The final publication is available at Springer via https://doi.org/10.1007/978-3-319-99813-8_39

Cellular Automata Pseudo-Random Number Generators and Their Resistance to Asynchrony

Luca Manzoni[1] and Luca Mariot[1]

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336, 20126 Milano (Italy)
{luca.manzoni, luca.mariot}@disco.unimib.it

Abstract. Cellular Automata (CA) have a long history being employed as pseudo-random number generators (PRNG), especially for cryptographic applications such as keystream generation in stream ciphers. Initially starting from the study of rule 30 of elementary CA, multiple rules where the objects of investigation and were shown to be able to pass most of the rigorous statistical tests used to assess the quality of PRNG. In all cases, the CA employed where of the classical, synchronous kind. This assumes a global clock regulating all CA updates which can be a weakness if an attacker is able to tamper it. Here we study how much asynchrony is necessary to make a CA-based PRNG ineffective. We have found that elementary CA are subdivided into three class: (1) there is a "state transition" where, after a certain level of asynchrony, the CA loses the ability to generate strong random sequences, (2) the randomness of the sequences increases with a limited level of asynchrony, or (3) CA normally unable to be used as PRNG exhibit a much stronger ability to generate random sequences when asynchrony is introduced.

1 Introduction

Cellular Automata (CA) are one of the oldest nature-inspired computational models in computer science [25,26]. Defined informally, CA are composed of a lattice of identical finite state automata (or *cells*) all updating at the same time according to their state and the state of their neighbours. CA have been successfully employed in multiple fields, like for instance the modelling of physical systems [5] such as fluids [4], natural ecosystems [1], traffic flows [13], and of pedestrians in crowds [2]. Here, we mainly deal with the cryptographic applications of CA. In particular, we consider the well-known problem of generating *pseudo-random sequences* by exploiting the dynamical behaviour of CA. Pseudo-random sequences play a fundamental role in cryptography, for example in keystream generation for stream ciphers [14]. Differently from other studies, we do not try to find new CA that works well as PRNG; instead, we study how asynchrony influences the ability of a CA to produce pseudo-random sequences.

The final publication is available at Springer via https://doi.org/10.1007/978-3-319-99813-8_39

In classical CA, all cells update at the same time (i.e., *synchronously*), the underlying assumption being that there is a single, global clock regulating all cells. This is, however, a strong assumption since real-world systems are usually not synchronous. Once this assumption is dropped, there are multiple ways to introduce asynchronous behaviours in CA. For example by using a probabilistic activation [9,10,11], updating a cell at a time according to a given sequence [18], having different areas of the CA update with different speeds [17,19], or even more general updating schemes [28,8]. Here we deal with a simplified model of asynchrony where the CA is partitioned in separate, contiguous sequences of cells, all cells inside the same sequence update in parallel, but the sequences update sequentially.

Our goal is to study what happens when the aforementioned assumption of a global clock is broken not by design, but by a malicious actor who wants to tamper with the PRNG. Since PRNG are used in cryptographic applications, limiting the amount of damage that can be carried on by damaging them (or, at least, the global clock governing their updates) is paramount. Here, in particular we experimentally study how different levels of asynchrony impacts the generation of pseudo-random sequences generated by *elementary CA*.

The paper is organised as follows: some necessary basic notions are recalled in Section 2. Section 3 briefly reviews the state of the art in CA-based PRNG, mostly focusing on the synchronous approach. Section 4 describes in the detail experiments we performed. In particular, Section 4.1 explains all the experimental settings used, while a general discussion of the experimental results is carried out in Section 4.2. The discussion of the results, particularly the classification of the observed behaviours in three broad classes, is given in Sections 4.3, 4.4 and 4.5. Some further considerations and directions for future works are presented in Section 5.

2 Basic Notions

In this section we recall some basic notions on CA, their properties, and how they can be employed as PRNG.

Definition 1. *A* cellular automaton *(CA) is a tuple* (Σ, f, r) *where* Σ *is a finite alphabet,* $r \in \mathbb{N}$ *is the radius, and* $f \colon \Sigma^{2r+1} \to \Sigma$ *is the local function of the CA. If the CA only has a finite number* $n \in \mathbb{N}$ *of cell, i.e., it is a* finite CA*, we say that it is a CA of size* n*.*

A CA is said to be an *elementary CA* (ECA) when its alphabet is $\{0, 1\}$ and it has radius 1. There are exactly 256 ECA, each one numbered with its Wolfram code, a number between 0 and 255 whose binary expansion represents the output column of the truth table defining the local function of the CA.

