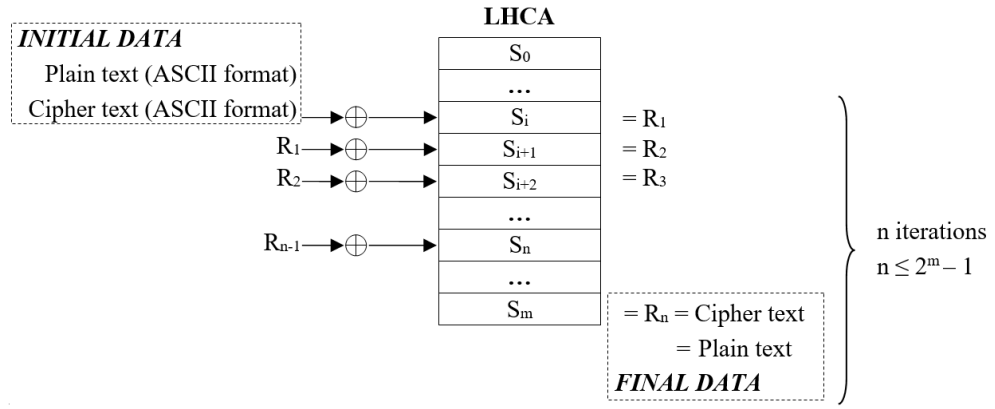


Figure 3. General encryption/decryption process.

The algorithm operates by encrypting each message character. The ASCII code of the character and the system key (initial state of CA) are XORed together in the first phase. The following stages involve changing the CA state and performing a new XOR operation on the prior result and the CA's new state. A certain number of iterations of this operation are performed, provided that the number is less than the maximum cycle length of the LHCA used, which is 255.

The encryption and decryption procedures are based on the data to be processed and the initial state of the LHCA. For this, a character is first converted into binary using the equivalent ASCII code. Then, XOR operation is performed on the binary value of the character and the LHCA states for a predetermined number of steps. An example of this procedure, with 3 iterations, is presented below:

Encryption Procedure
Character to encrypt: S (ASCII code for S = $53_{(16)}$ = $01010011_{(2)}$);
States of LHCA: S1 = 10110101 (initial state); S2 = 10100101; S3 = 10011101.
R1 = S ⊕ S1 = 01010011 ⊕ 10110101 = 11100110;
R2 = R1 ⊕ S2 = 11100110 ⊕ 10100101 = 01000011;
R3 = R2 ⊕ S3 = 01000011 ⊕ 10011101 = 11011110 = ▌.
▌ = Ciphertext equivalent of character S.
Decryption Procedure
Character to decrypt: ▌(ASCII code for ▌ = $DE_{(16)}$ = $11011110_{(2)}$);
States of LHCA: S1 = 10110101 (initial state); S2 = 10100101; S3 = 10011101.
R1 = ▌⊕ S1 = 11011110 ⊕ 10110101 = 01101011;
R2 = R1 ⊕ S2 = 01101011 ⊕ 10100101 = 11001110;
R3 = R2 ⊕ S3 = 11001110 ⊕ 10011101 = 01010011 = S.
S = Plaintext equivalent of character ▌.

The essential element of these ciphers is the creation of the key stream. The randomization of the stream key completely destroys any statistical features present in the message. In this system, the key is defined by the initial state of the LHCA and the number of iterations applied to the CA. The key of the system, depicted in Figure 3, consists of state Si (initial

state) and n (number of iterations). The proposed encryption algorithm was designed for 8-bit encryption, and therefore, an LHCA with 8 cells was used to generate the key.

To evaluate and study this encryption method, applications were developed for both hardware and software. CA-based models are extremely parallelizable since the new state in each time step is defined entirely by the preceding time step's neighboring state [26]. As a result, we can use parallel processing methods such as FPGA boards that are used to create encryption at the hardware level.

3.2. Software Implementation

The software application was created to enable the simulation of the chosen CA as well as the execution of the encryption method over various input data. Cellular automata simulation and encryption/decryption are the two sections of the program.

A variety of functions are offered by the CA simulation area presented in Figure 4 below to enable various configurations of the hybrid cellular automaton that is employed. In addition to utilizing the optimal evolution rules, it is also possible to input their various combinations to examine the evolution of the CA. Although the encryption algorithm presented here uses an 8-cell CA, the application provides the possibility of choosing the number of cells to simulate, varying from 4 to 10 CA cells. Both the cycle duration and the emergence of the states may be observed and verified.

