Natural Computing
https://doi.org/10.1007/s11047-023-09941-6

A reversible system based on hybrid toggle radius-4 cellular automata and its application as a block cipher

Everton R. Lira[1] · Heverton B. de Macêdo[2] · Danielli A. Lima[3] · Leonardo Alt[4] · Gina M. B. Oliveira[1]

Accepted: 23 February 2023

Abstract
The one-dimensional cellular automata (CA) system detailed herein uses a hybrid mechanism to attain reversibility, and this approach is adapted to create a novel block cipher algorithm called HCA (Hybrid Cellular Automata). CA are widely used for modeling complex systems and display inherently parallel properties. Therefore, applications derived from CA have a tendency to fit very well in the current computational paradigm where multithreading potential is very desirable. The HCA system has recently received a patent by the Brazilian agency INPI. Analyses performed on the model are presented here, including a theoretical discussion on its reversibility. Finally, the cryptographic robustness of HCA is empirically evaluated through avalanche property compliance and the NIST randomness suite.

Keywords Cellular automata · Reversibility · Cryptography · Block cipher

Mathematics Subject Classification 68Q80 · 94A60

1 Introduction

Cryptography is the everlasting study of methods that ensure confidentiality, integrity and authentication, when storing or transmitting data, to minimize security vulnerabilities. In such approaches, data is codified in a specific way so that only those for whom it is intended can read and process it.

Despite the successful application of reputed algorithms, such as AES (Daemen and Rijmen 2002) and RSA (Rivest et al. 1978), the evolution of hardware architectures imposes an ongoing race to develop more secure and effective encryption models. For example, with the popularization of portable electronic devices able to capture digital images, the exchange of such data between entities on private social networks or e-mails became more frequent, which led to the demand for methods that enable high throughput without loss of security.

Since the most popular symmetric encryption algorithms AES and DES (FIPS 1999) are of serial nature, this poses a challenge to massive data processing (Daemen and Rijmen 2005; Zeghid et al. 2007). This motivated a search for improving these classical algorithms (Prasad and Maheswari 2013) as an attempt to introduce parallelism on some costly or redundant steps in the process of encryption (Le et al. 2010). However, as they are inherently sequential systems, this customization is limited and does not allow the desirable level of parallelism to be reached. As such, the capacity of high-performance parallel architectures can become underutilized. In this context, cellular automata (CA) appear as a useful tool in the design of inherently parallel encryption systems.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The authors would also like to thank Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Fundação de Amparo a Pesquisa do Estado de Minas Gerais (Fapemig) for supporting this work.

✉ Everton R. Lira
evertonlira@gmail.com

[1] Computer Science Department, Federal University of Uberlândia, UFU, MG, Uberlândia, Brazil

[2] Computer Science Department, Goiano Federal Institute, IF Goiano, Rio Verde, GO, Brazil

[3] Informatics Department, Federal Institute of Triângulo Mineiro, IFTM, Patrocínio, MG, Brazil

[4] Berlin, Germany

Published online: 25 March 2023

CA are totally discrete mathematical models based on the livelihood of cellular organisms, a naturally occurring process which dictates the survival of such cells based on their behavior (implemented as CA rules) while interacting with the environment conditions they are exposed to (the cell neighborhood) (Rozenberg et al. 2012). CA are widely used in the literature (Sarkar 2000; Dennunzio et al. 2019) and, among the most known applications, the following can be mentioned: (i) modeling of biological and physical systems (Vichniac 1984; Ermentrout and Edelstein-Keshet 1993; Maerivoet and De Moor 2005; Alizadeh 2011; Ghimire et al. 2013; Feliciani and Nishinari 2016; Mattei et al. 2018; (ii) investigation of new computational paradigms (Hillis 1984; Lent et al. 1993; Morita 2008; Yilmaz 2015; (iii) proposition of tools for solving various computational problems, such as task scheduling (Swiecicka et al. 2006; Carneiro and Oliveira 2013; Carvalho et al. 2019), image processing (Rosin 2010), computational tasks (Mitchell 2005; Oliveira et al. 2009), robotics (Ioannidis et al. 2011; Lima and Oliveira 2017) and, more significantly related to this article, cryptographic models (Wolfram 1986; Gutowitz 1995; Sen et al. 2002).

The implementation simplicity and the ability to process data in parallel are some of the main advantages of applying CA-based models in the most diverse areas mentioned above (Vasantha et al. 2015). In addition, the discovery that even the simplest CA models, known as elementary, are capable of exhibiting chaotic-like dynamics (Wolfram 1986; Dennunzio et al. 2020) led researchers to see CA-based models as natural options for proposing fast, parallel and potentially secure encryption methods (Wolfram 1986; Gutowitz 1995; Sen et al. 2002).

A novel cryptographic model called HCA (Hybrid Cellular Automata), based on chaotic one-dimensional CA rules, is described herein. The HCA model recently received a patent registration in Brazil (Oliveira and Macêdo 2019), and this paper presents an unique detailed view on how the parameters of HCA were defined and on the investigations done to measure its cryptographic robustness.

This model employs preimage computation (the backward evolution of a CA configuration, defined in the next section) in the encryption process and applies one-dimensional permutive CA rules, the so-called toggle rules (which will be detailed in the following section), such as the method proposed by Gutowitz (Gutowitz 1995). Furthermore, this innovative method also addresses two problems of that approach (Gutowitz 1995): the spread of plaintext disturbances in only one direction and the block length increase during the successive preimage computations done in the encryption process.

Even though later models, also based on CA toggle rules, sought to reduce these problems (Wuensche 2008; Oliveira et al. 2004, 2008, 2010b; Silva et al. 2016), the solution proposed here is the only one that ensures an appropriate propagation of the disturbance over the entire lattice, as well as keeping the size of the ciphertext the same as the plaintext.

This paper is structured as follows: Sect. 2 brings an overview of CA concepts that are relevant to understand the HCA model description and when comparing it to predecessor systems. Section 3 presents a review of the main works in the literature related to the novel investigated method. Section 4 formally presents the HCA model and details all of the processes involved in its proposition. Section 5 presents a theoretical aspect related to HCA: the proof that the hybrid CA model used in HCA is reversible, unlike the model used by Gutowitz (Gutowitz 1995) that is irreversible. Section 6 describes methods established in the literature to verify the security of a cryptographic method. Experimental results obtained for them are presented in Sect. 7 for the validation of HCA security against cryptanalysis attacks: plaintext avalanche effect, key avalanche effect and NIST suite tests. Finally, Sect. 8 presents our main conclusions on the novel HCA cryptographic method and proposes some future directions to this research.

2 Formal concepts

A brief recollection of necessary formalism and definitions from the classic CA model is important before comparing the novel HCA system to other predecessor methods and describing its inner mechanism.

2.1 Basic CA model definitions

Since HCA is based on the one-dimensional CA specification, unless directly stated, consider CA references in this paper as adhering to this model.

Consider a set s of N cells. In an one-dimensional CA setting, s can be visually represented as a row of N neighboring cells. This row of cells is generally called a CA grid or a CA lattice, and each cell, in this s lattice, has its 0-based i-indexed position, where $0 \leq i < N$ as displayed in Fig. 1.

CA is a discrete time model, meaning every $s[i]$ cell has a state for each time step t. In classic CA, the state (or value) of a $s^t[i]$ cell is binary in nature. Therefore, it can be expressed as $s^t[i] \in \{0, 1\}$, a Boolean value. Also, the set of all lattice cell values at a certain time step, s^t, is called a

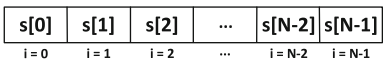

Fig. 1 One-dimensional CA lattice visual representation

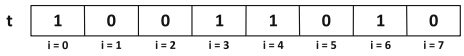

Fig. 2 One-dimensional CA configuration example

CA configuration. Figure 2 demonstrates one possible configuration for a CA grid of $N = 8$ cells.

The forward evolution of a s^t configuration, such as the one displayed in Fig. 2, is the process by which one or more evolution rules, which can be represented as Boolean functions, are applied to cell values from s^t resulting in a new s^{t+1} successor configuration. The value of every $s^{t+1}[i]$ cell is computed using a local evolution rule applied to the values of $s^t[i]$ and its neighboring cells, as will be further explained.

The arbitrary specification of a neighborhood radius directly affects the CA evolution process. For instance, if CA evolution will be processed considering a $r = 1$ neighborhood radius, the computation of cell $s^{t+1}[i]$ will be based on cell $s^t[i]$ and on both neighboring cells that are at a $r = 1$ index distance from it in the grid, which are $s^t[i-1]$ and $s^t[i+1]$. Likewise, when considering $r = 2$, the computation of $s^{t+1}[i]$ is based on the $(s^t[i-2], s^t[i-1], s^t[i], s^t[i+1], s^t[i+2])$ neighborhood, and so on.

2.2 Boundary condition

Some edge cases must be considered when the neighborhood set contains cells whose computed indexes are either negative or greater than $N - 1$. Such cases are treated by a boundary condition that is arbitrarily defined by the model designer and which, for this paper, unless explicitly stated, is the setting called 'periodic boundary condition'.

By this definition, despite the CA grid being visually represented as a row, it will be treated as a toroidal structure when computing the neighborhood of cells. So the first cell of the grid ($s^t[0]$) has the last cell of the grid ($s^t[N-1]$) as its left neighbor and, in a corresponding manner, this last cell of the grid ($s^t[N-1]$) has the first cell ($s^t[0]$) as its right neighbor. This system design choice allows a simple 1-bit disturbance to be propagated throughout the entire grid as CA evolution is performed, regardless of the position where this perturbation first occurs.

2.3 Wolfram's notation

Since the value of every $s^{t+1}[i]$ cell is computed using the values of a $(2 \times r + 1)$-sized neighborhood comprised of $s^t[i]$ and r neighboring cells from both sides, and due to the Boolean nature of cell values, local evolution rules for this CA must define a resulting value (output) for every one of the $2^{2\times r+1}$ possible combinations of input values the previously mentioned set could assume.

