Advances on Random Sequence Generation by Uniform Cellular Automata

Enrico Formenti[1], Katsunobu Imai[2], Bruno Martin[1(✉)], and Jean-Baptiste Yunès[3]

[1] I3S-CNRS, Univ. Nice Sophia Antipolis, Nice, France
{Enrico.Formenti,Bruno.Martin}@unice.fr
[2] Graduate School of Engineering, Hiroshima University, Hiroshima, Japan
imai@iec.hiroshima-u.ac.jp
[3] LIAFA, Univ. Paris Diderot, Paris, France
Jean-Baptiste.Yunes@univ-paris-diderot.fr

Abstract. The study of cellular automata rules suitable for cryptographic applications is under consideration. On one hand, cellular automata can be used to generate pseudo-random sequences as well as for the design of S-boxes in symmetric cryptography. On the other hand, Boolean functions with good properties like resiliency and non-linearity are usually obtained either by exhaustive search or by the use of genetic algorithms. We propose here to use some recent research in the classification of Boolean functions and to link it with the study of cellular automata rules. As a consequence of our technique, this also provides a mean to get Boolean functions with good cryptographic properties.

Keywords: Cellular automata · Random number generation · Boolean functions

1 Introduction

Cellular automata (CA) are models of finite state machines used in many applications. They form a discrete model of parallelism evolving within discrete time steps according to a local updating rule. CAs are employed for the generation of cryptographic binary pseudo-random sequences [18] and for solving the firing squad problem [10]. Pseudo-random sequences (PRS) have a long history of applications to computational (Monte Carlo sampling, numerical simulation) and communications problems (coding theory, stream ciphers). In the present work, we particularly focus on the search for good local CA rules by using mathematical tools from Boolean functions. For this, we consider a CA rule as a Boolean function in several variables (from three up to five) and we search for Boolean functions that fulfill good cryptographic properties such as non-linearity and resiliency. Next, we use those good Boolean functions as CA rules that can be iterated to provide 'extended' Boolean functions (in nine variables). This work (starting with Boolean functions in four variables) requires an exhaustive search among all possible Boolean functions. The methodology that we use can

C.S. Calude et al. (Eds.): Gruska Festschrift, LNCS 8808, pp. 56–70, 2014.
DOI: 10.1007/978-3-319-13350-8_5

provide Boolean functions satisfying non-linearity and resiliency properties with a large number of variables (up to nine here). Such Boolean functions are (yet) unreachable by a classical brute force search because of the combinatorial explosion.

Such Boolean functions (or CA rules) can be used in many applications. Either directly for pseudo-random sequences generation or as updating functions for providing lightweight random sources of good quality in sensor networks. Other target applications can be joined compression and data encryption (also called co*cryption [15]), or used in hardware devices like FPGA or GPU for quickly providing randomness.

The material is organized as follows. Section 2 introduces the definitions and notations from both cellular automata and Boolean functions theories. Section 3 recalls related results. More precisely, it provides evidence that there is no rule with three variables which provides cryptographic pseudo-random sequences. It also recalls a classification from [11] which lists all equivalence classes containing rules with four variables suitable for generating cryptographic pseudo-random sequences. Section 4 presents the main contribution of the paper and gives some of the five variable rules that can be used for generating cryptographic pseudo-random sequences. In Section 5, we present some statistical testing against the sequences generated by using five variable rules selected from Section 4. Finally, Section 6 concludes the paper and proposes future research directions.

2 Definition and Notation

This section recalls some basic notation and facts on pseudo-randomness, CAs and Boolean functions.

2.1 Pseudo-Randomness

In [20], three mechanisms responsible for random behavior in systems are described: (1) *Randomness from physics* like brownian motion; (2) *Randomness from the initial conditions* which is studied by chaos theory; and (3) *Randomness by design*, also called pseudo-randomness. Many algorithms generate PRS. The behavior of the system is fully determined by knowing the seed and the algorithm used. They are quicker methods than extracting "true" randomness from the environment, inaccessible to computers.

The applications of randomness have led to many different methods for generating random data. These methods may vary as to how unpredictable or statistically random they are, and how quickly they can generate random sequences. Before the age of computational PRS, generating large amount of random numbers required a lot of work and were distributed as random number tables.

In the sequel, we will consider *pseudo-random generators* (PRG). This corresponds to a deterministic algorithm which "stretches" a short truly random sequence (the *seed*) into a polynomially longer sequence that appears to be

"random" (although it isn't). In other words, although the output of a PRG is not really random, it is (polynomially for probabilistic distinguishers) unfeasible to tell the difference. It turns out that pseudorandomness and computational complexity are linked in a fundamental way (see [8] for further details). More practically, this corresponds to the behavior of random number generators implemented in operating systems. In this case, the short truly random sequence corresponds to the pseudo-device `/dev/random` and the output of the PRG to the pseudo-device `/dev/urandom` for producing more random bits of weaker quality.

2.2 Cellular Automata

One-dimensional binary CAs consist of a line of cells taking their states among binary values. For practical implementations, the number of cells is finite. There are two cases: a CA has *periodic boundary conditions* if the cells are arranged in a ring and it has *null boundary conditions* when both extreme cells are continuously fixed to zero. All the cells are finite state machines with an updating function which gives the new state of the cell according to its current state and the current state of its nearest neighbors. For a presentation of CAs, see [9].