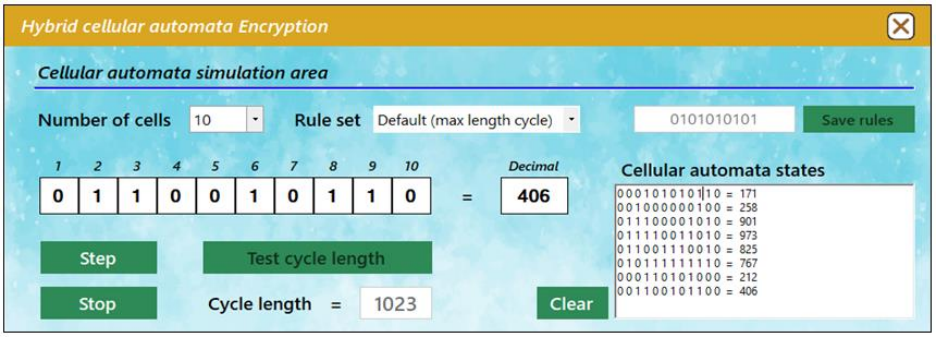


Figure 4. Application LHCA simulation area.

The second section provides controls for the encryption/decryption algorithm. The text message is loaded from a file selected by the user through the application window. Once the message is loaded, the user must specify the key of the system by setting the values of the cells in the first section of the application. When this section is activated, the CA is automatically configured as an 8-cell system. After setting the initial configuration, the operation is shown in two fields as follows:

- Original message: shows the initial message (plain text or cipher text can be inserted here);
- Processed message: shows the final message (encrypted or decrypted in accordance with the original message).

The encryption/decryption process is performed using the appropriate evolution rule (90 or 150) for each stage of the LHCA. This can be achieved using logic expressions of the two rules presented in Equations (2) and (3). For the implementation of LHCA evolution, a different technique was used. First, we encoded the order of the rules in binary form, '0'—for rule 90 and '1'—for rule 150. This allowed combining Equations (2) and (3) in a single Equation (5) that describes the behavior of the entire LHCA.

$$s_i(t+1) = s_{i-1}(t) \hat{} [R_o(s_i) \,\&\, s_i(t)] \hat{} s_{i+1}(t), \quad (5)$$

where $R_o(s_i)$ represents the rule applying to the current cell.

The controls provided in this section of the application allow for both encryption and decryption using the fields mentioned earlier. Once the encryption or decryption operation is completed, the fields can be cleared to process new messages. Figure 5 shows an example

of encryption in which the original message (plain text) is converted into cipher text using the algorithm.

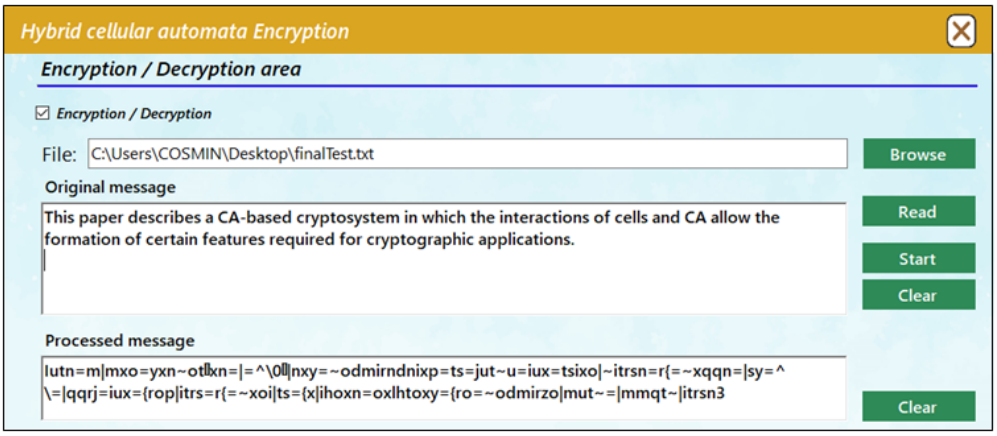


Figure 5. Results of the encryption process.

The plain text provided from a text file can be observed in the original message field. The process of encryption is carried out character by character and produces the ciphertext, which is displayed in the processed message field. Similarly, decryption is performed in the same way, starting from an encrypted message and processing it into plain text as shown in Figure 6 below.