Table 1 Boolean table for radius-1 Rule 30

$s^t[i-1]$	$s^t[i]$	$s^t[i+1]$	$s^{t+1}[i]$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

In fact, due to being mathematical functions, these local CA rules can be expressed using many conventional notations, such as Boolean functions or graphs. In this paper, a special notation for CA rules created by Wolfram will also be relevant (Wolfram 2002). Wolfram's notation is made by concatenating, in descending order, the resulting bits from a truth table representing the Boolean function of the CA rule. For instance, consider Table 1 for an arbitrary $r = 1$ CA rule.

In Wolfram's encoding, the CA rule expressed through Table 1 is called rule 30. This is due to the concatenation of the $2^{2\times r+1}$ resulting bits, in reverse order, being the '00011110' string which, in binary, represents the decimal number 30. This method of encoding CA rules as binary strings is vastly used in this paper, especially since the main CA rules applied in the HCA system are derived from the input encryption key through a process that will be explained in later sections.

2.4 Hybrid/heterogeneous cellular automata

When designing CA systems, there are two alternatives regarding rules used in CA evolution. The first option is using a single local CA rule to evolve all lattice cells, the latter is using distinct local CA rules to evolve separate regions (sites) of the CA grid. This second option of applying potentially different rules to non-overlapping sections of the lattice causes the CA in question to be called 'hybrid', 'heterogeneous', or 'non-uniform' (Dennunzio et al. 2014).

As described in (Dennunzio et al. 2012), non-uniform CA can present drastic changes in their dynamical behavior when compared to conventional CA. And thus, there are instances where adopting hybrid CA is considered a helping feature, such as pseudo-random number generation, which has close ties to cryptography (Dennunzio et al. 2013).

Consider, for instance, a s^t lattice comprised of $N = 8$ cells, and two distinct CA rules denoted by ϕ_m and ϕ_b. In a

hybrid CA setting, it could be arbitrarily established that cells $s^t[0]$, $s^t[1]$, $s^t[2]$, $s^t[3]$, $s^t[4]$ and $s^t[5]$ are to be evolved using ϕ_m while cells $s^t[6]$ and $s^t[7]$ are to be evolved using ϕ_b. In fact, the HCA method, to be further detailed in Sect. 4, employs a hybrid CA strategy similar to this.

2.5 Permutivity and toggle rules

CA rules are often classified according to global evolution patterns that can be visually observed in CA lattices they are applied to. Other approach is the classification based on mathematical properties they express. Some CA rules display a property called 'permutivity' (Dennunzio et al. 2014), which is relevant to the HCA system description.

As previously established, any one-dimensional CA rule ϕ is a Boolean function that maps every possible m-sized neighborhood word w (input) to a corresponding bit result value (output). Furthermore, this input can be expressed using a 1-based index $p \mid 1 \leq p \leq m$, like $(w[p=1], w[p=2], \ldots, w[p=m])$, where the resulting output bit is obtained by applying ϕ to it, as $\phi(w)$.

Now, considering every possible w, for any p index, either $w[p]=0$ or $w[p]=1$, due to the Boolean nature of the cell values in the CA grid. Let us denote, for simplicity, that $w(0, p)$ represents the variant of w where $w[p]=0$, and that $w(1, p)$ represents the variant of w where $w[p]=1$. For instance, consider $m=5$, and also arbitrarily, $w=(0,0,1,1,0)$. In this case, for $p=1$, $w(0,p=1)=(0,0,1,1,0)$ and $w(1,p=1)=(1,0,1,1,0)$. Meanwhile, for $p=3$, $w(0,p=3)=(0,0,0,1,0)$ and $w(1,p=3)=(0,0,1,1,0)$, and so on.

According to (Leporati and Mariot 2014), a CA rule ϕ is 'p-permutive' if, for every possible neighborhood word w, it holds that $\phi(w(0,p)) \neq \phi(w(1,p))$. This definition is relevant since both the novel HCA method, which will be defined in Sect. 4, and a predecessor system (Gutowitz 1995), that is described in Sect. 3, employ permutive rules under a different naming scheme 'toggle rules'. A leftmost permutive rule (1-permutive rule) can be also described as a 'left-toggle' rule, while a rightmost permutive rule is called a 'right-toggle' rule. This naming convention will be the one used in future sections for simplification.

Moreover, leftmost and rightmost toggle rules display unique characteristics. Take, for instance, a left-toggle CA rule ϕ and consider any w input neighborhood. Since ϕ is a leftmost permutive rule, for any w: $\phi(w(0,p=1)) \neq \phi(w(1,p=1))$. This also means the output bit $\phi(w(0,p=1))$ is the binary complement value of $\phi(w(1,p=1))$, represented as $\phi(w(0,p=1)) = \overline{\phi(w(1,p=1))}$.

Therefore, in a Boolean table for a left-toggle CA rule, the sequence of output values for inputs where $w[1]=0$ is the exact Boolean complement of the output values sequence for inputs where $w[1]=1$. And thus, for left-toggle rules, the second half of the corresponding Wolfram notation string can be inferred as the binary complement of the first half. Since rule 30 is leftmost permutive, Table 1 exemplifies such behavior, and the first half ('0001') of the corresponding Wolfram notation ('00011110') can be used as a compressed notation from which the rule can be easily derived. This holds, by definition, for every left-toggle rule, regardless of the neighborhood radius.

2.6 Preimage and reversibility

Finally, while CA evolution generally indicates a forward evolution procedure (denoted here as Φ), there also exists a CA backward evolution which represents its mathematical inverse (Φ^{-1}) function. And thus, while $\Phi(s^t)=s^{t+1}$, its inverse function is $\Phi^{-1}(s^{t+1})=s^t$ and s^t is considered a preimage of s^{t+1}.

Depending on the rule definition of the CA, there may exist configurations that are not reachable by forward evolution. They are called 'Garden of Eden' and, thus, it is not possible to determine a preimage for them (Moore 1962). On the other hand, there might also be cases in which a s^{t+1} configuration has more than one possible preimage.

By definition, a CA can only be considered reversible if every configuration can be reached by forward evolution from any other configuration and if, for each configuration, there is a single possible preimage. In this case, the preimage computation operation is a deterministic function, and CA backward evolution (Φ^{-1}) is always possible. The novel HCA model described herein is based on this backward evolution operation, as will be detailed in Sect. 4, and the reversibility of HCA will be formally proven in Sect. 5.

3 Related work

The first suggestion of using cellular automata models in cryptography was made by Wolfram (Wolfram 1985), after his studies on the statistical properties of CA chaotic rules with radius 1, which can be used as pseudo-random number generators (Wolfram 1986). Since then, various works on this topic have been published (Tomassini and Perrenoud 2000; Sen et al. 2002; Vasantha et al. 2015; Oliveira and Macêdo 2019; Benkiniouar and Benmohamed 2004; Nandi et al. 1994; Gutowitz 1995; Oliveira et al. 2004, 2008, 2010b; Silva et al. 2016; Wolfram 1985; Wuensche 2008; Oliveira et al. 2010a; Wuensche and Lesser 1992; Seredynski et al. 2004; Yang et al. 2016), where the cryptographic models can be classified into three kinds of approaches.

The first approach, proposed by Wolfram, takes advantage of the pseudo-random properties of known transition rules with chaotic behavior to generate random binary sequences. However, said rules are not used as the cryptographic key, which, in fact, corresponds to the initial lattice. This lattice is evolved by a predetermined chaotic rule (elementary rule 30) and the sequence of bits generated in a specific cell position is used as a pseudo-random sequence. Moreover, the effective ciphering process is made by a reversible function that mixes the plaintext with the random sequence, such as the XOR logical function (Wolfram 1985, 1986; Tomassini and Perrenoud 2000; Benkiniouar and Benmohamed 2004; Nandi et al. 1994). On the other hand, the HCA model discussed herein uses transition rules as secret keys and the plaintext as a initial lattice. More recently, this first approach was diversified, for example, by using different one-dimensional transition rules with radius 1 and 2 and also two-dimensional rules, and using evolutionary search for finding suitable chaotic rules (Seredynski et al. 2004; Tomassini and Perrenoud 2001; Sirakoulis 2016; Toffoli and Margolus 1957; Kari 1992; Machicao et al. 2012; John et al. 2020). Another line of investigation is the parallelization of cellular automata as pseudo-random number generators that can be applied in cryptographic schemes (Sirakoulis 2016).

The second approach is based on additive, hybrid and reversible CA rules. The cryptographic keys are typically a combination of known additive rules (Toffoli and Margolus 1957) that exhibit algebraic properties. When such rules are used together in a heterogeneous scheme, they exhibit a periodic dynamics with maximum and/or known cycle (Nandi et al. 1994; Kari 1992; Dennunzio et al. 2021). However, the parallelism and security of these models are limited, due to the additive property of the rules, which limits their chaoticity. The system proposed in Nandi et al. (1994) was broken in Blackburn et al. (1997) by analyzing the additive properties of the rules. More recently proposed systems based on this line of research have been mixing additive rules and nonlinear rules to circumvent the security problems of their predecessors (Das and Chowdhury 2010).

The last approach uses the backward evolution of the CA lattice to encrypt the plaintext. The cryptographic key is the CA transition rule and it must have some properties to ensure the preimage existence (Oliveira et al. 2008; Wuensche 2008; Oliveira et al. 2010a; Wuensche and Lesser 1992). Gutowitz was the first to propose a cryptographic model using such approach; it is based on the preimage computation of irreversible homogeneous CA (Gutowitz 1995). The cryptographic model discussed here also uses the backward evolution. However, in the novel HCA method, reversibility is attained by using a hybrid CA setting to ensure the existence of a single same-sized preimage s^{t+1} with the same length of the image for each possible s^t configuration. Due to similarities, we further detail the state of the art related to CA-based models that belong to the third approach.

Gutowitz's model employs CA toggle rules, which are used as cryptographic keys (or a part of these). Such rules are sensitive to the leftmost and/or to the rightmost cell in the neighborhood. That means any modification to the state of this cell necessarily causes a modification on the central cell. A preimage of an arbitrary lattice of size N is calculated adding R extra bits to each side and a preimage will be calculated with N + 2R cells. If a right-toggle rule transition is used as key, the preimage cells can be obtained in a deterministic way, step-by-step, from the leftmost side to the right Gutowitz (1994).

