International Journal of Pure and Applied Mathematics

Volume 4 No. 1 2003, 81-97

WOLFRAM CELLULAR AUTOMATA AND THEIR CRYPTOGRAPHIC USE AS PSEUDORANDOM BIT GENERATORS

R. Díaz Len[1], A. Hernández Encinas[2], L. Hernández Encinas[3] §
S. Hoya White[4], A. Martín del Rey[5], G. Rodríguez Sánchez[6]
I. Visus Ruíz[7]

[1]Departamento Matemática Aplicada, E.P.S.
Universidad de Salamanca
C/ Santo Tomás s/n, 05003-Ávila, SPAIN
e-mail: rauldlen@usal.es

[2,4,5,6,7]Departamento Matemática Aplicada, E.T.S.I.I.
Universidad de Salamanca
Avd. Fernando Ballesteros 2, 37700-Béjar, Salamanca, SPAIN
[2]e-mail: ascen@usal.es [4]e-mail: sarahw@usal.es
[5]e-mail: delrey@usal.es [6]e-mail: gerardo@usal.es
[7]e-mail: ivisus@usal.es

[3]Instituto de Física Aplicada, CSIC
C/Serrano 144, 28006-Madrid, SPAIN
e-mail: luis@iec.csic.es

Abstract: The cryptographic use of Wolfram cellular automata as pseudorandom bit generators and their properties, are presented. After to apply several statistical tests with cryptographic significance and the linear complexity test to these cellular automata, a classification of them is given. Moreover, a cryptanalytic attack to the sequences generated by the cellular automata with good pseudorandom properties is analyzed.

AMS Subject Classification: 94A60, 68W20, 68Q80, 11K45
Key Words: cellular automata, cryptography, pseudorandom bit generators

Received: November 19, 2002

§Correspondence author

1. Introduction

Random numbers have an important role in several fields, such as Monte Carlo methods, genetic algorithms, design of VLSI circuits, cryptographic systems, etc. (see Niederreiter [17] and the references therein). There exist mainly two methods for generating random numbers: by hardware and by software. In general, the first method is based on physical phenomena; whereas the second one is based on an algorithm (c.f. Goldreich et al [4], Lagarias [9], and Micali et al [14]). One of the characteristics of the second method is that the same sequence is obtained each time the same parameter of the algorithm is used. For this reason these sequences are called *pseudorandom*. Using deterministic algorithms is more easy and computationally cheaper; hence, more attractive. Moreover, in most cases, one only needs to guarantee that the algorithm has good random properties as number generator, *i.e.*, the obtained sequences seem to be random and have large period. Nevertheless, these properties are not sufficient in cryptography, for example.

As it is known, the main goal of cryptography is to assure the secrecy and confidentiality of communications among users, who interchange information by an insecure channel (c.f. Menezes et al [13] and Mollin [15]). Opposite, the goal of cryptanalysis is to break this secrecy and confidentiality. Pseudorandom bit generators are used to encrypt a plaintext by using a stream cipher. In this situation, the binary sequence defined by the plaintext is added, bit by bit, with the bit sequence generated by the pseudorandom bit generator. The secret key is the seed used in the generator to produce the sequence. Hence, good random properties of the generator are convenient to prevent statistical attacks, but moreover, it is necessary that the generator must be sure. The security, in this sense, means that the probability that an algorithm can produce in a polynomial time the next bit of a given sequence, is negligible.

In this paper, we are interested in the use of cellular automata as pseudorandom bit generators in relation to their cryptographic properties. Hence, we will study their pseudorandom properties, as well as their cryptographic security. The rest of this paper is organized as follows: In Section 2 we recall some of the most used pseudorandom number generators and the definition of the linear complexity of a sequence. The definition and properties of cellular automata, and in particular Wolfram cellular automata, are reviewed in Section 3. In Section 4 we present several statistical tests to determine the goodness of the pseudorandom bit generators. Later, a classification of Wolfram cellular automata according to their behavior in relation to these tests is determined in Section 5. A cryptanalytic attack against several classes of Wolfram cellular

automata is analyzed in Section 6. Finally, the conclusions are presented in Section 7.