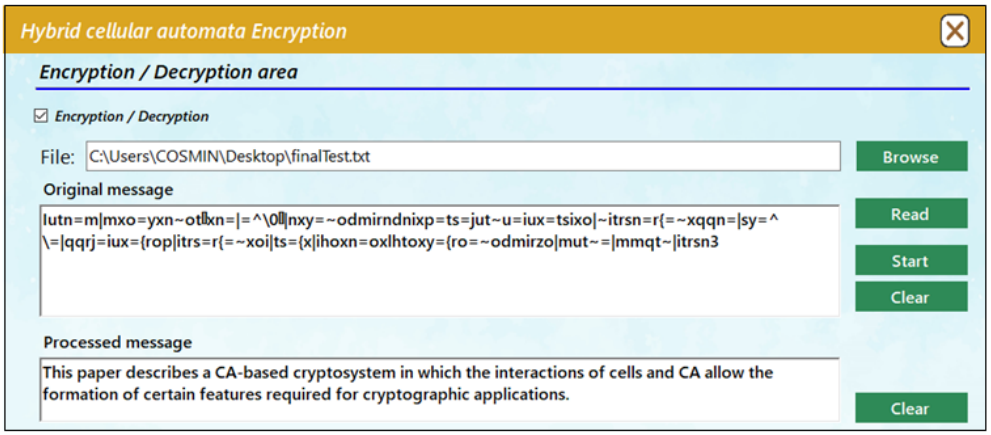


Figure 6. Results of the decryption process.

If the system's initial state (the private key) is the same, the suggested algorithm's uniqueness enables completing the same operation steps for encryption and decryption. The developed system provides an ideal framework for a reasonably detailed CA investigation of emergent properties.

3.3. Hardware Implementation

The suggested cryptosystem's general architecture is built on an FPGA device design, which handles encryption and decryption and a software desktop application, which represents the graphical tool that offers system's users an interface to manage hardware operation [27]. It is possible to synthesize logic architectures using several methods, including hardware description languages [28].

The C# programming language was used to fully implement the suggested solution and its functionality in the software applications. The hardware system was implemented as an active HDL application and was tested on a FPGA hardware device (XILINX Spartan3E XC3S500E).

Below is a block schematic of the proposed hardware cryptosystem (Figure 7). While the main (middle) module executes encryption or decryption depending on the kind of message received from the PC, the receiver and transmitter modules manage communication between PC software applications and the FPGA board.

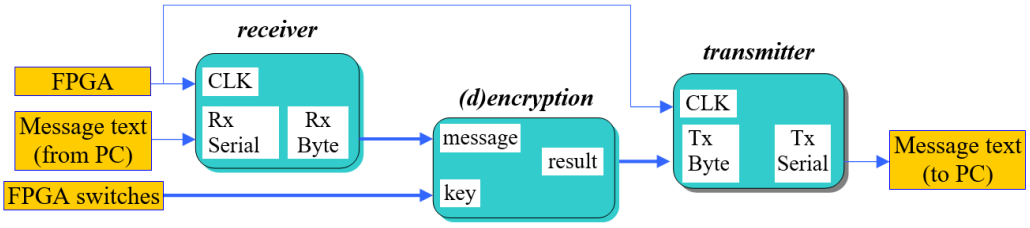


Figure 7. Hardware system block scheme.

Data from the FPGA chip must be gathered in order to evaluate the cryptosystem that has been built. The board uses a serial RS232 port to connect to the computer at a 9600 Baud rate. After character-by-character processing of the complete message, the FPGA transfers the encryption/decryption data to the PC for analysis. The current state of the eight switches on the board serves as the algorithm key.

The receiver module handles data reception from the computer in byte format and then it sends it further to the encryption/decryption module, which, together with the key, creates the processed message that is delivered to the transmitter module. The way the latter operates is by serializing the message and sending it back to the PC. Global clock (GCLK) sets the communication rate between PC and FPGA. The message to be processed is sent to FPGA from the software application by manually writing it in the appropriate field or by choosing a text file for longer messages. The application interface used to send messages to the FPGA board is presented in Figure 8 below.