In Gutowitz's model, the plaintext corresponds to the initial lattice and P preimages are calculated to obtain the ciphertext. As 2R bits are added to each preimage calculated, the size of the final lattice is given by N + 2RP. Such non-negligible increment is pointed as the major drawback of this model. Moreover, another flaw was identified in it, a high degree of similarity between ciphertexts was observed when the plaintext is submitted to a s perturbation. To deal with this problem, the model employs two phases where a left-toggle and a right-toggle rule are applied in each stage. Both rules are generated starting from the same cryptographic key, however, it needs more time steps to encrypt the plaintext. Later on, this model was altered by using bidirectional toggle CA rules (to the right and to the left simultaneously) in Oliveira et al. (2004), showing that the similarity flaw was solved with such a modification and that it is protected against differential cryptanalysis. However, the ciphertext length increment, when compared to the plaintext, remains in this model.

An algorithm known as 'reverse algorithm' was proposed in Wuensche and Lesser (1992) for a preimage computation starting from any lattice and applying an arbitrary transition rule (not only toggle rules). However, using a periodic boundary CA, the preimage computation is concluded verifying whether the initial bits can be equal to the final 2R rightmost ones. If so, the extra bits are discarded returning the preimage to the same size of the original lattice. If no, this preimage does not exist. This algorithm finds all the possible preimages for any arbitrary periodic boundary lattice, if at least one exists.

This reverse algorithm was evaluated as an encryption method in Oliveira et al. (2008) and Wuensche (2008). However, since there is no guarantee of preimage existence for all possible rule transitions, the major challenge these predecessor models faced was selecting rules that ensure the existence of at least one preimage for any possible lattice. An attempt to solve this problem was to use the Z parameter (Silva et al. 2016) in the rule specification.

The method proposed in Wuensche (2008) is very similar to the initial method proposed in Oliveira et al. (2008), despite being developed independently. The major conclusion in Wuensche (2008) is that the simple adoption of the reverse algorithm is not viable because the possible rules with 100% guarantee of preimage existence are not appropriate for ciphering, even when using the Z parameter to choose suitable secret keys. No treatment to this problem was addressed in Oliveira et al. (2008), that is, how to proceed if a failure occurs when computing preimages. This is an important distinction between the (Wuensche 2008) and Oliveira et al. (2008) works.

An alternative approach to use the reverse algorithm by adopting a wrap (contour) procedure was later investigated in Oliveira et al. (2010a). This contour procedure ensures any plaintext can be encrypted. However, it generates a variable size ciphertext, which can be larger than the plaintext. Later on, it was shown that an improved specification of the secret key reduces the occurrence rate of this failure (Oliveira et al. 2010b), which keeps the ciphertext length close to the plaintext. This specification was further investigated in Oliveira et al. (2010c) and Oliveira et al. (2011).

Additionally, a cryptographic model that employs a lattice with a fixed extra boundary was investigated in Silva et al. (2016), which applies the a variation of the reverse algorithm proposed by Wuensche. Even though the lattice size growth is less than in Gutowitz's model, the final lattice is still larger than the plaintext, which increases the cost of sending encrypted information, in addition to the aperiodic condition of the lattice hindering the good propagation of disturbances.

On a side note, even though producing a ciphertext larger than the plaintext is generally seen as a negative characteristic in symmetric cryptography, it is worth mentioning the existence of other cryptographic applications where having a preimage computation which increases the lattice size can be seen as beneficial. One such example is Mariot and Leporati (2014), where the text increase allows generating a sufficiently long configuration to be split among the participants of a secret sharing scheme.

As far as we know, the HCA model described herein is the first that simultaneously displays the following properties: uses backward evolution with chaotic toggle rules, preimage computation is valid for every possible configuration, and the ciphertext maintains the same size of the plaintext (using a periodic boundary condition). HCA was initially proposed in Macêdo (2007) and a patent registration was submitted to the Brazilian agency of patents (INPI) in 2007, which has been recently accepted in 2019 Oliveira and Macêdo (2019).

Meanwhile, other academic works have investigated different aspects of HCA and propose some adaptations of this CA-based model (Magalhães Júnior 2010; Lima 2012; Alt 2013). Some of the analyses over HCA investigated in these works are presented here. More recently, a new model inspired by HCA was proposed, replacing the cellular automata structure by complex networks connections (Barros de Macedo et al. 2014). In spite of some advantages related to the fast propagation of information promoted by non-local connections, the intrinsic parallelism of CA models is not presented in the model based on complex networks.

4 HCA method description

The HCA method consists of a symmetric block-based cryptographic system that uses the dynamic behavior of CAs to perform the encryption and decryption processes. Both forward and backward evolution of CAs are essential parts of this algorithm.

The original patented description of this system provides support for using right-toggle or left-toggle rules at will, but the focus of this current description will be on using left-toggle rules only, for simplicity.

4.1 HCA - block size definition

In HCA, 128-bit blocks are used for the encryption and decryption processes. The method could be easily adapted to be used with other block sizes, but this value was set to conform with the current standard for symmetric cryptography methods.

Like all block cipher methods, the HCA method is compatible with every mode of operation described in the literature, such as ECB, CBC, OFB, CFB, CTR, among others (NIST 2018). Despite this, the use of ECB and CBC are discouraged due to the publicly known inherent vulnerabilities caused by ECB Rogaway (2011) and to the existence of padding oracle attacks applicable when using CBC (Vaudenay 2004).

4.2 HCA - cryptographic key definition

In HCA, the plaintext is partitioned into 128-bit blocks which constitute CA lattice configurations and, as previously stated, the encryption and decryption processes are based on CA evolution. Therefore, encryption quality is directly related to the CA evolution rules used in these processes which must be derived from a K cryptographic key.

The Wolfram notation defined in Sect. 2.3 allows expressing radius r CA rules as $2^{2\times r+1}$-bit strings. However, since HCA is based on left-toggle rules and according to Sect. 2.5, half of that string length is enough for the compressed rule notation that will be the K cryptographic key.

Table 2 Key bits amount × discarded keys percentage

Radius	Length (K)	Keyspace	Discarded (%)
1	4-bits	$2^4 = 32$	25
2	16-bits	$2^{16} = 131072$	8.64
3	64-bits	2^{64}	≈ 0.113
4	256-bits	2^{256}	$\approx < 0.1 \times 10^{-8}$

Despite the HCA method being compatible with any left-toggle rule, some do not produce the desired dynamic behavior. The normalized spatial entropy (h), computed for any potential key K, can be used to evaluate the quality of rules Oliveira et al. (2010c). Consider k an integer that denotes the bit size of K, the h coefficient is computed by the Expression (1) (Shannon 1948), where p_i is the probability of a $\log_2(k)$-bit substring occurring in the K sequence, which is evaluated for every possible $\log_2(k)$-bit binary string through the summation.

$$h = \frac{-\sum_{i=1}^{k} (p_i \times \log_2(p_i))}{\log_2(k)} \tag{1}$$

Setting the acceptable cryptographic key entropy above 0.75 is a criteria that filters out CA rules without chaotic dynamics (Oliveira et al. 2010c). Rules extracted from K where $h(K) > 0.75$ do not produce an easily identifiable pattern during CA evolution and, therefore, make the resulting ciphertext hard enough to decrypt without previous knowledge of K.

Accordingly, keys with $h \leq 0.75$ are discarded. Table 2 presents the relation between the CA rule radius, the length of the K key, the corresponding keyspace, and the percentage of discarded keys when the $h(K) > 0.75$ criteria is enforced.

In Table 2 the percentages listed for $r = 1$ and $r = 2$ are absolute, since all the possible keys were tested. An analysis of the entire keyspace for $r \geq 3$ is impractical, but extrapolations based on random sampling are presented for $r = 3$ and $r = 4$. For both estimates, 2^{32} keys were randomly generated and evaluated in regard to this acceptance criteria ($h > 0.75$). An apparent correlation can be observed between increasing the CA radius and a reduction in the percentage of discarded keys.

Taking into account the speed of commercially available hardware, HCA employs radius-4 rules ($r = 4$) to ensure a large 2^{256} keyspace, which is deemed appropriate against a brute force attack. The estimated $r = 4$ discarded keys percentage suggests that only a minimal set of very homogeneous keys would be rejected in the vast 2^{256} keyspace.

Since HCA is a left-toggle rule CA model, the K cryptographic key is a $\frac{2^9}{2} = 256$-bit sequence which can be expanded to 512-bit (left-toggle) radius-4 CA rule, under the conventional Wolfram notation. An explanation on how CA rules are derived from keys will be provided in Sect. 4.3.1.

4.3 HCA - defining operations

Considering each 128-bit block, a partition of the plaintext is the initial lattice configuration $s^{t=0}$, for the CA. To encrypt means applying the reverse evolution operation (Φ^{-1}), also known as preimage computation, for 128 steps until the configuration $s^{t=-128}$ is reached. Two CA rules derived from the cryptographic key K are applied at each evolution step, which are the main rule ϕ_m and the border rule ϕ_b. Decryption is performed through the forward CA evolution operation (Φ), using the same set of rules employed in encryption.

At each t time step of the HCA procedure, from the 128 cells in lattice s^t, $2 \times r = 8$ consecutive cells are considered the border region, whose cells are evolved using border rule ϕ_b. All other 120 cells are the main region, evolved using main rule ϕ_m.