In [18], it was proposed to use CAs to produce PRS. Binary CAs with l cells ($l = 2N+1$ for $N \in \mathbb{N}$) were considered. For a CA, the values of the cells at time $t \geq 0$ are updated synchronously by a Boolean function f with $n = r_1 + r_2 + 1$ variables by the rule $x_i(t+1) = f(x_{i-r_1}(t), \ldots, x_i(t), \ldots, x_{i+r_2}(t))$. Elementary CAs are such that $r_1 = r_2 = 1$. For a fixed t, the sequence of the values $x_i(t)$ for $1 \leq i \leq 2N+1$, is the *configuration* at time t. It is a mapping $c : [\![1, l]\!] \to \mathbb{F}_2$ which assigns a Boolean state to each cell. The initial configuration ($t = 0$) $x_1(0), \ldots, x_l(0)$ is the *seed*, the sequence $(x_N(t))_t$ is the *output sequence* and, when $r_1 = r_2 = r$, the number r is the *radius* of the rule. The *Wolfram numbering* associates a rule number to any one of the 256 elementary CA; it takes the binary expansion of a rule number as the truth table of a 3-variable Boolean function.

2.3 Boolean Functions

A Boolean function is a mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. In the sequel, additions in $\mathbb{Z}$ (resp. $\mathbb{F}_2$) will be denoted by $+$ and Σ (resp. $\oplus$ and $\bigoplus$), products by $\times$ and $\prod$ (resp. $\cdot$ and $\prod$). When there is no ambiguity, $+$ will denote the addition of binary vectors. If x and y are binary vectors, their inner product is $x \cdot y = \sum_{i=1}^{n} x_i y_i$. A very handy representation of Boolean function is the *algebraic normal form*:

Definition 1 (ANF). *A Boolean function f with n variables is represented by a unique binary polynomial in n variables, called* algebraic normal form*: $f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u (\prod_{i=1}^{n} x_i^{u_i})$ $a_u \in \mathbb{F}_2$, u_i is the i-th projection of u.*

Example 1. The ANF of rule (30) is $x_1 \oplus x_2 \oplus x_3 \oplus x_2 x_3$ or 1+2+3+23.

The *degree of the ANF* or *algebraic degree* of f corresponds to the number of variables in the longest term $x_1^{u_1} \ldots x_n^{u_n}$ in the ANF of f. The *Hamming weight*

$w_H(f)$ of f is the number of $x \in \mathbb{F}_2^n$ such that $f(x)=1$. The *Hamming weight* $w_H(x)$ of $x \in \mathbb{F}_2^n$ counts the number of 1-valued coordinates in x. f is *balanced* if $w_H(f) = w_H(1 \oplus f) = 2^{n-1}$.

Definition 2. *f and g Boolean functions in n variables are* equivalent *iff*

$$f(x) = g\left((x \cdot A) \oplus a\right) \oplus \left(x \cdot B^T\right) \oplus b, \quad \forall x \in \mathbb{F}_2^n \tag{1}$$

where A is a non-singular binary $n \times n$ matrix, b a binary constant, a and $B \in \mathbb{F}_2^n$.

An important tool in the study of Boolean functions is the *Fourier-Hadamard transform*, a linear mapping which maps a Boolean function f to the real-valued function $\widehat{f}(u) = \sum_{x \in \mathbb{F}_2^n} f(x)(-1)^{u \cdot x}$, which describes the *spectrum* of the latter. When applied to the *sign function* $\mathrm{f}_\chi(x) = (-1)^{f(x)}$, the Fourier-Hadamard transform is the *Walsh transform*: $\widehat{\mathrm{f}_\chi}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus u \cdot x}$. Since $\mathrm{f}_\chi(u) = 1 - 2f(u)$, the Fourier-Hadamard transform is:

$$\widehat{f}(u) = \frac{1}{2} \sum_{x \in \mathbb{F}_2^n} (-1)^{u \cdot x} - \frac{1}{2} \widehat{\mathrm{f}_\chi}(u) \quad , \tag{2}$$

Using Eq. (2), we obtain that $\widehat{\mathrm{f}_\chi}(u) = 2^n \delta_0 - 2\widehat{f}(u)$, where δ_0 denotes the *Dirac symbol* defined by $\delta_0(u) = 1$ if u is the null vector and $\delta_0(u) = 0$ otherwise [4].

If f and g are two equivalent Boolean functions in n variables, it holds that:

$$\widehat{\mathrm{f}_\chi}(u) = (-1)^{a \cdot A^{-1}(u^t + B^T) + b} \, \widehat{\mathrm{g}_\chi}((u \oplus B)(A^{-1})^T) \ . \tag{3}$$

This property is used by [3] for counting the number of functions satisfying some cryptographic properties.

The Walsh transform allows to study the *correlation-immunity* of a function.

Definition 3. *A Boolean function f in n variables is* k-correlation-immune *$(0 < k < n)$ if, given any n independent and identically distributed binary random variables $x_1, \cdots, x_n$ according to a uniform Bernoulli distribution, then the random variable $Z = f(x_1, \ldots, x_n)$ is independent from any random vector $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$, $1 \le i_1 < \cdots < i_k < n$. When f is k-correlation immune and balanced, it is* k-resilient.

In [21], a spectral characterization of resilient functions was given:

Theorem 1. *A Boolean function f in n variables is k-resilient iff it is balanced and $\widehat{f}(u) = 0$ for all $u \in \mathbb{F}_2^n$ s.t. $0 < w_H(u) \le k$. Equivalently, f is k-resilient iff $\widehat{\mathrm{f}_\chi}(u) = 0$ for all $u \in \mathbb{F}_2^n$ s.t. $w_H(u) \le k$.*

Theorem 1 concerns both transforms (refer to [4] for further details).