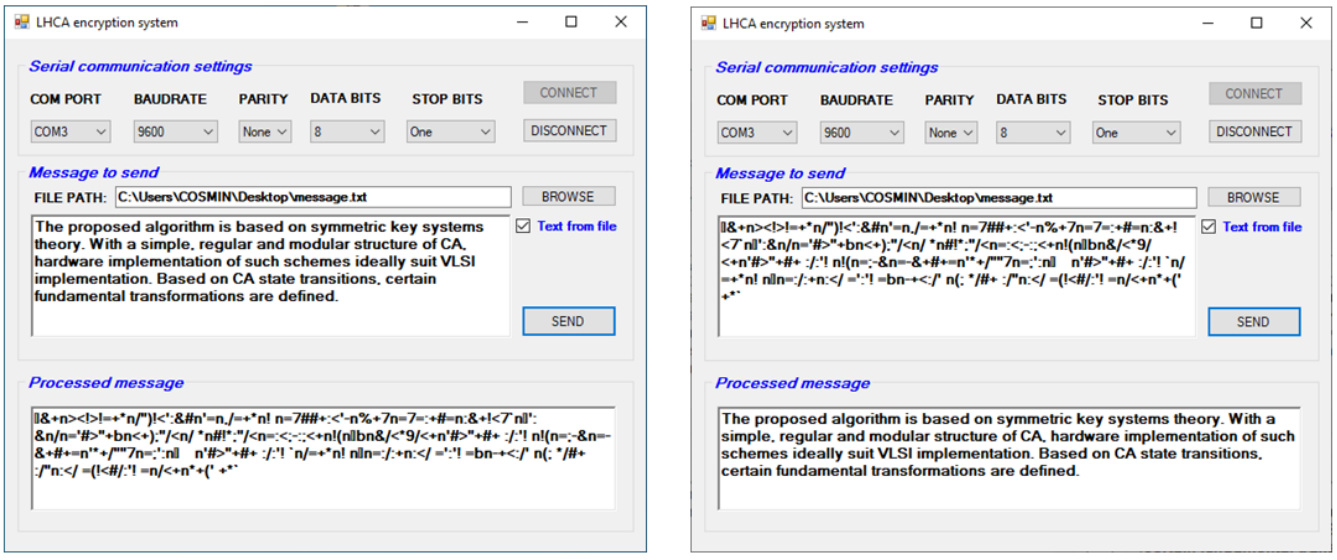


Figure 8. Interface with FPGA board: (**a**) Encryption process; (**b**) Decryption process.

The tests performed on the hardware system show a plain text sent to the FPGA system from the application to be encrypted. To check the correctness of the obtained result, the encrypted output data were fed back to the system for processing to test whether the original message could be reconstructed. After these steps, the plain text was obtained, proving the correct functionality of the hardware system.

Table 3 details the FPGA resources consumed and system timing for the experimental results of the hardware implementation. The data presented in the following table are based on an 8-bit system that includes modules for encryption, serial transmission, and reception.

Table 3. Overall system resource and time summary.

Resource Summary	
Slice Flip Flops	55/9312 (<1%)
4 input LUTs	147/9312 (<2%)
Occupied slices	86/4656 (<2%)
Timing Analyzer Summary	
Best case	8.138 ns
Worst case	14.314 ns
Clock setup	162.159 MHz (6.167 ns)

Time values from the above table are presented for the processing of one character in the message. The values are influenced by the key and the character to encrypt/decrypt. To test the functionality of the 8-bit LHCA used for encryption, different simulations were performed: simulation of its states evolution (Figure 9) and simulation of obtained results—encrypted data (Figure 10). The results are presented below as output waveforms.

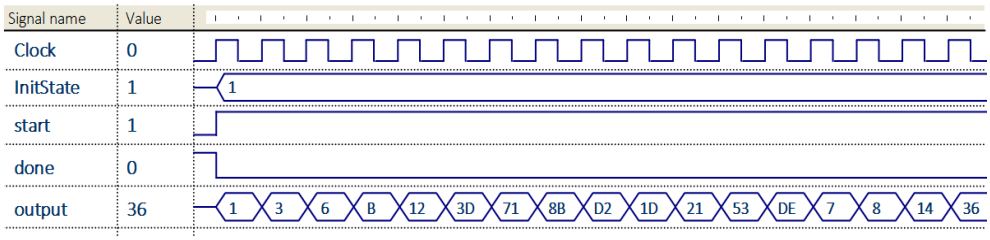


Figure 9. Output waveform for 8-bit 90/150 LHCA (decimal form).

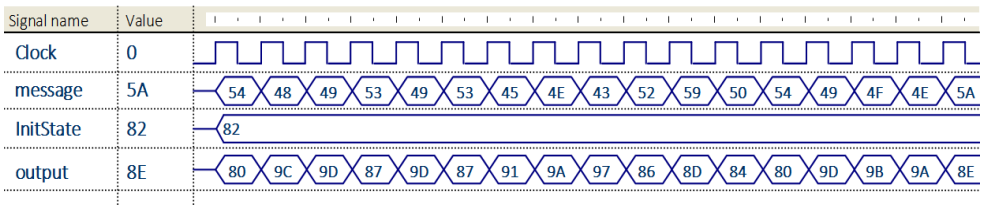


Figure 10. An 8-bit 90/150 LHCA output waveform for the encryption process (decimal form).

The entire cryptosystem developed in this work and the used FPGA are presented in Figure 11 below. The implementation in reconfigurable hardware makes the proposed cryptographic algorithm more efficient because of the parallelization obtained with the FPGA board and provides more security over the encryption technique due to the difficulty to obtain data from FPGA chip and hardware resources.