Here we only deal with CA of finite size with *periodic boundary conditions*, that is, the cell adjacent to the n-th one is the first one and vice versa. In the following we assume that the subscript denoting the cell position is to be interpreted modulo n, the size of the CA.

The *configuration* of a cellular automaton (Σ, f, r) of size n is a vector $c = c_0, \ldots, c_{n-1} \in \Sigma^n$. The CA updates its state using a global rule $F\colon \Sigma^n \to \Sigma^n$ where each cell updates its state at the same time using the local rule, thus giving the following global rule:

$$F(c)_i = f(c_{i-r}, \ldots, c_i, \ldots, c_{i+r}), \text{ for } 0 \leq i < n$$

Finite CA of length n with alphabet $\{0, 1\}$ are usually employed as PRNG in the following way [27]:

- A *random seed* of n bits is the initial configuration of the CA;
- To obtain an new pseudo-random bit the entire CA is updated and one cell (usually the central one) is sampled.

Since CA update all cells in parallel and each cell requires only access to local information, they can be easily parallelized and/or implemented in hardware [23].

2.1 The Asynchronous Model

While classical CA are inherently synchronous, in recent years multiple variations of CA were defined with the addition of some kind of asynchronous behaviour. In our work we deal with a very specific kind of asynchrony, where a finite CA of length n has its set of cells $\{0, \ldots, n-1\}$ partitioned into k contiguous segments $I_0, \ldots, I_{n-1}$ with $I_i = \left\{i\frac{n}{k}, \ldots, (i+1)\frac{n}{k} - 1\right\}$, where k is a divisor of n. At time 0 only the cells in the segment I_0 are updated; at the successive time step only the cells in the segment I_1 are updated, and so on. In general, at the t-th time step only the cells in the segment $t \bmod k$ are updated.

This kind of asynchrony can be tuned by using the parameter k: when $k = 1$ there is only one segment and the update is synchronous, like in classical CA. When $k = n$ only one cell updates at each time step, mimicking the behaviour of fully asynchronous CA [18]. It is also possible to obtain intermediate levels of asynchrony: for example, with $k = 2$ the CA is effectively split into two parts which update alternately.

In this paper we empirically study how increasing the value of k influences the ability of a CA to produce robust pseudo-random sequences (i.e., which pass rigorous statistical tests). To avoid the risk of sampling multiple times a cell that still has not updated, we perform the sampling every k steps. In this case for $k = 1$ the behaviour is the same as in classical CA and in all other cases we ensure that the sampled cell has always been updated between two samplings.

3 Related Work

In this section, we give a brief historical overview of the literature concerning pseudo-random sequence generation by means of cellular automata.

Wolfram [27] was the first to propose a PRNG based on a chaotic CA to be employed in cryptographic applications. Specifically, he suggested to use a

periodic CA equipped with rule 30, and to sample the value of a certain cell as a pseudo-random sequence. Some years later, Damgård [7] showed a concrete construction of iterated hash function based on Wolfram's PRNG.

Unfortunately, Wolfram's PRNG later turned out to be very weak from a cryptographic standpoint: Meier and Staffelbach [22] proved that it is vulnerable to a known plaintext attack, unless the CA is composed of at least 1000 cells. The attack exploits the *quasi-linearity* of rule 30, which allows to rewrite it in an equivalent way where the initial seeds are not equiprobable. Analogously, Daemen et al. [6] cryptanalysed Damgård's hash function, proving that it is computationally feasible to generate collisions in it.

Sipper and Tomassini [24] proposed a *cellular programming approach* based on a non-uniform CA, where the rule vector specifying which rule is applied in each cell is evolved by a *Genetic Algorithm* (GA). The fitness of each cell is evaluated by computing the entropy of the pseudo-random sequence generated by its current rule for 4096 time steps, averaging the results over 300 initial random configurations. The final rule vectors evolved through the cellular programming algorithm were then further investigated by testing longer sequences with the ENT statistical test suite.