Theorem 2 (Siegenthaler Bound). *For a k-resilient $(0 \le k < n-1)$ Boolean function in n variables, there is an upper bound for its algebraic degree d: $d \le n - k - 1$ if $k < n - 1$ and $d = 1$ if $k = n - 1$.*

2.4 Some Properties of the Fourier-Hadamard Transform

Computing the Fourier-Hadamard transform We use the Fourier-Hadamard transform from [6] called the *Walsh or Sequency Ordered Transform* $(\mathrm{WHT})_w$. This transform is used to study the CA rules in order to find the best rules for generating PR sequences, like in [7,14]. To check the rules, we use the fast transform algorithm whose time complexity is $O(n \log n)$. The algorithm receives as an input an array $\mathtt{F}$ of size $2^{\mathtt{n}}$ which contains the images by the t iterates of the local rule f of all the configurations of $\mathtt{n}$ cells naturally ordered: $f^t(0), \ldots, f^t(2^{\mathtt{n}}-1)$ and outputs the transform $\widehat{\mathtt{F}}$ in the reverse order: $\widehat{f^t}(2^{\mathtt{n}}-1), \ldots, \widehat{f^t}(0)$.

Application to CA rules We proceed step by step with increasing values of t, which counts the number of times the local rule in 5 variables, supposed to be 1-resilient, is iterated on an initial configuration. In this way, we consider the natural extension of $f : \mathbb{F}_2^5 \to \mathbb{F}_2$ to $f : \mathbb{F}_2^{n+4} \to \mathbb{F}_2^n$ where:

$$f(x_0, \ldots, x_{n+4}) = (y_1, \ldots, y_n) \text{ s.t. } y_j = f(x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+2}), j \in [\![1, n]\!]$$

Using the extended f, one can define the t-th iterate of f which is a function $f^t : \mathbb{F}_2^{4t+1} \to \mathbb{F}_2$. We compute next the maximum absolute value of the Fourier-Hadamard transform of the t^{th}-iterate of f at all the points u of Hamming weight 1 and we select the rules with a minimum spectral value.

The computation is repeated with increasing values of t until we identify rules with flat spectral or relatively small values which are slowly growing.

Some properties on the iterates In order to find other CA rules which preserve resiliency upon iterates, one can remark that the Fourier-Hadamard transform is preserved under some transformations like the *reflection* (which just takes the mirror-image of the initial configuration). Unfortunately, the other classical transformations on CAs (conjugation and conjugation-reflection) do not preserve the resiliency upon iterates in general.

Let Φ denote the *reverse operator* $\Phi : \mathbb{F}_2^m \to \mathbb{F}_2^m$, $\Phi((v_1, \ldots, v_m)) = (v_m, \ldots, v_1)$.

Definition 4. *Let* $f : \mathbb{F}_2^{2m+1} \to \mathbb{F}_2$ *be the local function of a CA. Then,* $f_R(x_{-m}, \ldots, x_0, \ldots, x_m) = \Phi \circ f(x_m, \ldots, x_0, \ldots, x_{-m})$ *is the* reflection *of* f.

Another basic transformation is given by $\Psi(x) = 1 \oplus x$ for $x \in \mathbb{F}_2$. It corresponds to the negation of the variable and is used for designing the conjugation and the conjugation-reflection introduced in [19, p. 492]. With some abuse of notation, Ψ is extended to sequences of Boolean variables: for $u = (u_1, u_2, \ldots, u_n)$ with $u_i \in \mathbb{F}_2$, $\Psi(u) = (\Psi(u_1), \Psi(u_2), \ldots, \Psi(u_n))$. Moreover, $\Psi^{-1} = \Psi$.

Definition 5. *Let* $f : \mathbb{F}_2^{2m+1} \to \mathbb{F}_2$ *be the local function of a CA. Then* $f_N(x_{-m}, \ldots, x_0, \ldots, x_m) = \Psi \circ f(\Psi(x_m, \ldots, x_0, x_{-m}))$ *is the* negation *of* f.

One can see that for any $t \in \mathbb{N}$, $f^t \circ H = H \circ f_R^t$ for $H = \Psi$ or $H = \Phi$.

Lemma 1. *Let* $\Xi : \mathbb{F}_2^{2m+1} \to \mathbb{F}_2^{2m+1}$ *be 1:1 and* $f \colon \mathbb{F}_2^{2m+1} \to \mathbb{F}_2$ *a CA. Then,* $w_H(f) = w_H(f \circ \Xi)$.

Proposition 1 shows that resiliency is preserved by the reflection when the local rule is iterated.

Proposition 1. *Let* $f : \mathbb{F}_2^{2m+1} \to \mathbb{F}_2$ *be the local function of a CA. For any* $t \in \mathbb{N}$, *let* $0 < k \leq 2mt + 1$. *Then,* f_R^t *is* k*-resilient iff* f^t *is* k*-resilient.*

Proof. The transformation Φ is bijective. Hence, by Lemma 1, we have $w_H(f_R) = w_H(f \circ \Phi) = w_H(f)$. Since f is balanced, $w_H(f_R) = w_H(\Psi \circ f)$. Now, applying Lemma 1 to $\Psi \circ f$ and using last equation, it holds $w_H(\Psi \circ f) = w_H(\Psi \circ f \circ \Phi) = w_H(\Psi \circ f_R)$. Let $a = B = (0, 0, \ldots, 0)$, $b = 0$ and A the reverse identity matrix. Remark that A is non-singular, then, by using Eq. 3, one obtains $\widehat{(\mathrm{f}_R^t)_\chi}(u) = \widehat{\mathrm{f}}_\chi^t(u \cdot (A^{-1})^T) = \widehat{\mathrm{f}}_\chi^t(u \cdot A) = \widehat{\mathrm{f}}_\chi^t(A \cdot u)$ which entails

$$\widehat{f_R^t}(u) = \widehat{f}^t(A \cdot u) \ . \tag{4}$$

Now, assume that f^t is k-resilient. Remark that $w_H(A \cdot u) = w_H(u)$ for any u, therefore, by Theorem 1, if $\widehat{f}^t(u) = 0$ for $0 < w_H(u) \leq k$, then, by Eq. 4, $\widehat{f_R^t}(u) = 0$ too. For the converse, just remark that A^2 is the identity transformation and then, by Eq. 4, one finds $\widehat{f_R^t}(\Phi(u)) = \widehat{f}^t(u)$. Therefore if $\widehat{f_R^t}(\Phi(u)) = 0$, we have $\widehat{f}^t(u) = 0$. Since Φ is a bijection we have the thesis.