The hybrid cellular automata with the best statistical properties were analyzed, implemented, tested and evaluated in software and hardware (FPGA). Due to their properties, cellular automata are suitable for hardware implementation, and by using some instruction parallelization algorithms [29], increased speeds can be obtained for securing data transmitted in heterogeneous telecommunication networks.



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

Figure 11. (**a**) Entire cryptosystem; (**b**) FPGA board.

4. Testing and Results

This section focuses on testing the quality of the proposed encryption algorithm. Statistical tests, along with graphical representations, were performed to assess the randomness of the produced results, ending with a discussion about the cryptosystem and applications developed in this work.

For analyzing cryptographic algorithm results, we used a series of statistical tests provided by the National Institute of Standards and Technology (NIST) to check whether the produced sequence complies with the requirements of a pseudorandom number generator [30]. The NIST Test Suite is a collection of 15 statistical tests designed to assess the randomness of binary sequences generated by cryptographic random or pseudorandom number generators, regardless of whether they are hardware- or software-based. These tests are designed to detect various forms of non-randomness that may be present in a sequence.

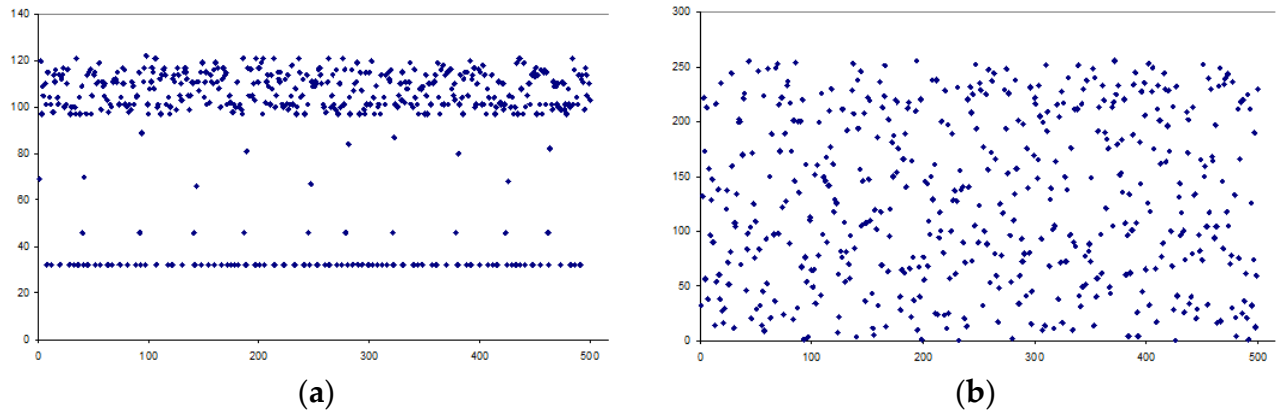
To evaluate the randomness of a sequence, a range of statistical tests can be employed. These tests are designed to identify non-randomness in the sequence and produce a test statistic value as a function of the data. Based on the test statistic value, a p-value is calculated, which represents the probability that a perfect random number generator would produce a less random sequence than the one tested. A significance level (α) is determined, and if the p-value is greater than or equal to α, the sequence is considered random, while a p-value of less than α indicates non-randomness.

In this work, α has been chosen to be 0.01. An α of 0.01 indicates that it is expected that one sequence in 100 sequences to be rejected. If a p-value is equal to or greater than 0.01, it implies that the sequence can be considered random with a confidence level of 99%. Each test has certain specifications for its usage. The most important of these is the input size recommendation that specifies the length in bits of the tested sequences. We used an encryption result of our proposed algorithm consisting of about 1,000,000 bits that were tested by splitting them into ten bitstreams. That means that 10 streams of 100,000 bits were tested. For each of the 15 tests, the software provides the p-value and the proportion of passing sequences (pass rate). Table 4 presents the results of the performed tests.

National Institute of Standards and Technology tests asses the most significant proprieties that may indicate that a sequence is potentially random, namely uniformity, scalability and consistency. The test results indicate that the sequence has the propriety of being possibly random. For better visualization of the obtained algorithm results, graphical representations were produced by analyzing the rate of diffusion of original text throughout the ASCII interval. Results are presented in Figure 12 below.

Table 4. Statistical testing results using NIST software.