2. Pseudorandom Number Generators and Linear Complexity

There are many pseudorandom generators (PRG) and a general review of them is presented in Knuth [8, Chapter 3]. Among them, the more extended due to their good random properties, long periods, easy implementation and computational efficiency, are the following:

1. *Linear congruential generators*: They are based on the formula

$$x_{i+1} \equiv ax_i + b \,(\mathrm{mod}\, m)\,, \quad i \geq 0,$$

where a is the multiplier, b is a constant, m is the modulus, and x_0 is the seed.

2. *Fibonacci generators*: This class of generators are defined by the formula:

$$x_{i+1} \equiv (x_{i-j} * x_{i-k})\,, \quad i \geq \min(j,k)\,,$$

where $*$ stands for some of the following binary operators: $+$, $-$, $\cdot$, $/$, or $\oplus$ (XOR).

3. *Linear feedback shift register (LFSR)*: A linear feedback shift register of length L consists of L stages, $0, 1, \ldots, L-1$, each of them can store one bit and have one input and one output (see Golomb [5]). A clock controls the movement of data and during each unit of time the following operations are performed: (a) The content of stage 0 is output and forms part of the output sequence; (b) the content of stage i is moved to stage $i-1$, $1 \leq i \leq L-1$; and (c) the content of stage $L-1$ is the feedback bit. The output of a sequence is determined by the following recursive expression:

$$s_{i+1} \equiv (a_1 s_{i-1} + a_2 s_{i-2} + \ldots + a_L s_{i-L}) \,(\mathrm{mod}\, 2)\,,$$

where $a_i \in \mathbb{Z}_2$, $1 \leq i \leq L-1$, and $a_L = 1$. If the initial content of stage i is $s_i \in \{0,1\}$, $1 \leq i \leq L-1$, then $[s_{L-1}, \ldots, s_1, s_0]$ is called the *initial state* of the LFSR. It is said that a LFSR generates a sequence s (finite or infinite) if there exists some initial state for which the output sequence of the LFSR is s.

The *linear complexity* of a bit sequence s is defined as the length of the shortest LFSR that generates the given sequence, and it is denoted by $L(s)$. If L_i, $i \geq 1$, is the linear complexity of the finite subsequence $s^i = s_1, s_2, \ldots, s_i$, of the sequence s, then the sequence $L_1, L_2, \ldots$ is called the *linear complexity profile* of s. This sequence can be plotted by representing the points (i, L_i), $i \geq 1$, and joining them by horizontal and vertical segments. It is clear that the graph of a linear complexity profile is not decreasing and that the expected linear complexity of a random sequence should be closely follow the line $L = i/2$. The Berlekamp-Massey algorithm (c.f. Menezes et al [13, §6.2.3]) is an efficient algorithm to determine the linear complexity of a finite sequence of length n.

Remark that, as in the case of statistical tests for pseudorandomness, for a sequence to have a linear complexity profile closely to that of a random sequence, is a necessary but not a sufficient condition to be considered as random. For example, linear congruential generators pass the statistical tests of pseudorandomness (see Menezes et al [13, §5.4.4]), but it has been proved (see Lagarias et al [10] and Plumstead [19]) that these generators are predictable. These results are still true if only the sequence of the less significant bits of x_i, $i \geq 1$, is known (see Boyar [1, 2]). For these reasons, linear generators are not suitable to be use in cryptography.

In the last years, generators defined by polynomials recurrences of the form $x_{i+1} \equiv f(x_i) \pmod{m}$, $i \geq 0$, where $f(x)$ is a polynomial function, have been studied. In this way, the properties of their multidimensional distribution are studied in Gutierrez et al [7] and Shparlinski [20]. Moreover, number generators based on the difficulty of mathematical and computational problems, as the discrete logarithm problem, are proposed (see Genaro [3]).

3. Linear Finite Cellular Automata as Pseudorandom Bit Generators

Cellular automata –CA for short– are discrete dynamical systems formed by a finite or infinite number of identical objects called *cells*. These cells are endowed with a state, which changes at every discrete step of time according to a deterministic rule. The most used CA are finite and linear. More precisely, they can be defined as a 4-uplet $\mathcal{A} = (C, S, V, f)$, where C is the *cellular space*, which is a linear array of m cells. Each cell is denoted by $\langle i \rangle$, $0 \leq i \leq m-1$. S is the finite *state set*, that is, it is the set of all possible values of the cells. In general is taken $S = \mathbb{Z}_k$. The *set of indices* of C is the ordered finite set $V \subset \mathbb{Z}$, such that for every cell $\langle i \rangle \in C$, its neighborhood $V_{\langle i \rangle}$ is the ordered set of m cells given by

$$V_{\langle i \rangle} = \{\langle i + \alpha_1 \rangle, \ldots, \langle i + \alpha_m \rangle : \alpha_i \in V\}.$$

The most used neighborhoods are symmetric in the sense that the cell $\langle i \rangle$ is the central cell of the set $V_{\langle i \rangle}$, and in this case, the radius is the value $r = (m-1)/2$.