Lemma 2. *Let* $f : \mathbb{F}_2^{2m+1} \to \mathbb{F}_2$ *be the local function of a CA. For any* $t \in \mathbb{N}$, f_N^t *is balanced iff* f^t *is balanced.*

Proof. Assume f^t balanced for some $t \in \mathbb{N}$. By definition of f_N^t, $w_H(f_N^t) = w_H(\Psi \circ f^t \circ \Psi)$. Remark that $\Psi \circ f^t$ is a CA; then by Lemma 1, $w_H(\Psi \circ f^t \circ \Psi) = w_H(\Psi \circ f^t)$. Since f^t is balanced, $w_H(\Psi \circ f^t) = w_H(f^t)$. Finally, observing that Ψ^2 is the identity and by Lemma 1 again, it holds $w_H(f^t) = w_H(\Psi^2 \circ f^t) = w_H(\Psi^2 \circ f^t \circ \Psi) = w_H(\Psi \circ f_N^t)$. For the converse, assume that f_N^t is balanced for some $t \in \mathbb{N}$. Then, $w_H(f_N^t) = w_H(\Psi \circ f_N^t) = w_H(\Psi^2 \circ f^t \circ \Psi) = w_H(f^t \circ \Psi)$. By Lemma 1, $w_H(f^t \circ \Psi) = w_H(f^t)$ and therefore $w_H(f^t) = w_H(f_N^t)$. Again, by Lemma 1, $w_H(\Psi \circ f^t) = w_H(\Psi \circ f^t \circ \Psi) = w_H(f_N^t)$. Hence $w_H(f^t) = w_H(\Psi \circ f^t)$.

Proposition 2. *Let* $f : \mathbb{F}_2^{2m+1} \to \mathbb{F}_2$ *be the local function of a CA. For any* $t \in \mathbb{N}$, *let* $0 < k \leq 2mt + 1$. *Then,* f^t *is* k*-resilient iff* f_N^t *is* k*-resilient.*

Proof. Fix $k \in \mathbb{N}$ as in the hypothesis. By Lemma 2, it suffices to prove that $\widehat{f_N}(u) = h(u) \cdot \widehat{f}(u)$ for any $u \in \mathbb{F}_2^{2m+1}$ such that $0 < w_H(u) \leq k$ and $h : \mathbb{F}_2^{2m+1} \to \mathbb{R}^+$. Let $A = \mathrm{Id}$, $a = (1, 1, \ldots, 1)$, $b = 1$ and $B = (0, 0, \ldots, 0)$. Then, by using Eq. 3, one obtains $\widehat{(\mathrm{f}_N^t)_\chi}(u) = (-1)^{1+a\cdot u}\widehat{\mathrm{f}_\chi^t}(u)$ for any $u \in \mathbb{F}_2^{2mt+1}$ with $0 < w_H(u) \leq 2mt + 1$. This entails $\widehat{f_N^t}(u) = (-1)^{1+a\cdot u}\widehat{f}^t(u)$.

Consider the equivalence relation $\mathcal{R}$ on CA rules such that $f\mathcal{R}g$ iff $g = f_R$ or $g = f_N$ or $g = f_{RN}$. According to [5], there are $2^{2^m}(6 + 2^{2^m})$ distinct $\mathcal{R}$-classes. Propositions 1 and 2 say that all elements in a class have the same resiliency and hence only one element per class should be tested for studying this property. However the gain obtained by this quotient of the set of local rules is minor. Section 5 proposes (among other things) to consider affine transformations instead. Indeed, even if f and its Boolean equivalent, say f_A have the same resiliency, this does not hold, in general, for their iterates. This is essentially due to the fact that the above proofs are based on the existence of a bijection ϕ and a transformation τ on the local rules such that for any local rule f, it holds that $\forall t \in \mathbb{N}$, $[\tau(f)]^t \circ \phi = \phi \circ f^t$. This property is not true, in general for transformations different from negation or reflection.

3 Related Results

3.1 3-variable Boolean Update Function

An exhaustive search of 3-variable Boolean update function was done in [16]:

Theorem 3. *There is no non-linear correlation-immune elementary CA.*

The same result can be obtained by applying the Siegenthaler bound with $n = 3$ variables and testing for $k = 1$-resiliency. It tells that the algebraic degree is $d \leq n - k - 1 = 1$. Thus, only linear functions can be resilient.

Despite this, CA may be used for generating PRS by increasing the number of variables in the Boolean function which is used as a local CA rule. In the sequel, we recall which functions in four variables are suitable and we present a way to gather five variable functions for cryptographic purposes.

3.2 4-variable Boolean Update Function

In [11] the $2^{16} = 65536$ elementary CA rules with 4 variables were classified according to their resiliency and non-linearity. An exhaustive search by the Walsh transform of all Boolean functions with 4 variables was realized, to find a list of 1-resilient functions, with high non-linearity. There are exactly 200 non linear balanced functions which are 1-resilient.