NIST Statistical Test	*p*-Value	Proportion (Pass Rate)
Frequency (monobits)	0.534146	10/10
Block frequency	0.752134	9/10
Runs	0.350485	10/10
Longest run of one in a block	0.911413	10/10
Rank	0.739918	10/10
FFT	0.759145	10/10
Non-overlapping template	0.911413	9/10
Overlapping template	0.213309	10/10
Linear complexity	0.739918	10/10
Serial 1	0.213309	10/10
Serial 2	0.534146	10/10
Universal	undefined	undefined
Approximate Entropy	0.122325	10/10
Cumulative Sum (forward)	0.534146	10/10
Cumulative Sum (reverse)	0.350485	10/10



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

Figure 12. Dispersion of encrypted data over ASCII interval (**a**) Plaintext; (**b**) Ciphertext.

The graph above depicts the distribution of original data (left side) and encrypted data (right side) across ASCII intervals. It is worth noting that the dispersion of the encrypted data is consistent across all intervals, implying that encrypted text is dispersed almost evenly throughout all ASCII intervals, not only alphanumeric.

Another important aspect of the system consists in the number of iterations produced to generate the pseudorandom keystream sequence. The iterations are a crucial part of the cryptosystem meaning that their number affects the efficiency of the algorithm in terms of security provided by the system. No iterations or low number of iterations provides little or no efficiency in security of the system, thus a higher number of iterations are required to provide security. This is because the generated key stream is used to encrypt the plaintext, and if the key stream is not sufficiently unpredictable or large, it can be more susceptible to attacks.

In Table 5, we present an example consisting in a statistical analysis using NIST Tests on the system results produced using a small number of iterations in comparison to a higher number of iterations.

We can observe that performing a low number of iterations results in failure of the statistical tests, thus indicating that the produced sequence does not offer pseudorandom characteristics resulting in low security provided by the cryptosystem.

Therefore, while we prioritize efficiency in generating the pseudorandom numbers, we also agree that the number of iterations affects the overall efficiency of the cryptosystem in terms of security.

Table 5. Statistical testing of system with different number of iterations (NIST software).

NIST Statistical Test	3 Iterations		20 Iterations	
	p-Value	Pass Rate	*p*-Value	Pass Rate
Frequency (monobits)	0.000164	Failure	0.122325	10/10
Block frequency	0.000025	Failure	0.911413	9/10
Runs	0.000000	Failure	0.991468	10/10
Longest run of one in a block	0.000469	Failure	0.911413	10/10
Rank	0.000245	Failure	0.350485	10/10
FFT	0.000054	Failure	0.739918	10/10
Non-overlapping template	0.000000	Failure	0.035174	9/10
Overlapping template	0.000000	Failure	0.739918	10/10
Linear complexity	0.000047	Failure	0.213309	9/10
Serial 1	0.000000	Failure	0.534146	10/10
Serial 2	0.000000	Failure	0.122325	10/10
Universal	undefined	undefined	undefined	undefined
Approximate Entropy	0.000122	Failure	0.122325	9/10
Cumulative Sum (forward)	0.000000	Failure	0.534146	10/10
Cumulative Sum (reverse)	0.000000	Failure	0.122325	10/10

5. Discussion

According to NIST's guidelines, the minimum requirement for passing each statistical test is around 8 for a sample size of 10 binary sequences. Randomness is a characteristic that can be defined and described in terms of probability. Many statistical tests are available to determine the randomness of a sequence by detecting the presence or absence of patterns. If a pattern is detected, it indicates that the sequence is not random. However, since there are numerous tests available to evaluate randomness, no particular finite set of tests is considered to be "complete." Concerning that and the above results, we can state that our algorithm produces potentially random data.

Various types of stream ciphers are currently in use across different networks, many of which were created as part of the eSTREAM initiative. Among the most popular ones are Grain and Trivium, which feature distinct specifications such as key length. Subsequent to these designs, a group of stream ciphers based on cellular automata (CA) were developed, including CAvium, CAR30, and FResCA, among others. CAvium is a modification of Trivium that relies on a hybrid CA to replace Trivium's shift registers, incorporating a range of rules (30, 60, 90, 120, 150, 180, 210, 240) to enhance the nonlinearity of keystream generation.

CAR30 is another stream cipher based on cellular automata, featuring a three-neighborhood CA rule that draws inspiration from Grain-128. This cipher serves as a pseudorandom number generator in a modified version of the Grain algorithm. FResCA is another CA-based stream cipher, also inspired by Grain, but with a distinct design. While other CA-based stream ciphers rely solely on three-neighborhood CA, FResCA uses a hybrid approach that combines both three- and four-neighborhood CA. A more recent CA stream cipher algorithm, known as CARPenter and described in [31], employs a non-linear mixing function and combines a non-linear CA with a maximum length null-boundary linear CA.