Moreover, the *local transition function* $f\colon S^m \to S$ is the function determining the evolution of the CA throughout the time, *i.e.*, the changes of the states of every cell taking the states of its neighbors into account. Hence, if $a_i^{(t)} \in S$ stands for the state of the cell $\langle i \rangle$ at time t, its next state is given by the following expression:

$$a_i^{(t+1)} = f(a_{i+\alpha_1}^{(t)}, \ldots, a_{i+\alpha_m}^{(t)}).$$

If the neighborhood is symmetric of radius r, this expression becomes

$$a_i^{(t+1)} = f\left(a_{i-r}^{(t)}, \ldots, a_i^{(t)}, \ldots, a_{i+r}^{(t)}\right). \tag{1}$$

The set of states of all cells in a time t is called the *configuration at time* t and it is represented by the vector:

$$C^{(t)} = \left(a_0^{(t)}, a_1^{(t)}, \ldots, a_{n-1}^{(t)}\right) \in S \times \overset{(n}{\cdots} \times S.$$

In particular, $C^{(0)}$ is the *initial configuration*. Finally, as the cellular space is finite, boundary conditions must be established in order to assure that the evolution of the cellular automata is well-defined.

An interesting property of the CA is to be left-toggle, right-toggle or both. A CA is *left-toggle* if the equation (1) holds and it is verified that

$$1 - a_i^{(t+1)} = f(1 - a_{i-r}^{(t)}, \ldots, a_i^{(t)}, \ldots, a_{i+r}^{(t)}). \tag{2}$$

In a similar way, a CA is *right-toggle* if the equation (1) holds and it verifies

$$1 - a_i^{(t+1)} = f(a_{i-r}^{(t)}, \ldots, a_i^{(t)}, \ldots, 1 - a_{i+r}^{(t)}). \tag{3}$$

In the particular case of *Wolfram cellular automata* (WCA), the state set is $S = \mathbb{Z}_2$ (see Wolfram [23]), the neighborhoods are symmetric of radius $r = 1$, hence $V = \{-1, 0, 1\}$, the transition rule is given by the formula

$$a_i^{(t+1)} = f(a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}), \tag{4}$$

and the boundary conditions are defined by:

$$a_i^{(t)} = a_j^{(t)} \Leftrightarrow i \equiv j \mod (n-1).$$

As for WCA is $|S| = 2$ and $|V| = 3$, there are 2^8 different values for the rule $f\colon S^3 \to S$, say $f(0,0,0) = f_0$, $f(0,0,1) = f_1$, $f(0,1,0) = f_2$, $f(0,1,1) = f_3$, $f(1,0,0) = f_4$, $f(1,0,1) = f_5$, $f(1,1,0) = f_6$, and $f(1,1,1) = f_7$. Hence, it is possible to assign a unique value w, $0 \leq w \leq 255$, to each Wolfram cellular automaton WCA(w), such that this value will be the number of the CA. The number is determined by the following formula

$$w = \sum_{i=0}^{7} 2^i f_i.$$

For example, if $f_0 = f_5 = f_6 = f_7 = 0$, and $f_1 = f_2 = f_3 = f_4 = 1$, we have $w = 30$, whose expression is any of the following:

$$\begin{aligned} a_i^{(t+1)} &= \left(a_{i-1}^{(t)} + a_i^{(t)} + a_{i+1}^{(t)} + a_i^{(t)} \cdot a_{i+1}^{(t)}\right) (\mathrm{mod}\, 2) \\ &= a_{i-1}^{(t)} \text{ XOR } \left(a_i^{(t)} \text{ OR } a_{i+1}^{(t)}\right). \end{aligned} \tag{5}$$

In Section 2 we have presented some of the most important PRG. Here, we consider the use of WCA as pseudorandom bit generators (PRBG). WCA(30) was the first WCA proposed as a PRBG by Wolfram [24], and from then, many other works have studied the problem of generating numbers in a pseudorandom way by using CA (c.f. de la Guía et al [6], Nandi et al [16], Tomassini et al [21], and Tomassini et al [22]).