A Boolean function in 4 variables is defined by an integer between 0 and 65536, extending Wolfram's notation for CA rules with 3 variables. For classifying the functions, we use their ANF. For instance the ANF of rule (280) (=100011000 in binary) corresponds to the polynomial $f(x_1, x_2, x_3, x_4) = x_1x_2 \oplus x_3 \oplus x_4 = 12 + 3 + 4$.

For the classification of these functions, let σ denote a 4×4 permutation matrix. Recall that two Boolean functions f and g are affine equivalent if there exists a permutation σ such that $f(x) = g(\sigma(x))$ or $g(\sigma(x)) + 1$.

The following table gives the set of all 1-resilient function, with a representative of each class f, its corresponding ANF and the cardinal of each class:

f	ANF	ANF	card.
34680	280	12+3+4	12
6120	360	4+12+13+23	8
7140	300	2+4+12+13	48
11730	282	1+3+4+12	24
34740	1308	2+3+4+12+24	48
39318	4374	1+2+3+4+34	12
7128	5432	3+4+12+13+24+34	24
11220	380	2+3+12+13+24	24

The non-linearity of these functions is computed for an evaluation of the resistance against the attack of [1]. The 200 1-resilient Boolean functions with 4 variables have a non-linearity equal to 4.

4 Exploring Radius 2, 1-Resilient Elementary CA Rules

Unlike 3 and 4 variable Boolean update function, we will not explore the whole class of radius 2 elementary uniform CA rules. Instead, we use the classification of Boolean functions in 6 variables or less with respect to some cryptographic properties from [3] where an efficient algebraic approach to the classification of the affine equivalence classes of the cosets of the first order Reed-Muller error correcting code is proposed. Indeed, the study of the properties of Boolean functions is related to the study of Reed-Muller codes. The code-words of the r-th order Reed-Muller code of length 2^n, denoted by $RM(r, n)$ correspond to the truth tables of Boolean functions with degree less or equal to r. [2] classified all the 2^{26} cosets of $RM(1, 5)$ into 48 equivalence classes under the action of the group $AGL(2, 5)$. The method is used to classify with respect to the 48 classes into which the general affine group $AGL(2, 5)$ partitions the cosets of $RM(1, 5)$. The cryptographic properties considered by [3] are correlation immunity (CI), resiliency (R) and propagation characteristics as well as their combination.

Table 1. Number of functions satisfying $CI(1)$ and $R(1)$

Representative	$\mathcal{N}_{CI(1)}$	$\mathcal{N}_{R(1)}$
12	4840	4120
123	16640	11520
123+14	216 000	133 984
123+14+25	69120	24960
123+145+23	1 029 120	537600
123+145+23+24+35	233 472	96 960

Table 1 is a selection of the representatives of Boolean functions taken out from [3] which lists the representative and counts the number of equivalent Boolean functions in the equivalence class which satisfy 1-resiliency (denoted by $R(1)$ in the table) and correlation immunity of first order (denoted by $CI(1)$

in the table). In Table 1, for a property P, $\mathcal{N}_P$ counts the number of Boolean functions which fulfills P.

From the original table, we only select representatives of Boolean functions of algebraic degrees 2 and 3 since, because of the Siegenthaler bound, there cannot be 1-resilient Boolean function of degree one. The classification done by [3] also removes Boolean functions of degree 4 if 1-resiliency is considered. Thus, there are only 6 equivalence classes containing 1-resilient Boolean functions which are listed in Table 1; 12 is the single equivalence class of degree two Boolean functions and the remaining 5 are all of degree three.

4.1 Finding the Rules

From the classification by [3], representatives of Boolean functions fulfilling the property of 1-resiliency were found. We restricted the search of Boolean functions in the same algebraic coset instead of the equivalence class in order to limit the combinatorial explosion. Our goal was to find 1-resilient elements in the cosets. For this, we first explored the elements of the cosets listed in Table 1 by considering all the linear combinations of all possible linear/affine functions and by computing the Fourier-Hadamard transform on all those elements in the coset. More precisely, the first step is to generate all coset elements. If we denote by $R(x_1, x_2, x_3, x_4, x_5)$ the coset leader (which is the representative), we consider all elements of the form:

$$R(x_1, x_2, x_3, x_4, x_5) \oplus (ax_1) \oplus (bx_2) \oplus (cx_3) \oplus (dx_4) \oplus (ex_5) \oplus h$$

for a, b, c, d, e, h Boolean, spanning all the 2^6 elements of the coset. Then, for each element, we compute the Fourier-Hadamard transform, we next only select the balanced Boolean functions and finally the Boolean functions which are 1-correlation immune among the balanced Boolean functions. That is, among the balanced Boolean functions, all functions with zero spectral values at points whose binary decomposition has a Hamming weight of 1. This first step was done with Mathematica 9.0 and gave us Table 2.

Reading Table 2, we notice that two cosets seem not to contain 1-resilient functions, although listed in the table by [3]. The reason for this is that we did not make the complete exploration of the equivalence class. Recall that the table by [3] classifies the 48 equivalence classes of $RM(1, 5)$ under the action of the general affine group $AGL(2, 5)$. At first, to check if our approach is valid, we only generated the coset elements and not the Boolean functions which could be obtained by the action of $AGL(2, 5)$ and which can be generated using Eq. (1). The size of the set of functions to explore is thus smaller. We run the fast transform algorithm on a set containing 6.2^6 elements which has to be compared with the whole set with 2^{32} elements. If we had taken into account the action of $AGL(2, 5)$, we should have explored 6 classes among the 48 equivalence classes (a ratio of 1/8) on the whole set.