All the stream ciphers mentioned above except CAvium had their inspiration from the design of Grain-128, a lightweight stream cipher that had high throughput and fast initialization, which is still developed during the lightweight stream cipher competition initiated by NIST in response to the increasing number of connected IoT devices.

In comparison to these existing ciphers, our algorithm has a relatively low computational complexity due to the simplicity of the rules used, which allows for easy implementation logic in hardware structures without usage of numerous resources. This results in faster encryption and decryption times. A comparison between our proposed scheme of encryption and other similar ones is presented in Table 6.

Table 6. Comparison with related stream ciphers.

Encryption Scheme	Grain	CAvium	CAR30	CARpenter	Our Algorithm
LUTs used	278	562	936	354	147
Keystream size	256	256	256	256	256

Due to differences in implementation, the values depicted in the table may exhibit minor variations. Nevertheless, the results point out the advantages of our cipher, which notably reduces the number of required rules and the radius of the neighborhood while maintaining the same properties as other ciphers. This represents a key highlight of our approach.

Another advantage a relatively small key size, which makes it easier to transmit and store. An attack on the system assumes not only finding the initial state of the system to decipher the message, but also the number of iterations performed from that state through the LHCA. Therefore, even if the proposed system only uses 8-bit LHCA to perform encryption and decryption, it provides security through the two components—the initial state of the CA and the number of evolution steps—that make up the system secret keys.

6. Conclusions

This paper demonstrated data encryption and decryption solutions using linear models based on cellular automata theory with a combination of rules 90 and 150. It is interesting that these encryption techniques based on discrete, dynamical systems with simple evolution rules which employ keystream sequences generated by pseudorandom generators have a direct cryptographic application in stream ciphers.

Since the hardware implementation of the established CA-based models is simple and particularly suitable for reconfigurable hardware logic (e.g., FPGA), the encryption approach is suitable for cryptographic systems in various practical applications. It is suited for innovations where time execution is essential, such as in communication systems with high transmission rates. Still, it could also be integrated into most of telecommunication networks.

The cryptosystem developed in this work will be enhanced so that the CA may operate with 16 or 32 cells. Due to the intrinsic properties of the CA, a 16-bit or 32-bit CA can still have a simple structure while operating the same way (e. g., all CA cells are updated at the same time, and this is possible in a single clock). Furthermore, we will focus on incorporating this encryption algorithm into numerous applications (e.g., telemedicine) and data communication channels, such as the Internet. Another research extension to this system focuses on improving the speed of encryption by integrating data directly into LHCA.

Author Contributions: The authors contributed equally to this work, from conceptualization of the problem to the software and hardware implementations that were used to analyze and test our proposed solution and to the funding acquisition that supported this research. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI–UEFISCDI, project number PN-III-P2-2.1-PED-2021-3669, within PNCDI III, contract number 693PED/28.06.2022.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: We would like to acknowledge the usage of the NIST Statistical Test Suite as a contribution to this work. The software was used to test our proposed algorithm output against the random number generator theory.