Here we are interested in the use of WCA as PRBG from the point of view of cryptography. A process to obtain a bit sequence by using a WCA is to consider an initial configuration of n cells of the WCA, $C^{(0)}$, and to iterate it k times. As the state set is $S = \mathbb{Z}_2$, after determining the evolution of this WCA,

k configurations of n bits are obtained, $C^{(i)} = \left(a_0^{(i)}, \ldots, a_{n-1}^{(i)}\right)$, $1 \leq i \leq k$. Hence, by linking together the configurations, a bit sequence of $k \cdot n$ bits is obtained:

$$s = a_0^{(1)}, \ldots, a_{n-1}^{(1)}, a_0^{(2)}, \ldots, a_{n-1}^{(2)}, \ldots, a_0^{(k)}, \ldots, a_{n-1}^{(k)}.$$

Nevertheless, knowing the whole evolution of a WCA is a weakness for its cryptographic use. Hence, it is necessary to determine a bit sequence by a different process. For example, one can consider the time evolution of the central cell. In this way, after iterating k times the WCA whose initial configuration has $2n + 1$ cells, a sequence of k bits is obtained:

$$s = a_n^{(1)}, a_n^{(2)}, \ldots, a_n^{(k)}.$$

This process is more expensive than the previous one, but it is safer for cryptographic purposes, because only one bit of each time iteration is known.

4. Statistical Tests for Pseudorandomness

To assure good pseudorandom properties of a bit sequence, it has to pass several statistical tests. There are many tests designed for this purpose (see Knuth [8, Chapter 3], Marsaglia [11], and Niederreiter [18, Chapter 7]). Nevertheless, we will use the five basic tests proposed by Menezes et al [13, §5.4.4]. These tests have been developed *ad hoc* for cryptographic use and they are based on Golomb's randomness postulates (see Golomb [5]).

The *frequency test* has the purpose of determining whether the number of 0's and 1's in the sequence $s = s_0, \ldots, s_{n-1}$ are approximately the same, as it is expected for a truly random sequence. If n_0, n_1 denotes the number of 0's and 1's in s, respectively, the statistic considered (which follows a χ^2 distribution with 1 degree of freedom if $n \geq 10$) is:

$$X_f = \frac{(n_0 - n_1)^2}{n}.$$

The *serial test* tries to determine if the number of pairs 00, 01, 10, and 11 in s, are approximately the same. If n_{00}, n_{01}, n_{10}, and n_{11} are, respectively, the number of such occurrences, the statistic used (which follows a χ^2 distribution with 2 degrees of freedom if $n \geq 21$) is:

$$X_s = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1.$$

Let m be an integer, such that $\lfloor n/m \rfloor \geq 5 \cdot 2^m$ and let $k = n/m$. In the *poker test* the sequence s is divided into k non-overlapping parts of length m. Let n_i be the number of occurrences of the i-th type of sequences of length m, such that each of them appears the same number of times in s. Then, the statistic considered (which follows a χ^2 distribution with $2^m - 1$ degrees of freedom) is:

$$X_p = \frac{2^m}{k} \left(\sum_{i=1}^{2^m} n_i^2 \right) - k.$$

A *gap* (resp. a *block*) of a sequence is a subsequence of s consisting of 0's (resp. 1's) between 1's (resp. 0's). Gaps and blocks are called *runs*. As the expected number of runs of length i in a random sequence of length n is $e_i = (n - i + 3)/2^{i+2}$, the purpose of the *runs test* is to check if the number of runs of several lengths in s is as expected in a random sequence. Let k be the largest integer i for which $e_i \geq 5$ and let G_i and B_i the number of gaps and blocks of length i, such that $1 \leq i \leq k$. The statistic used for this test (which follows a χ^2 distribution with $2k - 2$ degrees of freedom) is:

$$X_r = \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i}.$$

The *autocorrelation test* determines the correlations between the sequence s and non-cyclic shifted versions of s. Let d be an integer such that $1 \leq d \leq \lfloor n/2 \rfloor$. The number of bits in s not equal to their d-shifts is given by