4.3.1 Generating CA rules from a key

As previously established in Sect. 4.2, HCA is a radius-4 CA model that uses left-toggle rules, which means the 256-bit cryptographic key K can be expanded into a 512-bit string K_m that represents the ϕ_m radius-4 CA rule in Wolfram notation. This expansion procedure, exemplified at the end of Sect. 2.5, is presented as Expression 2, where the + sign stands for string concatenation and the upper slash indicates a binary complement operation.

$$K_m = K + \overline{K} \tag{2}$$

While ϕ_m is derived from K as shown in Expression (2), the 512-bit border rule, ϕ_b is determined between two rules which are rule $\{11\cdots11 + 00\cdots00\}$ and rule $\{00\cdots00 + 11\cdots11\}$. The Wolfram notation for these two left-toggle 512-bit rules start with a 256-bit consecutive repetition of a certain bit value and, thus, are bitwise complements one to another. The determination criteria for ϕ_b is denoted in Expression by its Wolfram representation string K_b.

$$K_b = \begin{cases} 11\cdots11 + 00\cdots00, & K[0] = 0 \\ 00\cdots00 + 11\cdots11, & K[0] = 1 \end{cases}$$

According to Expression , K_b will be the single rule from this subset which satisfies $K_b[0] = \overline{K[0]}$. This subset from which K_b is selected contains left-toggle rules that generate simple, but very unique, behaviors. If $K_b = 00\cdots00 + 11\cdots11$, then $\phi_b((s^t[i-4], \ldots, s^t[i], \ldots, s^t[i+4])) = s^t[i-4]$, which

means the resulting output $s^{t+1}[i]$ copies the value of the leftmost cell of the $s^t[i]$ neighborhood. Otherwise, if $K_b = 11\cdots11 + 00\cdots00$, then $\phi_b((s^t[i-4], \ldots, s^t[i], \ldots, s^t[i+4])) = \overline{s^t[i-4]}$, which means the resulting output $s^{t+1}[i]$ is the bitwise complement of the leftmost cell of the $s^t[i]$ neighborhood. The simplistic behavior expressed by border rule ϕ_b is crucial to the deterministic preimage computation that allows HCA to achieve reversibility, as described in following sections.

4.3.2 Backward evolution operation

Given a lattice s at step t (represented as s^t), consider the backward evolution operation (preimage computation) as $\Phi^{-1}(s^t, \phi_m, \phi_b) = s^{t-1}$. Applying Φ^{-1} to s^t means computing all bits of the predecessor configuration s^{t-1} using rules ϕ_m and ϕ_b, where $s^{t-1} = s^{t-1}[0], s^{t-1}[1], \cdots, s^{t-1}[127]$.

The preimage computation is started by determining the value of 8 consecutive bits ($b1$, $b2$, $b3$, $b4$, $b5$, $b6$, $b7$, $b8$) of the preimage using ϕ_b. As stated in the previous section, for any output bit $s^t[i]$ computed from the $s^{t-1}[i]$ neighborhood through ϕ_b, there is a relation of value equality or complement between $s^t[i]$ and the leftmost cell of the $s^{t-1}[i]$ neighborhood, which is $s^{t-1}[i-4]$.

Therefore, the knowledge of ϕ_b and of cell value $s^t[i]$, allows the deterministic computation of $s^{t-1}[i-4]$ if $s^{t-1}[i]$ is a border cell, according to Expression (3).

$$s^{t-1}[i-4] = \phi_b((s^t[(i), \ldots)) \tag{3}$$

This procedure is used to compute 8 cells of the s^{t-1} preimage. This is only possible due to the simplicity of border rule ϕ_b that imposes a non-chaotic dynamic behavior to these cells. Such non-chaotic behavior will not affect the quality of the algorithm, since the border rule will not have a significant influence on the CA dynamic as a whole. The border rule is only used to ensure the existence of a single preimage for any possible configuration, as proved in Sect. 5.

All the other 120-bits of the preimage are obtained from main rule ϕ_m, responsible for providing the desired chaotic behavior to the algorithm. The values of these 120 remaining bits are determined, one by one, in the toggle direction (left) as displayed in Fig. 3. Also, Fig. 4 represents the computation of a single cell value using main rule ϕ_m.

An initial supposition for the context presented in Fig. 4 is that the bit value in position $s^t[i]$ has been computed using main rule ϕ_m. Therefore, there is a valid mapping, specified by ϕ_m, from the $(m1, b1, \cdots, b8)$ values in the $s^{t-1}[i]$ neighborhood $(s^{t-1}[i-4], s^{t-1}[i-3], \cdots, s^{t-1}[i+4])$ to the output value in $s^t[i]$, which is o.

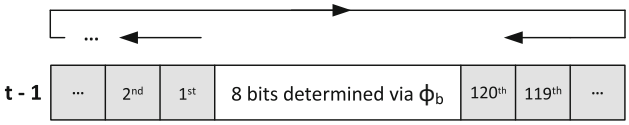

Fig. 3 Order of preimages bits computation using main rule ϕ_m

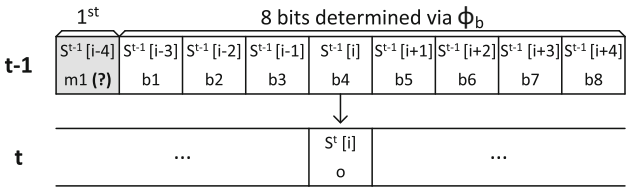

Fig. 4 First main bit determination

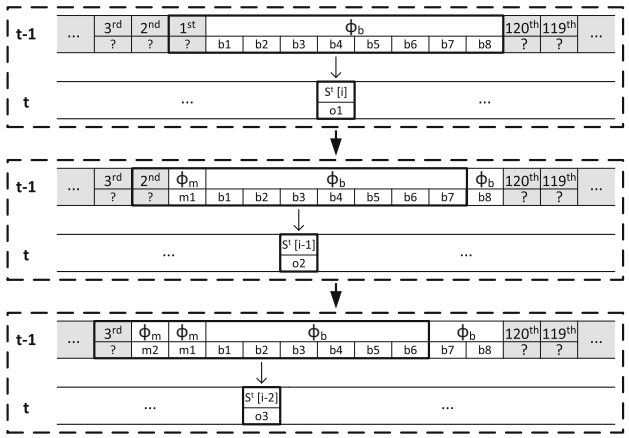

Fig. 5 Progressive computation of remaining preimage bits

Since the main rule, ϕ_m, is a radius-4 CA rule, it denotes a Boolean function that maps every possible 9-bit neighborhood values to their corresponding output bits. Due to the characteristics of ϕ_m as a left-toggle rule, and due to knowledge of the $b1, \ldots, b8$ cells values, there is only one value that permutive cell $m1$ could assume in Fig. 4 so that $\phi_m((m1, b1, b2, b3, b4, b5, b6, b7, b8)) = o$.

After determining the $m1$ value for position $s^{t-1}[i-4]$, this same logic can be used to compute $m2$ and, progressively, to determine every one of the 120 remaining bits in the s^{t-1} preimage. The first steps, and their respective considered neighborhoods, for this operation are presented on Fig. 5.

The first backward evolution $(\Phi^{-1}(s^t))$ is concluded after 120 main bits of the preimage are evaluated, since this means all the 128 bits in configuration s^{t-1} have been determined.

4.3.3 Forward evolution operation

Given a lattice s at step t, consider the forward evolution as $\Phi(s^t, \phi_m, \phi_b) = s^{t+1}$. Applying Φ to s^t means finding all

bits of s at step $t+1$, using rules ϕ_m and ϕ_b, where $s^{t+1} = (s^{t+1}[0], s^{t+1}[1], \cdots, s^{t+1}[127])$. For each i position, the bit value of the cell $s^{t+1}[i]$ is determined using main rule ϕ_m or border rule ϕ_b.

In this forward evolution procedure, the values of all s^{t+1} cells can be computed simultaneously. As previously stated, from the 128 grid cells, 8 are determined using the border rule, ϕ_b, and 120 cells are determined using the main rule, ϕ_m. A relevant explanation on which cell values are computed by each rule, for a certain time step t, is provided in Sect. 4.4.

4.4 HCA - lattice regions

Since the behavior imposed by border rule ϕ_b is very simplistic, in forward evolution the position of this border region suffers a circular shift by 8 positions to the right at each time step to ensure every cell position of the CA lattice is evolved multiple times by the main rule ϕ_m which is the source of chaotic behavior for the system.

There are 128 steps of evolution in HCA, so encryption starts with the $s^{t=0}$ plaintext and results in the $s^{t=-128}$ ciphertext. The first preimage computation Φ^{-1}, which determines $s^{t=-1}$ from $s^{t=0}$, is done under the premise that in lattice $s^{t=-1}$ the border region was comprised of cells $(s^{t=-1}[0], \ldots, s^{t=-1}[7])$. The following Expression 4 computes, for every $t<0$, which i lattice index should be considered the leftmost cell of the 8-bit border region in s^t.

$$i = (128 - ((|t| - 1) \times 8) \bmod 128) \bmod 128 \tag{4}$$

This Expression 4 is also helpful during HCA decryption to determine the border region and, thus, validate for each $s^t[i]$ cell, if ϕ_m or ϕ_b should be applied to its neighborhood during CA evolution Φ. For instance, when applying $t = -128$ to Expression 4, the result is $i = 0$, which means the border region of the ciphertext is comprised of cells $(s^{t=-128}[0], s^{t=-128}[1], \ldots, s^{t=-128}[7])$ and they should be evolved using rule ϕ_b while all other 120 cells will be evolved using ϕ_m to generate $s^{t=-127}$.

4.5 HCA - parallelism

A relevant characteristic of cellular automata is the inherent parallelism of these systems. In a conventional forward evolution procedure, all the s^t cells can be evolved simultaneously to generate the s^{t+1} configuration. Since the decryption process of HCA is based on the forward evolution operation (Φ), with proper hardware it is possible to evolve all 128 cells from the s^t lattice to the s^{t+1} lattice in parallel, with a considerable performance gain.

On the other hand, since the encryption process of HCA is based on the backward evolution operation (Φ^{-1}), a distinct way of achieving parallelism was devised. The HCA encryption is based on applying 128 successive preimage computation operations to each block, and, conventionally, the computation of a s^{t-1} preimage would only be started after all the bits of s^t are known. However, some level of parallelism can be attained by allowing bits from distinct preimages to be determined simultaneously. In the following subsections, the first steps of a HCA decryption process will be detailed according to the execution *cycle* at which cell values can be computed, considering constant time basic CA operations.