Table 2. 1-resilient Boolean functions in the cosets. Hexadecimal numbers refer to the truth table.

Representative	1-resilient functions
12	3c3c3cc3 3c3cc33c 3cc33c3c 3cc3c3c3 5a5a5aa5 5a5aa55a 5aa55a5a 5aa5a5a5 66666699 66669966 66996666 66999999 69696996 69699669 69966969 69969696 96696969 96699696 96966996 96969669 99666666 99669999 99996699 99999966 a55a5a5a a55aa5a5 a5a55aa5 a5a5a55a c33c3c3c c33cc3c3 c3c33cc3 c3c3c33c
123	66696996 66699669 66966969 66969696 69666699 69669966 69996666 69999999 96666666 96669999 96996699 96999966 99696969 99699696 99966996 99969669
123+14	66695aa5 6669a55a 66965a5a 6696a5a5 696655aa 6966aa55 969955aa 9699aa55 99695a5a 9969a5a5 99965aa5 9996a55a
123+14+25	∅
123+145+23	1eb4663c 1eb499c3 e14b663c e14b99c3
123+145+23+24+35	∅

4.2 Testing the Iterates

We use the results from section 4.1 to select rules susceptible of preserving 1-resiliency, with the same procedure we used in section 2.4. More precisely, from the set of elementary, radius 2 rules (with a generic element denoted by f) , we consider the natural extension of $f : \mathbb{F}_2^5 \to \mathbb{F}_2$ to $f : \mathbb{F}_2^{n+4} \to \mathbb{F}_2^n$ (with $n > 0$) where: $f(x_1, \ldots, x_{n+4}) = (y_1, \ldots, y_n)$ such that $y_j = f(x_j, x_{j+1}, x_{j+2}, x_{j+3}, x_{j+4}), j \in [\![1, n]\!]$. Using the extended f, one can define the t-th iterate of f which is a function $f^t : \mathbb{F}_2^{4t+1} \to \mathbb{F}_2$. We next test the second iterate for selecting rules preserving the 1-resiliency. In other words, we compute the maximum absolute value of the Fourier-Hadamard transform of the t^{th}-iterate of f at all the points u of Hamming weight 1 and we select the balanced rules with a flat spectral value at those points (by Theorem 1).

For every f of Tab. 2, we built f^2 and tested its 1-resiliency property. This property is easily observable on the Fourier-Hadamard spectrum $\widehat{f^2} : f^2$ is m-resilient if $\forall u \in \mathbb{F}_2^9 \,/\, w_H(u) \leq m$, then $\widehat{f^2}(u) = 0$. The spectrum has been computed by the algorithm defined in subsection 2.4 and implemented in `C`. The results are in Tab. 3 and shows that few functions (exactly 4 of them) of coset 12 are not 1-resilient, that every function of coset 123 and coset 123+14 preserves 1-resiliency, and no function of coset 123+145+23 are 1-resilient after 2 iterations.

5 PRNG Testing

The quality of pseudo-randomness generated by the above mentioned Boolean functions has been evaluated by using the Diehard test suite, a widely used tool. It has been developed by Marsaglia from the Florida State University and consists of 17 different tests which have become something which could be considered as a "benchmarking tool" for PR number generators (see [12]). It is meant to evaluate if a stream of numbers is a good PRS. We will not explain how Diehard really works and we refer the reader to [13] for further details. Basically, Diehard uses Kolmogorov-Smirnov normality test to quantify the distance between the distribution of a given data set and the uniform distribution; and as the documentation says:

Table 3. 1-resilient Boolean functions after 2 iterations

Coset 12	0x3C3C3CC3	yes	0x3C3CC33C	no	0x3CC33C3C	no
	0x3CC3C3C3	yes	0x5A5A5AA5	yes	0x5A5AA55A	yes
	0x5AA55A5A	yes	0x5AA5A5A5	yes	0x66666699	yes
	0x66669966	yes	0x66996666	yes	0x66999999	yes
	0x69696996	yes	0x69699669	yes	0x69966969	yes
	0x69969696	yes	0x96696969	yes	0x96699696	yes
	0x96966996	yes	0x96969669	yes	0x99666666	yes
	0x99669999	yes	0x99996699	yes	0x99999966	yes
	0xA55A5A5A	yes	0xA55AA5A5	yes	0xA5A55AA5	yes
	0xA5A5A55A	yes	0xC33C3C3C	yes	0xC33CC3C3	no
	0xC3C33CC3	no	0xC3C3C33C	yes		
Coset 123	0x66696996	yes	0x66699669	yes	0x66966969	yes
	0x66969696	yes	0x69666699	yes	0x69669966	yes
	0x69996666	yes	0x69999999	yes	0x96666666	yes
	0x96669999	yes	0x96996699	yes	0x96999966	yes
	0x99696969	yes	0x99699696	yes	0x99966996	yes
	0x99969669	yes				
Coset 123+14	0x66695AA5	yes	0x6669A55A	yes	0x66965A5A	yes
	0x6696A5A5	yes	0x696655AA	yes	0x6966AA55	yes
	0x969955AA	yes	0x9699AA55	yes	0x99695A5A	yes
	0x9969A5A5	yes	0x99965AA5	yes	0x9996A55A	yes
Coset 123+145+23	0x1EB4663C	no	0x1EB499C3	no	0x2D7855F0	no
	0x2D78AA0F	no	0x44EE3C66	no	0x44EEC399	no
	0x4B1ECC69	no	0x77220FAA	no	0x7722F055	no
	0x88DD0FAA	no	0x88DDF055	no	0xB4E13396	no
	0xBB113C66	no	0xBB11C399	no	0xD28755F0	no
	0xD287AA0F	no	0xE14B663C	no	0xE14B99C3	no

> Each Diehard test is able to provide probability values (p-value) which should be uniformly distributed on $[0, 1)$ if the sequence is made of truly independent bits. Those p-values are obtained by $p = F(X)$ where F is the assumed distribution of the sample random variable X–often normal. But that assumed F is just an asymptotic approximation, for which the fit will be worse in the tail of the distribution. Thus, we should not be surprised with occasional p-values close to 0 or 1. When a stream really fails, one gets p-values of 0 or 1 to six or more places. Otherwise, for each test, its p-value should lie in the interval $(0.025, 0.975)$.