$$A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}.$$

The statistic used (which follows a $N(0,1)$ distribution if $n - d \geq 10$) is the following:

$$X_a = 2 \frac{A(d) - \frac{n-d}{2}}{\sqrt{n-d}}.$$

WCA number	X_f	X_s	X_p	X_r	X_a
30	4	5	4	6	8
45	9	3	9	7	3
60	8	5	10	11	6
75	5	8	5	9	8
86	3	6	6	7	6
89	1	3	2	6	6
90	4	6	5	7	5
101	5	5	5	6	5
102	6	4	1	6	1
105	3	0	3	4	5
106	7	8	7	7	7
120	4	3	5	11	5
122	4	4	2	6	4
126	7	6	7	3	11
129	6	5	6	6	5
135	8	8	5	5	1
149	8	7	7	6	1
150	4	3	12	7	4
153	6	5	8	10	7
161	5	6	8	12	6
165	8	7	4	12	7
169	8	7	3	5	3
195	3	3	7	11	5
225	7	5	5	6	7

Table 1. % of rejected sequences for the WCA, which have passed the first sieve

5. Pseudorandomness Classification of Wolfram Cellular Automata

In this section we present the behavior of all WCA with relation to the statistical test presented in Section 4 and their linear complexity. In relation to the pseudorandom tests, we have considered four sieves such that in each of them, 100 sequences for each WCA studied have been generated. The significance level considered for each test has been $\alpha = 0.05$ and we have rejected a WCA if the number of sequences, which do not pass any of the tests is bigger than 20%.

In the first sieve all the 256 WCA have been studied and for each of them we have analyzed 100 different sequences. The length of the initial configuration of each WCA was of 300 cells and they have been iterated 1000 times. Hence, each bit sequence has a length $k = 1000$. Only 24 of the all 256 WCA have pass this sieve and they are shown in Table 1.

WCA number	X_f	X_s	X_p	X_r	X_a
30	7	3	6	3	5
45	2	6	3	12	9
60	3	6	6	6	5
75	9	5	5	3	4
86	5	5	4	6	8
89	5	4	6	4	2
90	4	2	6	6	4
101	5	5	3	7	8
102	4	5	8	4	6
105	7	9	7	4	2
106	5	3	6	8	3
120	3	3	3	2	6
135	8	5	6	4	3
149	2	5	2	1	6
150	6	14	6	11	8
153	6	5	8	10	7
161	10	6	10	4	8
165	4	4	4	7	7
169	6	6	6	10	4
195	5	7	8	4	2
225	4	6	5	10	5

Table 2. % of rejected sequences for the WCA, which have passed the fourth sieve

In the second sieve only the remaining 24 WCA have been studied. For this occasion, the length of each initial configuration was of 500 cells and each of the 100 sequences for each WCA have been iterated 2500 times. After this study, only the WCA(126) did not pass this sieve. The third sieve was similar to the previous ones. In this case the length of each sequence was of 5000 bits and only the WCA(129) did not pass the sieve. Hence, only rest 22 WCA in the study. For the fourth sieve the number of bits of each sequence for the rest of the WCA was of 10000. Again, only one WCA did not pass this fourth sieve, the given by the number 122. After to perform the fourth sieve, the results obtained for

the remaining WCA are shown in Table 2. Hence, we can conclude that only 21 WCA have pass the five pseudorandom tests considered.

After this analysis, we have studied the linear complexity of each WCA survivor in order to guarantee another necessary condition of pseudorandomness as it was mentioned in Section 2. In this way, we have determined the linear complexity for several sequences of $k = 10000$ bits for each WCA and these results are shown in Table 3.

WCA number	Interval of the linear complexity
30	$4998 - 5002$
45	$4999 - 5002$
60	$992 - 1000$
75	$5000 - 5002$
86	5001
89	$5001 - 5002$
90	496
101	5001
102	$992 - 1000$
105	498
106	$5000 - 5001$
120	$4999 - 5001$
135	5000
149	5000
150	494
153	$993 - 1000$
161	$2616 - 2895$
165	497
169	$4999 - 5000$
195	$993 - 1000$
225	$4999 - 5000$

Table 3. Linear complexity of the WCA survivor

The linear complexity profiles of two extreme cases in Table 2 –numbers 30 and 150– are shown in Figures 1 and 2.