A common trend that can be noticed in the CA-based PRNG literature is that the cryptographic quality of the pseudo-random sequences is usually assessed by means of statistical tests. A more refined approach which emerged in the last years consists in analysing the *cryptographic properties* of the local rules underlying the CA, by interpreting them as *Boolean functions*. Considering Wolfram's PRNG, it turns out that rule 30 is both *balanced* and *nonlinear*, but it is not *first order correlation-immune* [21]. This is the reason why Meier and Staffelbach's attack proved to be successful. As a consequence, recent works like Formenti et al. [12] and Leporati and Mariot [15,16] focused on the search of local functions of radius 2 and 3 in order to find new rules with a better trade-off of balancedness, nonlinearity and correlation-immunity, and which can also pass stringent statistical tests (such as the NIST suite [3]) when plugged into Wolfram's PRNG model.

4 Experiments

4.1 Experimental Settings

For performing the experiments we considered only *balanced ECA*, meaning that the truth table of the local rules is composed of an equal number of zeros and ones. The reason behind this choice is that balancedness is a fundamental cryptographic criterion, and CA with unbalanced rules have an inherent statistical bias in their dynamics [16]. Hence, since our aim in this work is to investigate the resilience against asynchrony of local rules which already yield good pseudo-random sequences in the classical synchronous update scheme, we focused only on the subset of balanced rules.

The initial random seed was chosen using `https://random.org` to obtain a 64 bit initial configuration for the CA. For each CA 1000 runs with different

initial configurations were performed and 10^6 bits were generated in each run, thus producing sequences of 10^9 bits. The values k governing the asynchrony of the CA were all divisors of 64, the length of the CA configuration: 1 (synchronous behaviour), 2, 4, 8, 16, 32, and 64 (fully asynchronous behaviour).

The randomness of each sequence was assessed using the NIST test suite [3], consisting of 188 statistical tests. The quality of the pseudo-random sequences generated is thus expressed using a value from 0 (no test passed) to 188 (all tests passed). It is important to remember that, while not passing a large enough number of statistical tests indicates a weakness, even passing them all does not ensure that the PRNG employed is robust.

4.2 Experimental Results

We have experimentally observed three main behaviours depending on the level of asynchrony in CA:

1. A "phase transition" happens when enough asynchrony is present. Before the cutoff value the CA retains its ability to generate strong pseudo-random sequences. After the cutoff value most of the statistical tests fail (Section 4.3).
2. The CA ability to generate strong pseudo-random sequences increases with a limited amount of asynchrony and decreases with a large amount of it (Section 4.4).
3. A CA that is usually unsuitable to be used as a PRNG generates sequences with better pseudo-randomness once a limited amount of asynchrony is added (Section 4.5).

We are excluding from this classification the CA that did not pass a high enough level of statistical tests with any level of asynchrony. The subdivision of the remaining balanced ECA rules in the three classes is presented in Table 1. In the following we discuss the results obtained for each one of these classes.

Table 1. The subdivision in classes of balanced ECA.

Type 1	30, 45, 75, 86, 89, 101, 135, 149
Type 2	106, 120, 169, 225
Type 3	60, 90, 105, 150, 154, 165, 166, 180, 195, 210

4.3 Type 1 rules

Type 1 rules includes rule 30, which as remarked in Section 3 was among the first employed as a PRNG, even if later it was found to have some weaknesses [22]. The results for this class of rules is shown in Figure 1.

It is possible to observe that most of the rules pass all or almost all the tests when the parameter k is below 32, with 188 or 187 tests passed by each rule. The first difference can be observed when $k = 32$:

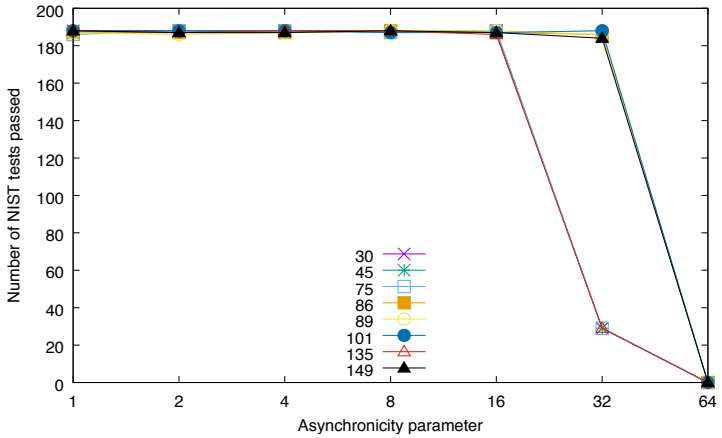

Fig. 1. The number of NIST test passed by type 1 rules with the change in asynchrony

- rules 86, 89, 101, and 149 still pass most of the tests (from a maximum of 188 for rule 101 to a minimum of 184 for rule 149);
- rules 30, 45, 75, and 135 have a sharp decrease in the number of tests passed, which is 29.