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

References

1. Von Neumann, J. *Theory of Self-Reproducing Automata*; University of Illinois: Urbana, IL, USA, 1966.
2. Sipper, M. *Evolution of Parallel Cellular Machines—The Cellular Programming Approach*; Springer: Berlin/Heidelberg, Germany, 2004.
3. Cho, S.J.; Choi, U.S.; Kim, H.D.; Hwang, Y.H.; Kim, J.G. Analysis of 90/150 Two Predecessor Nongroup Cellular Automata. In Proceedings of the Cellular Automata, 8th International Conference on Cellular Automata for Research and Industry (ACRI '08), Yokohama, Japan, 23–26 September 2008; Springer: Berlin/Heidelberg, Germany, 2008. [CrossRef]
4. Wolfram, S. Random Sequence Generation by Cellular Automata. *Adv. Appl. Math.* **1986**, *7*, 123–169. [CrossRef]
5. 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]
6. Anghelescu, P. Parallel Optimization of Program Instructions Using Genetic Algorithms. *Comput. Mater. Contin.* **2021**, *67*, 3293–3310. [CrossRef]
7. Aumasson, J.P. *Serious Cryptography*; No Starch Press: San Francisco, CA, USA, 2018.
8. Menezes, A.; van Oorschot, P.; Vanstone, S. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, MA, USA, 1996.
9. Wolfram, S. Cryptography with Cellular Automata. In Proceeding of the Advances in Cryptology: Crypto '85, Linz, Austria, 9–11 April 1985. [CrossRef]
10. Tomassini, M.; Perrenoud, M. Stream Ciphers with One and Two-Dimensional Cellular Automata. In Proceedings of the Parallel Problem Solving from Nature—PPSN VI, Paris, France, 18–20 September 2000. [CrossRef]
11. Seredynski, F.; Bouvry, P.; Zomaya, A.Y. Cellular Programming and Symmetric Key Cryptography Systems. In Proceedings of Genetic and Evolutionary Computation—GECCO 2003, Chicago, IL, USA, 12–16 July 2003. [CrossRef]
12. Kumar, A.; Raghava, N.S. An efficient image encryption scheme using elementary cellular automata with novel permutation box. *Multimed. Tools Appl.* **2021**, *80*, 21727–21750. [CrossRef]
13. Anghelescu, P.; Stirbu, C. Cellular automata based algorithm for image density classification task. In Proceedings of the 2014 6th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Bucharest, Romania, 23–25 October 2014. [CrossRef]
14. Hortensius, P.; McLeod, R.; Card, H. Parallel random number generation for VLSI systems using cellular automata. *IEEE Trans. Comput.* **1989**, *38*, 1466–1473. [CrossRef]
15. Nandi, S.; Kar, B.; Chaudhuri, P.P. Theory and applications of cellular automata in cryptography. *IEEE Trans. Comput.* **1994**, *43*, 1346–1357. [CrossRef]
16. Wolfram, S. *A New Kind of Science*; Wolfram Media: Champaign, IL, USA, 2002.
17. Anghelescu, P.; Ionita, S.; Sofron, E. Encryption Technique with Programmable Cellular Automata (ETPCA). *J. Cell. Autom.* **2010**, *5*, 79–105.
18. Treml, L.M.; Bartocci, E.; Gizzi, A. Modeling and Analysis of Cardiac Hybrid Cellular Automata via GPU-Accelerated Monte Carlo Simulation. *Mathematics* **2021**, *9*, 164. [CrossRef]
19. Wang, W.; Zhou, Y.; Liu, J.; Sun, B. On-Street Cruising for Parking Model in Consideration with Gaming Elements and Its Impact Analysis. *Mathematics* **2022**, *10*, 3423. [CrossRef]
20. Anghelescu, P.; Ionita, S.; Sofron, E. FPGA Implementation of Hybrid Additive Programmable Cellular Automata Encryption Algorithm. In Proceedings of 8th International Conference on Hybrid Intelligent Systems, Barcelona, Spain,, 10–12 September 2008. [CrossRef]
21. Angulo, K.V.; Gil, D.G.; Espitia, H.E. Modeling and Numerical Validation for an Algorithm Based on Cellular Automata to Reduce Noise in Digital Images. *Computers* **2022**, *11*, 46. [CrossRef]
22. Perestrelo, S.A.; Grácio, M.C.; Ribeiro, N.D.A.; Lopes, L.M. A Multi-Scale Network with Percolation Model to Describe the Spreading of Forest Fires. *Mathematics* **2022**, *10*, 588. [CrossRef]
23. Li, X.-W.; Kim, S.-T.; Wang, Q.-H. Designing Three-Dimensional Cellular Automata Based Video Authentication With an Optical Integral Imaging Generated Memory-Distributed Watermark. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 1200–1212. [CrossRef]
24. Nanda, S.K.; Mohanty, S.; Pattnaik, P.K.; Sain, M. Throughput Optimized Reversible Cellular Automata Based Security Algorithm. *Electronics* **2022**, *11*, 3190. [CrossRef]
25. Devipriya, M.; Sreenivasan, M.; Brindha, M. Reconfigurable Architecture for Image Encryption Using a Three-Layer Artificial Neural Network, IETE J. *Res.* **2022**, *2022*, 1–14. [CrossRef]
26. Anghelescu, P.; Sofron, E.; Ionita, S. VLSI Implementation of High-Speed Cellular Automata Encryption Algorithm. In Proceedings of International Semiconductor Conference (CAS), Sinaia, Romania, 15 October–17 September 2007. [CrossRef]
27. Anghelescu, P. FPGA implementation of programmable cellular automata encryption algorithm for network communications. *Comput. Syst. Sci. Eng.* **2016**, *31*, 361–370.
28. Harshala, P.; Shyam Sunder, P. Cellular automata for built in test pattern generation and test response analyzer. *Int. J. Electr. Electron. Res. 2* **2014**, *2*, 44–50.
29. Dascălu, M. Cellular Automata Hardware Implementations—An Overview. *Sci. Technol.* **2016**, *19*, 360–368.

30. 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.
31. John, A.; Lakra, R.; Jose, J. On the design of stream ciphers with Cellular Automata having radius = 2. *Cryptol. Eprint Arch.* **2020**, *2020*, 327.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.