So in order to test our data, we designed a `C` program in which we included the Diehard functions that were slightly modified to fit well with our needs. That is to directly use the results of the CA as a PRG. The 17 different and independent statistical tests require about 16 Mbyte of PR values in binary format.

Our goal was to generate different number sequences from the CA and test them against Diehard. Two different tests were made.

5.1 Randomness Preservation

In this section we describe the experimentation we made to test if a CA "preserves" the randomness through its dynamics. For this experiment, we consider a CA, whose transition function is $f : \mathbb{F}_2^5 \to \mathbb{F}_2$. Given such a CA, we set up an initial sequence of bits $(b_i)_{i\geq 0}$ that we extract from the `/dev/random` pseudo-device of a MacOSX system[1]. Then we compute the sequence of bits $(b'_i)_{i\geq 0}$ such that $\forall i \leq 0$, $b'_i = f(b_{5i}, b_{5i+1}, b_{5i+2}, b_{5i+3}, b_{5i+4})$. To ensure some statistical soundness, for a single CA we build 30[2] of such sequences from the same entropic source (each sequence being 16 Mbyte long as required by Diehard).

The measure, illustrated in Fig. 1, shows all the distributions of the indicators produced by each single sequence passing all the Diehard tests. And it can be observed that the p-values are well distributed for every data pack. Indeed, there are no accumulation points near zero or one.

This means that the input to the tests is made of independent bits. Thus, we can deduce that these functions are good at preserving the randomness. Or, in other terms, if we feed a CA with a truly random sequence (obtained by the entropy collector of the BSD kernel) as an input configuration and let the CA run, the output configuration is still PR, according to the Diehard test suite.

5.2 Random Number Generation

Much more classically, these tests were built to evaluate the possible generation of a good PR sequence by CAs. While it is well known that radius 1 elementary CAs are not suitable for generating PR sequences, it is not impossible to build good PR sequence from simple CAs. As we already tested if the radius-2 functions are good to preserve the randomness, it would be interesting to consider them as PRNG. So, we tried something very similar to [17].

We set up two rings of cells. Although Wolfram used a ring of 127 cells and Preneel (1993) suggested a ring of 1024 cells to ensure a better quality (both used a slightly different mechanisms for random bit extraction), we use perimeters 64 and 65 as done in [17]. The initial configuration of these rings is of Hamming weight 1. We let the CA iterate about 2 million times. Then, from each configuration obtained, we extract two 32-bits words: the "even" (resp. "odd"), word is built with the state of the first 32 "even" (resp. "odd") cells. The sequences of these "even" (resp. "odd") words constitute two different sequences of 16 Mbyte.

Then, we use Diehard to produce p-values for each test. We were able to find some CAs (like the one with rule `0x69999999` given as an example in Fig. 2). This suggests that it may be possible to obtain a good PRNG from such a CA.

[1] The entropy collector of the BSD kernel family is considered as a pretty good source of random numbers and MacOSX is built on top of a BSD kernel.

[2] The repetition of 30 independent experiments comes from statistics. Indeed sample sizes of at least 30 are for many tests considered as "large" and allows a better statistical treatment.

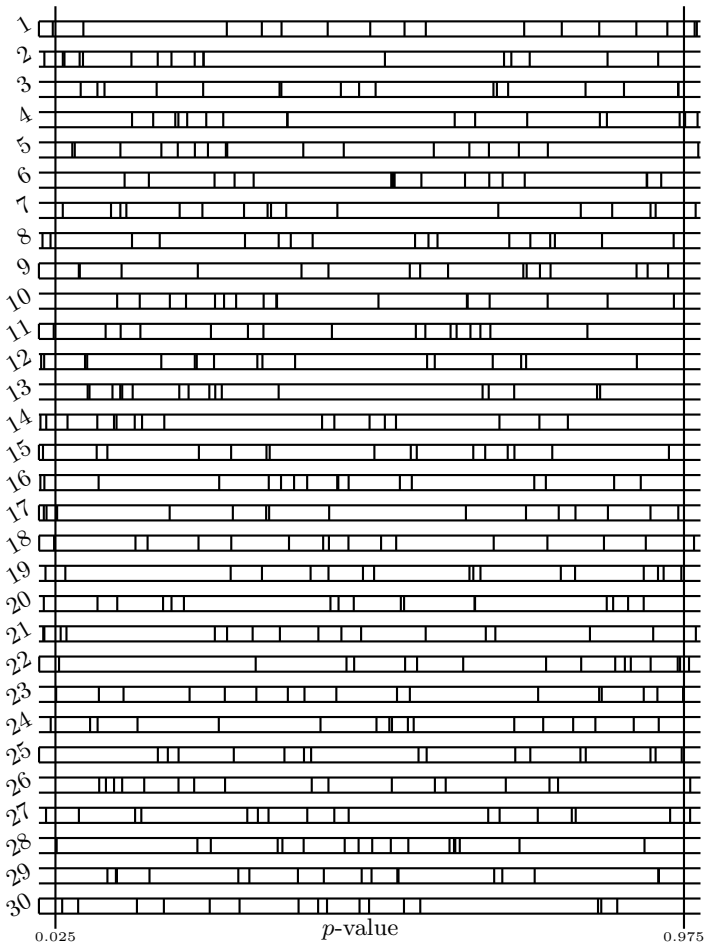

Fig. 1. `0x3C3C3CC3`: distribution of the p-values for each data pack. p-values between the two lines (at 0.025 and 0.975) mean that the corresponding statistical test was successful.