In all cases, when full asynchrony is present, none of the rules in this class can pass even one of the tests, showing that full asynchrony completely changes the behaviour of the CA.

4.4 Type 2 rules

Type 2 rules are, in some sense, similar to the ones of type 1, as it can be observed in Figure 2. With a high enough level of asynchrony (i.e., $k = 32$ or $k = 64$), they are unable to pass any statistical test of the NIST suite. It is for small levels of asynchrony that their behaviour differ. In fact, when updates happen synchronously the rules of this class are not as good as the ones of type 1, with the number of tests passed ranging from 167 (rule 169) to 172 (rules 106 and 120). When a small amount of asynchrony is added (k between 2 and 16) their ability to generate strong pseudo-random sequences increases. This is a quite interesting behaviour since it shows that asynchrony is not always an hindering factor for using CA as PRNG, but can be also employed to strengthen them.

4.5 Type 3 rules

Possibly the most interesting class of rules is the one where asynchrony is an essential factor in enabling the generation of strong pseudo-random sequences, as it can be observed in Figure 3.

All of these rules have in common the fact that they pass none of the NIST tests when the updates are synchronous. Once asynchrony is added the behaviour

The final publication is available at Springer via https://doi.org/10.1007/978-3-319-99813-8_39

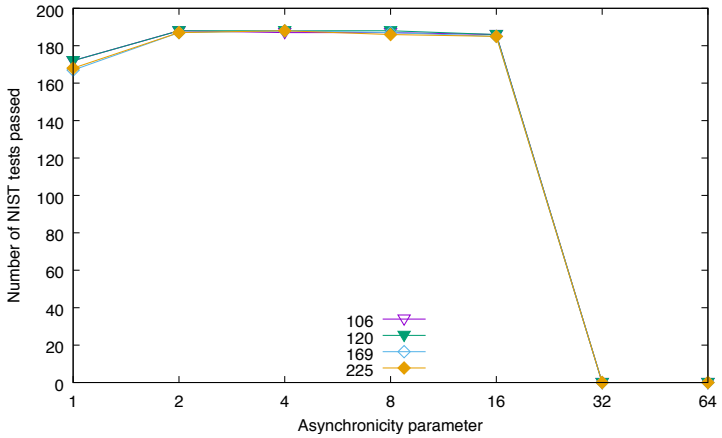

Fig. 2. The number of NIST test passed by type 2 rules with the change in asynchrony

changes drastically. As it is possible to observe in Table 2, the maximum number of NIST statistical tests passed is, for most of the rules neat the maximum (188). Among the rules considered, there are simple ones, like rule 90, the "traffic rule", whose behaviour is, in the synchronous case, extremely predictable since, after n steps (in our case $n = 64$) an attacker has enough information on the CA to predict exactly its dynamics. When asynchrony is introduced this ceases to be true and, while there is no assurance that similar predictions are not possible, the statistical tests are unable to expose any clear regularity in the resulting data.

An observation of the results, however, shows that not all rules in this class share exactly the same behaviour, even if, in the general trend, they are all quite similar. Therefore, we can further subdivide the rules of this class into four distinct sub-classes:

1. rules 154 and 166 already show increased scores in the tests with $k = 2$, showing that even a limited amount of asynchrony is sufficient;
2. rules 180 and 210 perform similarly to 154 and 166, but they show a decrease for $k = 32$ that is not present in the latter two rules;
3. rules 60, 105, 150, and 195 require more asynchrony ($k = 4$) before reaching high enough scores in the statistical tests;
4. rules 90 and 165 are able to pass more tests than any other rule for full asynchrony (51 and 52 tests, respectively).

In particular, the last case is of particular interest, since it seems to highlight that the two considered rules have some characteristic that is able to counteract, in a limited way, the effect of full asynchrony. It could be interesting to understand what this characteristic is in order to take it into account in the design of new CA-based PRNG.