4.5.1 First preimage computation

Consider, for instance, the initial step of HCA encryption, where Φ^{-1} is applied to the $s^{t=0}$ plaintext lattice, resulting in the $s^{t=-1}$ preimage. According to Sect. 4.4, the border region for $t = -1$ has cells $(s^{t=-1}[0], \ldots, s^{t=-1}[7])$. As established in Sect. 4.3.2, for any t time step and 0-based cell index i, if $s^{t-1}[i]$ is a border cell, the value of $s^{t-1}[i-4]$ can be computed as soon as the values of ϕ_b and cell $s^t[i]$ are known. Therefore, since ϕ_b can be determined for any t as soon as the K encryption key is known, and due to all $s^{t=0}$ cell values being available from the beginning ($cycle = 0$), then the values of 8 $s^{t=-1}$ cells $(s^{t=-1}[3], s^{t=-1}[2], \ldots, s^{t=-1}[124])$ can be made available at $cycle = 1$ using ϕ_p. After that, all the other 120 cell values $(s^{t=-1}[123], s^{t=-1}[122], \ldots, s^{t=-1}[4])$ can be sequentially computed (from $cycle = 2$ to $cycle = 121$) using main rule ϕ_m.

4.5.2 Second preimage computation

In the second step of HCA encryption, $\Phi^{-1}(s^{t=-1}) = s^{t=-2}$, the border region for $t = -2$ is comprised of cells $(s^{t=-2}[120], \ldots, s^{t=-2}[127])$ and, again, for any t time step and 0-based cell index i, if $s^{t-1}[i]$ is a border cell, the value of $s^{t-1}[i-4]$ can be computed as soon as the values of ϕ_b and cell $s^t[i]$ are known. Therefore, 4 cell values $(s^{t=-2}[123], s^{t=-2}[122], \ldots, s^{t=-2}[120])$ can be made available at $cycle = 2$ using ϕ_b. The other 4 cells computed using ϕ_b, $(s^{t=-2}[119], s^{t=-2}[118], \ldots, s^{t=-2}[116])$ can be made available consecutively from $cycle = 3$ to $cycle = 6$, due to the availability of corresponding $s^{t=-1}$ values. And the remaining 120 cell values computed using main rule ϕ_m,

$$(s^{t=-2}[115], s^{t=-2}[114], \ldots, s^{t=-2}[0], s^{t=-2}[127], \ldots, s^{t=-2}[124])$$

are made available sequentially (from $cycle = 7$ to $cycle = 126$).

4.5.3 Third preimage computation

At the third step of HCA encryption, $\Phi^{-1}(s^{t=-2}) = s^{t=-3}$, the border region for $t = -3$ is comprised of cells $(s^{t=-3}[112], \ldots, s^{t=-3}[119])$. From this preimage computation onward, a pattern can be established, since all the cell values will be available for computed in consecutive cycles, due to the availability of cell values from the image. And thus, the 8 cell values computed using ϕ_b, $(s^{t=-3}[115], s^{t=-3}[114], \ldots, s^{t=-3}[108])$, are made available in consecutive cycles, from $cycle = 4$ to $cycle = 11$. As for the 120 cell values computed using ϕ_m,

$$(s^{t=-3}[107], s^{t=-3}[106], \ldots, s^{t=-3}[0], s^{t=-3}[127], \ldots, s^{t=-3}[116])$$

, they are made available sequentially (from $cycle = 12$ to $cycle = 131$).

4.5.4 Fourth preimage computation

At the forth step of HCA encryption, $\Phi^{-1}(s^{t=-3}) = s^{t=-4}$, the border region for $t = -4$ is comprised of cells $(s^{t=-4}[104], \ldots, s^{t=-4}[111])$. This is the last computation being described in detail, since the cycle pattern mentioned above will remain constant and conclusions can be drawn from it. The 8 cells computed using ϕ_b, $(s^{t=-4}[107], s^{t=-4}[106], \ldots, s^{t=-4}[100])$, are made available consecutively, from $cycle = 9$ to $cycle = 16$. As for the 120 cell values computed using ϕ_m,

$$(s^{t=-4}[99], s^{t=-4}[98], \ldots, s^{t=-4}[0], s^{t=-4}[127], \ldots, s^{t=-4}[108])$$

, they are made available sequentially (from $cycle = 17$ to $cycle = 136$).

4.5.5 Conclusions on HCA decryption parallelism

The cycle pattern established from $\Phi^{-1}(s^{t=-2})$ onward indicates the last cell computation, for a certain s^t preimage where $t \leq -3$, will at minimum happen at cycle $131 + ((-t-3) \times 5)$. The 131 and -3 terms are due to the cycle where the pattern starts being constant, as for 5, it is the growth constant observed for each $t \leq -3$ which is related to the CA radius. Assuming the possibility, with specialized hardware, of starting each preimage cell computation as soon as its conditions are fulfilled, theoretically the computation of the $t = -128$ ciphertext could be done in $cycle = 756$ according to the expression presented above, which is a relevant improvement when compared to the traditional approach of only starting the next preimage computation when all the image cell values are available, an approach that takes $128^2 = 16.384$ cycles.

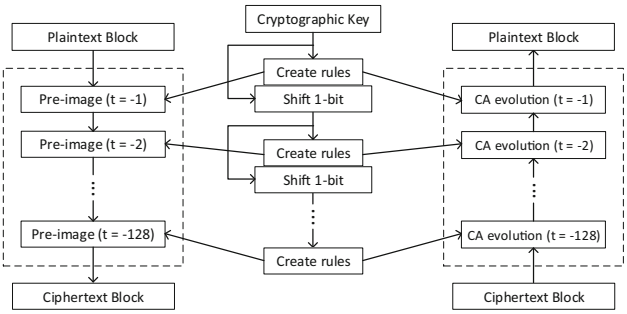

Fig. 6 Scheme illustrating the HCA encryption and decryption processes

4.6 HCA - method overview

The cryptographic key K, used in the process of generating the ϕ_m and ϕ_b rules, must be applied in a way that generates the same rules for equivalent encryption and decryption steps. Thus, the key K used at step $t = 1$ in encryption should be the same as the one used at step $t = 128$ during decryption. A scheme illustrating this relation is presented in Fig. 6, where the left side displays the encryption process being performed from top to bottom, and the right side of the figure shows the decryption process being performed in the opposite direction.

During encryption, at each CA step, the cryptographic key K is circularly shifted to the left by 1 position, generating two new rules ϕ_m and ϕ_b.

It is important to note that, in decryption, the ciphertext (s^{-128}) will be used as the initial configuration for the procedure, but the cryptographic key used in this step will be $Shift_{127}(K, left)$. The original cryptographic key K must be circularly shifted to the left by 127 positions before rules ϕ_m and ϕ_b are derived from it. At each decryption step it will be necessary to circularly shift this cryptographic key used in the previous step to the right by 1 position, so that it becomes equivalent to the key used in the corresponding encryption step.

It is expected that all main rules derived from K during encryption have chaotic dynamic behavior, and since many distinct rules are employed in the method, it is harder for a cryptographic attack to exploit the dynamic behavior of a specific rule.

The Algorithm 1 presents in pseudo-code the operations performed during the encryption process (backward evolution) and the Algorithm 2 shows the decryption process operations (forward evolution).

Algorithm 1: HCA algorithm - encryption

input: cryptographic key K and plaintext block s
output: ciphertext block

```
for i ← 1 to 128 do
    φ_m ← createRule_m(K);
    φ_b ← createRule_b(K);
    s ← Φ^{-1}(s, φ_m, φ_b);
    K ← Shift_1(K, left);
end
```

Algorithm 2: HCA algorithm - decryption

input : cryptographic key K and ciphertext block s
output: plaintext block

$K \leftarrow Shift_{127}(K, left)$;
for $i \leftarrow 1$ **to** 128 **do**
  $\phi_m \leftarrow createRule_m(K)$;
  $\phi_b \leftarrow createRule_b(K)$;
  $s \leftarrow \Phi(s, \phi_m, \phi_b)$;
  $K \leftarrow Shift_1(K, right)$;
end

5 Reversibility of HCA

As described in Sect. 4, for each plaintext block, the HCA symmetric cryptography method employs a series of preimage computation operations (Φ^{-1}) to encrypt it, and a series of CA forward evolution operations (Φ) to reverse the resulting ciphertext back to the original plaintext.

Such mechanism can only be effective if the HCA model provides CA reversibility, so that any lattice configuration will only have a single preimage. And thus, a required formal analysis of the reversibility property for HCA is provided in this section.

Theorem 1 *Let ϕ_m and ϕ_b be, respectively, the main and boundary rules used at any step of the HCA model. For any configuration s^t, s^t has one and only one preimage, which is s^{t-1}.*

Proof Let $s^t[0,\ldots,7]$ be the cells in s^t under ϕ_b, and $s^t[8,\ldots,127]$ the cells in s^t under ϕ_m. For simplicity, the theorem description is demonstrated for the lattice cell positions listed above. However, this is done without loss of generality due to the toroidal arrangement (wrap-around) of the CA lattice: circular shift could be applied at will and this description is still valid regardless of lattice cell positions. Let $s^{t-1}[0,\ldots,127]$ be the cells in s^{t-1}. Also w.l.o.g., let us consider only left-toggle rules.

By definition, we have that the output bit of left-toggle rules mappings necessarily changes when the leftmost bit of the input neighborhood is changed if the others neighborhood bits remain unchanged. From the definition of the HCA model, we know that ϕ_m is left-toggle rule, and that ϕ_b is even stricter: only the leftmost bit matters when computing the output, that is, the rule either always copies or always inverts the leftmost input bit. Thus, even though $s^t[0]$ is the result of the application of ϕ_b over the neighborhood $s^{t-1}[124,\ldots,4]$, it depends solely on $s^{t-1}[124]$, as shown in Fig. 7.