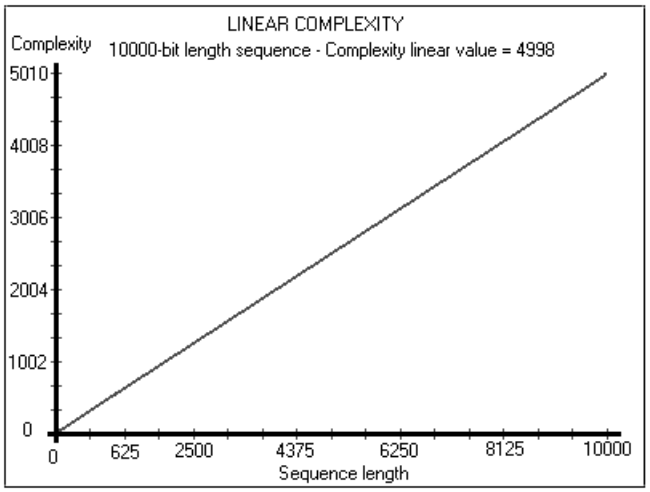

Figure 1: Linear complexity profile of WCA(30)

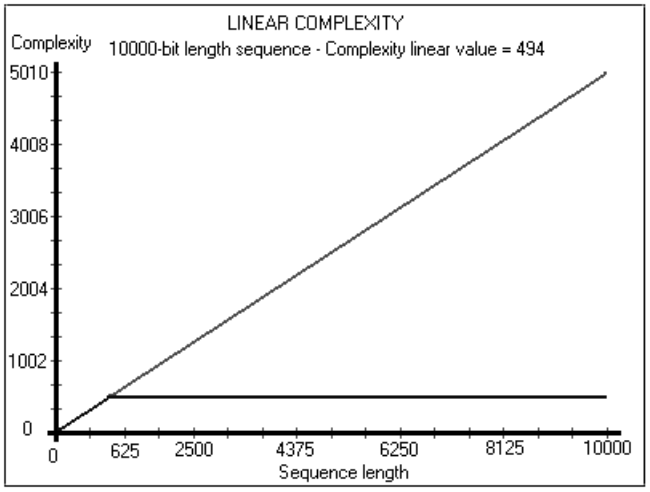

Figure 2: Linear complexity profile of WCA(150)

After this study, we conclude that the only WCA with good pseudorandom properties and linear complexity close to the line $k/2$ are those given by following numbers:

$$30, 45, 75, 86, 89, 101, 106, 120, 135, 149, 169, 225.$$

Note that the WCA of numbers $30, 45, 75, 120, 135,$ and 225 are left-toggle, whereas the WCA of numbers $86, 89, 101, 106, 149,$ and 169 are right-toggle. Hence, from formulas (2) (resp. (3)) and (4) it is possible to obtain $a_{i-1}^{(t)}$ (resp. $a_{i+1}^{(t)}$) as a function of $a_i^{(t)}$, $a_i^{(t+1)}$, and $a_{i+1}^{(t)}$ (resp. $a_i^{(t)}$, $a_i^{(t+1)}$, and $a_{i-1}^{(t)}$).

6. Cryptanalytic Attack

Meier and Staffelbach [12] presented an attack to the sequences generated by WCA(30), which is successful if the length of the secret key (the initial configuration of the CA) is $n < 500$ bits and it is based on the transition function of WCA(30). From formula (5) it is possible to obtain the value of $a_{i-1}^{(t)}$ as a linear function of the values of $a_i^{(t)}$, $a_i^{(t+1)}$, and $a_{i+1}^{(t)}$:

$$a_{i-1}^{(t)} = \left(a_i^{(t+1)} + a_i^{(t)} + a_{i+1}^{(t)} + a_i^{(t)} \cdot a_{i+1}^{(t)}\right) (\mathrm{mod}\, 2). \tag{6}$$