The final publication is available at Springer via https://doi.org/10.1007/978-3-319-99813-8_39

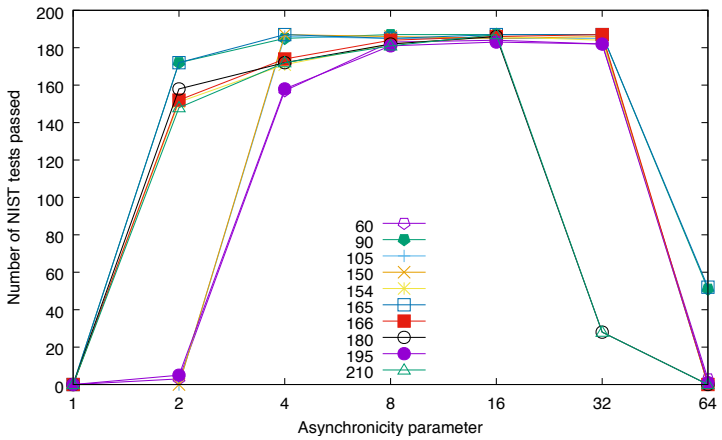

Fig. 3. The number of NIST test passed by type 3 rules with the change in asynchrony

Table 2. The maximum number of tests passed by the rules of type 3 together with the value of the asynchrony parameter where the maximum was reached.

Rule	*60*	*90*	*105*	*150*	*154*	*165*	*166*	*180*	*195*	*210*
Max score	184	187	186	187	186	187	187	186	183	187
k	16	8, 16, 32	4	4	16	4, 16, 32	32	16	16	16

5 Conclusions

In this paper we have explored the effect of increasing levels of asynchrony in ECA used as PRNG. Since they can be employed in cryptographic applications, it is important to understand what is the edge that an attacker can gain by disturbing the global clock regulating the update of the cells. Three different interesting behaviours were found. The least unexpected one is the type 1 behaviour, where there is an abrupt decrease in the pseudo-randomness quality of the sequences generated when asynchrony increases. Similar to the first class, type 2 CA exhibit a more complex behaviour, where a limited amount of asynchrony produces an increase in the pseudo-randomness quality of the generated sequences, while a further increase greatly reduces it. Finally, CA of type 3 are usually unsuited to be used as PRNG, but a limited amount of asynchrony make them competitive with the traditional rules employed for pseudo-random number generation. It is noticeable the fact that there are no CA where the decrease in quality is smoother; it appears as if the qualities necessary for obtaining a good PRNG are "binary": they are either almost all present or almost all absent.

This preliminary study opens many different possibilities for exploring the relationship between pseudo-randomness and asynchrony. It is currently unknown if the same behaviours can also be found in CA with radius grater than 1 or if new

behaviours will appear. The results found for CA of type 3 open a lot of questions on why such CA need asynchrony to generate pseudo-random sequences: what are the factors that make them predictable when synchronous and unpredictable when asynchronous? Moreover, the way asynchrony has been introduced in this study is quite limited: the updates are always performed in contiguous blocks of the same size. It would be interesting to study if different updating patterns produce different behaviours or if the observed ones are all the possible ones.

Finally, another direction for further research is to relate the results presented in this paper with the cryptographic properties of the considered local rules. An interesting starting point could be to compare the three classes of rules observed in our experiments with respect to the property of *asynchrony immunity*, recently introduced in [20]. Of course, this line of research could also be generalised to rules of higher radius.

References

1. Balzter, H., Braun, P.W., Köhler, W.: Cellular automata models for vegetation dynamics. Ecological modelling 107(2), 113–125 (1998)
2. Bandini, S., Rubagotti, F., Vizzari, G., Shimura, K.: A cellular automata based model for pedestrian and group dynamics: Motivations and first experiments. In: Parallel Computing Technologies - 11th International Conference, PaCT 2011, Kazan, Russia, September 19-23, 2011. Proceedings. pp. 125–139 (2011)
3. Bassham III, L.E., Rukhin, A.L., Soto, J., Nechvatal, J.R., Smid, M.E., Barker, E.B., Leigh, S.D., Levenson, M., Vangel, M., Banks, D.L., et al.: Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications (2010)
4. Cappuccio, R., Cattaneo, G., Erbacci, G., Jocher, U.: A parallel implementation of a cellular automata based model for coffee percolation. Parallel Computing 27(5), 685–717 (2001)
5. Chopard, B.: Cellular automata modeling of physical systems. In: Encyclopedia of Complexity and Systems Science, pp. 865–892. Springer (2009)
6. Daemen, J., Govaerts, R., Vandewalle, J.: A framework for the design of one-way hash functions including cryptanalysis of damgård's one-way function based on a cellular automaton. In: Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings. pp. 82–96 (1991)
7. Damgård, I.: A design principle for hash functions. In: Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. pp. 416–427 (1989)
8. Dennunzio, A., Formenti, E., Manzoni, L., Mauri, G.: m-Asynchronous cellular automata: From fairness to quasi-fairness. Natural Computing 12(4), 561–572 (2013)
9. Fatès, N., Morvan, M.: An experimental study of robustness to asynchronism for elementary cellular automata. Complex Systems 16(1), 1–27 (2005)
10. Fatès, N., Morvan, M., Schabanel, N., Thierry, E.: Fully asynchronous behaviour of double-quiescent elementary cellular automata. Theoretical Computer Science 362, 1–16 (2006)
11. Fatès, N., Regnault, D., Schabanel, N., Thierry, E.: Asynchronous behavior of double-quiescent elementary cellular automata. In: Correa, J.R., Hevia, A., Kiwi,