Therefore, $s^t[0] = \phi_b(s^{t-1}[124])$, but also $s^{t-1}[124] = \phi_b(s^t[0])$, since ϕ_b can only express a copy or complement operation. The same is true for each lattice cell in $s^t[0,\ldots,7]$ with respect to each lattice cell in $s^{t-1}[124,\ldots,3]$. Therefore, we can apply ϕ_b over each element in $s^t[0,\ldots,7]$ to uniquely determine $s^{t-1}[124,\ldots,3]$, as shown in Fig. 8.

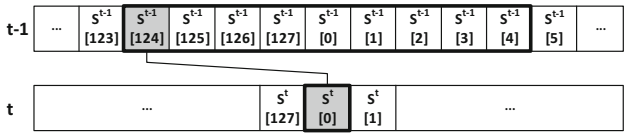

Fig. 7 Preimage computation on single bit of the border

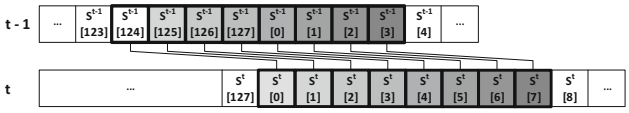

Fig. 8 Preimage computation on all bits of the border

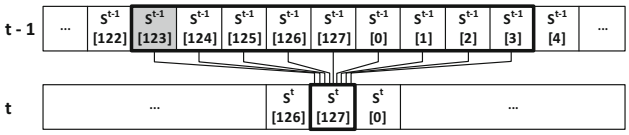

Fig. 9 Preimage computation in a main bit

Since $s^t[127] = \phi_m(s^{t-1}[123,\ldots,3])$, and considering the values of cells $s^{t-1}[124,\ldots,3]$ are already known, $s^{t-1}[123]$ can be computed by checking which bit value placed in $s^{t-1}[123]$ would lead to $\phi_m(s^{t-1}[123,\ldots,3]) = s^t[127]$, pictured in Fig. 9.

Since ϕ_m is a left-toggle rule, any value change in $s^{t-1}[123]$ would result in a change to $s^t[127]$, therefore $s^{t-1}[123]$ is unique. Cells $s^{t-1}[122,121,\ldots,5,4]$ are sequentially computed in an analogous manner and therefore are also unique.

Since the preimage s^{t-1} is computed deterministically and uniquely, we have that the theorem holds. □

From Theorem 1 it follows that the HCA model is reversible.

6 Security

In the literature there are many methods that help evaluating the quality of a cryptographic algorithm, and some that apply to this context of symmetric cryptography based on hybrid cellular automata will be listed and explained in this section.

6.1 Avalanche effect

Initially coined by Feistel (1973), the "Avalanche Effect" is an expected property in cryptographic systems which can be measured in two cases (Gustafson et al. 1994):

– *Plaintext Avalanche*: Using the same key, what is the impact of flipping a single bit in the plaintext?

– *Key Avalanche*: Using the same plaintext, what is the impact of flipping a single bit in the key?

This procedure is detailed in Fig. 10. When the Plaintext Avalanche is being evaluated we have $K = K'$, otherwise when Key Avalanche is being measured we have $X = X'$. For both cases, $Z = Y \oplus Y'$.

If an algorithm does not exhibit sufficient Avalanche Effect compliance it would be extremely vulnerable to chosen-plaintext attacks, so this kind of analysis is regarded as a conventional test ran to evaluate cryptographic algorithms' strength (Ramanujam and Karuppiah 2011; Mishra et al. 2011; Nadu 2018). The method presented in Sect. 4 was tested according to the specifications listed in 6.1.1 and 6.1.2. Results are presented on Sect. 7.1.

6.1.1 Avalanche effect - standard deviation analysis

If the algorithm presents strong Avalanche Effect, then we should expect the minimal difference between X and X' (for the Plaintext Avalanche test) or between K and K' (for the Key Avalanche test) to cause a significant difference between ciphertexts Y and Y' and thus, in ideal conditions, the percentage of '1' bits in Z should be around 50% (as would also be expected from a randomly generated binary sequence). It would also be relevant to know, considering many distinct avalanche evaluations in a diverse population, how consistent are these results. In this case, the standard deviation analysis is considered a proper way to measure this, and a lower StdDev value would be desirable.

6.1.2 Avalanche effect - entropy analysis

When evaluating the Avalanche Effect on an algorithm, besides counting how many bits of the ciphertext were affected by a minimal change, its also important to quantify how well propagated were the effects of said change. It would be desirable for this impact (bits changed) to be strongly distributed through the entire resulting ciphertext, and the concept of Information Entropy presented in 4.2 is a means to evaluate this.

So, when using the normalized entropy formula 1 to analyze sets of Z strings obtained from many Avalanche experiments, resulting values closer to 1.00 are desirable, as they would indicate the propagated changes were highly dispersed across the resulting ciphertext. Meanwhile, a result close to 0.00 is highly undesirable since it means the initial change made small or no difference in the ciphertext, or that it caused all the bits in Y and Y' to be the exact opposites.

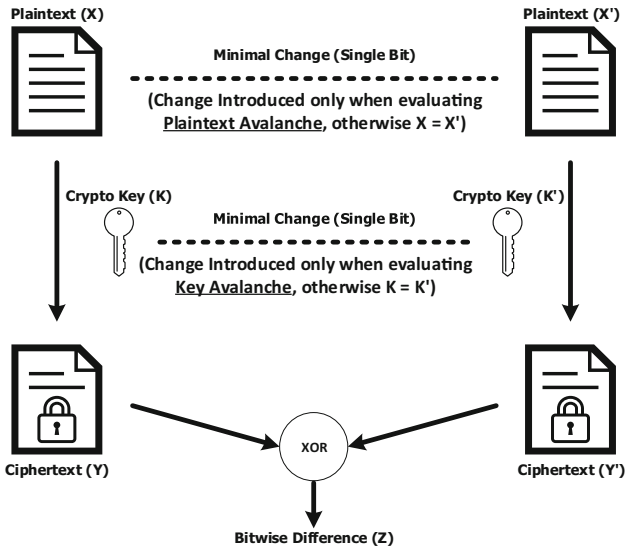

Fig. 10 Avalanche Effect evaluation explanation

6.2 NIST PRNG statistical test suite

The NIST (National Institute of Standards and Technology) is an American Institute founded in 1901 that provides guidelines on security and innovation. In 2010, NIST released their latest version of a Statistical Test Suite that evaluates the statistical quality of sequences generated by pseudo-random number generators (PRNG) (Zaman and Ghosh 2012).

Since there is a known correlation between PRNG and encryption, this test suite is also being used to measure the quality of encryption algorithms by evaluating the statistical difference between the plaintext and its resulting ciphertext.

The NIST suite consists of 15 tests, and each test can be comprised of many subtests, which is why the suite is sometimes listed as having 15 tests (Lakra et al. 2018) and in other times as being a set of 188 or more tests (Manzoni and Mariot 2018).

7 Evaluations

7.1 Avalanche effect

The Avalanche Effect test was applied to HCA in the following conditions:

– Each CA lattice is comprised of N = 128 bits
– 655, 360 random lattices are generated, totalling 10 megabytes of data
– Each execution ran for N evolution steps (except for RNG)

Each of the randomly generated initial lattices was encrypted by the HCA algorithm using random valid HCA cryptographic keys (with spatial entropy > 0.75) and the results are presented in Table 3. In each table, values obtained from sequences generated by the Mersenne Twister PRNG (Matsumoto and Nishimura 1998) are included for comparison purposes.

In Table 3 the test results are displayed for the plaintext and key avalanche evaluations. A good encryption algorithm should present a modification rate in the final ciphertext around 50%, with a low standard deviation (σ), as is the case in all average values found for both instances. The result values are also similar to the bit distribution rate in the PRNG generated sequences.

It is also important to ensure a random spatial dispersion of the changed bits, so a spatial entropy analysis is done on the resulting difference (XOR) lattice. These results follow in Tables 4 and 5 for the plaintext and key avalanche evaluations, respectively.

The average entropy results for each tested radius value should be as close as possible to 1.00 and the evaluated results, presented above, were comparable in all instances to the average entropy found in randomly generated sequences.

Table 3 HCA - Avalanche Effect Result Statistics

HCA - text aval.		HCA - key aval.		RNG	
Avg (%)	σ	Avg (%)	σ	Avg (%)	σ
49.986	4.421	49.996	4.426	50.002	4.417

Table 4 HCA - plaintext avalanche entropy analysis

HCA				RNG			
Min	Max	Avg	σ	Min	Max	Avg	σ
0.746	0.948	0.883	0.018	0.750	0.955	0.883	0.018

Table 5 HCA - Key Avalanche entropy analysis

HCA				RNG			
Min	Max	Avg	σ	Min	Max	Avg	σ
0.741	0.950	0.883	0.018	0.750	0.955	0.883	0.018

7.2 NIST PRNG suite

The NIST suite (Zaman and Ghosh 2012) tests were run on sequences that directly convey the changes an encryption method causes on random plaintexts. Each test has a minimum input size recommendation, and since some of them are as large as 10^6 bits, the results presented here were obtained using sequences consisting of 10 Megabytes. The input sequence construction procedure is explained on Fig. 11.

As presented in Fig. 11, building each 10 Megabytes sequence begins by initializing a single 128-bit block sized lattice using a pseudo-random seed, this initial plaintext is regarded as "P_1". The encryption algorithm is applied to P_1, generating a ciphertext called "P_2", and their binary difference, $P_1 \oplus P_2$, represents the effect of the encryption procedure. After $P_1 \oplus P_2$ is calculated, this 128-bit sequence is the first part of the 10 megabytes input sequence used for NIST evaluation; the next part will be $P_2 \oplus P_3$, where "P_3" is the new ciphertext obtained by running the encryption algorithm with P_2 as the plaintext. This iterative procedure is repeated until the 10 Megabytes sequence is complete by appending the last part, $P_{N-1} \oplus P_N$, where, accordingly, $N = (10\,megabytes)/(128\,bits)$.

The NIST evaluation ran for each algorithm was executed for 1,000 distinct 10 Megabytes sequences generated using the procedure listed above. The percentage of passing sequences for each NIST test follow in Table 6.