Hence, from n consecutive values of the cell $\langle i \rangle$, and $n-1$ consecutive values of its adjacent cell $\langle i+1 \rangle$, it is possible to obtain $n-1$ values of its another adjacent cell $\langle i-1 \rangle$. If the values of the cell $\langle i+1 \rangle$ are known for each time; it is possible to recover the whole evolution of the WCA from formula (6); and, in particular, the initial configuration; *i.e.*, the secret key. Hence, to know the evolution of the cell $\langle i+1 \rangle$ is equivalent to know the key. This attack permits to obtain the secret key of length $2n-1$ if n bits of the key stream sequence are known. This algorithm supposes that the evolution of the central cell is given; that is, the n values $a_i^{(t+k)}$, $k = 0, \ldots, n-1$, are known. Then $n-1$ random values for $a_{i+j}^{(t)}$, $j = 1, \ldots, n-1$, are generated. From these values, it is possible to determine the values of all cells forming the right triangle of the configuration of the WCA considered, *i.e.*, the values of the cells $a_{i+j}^{(t+k)}$, where $j = 1, \ldots, n-k$, and $k = 1, \ldots, n-1$. Later the values of the cells forming the left triangle of the evolution of the WCA are computed: $a_{i-j}^{(t+k)}$, with $k = 0, \ldots, n-j$, and $j = 1, \ldots, n-1$. In this way, an initial configuration of the WCA is obtained, and hence, a secret key that generates the same evolution of the central cell $\langle i \rangle$. Note, that probably the whole evolution of the WCA is not the same that the initial evolution of the WCA, because the initial configurations can be different, but in both cases, the evolution of the central cell is the same.

Due to the fact that all the WCA, which have passed the statistical tests and have good linear complexity, are left-toggle or right-toggle, we have studied the algorithm defined for WCA(30) in order to apply it for the rest of WCA.

In fact the attack can be extended to any left-toggle WCA because for all of them one can recover the value of the cell $\langle i-1\rangle$ at time t from the values of the cells $\langle i\rangle$ and $\langle i+1\rangle$ at the same time and the value of the cell $\langle i\rangle$ at time $t+1$. Moreover, for the right-toggle WCA we have implemented an algorithm, similar to the previous one for left-toggle, from which it is possible to attack successfully these WCA. The algorithm for both class of WCA can be resumed as follows:

Input: A bit sequence of length n, $s = s_0, s_1, \ldots, s_{n-1}$, and the number w of the left-toggle or right-toggle WCA.

Output: The initial configuration of the WCA of length $2n-1$.

1. `Compute the transition function of WCA`(w)`:`
$$a_i^{(t+1)} = f(a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)})$$
2. `If WCA`(w) `is left-toggle` $t \leftarrow -1$ `else` $t \leftarrow 1$
3. `Compute the function` $a_{i+t}^{(t)} \leftarrow F(a_i^{(t+1)}, a_i^{(t)}, a_{i-t}^{(t)})$
`from the formula:` $a_i^{(t+1)} = f(a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)})$
4. `For` j `from` 0 `to` $n-1$ `do` $a_n^j \leftarrow s_j$
5. `For` j `from` 1 `to` $n-1$ `do` $a_{n-t\cdot j}^0 \leftarrow \mathrm{rand}(0..1)$
6. `For` k `from` 1 `to` $n-2$ `do`
`For` j `from` 1 `to` $n-1-k$ `do`
$a_{n-t\cdot j}^k \leftarrow F\left(a_{n-t\cdot j-1}^{k-1}, a_{n-t\cdot j}^{k-1}, a_{n-t\cdot j+1}^{k-1}\right)$
7. `For` j `from` 1 `to` $n-1$ `do`
`For` k `from` $n-1-j$ `by` -1 `to` 0 `do`
$a_{n+t\cdot j}^k \leftarrow F\left(a_{n+t(j-1)}^{k+1}, a_{n+t(j-1)}^k, a_{n+t(j-2)}^k\right)$
8. `Return`$\left(a_1^0, a_2^0, \ldots, a_{2n-1}^0\right)$

In this way, we have proved that it is possible to determine the initial configuration of the WCA considered, *i.e.*, the secret key used for generating the bit sequence. Hence WCA are not suitable to be used as pseudorandom bit generators in cryptography, and their use must be restricted to others applications.