M. (eds.) LATIN 2006: Theoretical Informatics, Lecture Notes in Computer Science, vol. 3887, pp. 455–466. Springer (2006)
12. Formenti, E., Imai, K., Martin, B., Yunès, J.: Advances on random sequence generation by uniform cellular automata. In: Computing with New Resources - Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday. pp. 56–70 (2014)
13. Kanai, M., Nishinari, K., Tokihiro, T.: Stochastic cellular-automaton model for traffic flow. In: Cellular Automata, 7th International Conference on Cellular Automata, for Research and Industry, ACRI 2006, Perpignan, France, September 20-23, 2006, Proceedings. pp. 538–547 (2006)
14. Klein, A.: Stream ciphers. Springer (2013)
15. Leporati, A., Mariot, L.: 1-resiliency of bipermutive cellular automata rules. In: Cellular Automata and Discrete Complex Systems - 19th International Workshop, AUTOMATA 2013, Gießen, Germany, September 17-19, 2013. Proceedings. pp. 110–123 (2013)
16. Leporati, A., Mariot, L.: Cryptographic properties of bipermutive cellular automata rules. J. Cellular Automata 9(5-6), 437–475 (2014)
17. Luca Manzoni, Hiroshi Umeo: The firing squad synchronization problem on CA with multiple updating cycles. Theoretical Computer Science 559, 108–117 (2014)
18. Manzoni, L.: Asynchronous cellular automata and dynamical properties. Natural Computing 11(2), 269–276 (2012)
19. Manzoni, L., Porreca, A.E., Umeo, H.: The Firing Squad Synchronization Problem on Higher-dimensional CA with Multiple Updating Cycles. In: 4th International Workshop on Applications and Fundamentals of Cellular Automata - AFCA 2016. Hiroshima, Japan (November 2016)
20. Mariot, L.: Asynchrony immune cellular automata. In: Cellular Automata - 12th International Conference on Cellular Automata for Research and Industry, ACRI 2016, Fez, Morocco, September 5-8, 2016. Proceedings. pp. 176–181 (2016)
21. Martin, B.: A walsh exploration of elementary CA rules. J. Cellular Automata 3(2), 145–156 (2008)
22. Meier, W., Staffelbach, O.: Analysis of pseudo random sequence generated by cellular automata. In: Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. pp. 186–199 (1991)
23. Shackleford, B., Tanaka, M., Carter, R.J., Snider, G.: FPGA implementation of neighborhood-of-four cellular automata random number generators. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA 2002, Monterey, CA, USA, February 24-26, 2002. pp. 106–112 (2002)
24. Sipper, M., Tomassini, M.: Computation in artificially evolved, non-uniform cellular automata. Theor. Comput. Sci. 217(1), 81–98 (1999)
25. Ulam, S.: Random processes and transformations. In: Proceedings of the International Congress on Mathematics. vol. 2, pp. 264–275 (1952)
26. Von Neumann, J.: Theory of self-reproducing automata. Edited by Burks, Arthur W. University of Illinois Press (1966)
27. Wolfram, S.: Cryptography with cellular automata. In: Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings. pp. 429–432 (1985)
28. Worsch, T.: A note on (intrinsically?) universal asynchronous cellular automata. In: Proceedings of Automata 2010, pp. 339–350. 14-16 June 2010, Nancy (France) (2010)