Besides the results found for sequences generated by the HCA method, Table 6 also contains the results for sequences similarly generated using the AES algorithm.

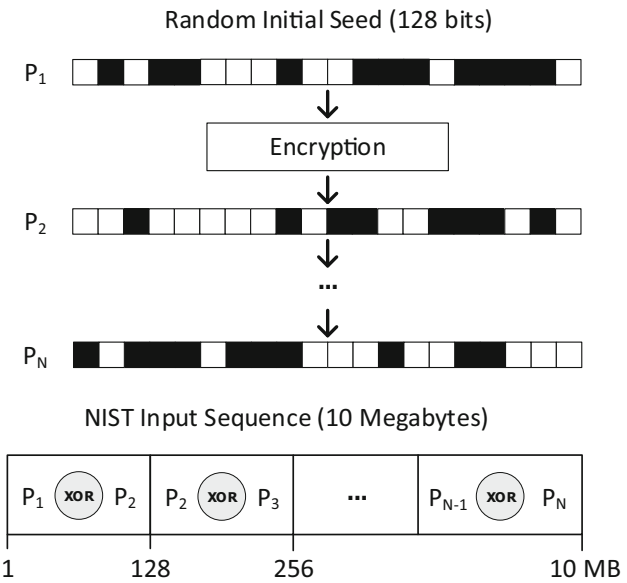

Fig. 11 NIST input sequence building

Table 6 NIST suite tests

NIST test	HCA (%)	AES (%)
T01 - Frequency (Monobits) test	99.4	99.1
T02 - Frequency test within a block	98.8	99.4
T03 - Runs test	98.9	98.7
T04 - Test for the longest run of ones in a block	99.0	98.3
T05 - Binary matrix rank test	99.2	99.2
T06 - Discrete fourier transform (Specral) test	99.5	98.6
T07 - Non-overlapping template matcing test	96.6	98.0
T08 - Overlapping template matching test	99.2	99.2
T09 - Maurer's "Universal Statistical" test	99.4	99.0
T10 - Linear complexity test	99.0	98.4
T11 - Serial test	98.1	98.1
T12 - Approximate entropy test	98.6	99.1
T13 - Cumulative sums (Cusum) test	99.0	98.8
T14 - Random excursions test	93.9	93.2
T15 - Random excursions variant test	92.5	92.7

The proximity between the passing rates of both algorithms, for all tests in the NIST suite, suggests HCA is a promising method, since AES is the current symmetric encryption standard.

8 Conclusions and future work

This paper describes a symmetrical block cipher model based on reversible heterogeneous cellular automata that employs two radius-4 left-toggle rules. The main rule is chaotic and non-additive; it is applied to the majority of bits at each time step to provide the necessary entropy to the encryption process. The second one is periodic (more specifically, fixed-point with a spatial displacement) and additive; it is applied to a small set of consecutive bits (the lattice border) and is used to ensure the existence of a preimage. This model was named HCA (Hybrid Cellular Automata) and it was firstly proposed in 2007, when a patent registration was submitted in Brazil Oliveira and Macêdo (2019). It is the first time that HCA is presented and evaluated in a wide range scientific forum. In the past, just the brazilian patent registration (whose process was finalized in 2019) and some local academic works (master's thesis), written in portuguese, have focused on aspects of, and extensions to, the HCA model (Magalhães Júnior 2010; Lima 2012; Alt 2013).

The adopted block size is a sequence of 128 bits, and the secret key has 256 bits which define the main rule to be applied. Moreover, as presented here, forward and backward CA evolution procedures correspond to the decryption and encryption processes, respectively. However, the converse is also possible; HCA enables one to use forward evolution in ciphering, while the receiver must use backward evolution to decipher. In general, forward is faster than backward evolution and in the specification discussed here the receiver will employ the faster process to decipher. It would also be easy to increase the size of the block for 256 bits or more. If one wants to use a larger key space, it is also simple to adapt the model to use radius-5 toggle rules or more; however this would increase the complexity of implementing the solution in HPC systems, such as FPGAs (Halbach and Hoffmann 2004).

Although the statistical methods presented in Sect. 6 are not capable of fully attesting the security of HCA, they are the standard validations usually ran against novel cryptographic methods and pseudo-random number generators. Therefore, their results provide a basic sanity check that could reveal existing vulnerabilities. The positive results obtained for HCA are very welcome, but only rigorous cryptanalysis would be capable of certifying the robustness of HCA, which is an ongoing work of our research group.

When compared to other similar CA-based methods (Gutowitz 1995; Oliveira et al. 2004, 2008, 2010b, c, 2011; Wuensche 2008) which also apply toggle (permutive) rules, the HCA algorithm has the advantage of keeping the ciphertext size equal to the plaintext, whereas being valid for any possible CA initial configuration since the existence of a preimage is ensured due to the heterogeneous arrange of the two toggle rules (chaotic and additive) (Oliveira et al. 2008). As a symmetric algorithm, this model can be applied to any kind of data (text, images, etc.) by defining a safe padding strategy and a secure mode of operation.

Despite the parallelism mechanisms of HCA having already been explained in this paper, implementation in specialized hardware was not possible at the time of this publication. The method was implemented in conventional

x86 software and only inter-block parallelism, which is available to any block cipher algorithm, was explored. Therefore, the efficient implementation of HCA in High Performance Computing (HPC) systems, such as FPGA architectures, is another ongoing work of our research group.

The conception of the reversibility analysis presented in Sect. 5 gave insight into the possibility of extending this concept to HCA alternatives with an even higher heterogeneity level. This could also allow the use of rules with radius lower than 4, which would lead to more performance and ease to implement. Another expected development is a forthcoming work that investigates an HCA adaptation using multidimensional cellular automata.

Data availability Not applicable

Code availability An implementation of the featured algorithm is made available at https://github.com/evertonrlira/HCA. Any updates will also be published on the linked repository.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Consent for publication The authors declare that they give consent for publication.

References

Alizadeh R (2011) A dynamic cellular automaton model for evacuation process with obstacles. Safety Sci 49(2):315–323

Alt LdS (2013) Propriedades decidíveis de autômatos celulares finitos, híbridos, não-lineares, sensíveis e reversíveis (in portuguese). Master's thesis, Federal Univ. of Uberlândia

Benkiniouar M, Benmohamed M (2004) Cellular automata for cryptography. In: Proceedings. 2004 international conference on information and communication technologies: from theory to applications, 2004., IEEE, pp 423–424

Blackburn SR, Murphy S, Paterson KG, Nandi S, Chaudhuri P (1997) Comments on'' theory and applications of cellular automata in cryptography''[with reply]. IEEE Trans Comput 46(5):637–639

Carneiro MG, Oliveira GM (2013) Synchronous cellular automata-based scheduler initialized by heuristic and modeled by a pseudo-linear neighborhood. Natural Comput 12(3):339–351

Carvalho TI, Carneiro MG, Oliveira GM (2019) Improving cellular automata scheduling through dynamics control. Int J Parallel Emergent Distrib Syst 34(1):115–141

Daemen J, Rijmen V (2002) The design of Rijndael: AES - the advanced encryption standard. Springer Verlag, Berlin, Heidelberg, New York

Daemen J, Rijmen V (2005) Rijndael/aes. In: Encyclopedia of Cryptography and Security, Springer, pp 520–524

Das S, Chowdhury DR (2010) Generating cryptographically suitable non-linear maximum length cellular automata. In: International conference on cellular automata for research and industry, Springer, pp 241–250

Dennunzio A, Formenti E, Provillard J (2012) Non-uniform cellular automata: classes, dynamics, and decidability. Inf Comput 215:32–46

Dennunzio A, Formenti E, Provillard J (2013) Local rule distributions, language complexity and non-uniform cellular automata. Theor Comput Sci 504:38–51

Dennunzio A, Formenti E, Provillard J (2014) Three research directions in non-uniform cellular automata. Theor Comput Sci 559:73–90

Dennunzio A, Formenti E, Weiss M (2014) Multidimensional cellular automata: closing property, quasi-expansivity, and (un)decidability issues. Theor Comput Sci 516:40–59. https://doi.org/10.1016/j.tcs.2013.11.005

Dennunzio A, Formenti E, Manzoni L, Margara L, Porreca AE (2019) On the dynamical behaviour of linear higher-order cellular automata and its decidability. Inf Sci 486:73–87

Dennunzio A, Formenti E, Grinberg D, Margara L (2020) Chaos and ergodicity are decidable for linear cellular automata over (z/mz)n. Inf Sci 539:136–144

Dennunzio A, Formenti E, Grinberg D, Margara L (2021) Decidable characterizations of dynamical properties for additive cellular automata over a finite abelian group with applications to data encryption. Inf Sci 563:183–195

Ermentrout GB, Edelstein-Keshet L (1993) Cellular automata approaches to biological modeling. J Theor Biol 160(1):97–133

Feistel H (1973) Cryptography and computer privacy. Sci Am 228(5):15–23

Feliciani C, Nishinari K (2016) An improved cellular automata model to simulate the behavior of high density crowd and validation by experimental data. Physica A Statist Mech Appl 451:135–148

Fips P (1999) 46–3. data encryption standard (des). Nat Inst Standards Technol 25(10):1–22

Ghimire B, Chen AS, Guidolin M, Keedwell EC, Djordjević S, Savić DA (2013) Formulation of a fast 2d urban pluvial flood model using a cellular automata approach. J Hydroinf 15(3):676–686

Gustafson H, Dawson E, Nielsen L, Caelli W (1994) A computer package for measuring the strength of encryption algorithms. Comput Secur 13(8):687–697

Gutowitz H (1995) Cryptography with dynamical systems. Kluwer Acad Press, Dordrecht

Gutowitz HA (1994) Method and apparatus for encryption, decryption and authentication using dynamical systems. US Patent 5,365,589

Halbach M, Hoffmann R (2004) Implementing cellular automata in fpga logic. In: 18th International parallel and distributed processing symposium, 2004. Proceedings, IEEE, p 258

Hillis WD (1984) The connection machine: a computer architecture based on cellular automata. Physica D 10(1–2):213–228