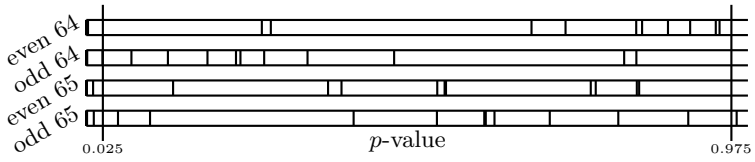

Fig. 2. Distribution of the p-values for the ring CA with rule `0x69999999`. p-values between the two lines (at 0.025 and 0.975) mean that the corresponding statistical test was successful.

6 Conclusion

The main interest of this work concerns the hardware implementation. The target hardware model of CAs is the Field Programmable Gate Arrays (known as FPGAs). FPGAs are now a popular implementation style for digital logic systems and subsystems. These devices consist of an array of uncommitted logic gates whose function and interconnection is determined by downloading information to the device. When the programming configuration is held in static RAM, the logic function implemented by those FPGAs can be dynamically reconfigured in fractions of a second by rewriting the configuration memory contents. Thus, the use of FPGAs can speed up the computation done by the cellular automata. Putting all together allows high-rate pseudo-random generation of good quality that can be used as a basic component for lightweight cryptography requiring pseudo-random sources.

These results can be extended in many directions. If the number of variables of a Boolean function must be increased, our approach for extending good updating rules can be helpful. Increasing the number of variables in a Boolean function is a classical problem in symmetric cryptography.

Acknowledgments. The authors are grateful to C. Carlet who pointed out Reference [4] for explaining the difference between Fourier-Hadamard and Walsh transforms and to J. Mairesse for its help with statistical testing.

References

1. Apohan, A.M., Koc, C.K.: Inversion of cellular automata iterations. Computer and Digital Techniques **144**, 279–284 (1997)
2. Berlekamp, E., Welch, L.: Weight distribution of the cosets of the $(32, 6)$ Reed-Muller code. IEEE Trans. Inf. Theory **18**, 203–207 (1972)
3. Braeken, A., Borissov, Y., Nikova, S., Preneel, B.: Classification of Boolean functions of 6 variables or less with respect to some cryptographic properties. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 324–334. Springer, Heidelberg (2005)
4. Carlet, C.: Boolean functions for cryptography and error-correcting codes. Technical report, University of Paris 8 (2011)
5. Cattaneo, G., Formenti, E., Margara, L., Mauri, G.: Transformations of the one-dimensional cellular automata rule space. Parallel Computing **23**(11), 1593–1611 (1997)
6. Elliott, D.E., Rao, K.R.: Fast transforms, algorithms, analysis, applications. Academic press (1982)
7. Formenti, E., Imai, K., Martin, B., Yunès, J.-B.: On 1-resilient, radius 2 elementary CA rules. In: Fatès, N., Goles, E., Maass, A., Rapaport, I. (eds.) Automata 2011, pp. 41–54 (2011)
8. Goldreich, O.: Pseudorandomness. Notices of the AMS **46**(10), 1209–1216 (1999)
9. Gruska. J.: Foundations of Computing. International Thomson Publishing (1997)
10. Gruska, J., La Torre, S., Parente, M.: The firing squad synchronization problem on squares, toruses and rings. Int. J. Found. Comput. Sci. **18**(3), 637–654 (2007)

11. Lacharme, P., Martin, B., Solé, P.: Pseudo-random sequences, boolean functions and cellular automata. In: Proceedings of Boolean Functions and Cryptographic Applications (2008)
12. Marsaglia, G.: A current view of random number generators. In: Computer Sciences and Statistics, pp. 3–10 (1985)
13. Marsaglia, G.: Diehard (1995). http://www.stat.fsu.edu/pub/diehard/
14. Martin, B.: A Walsh exploration of Wolfram CA rules. In: International Workshop on Cellular Automata, pp. 25–30. Hiroshima University, Japan (2006)
15. Martin, B.: Mixing compression and CA encryption. In: Bonnecaze, A., Leneutre, J., State, R. (eds.) SAR-SSI 2007, pp. 255–266. Université Jean Moulin, Lyon (2007)
16. Martin, B.: A Walsh exploration of elementary CA rules. Journal of Cellular Automata **3**(2), 145–156 (2008)
17. Shackleford, B., Tanaka, M., Carter, R.J., Snider, G.: FPGA implementation of neighborhood-of-four cellular automata random number generators. In: Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays, FPGA 2002, pp. 106–112. ACM (2002)
18. Wolfram, S.: Cryptography with cellular automata. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 429–432. Springer, Heidelberg (1986)
19. Wolfram, S.: Theory and applications of cellular automata. World Scientific, Singapore (1986)
20. Wolfram, S.: A new kind of science. Wolfram Media Inc., Champaign (2002)
21. Xiao, G.-Z., Massey, J.L.: A spectral characterization of correlation-immune combining functions. IEEE Trans. on Information Theory **34**(3), 569 (1988)