7. Conclusions

We have studied all the 256 Wolfram cellular automata for their use in cryptography as pseudorandom bit generators. To do this, we have analyzed their pseudorandom properties by means of several statistical tests with cryptographic significance, and we have determined their linear complexity. From the results obtained, we have considered only 12 WCA with good pseudorandom properties and high linear complexity. For these WCA, which are left-toggle or right-toggle, we have implemented an efficient algorithm, which computes the initial configuration of the WCA considered, and we conclude that none WCA must be used for cryptographic purposes. Hence, the future work consists in studying if hybrid or non-uniform cellular automata (their local transition function is not the same for all cells) are suitable for their use in cryptography.

Acknowledgments

The authors thanks Jaime Muñoz Masqué for his valuable comments in preparing the manuscript. This work is supported by "Memoria Samuel Solórzano Barruso" Foundation (Spain) and Ministerio de Ciencia y Tecnología (Spain) under grant TIC2001–0586.

References

[1] J. Boyar, Infering sequences produced by pseudorandom number generators, *Technical Report*, **86-002**, University of Chicago (1986).

[2] J. Boyar, Infering sequences produced by a linear congruential generator mising low order bits, *J. Cryptology*, **1** (1989), 177–184.

[3] R. Gennaro, An improved pseudo-random generator based on discrete log, *Proc. of Crypto'00*, *LNCS*, **1880** (2000), 469–481.

[4] O. Goldreich, H. Krawczyk, M. Luby, On the existence of pseudorandom generators, *Proc. of Crypto'88*, *LNCS*, **403** (1990), 57–75.

[5] S.W. Golomb, *Shift Register Sequences*, Holden-Day, San Francisco (1967).

[6] D. de la Guía Martínez, A. Peinado Domínguez, Pseudorandom number generation based on nongroup cellular automata, In: *Proc. of the 33rd*

Annual IEEE International Carnahan Conference on Security Technology, 370–376 (1999).

[7] J. Gutierrez, I. Shparlinski, A. Winterhof, On the linear and nonlinear complexity profile of nonlinear pseudorandom number generators, *Preprint* (Available at http://www.comp.mq.edu.au/˜igor/Publ.html).

[8] D.E. Knuth, *The Art of Computer Programming, Vol 2. Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Reading, MA (1998).

[9] J.C. Lagarias, Pseudorandom number generators in cryptography and number theory, *Proc. Symp. Appl. Math.*, **42** (1990), 115–143.

[10] J.C. Lagarias, J. Reeds, Unique extrapolation of polynomial recurrences, *SIAM J. Comput.*, **2**, No. 17 (1988), 342–362.

[11] G. Marsaglia, *Diehard* (Available at http://stat.fsu.edu/˜geo/diehard.html).

[12] W. Meier, O. Staffelbach, Analysis of pseudo random sequences generated by cellular automata, *Proc. of Eurocrypt'91*, *LNCS*, **547** (1991), 186–199.

[13] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL (1997).

[14] S. Micali, C.P. Schnorr, Efficient perfect polynomial random number generators, *J. Cryptology*, **3** (1991), 157–172.

[15] R.A. Mollin, *An Introducction to Cryptography*, Chapman & Hall/CRC, Boca Raton, FL (2001).

[16] S. Nandi, B.K. Kar, P.P. Chaudhuri, Theory and applications of cellular automata in cryptography, *IEEE Trans. Comput.*, **43**, No. 12 (1994), 1346–1357.

[17] H. Niederreiter, Quasi-Monte Carlo methods and pseudo-random numbers, *Bul. Amer. Math. Soc.*, **84** (1978), 957–1041.

[18] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia (1992).

[19] J. Plumstead, Infering a sequence generated by a linear congruence, In: *Proc. 23rd Ann. Symp. Found. Comput. Sci.*, 153–159 (1982).

[20] I. Shparlinski, On the linear complexity of the power generator, *Des. Codes Cryptogr.*, **23** (2001), 5–10.

[21] M. Tomassini, M. Perrenoud, Cryptography with cellular automata, *Appl. Soft Comput.*, **1** (2001), 151–160.

[22] M. Tomassini, M. Sipper, M. Perrenoud, On the generation of high-quality random numbers by two-dimensional cellular automata, *IEEE Trans.Comput.*, **49** (2000), 1146–1151.

[23] S. Wolfram, Cellular automata, *Los Alamos Science*, **9** (1983), 2–21.

[24] S. Wolfram, Cryptography with cellular automata, *Proc. of Crypto'85*, *LNCS*, **435** (1986), 416–427.