Ioannidis K, Sirakoulis GC, Andreadis I (2011) Cellular ants: a method to create collision free trajectories for a cooperative robot team. Robot Auton Syst 59(2):113–127

John A, Lakra R, Jose J (2020) On the design of stream ciphers with cellular automata having radius= 2. IACR Cryptol ePrint Arch 2020:327

Kari J (1992) Cryptosystems based on reversible cellular automata. https://www.researchgate.net/profile/Jarkko-Kari/publication/2350190_Cryptosystems_Based_on_Reversible_Cellular_Automata/links/55ca14dc08aeca747d69de82/Cryptosystems-Based-on-Reversible-Cellular-Automata.pdf

Lakra R, John A, Jose J (2018) Carpenter: A cellular automata based resilient pentavalent stream cipher. International Conference on Cellular Automata, Springer, pp 352–363

Le D, Chang J, Gou X, Zhang A, Lu C (2010) Parallel aes algorithm for fast data encryption on gpu. In: 2010 2nd international conference on computer engineering and technology (ICCET), IEEE, vol 6, pp V6–1

Lent CS, Tougaw PD, Porod W, Bernstein GH (1993) Quantum cellular automata. Nanotechnol 4(1):49

Leporati A, Mariot L (2014) Cryptographic properties of bipermutive cellular automata rules. J Cell Autom 9:437–475

Lima DA (2012) Modelo criptográfico baseado em autômatos celulares tridimensionais híbridos (in portuguese). Master's thesis, Federal Univ. of Uberlândia

Lima DA, Oliveira GM (2017) A cellular automata ant memory model of foraging in a swarm of robots. Appl Math Model 47:551–572

Barros de Macedo H, Barbosa de Oliveira GM, Costa Ribeiro CH (2014) Dynamic behaviour of network cellular automata with non-chaotic standard rules. In: Complex systems, 2nd World conference on complex systems (WCCS), IEEE, pp 451–456

Macêdo HBd (2007) Um novo método criptográfico baseado no cálculo de pré-imagens de autômatos celulares caóticos, não-homogêneos e não-aditivos (in portuguese). Master's thesis, Federal Univ. of Uberlândia

Machicao J, Marco AG, Bruno OM (2012) Chaotic encryption method based on life-like cellular automata. Expert Syst Appl 39(16):12626–12635

Maerivoet S, De Moor B (2005) Cellular automata models of road traffic. Phys Rep 419(1):1–64

Magalhães Júnior TAd (2010) Método criptográco baseado em autômatos celulares bidimensionais para cifragem de imagens (in portuguese). Master's thesis, Federal Univ. of Uberlândia

Manzoni L, Mariot L (2018) Cellular automata pseudo-random number generators and their resistance to asynchrony. International conference on cellular automata, Springer pp 428–437

Mariot L, Leporati A (2014) Sharing secrets by computing preimages of bipermutive cellular automata. In: Cellular Automata: 11th International conference on cellular automata for research and industry, ACRI 2014, Krakow, Poland, September 22-25, 2014. Proceedings 11, Springer, pp 417–426

Matsumoto M, Nishimura T (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans Model Comput Simul (TOMACS) 8(1):3–30

Mattei M, Frunzo L, D'acunto B, Pechaud Y, Pirozzi F, Esposito G (2018) Continuum and discrete approach in modeling biofilm development and structure: a review. J Math Biol 76(4):945–1003

Mishra P, Gupta I, Pillai NR (2011) Generalized avalanche test for stream cipher analysis. In: Security Aspects in Information Technology: First International Conference, InfoSecHiComNet 2011, Haldia, India, October 19-22, 2011. Proceedings, Springer, pp 168–180

Mitchell M, et al. (2005) Computation in cellular automata: a selected review

Moore EF (1962) Machine models of self-reproduction. In: Proceedings of symposia in applied mathematics, American Mathematical Society New York, pp 17–33

Morita K (2008) Reversible computing and cellular automata-a survey. Theor Comput Sci 395(1):101–131

Nadu ST (2018) A block cipher algorithm to enhance the avalanche effect using dynamic key-dependent s-box and genetic operations. Int J Pure Appl Math 119(10):399–418

Nandi S, Kar B, Chaudhuri PP (1994) Theory and applications of cellular automata in cryptography. IEEE Trans Comput 43(12):1346–1357

NIST (2018) Block cipher modes. https://csrc.nist.gov/projects/block-cipher-techniques, accessed: 2019-02-05

Oliveira G, Macêdo H (2019) Sistema criptográfico baseado no cálculo de preimagem em autômatos celulares não-homogêneos, não-aditivos e com dinâmica caótica. Patent dep at INPI-Brazil under number PI0703188-2

Oliveira G, Martins LG, Alt LS, Ferreira GB (2010a) A cellular automata-based cryptographic model with a variable-length ciphertext. In: The 2010 International conference on scientific computing pp 1 – 10

Oliveira GM, Martins LG, de Carvalho LB, Fynn E (2009) Some investigations about synchronization and density classification tasks in one-dimensional and two-dimensional cellular automata rule spaces. Electron Notes Theor Comput Sci 252:121–142

Oliveira GM, Martins LG, Alt LS, Ferreira GB (2010) Exhaustive evaluation of radius 2 toggle rules for a variable-length cryptographic cellular automata-based model. In: International conference on cellular automata, Springer, pp 275–286

Oliveira GM, Martins LG, Ferreira GB, Alt LS (2010c) Secret key specification for a variable-length cryptographic cellular automata model. In: PPSN, Springer, pp 381–390

Oliveira GM, Martins LG, Alt LS (2011) Deeper investigating adequate secret key specifications for a variable length cryptographic cellular automata based model. Cellular Automata: Innov Model for Sci and Eng p 265

Oliveira GMB, Coelho A, Monteiro L (2004) Cellular automata cryptographic model based on bi-directional toggle rules. Int J Modern Phys C 15(08):1061–1068

Oliveira GMB, Lima M, Macedo H, Branquinho A (2008) A cryptographic modelo based on the pre-image computation of cellular automata. In: Adamatzky A,Alonso-Sanz R, Lawniczak A (eds) Automata-2008: Theory and Applications of Cellular Automata, Luniver Press, pp 139-155. https://books.google.com.br/books?id=poMaluGfOnsC

Prasad VC, Maheswari S (2013) Robust watermarking of aes encrypted images for drm systems. In: 2013 International conference on emerging trends in computing, communication and nanotechnology (ICECCN), IEEE, pp 189–193

Ramanujam S, Karuppiah M (2011) Designing an algorithm with high avalanche effect. IJCSNS Int J Comput Sci Netw Secur 11(1):106–111

Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126

Rogaway P (2011) Evaluation of some blockcipher modes of operation. Crypto Research and Eval Committees (CRYPTREC) for the Gov of Japan

Rosin PL (2010) Image processing using 3-state cellular automata. Comput Vision Image Underst 114(7):790–802

Rozenberg G, Bäck T, Kok JN (2012) Handbook of natural computing. Springer, New york

Sarkar P (2000) A brief history of cellular automata. Acm Comput Surv (csur) 32(1):80–107

Sen S, Shaw C, Chowdhuri DR, Ganguly N, Chaudhuri PP (2002) Cellular automata based cryptosystem (cac). In: Information and communications security: 4th International conference, ICICS 2002 Singapore, December 9–12, 2002 Proceedings 4, Springer, pp 303–314

Seredynski F, Bouvry P, Zomaya AY (2004) Cellular automata computations and secret key cryptography. Parallel Comput 30(5–6):753–766

Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423

Silva EC, Soares JA, Lima DA (2016) Autômatos celulares unidimensionais caóticos com borda fixa aplicados à modelagem de um sistema criptográfico para imagens digitais (in portuguese). Informática Teórica e Aplicada pp 250–276

Sirakoulis GC (2016) Parallel application of hybrid dna cellular automata for pseudorandom number generation. J Cell Autom 11(1):63–89. http://www.oldcitypublishing.com/journals/jca home/jca-issue-contents/jca-volume-11-number-1-2016/jca-11-1-p-63-89/

Swiecicka A, Seredynski F, Zomaya AY (2006) Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. IEEE Trans Parallel Distrib Syst 17(3):253–262

Toffoli T, Margolus N (1987) Cellular automata machines: a new environment for modeling. MIT press, USA

Tomassini M, Perrenoud M (2000) Stream cyphers with one-and two-dimensional cellular automata. In: PPSN, Springer, pp 722–731

Tomassini M, Perrenoud M (2001) Cryptography with cellular automata. Appl Soft Comput 1(2):151–160

Vasantha S, Shivakumar N, Rao DS (2015) A new encryption and decryption algorithm for block cipher using cellular automata rules. Int J Emerg Eng Res Tech 3(8):130–136

Vaudenay S (2004) Security flaws induced by cbc padding. Adv in Crypto-Proc of EUROCRYPT'02 pp 534–545

Vichniac GY (1984) Simulating physics with cellular automata. Physica D 10(1–2):96–116

Wolfram S (1985) Cryptography with cellular automata. In: Conference on the theory and application of cryptographic techniques, Springer, pp 429–432

Wolfram S (1986) Random sequence generation by cellular automata. Adv Appl Math 7(2):123–169

Wolfram S (2002) A new kind of science. Wolfram Media - (1st Ed.)

Wuensche A (2008) Encryption using cellular automata chain-rules. In: Automata, pp 126–138

Wuensche A, Lesser M (1992) The global dynamics of cellular automata. Andrew Wuensche

Yang YG, Tian J, Lei H, Zhou YH, Shi WM (2016) Novel quantum image encryption using one-dimensional quantum cellular automata. Inf Sci 345:257–270

Yilmaz O (2015) Symbolic computation using cellular automata-based hyperdimensional computing. Neural Comput 27(12):2661–2692

Zaman J, Ghosh R (2012) A review study of nist statistical test suite: development of an indigenous computer package. arXiv preprint arXiv:1208.5740

Zeghid M, Machhout M, Khriji L, Baganne A, Tourki R (2007) A modified aes based algorithm for image encryption. Int J Comput Sci Eng 1(1):70–75