Pseudo-random sequences, boolean functions and cellular automata

Patrick Lacharme, Bruno Martin, Patrick Solé

To cite this version:

Patrick Lacharme, Bruno Martin, Patrick Solé. Pseudo-random sequences, boolean functions and cellular automata. Boolean Functions: Cryptography & Applications, May 2008, Copenhague, Denmark. pp.80-95. hal-00305493

HAL Id: hal-00305493

https://hal.archives-ouvertes.fr/hal-00305493

Submitted on 24 Jul 2008

PSEUDO-RANDOM SEQUENCES, BOOLEAN FUNCTIONS AND CELLULAR AUTOMATA

Patrick Lacharme[1], Bruno Martin[2] and Patrick Solé[2]

Abstract. Generation of pseudo-random binary sequences by one-dimensional cellular automata is surveyed using both uniform and hybrid automata. The updating function is a Boolean function that must satisfy criteria of resilience and nonlinearity for the sequence they generate to be secure for a stream cipher application.

Keywords : Cellular automata, pseudo-random sequences, resilience, nonlinearity.

Introduction

Cellular automata (CA) are models of finite state machines used in many applications of computer science. In particular, they are is employed for the generation of binary pseudo-random sequences in cryptography. In the present work, we survey methods to generate binary pseudo-random sequences with cellular automata. We particularly focus on the search for good updating functions.

Section 1 presents the different cellular automata for sequence generation and a surveys their cryptographic applications. Section 2 concentrates on uniform cellular automata, particularly on the properties required for the updating function. An exhaustive list of rules satisfying the properties is given. Section 3 surveys

[1] IMATH, Université Toulon/Var, BP 132, F–83957 La Garde cedex.
[2] I3S, Université de Nice Sophia-Antipolis, CNRS, 2000 route des Lucioles, BP 121, F–06903 Sophia–Antipolis Cedex.

the generation of pseudo-random sequences by hybrid cellular automata and proposes the generation of classical sequences by CA synthesis. At last, cryptographic applications of hybrid cellular automata are sketched.

1. **Cellular automata and pseudo-random generation**

1.1. **Uniform cellular automata**

One-dimensional binary cellular automata consist of a line of cells with binary values. For practical implementations, the number of cells is finite. There are two cases: a CA is *boundary* if the cells are arranged in a circular register and it is *null-boundary* when both extremal cells are fed with zeroes. 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.

S. Wolfram proposes in [31] to use cellular automata to produce pseudo-random sequences. He considers one-dimensional binary cellular automata with ℓ cells with $\ell = 2N + 1$. All the cells are identical finite state machines with binary states. For an uniform CA, the values of the cells at time t are updated synchronously with a Boolean function f with $n = r_1 + r_2 + 1$ variables by the rule

$$x_i(t+1) = f(x_{i-r_1}(t), \ldots, x_i(t), \ldots, x_{i+r_2}(t)) \ .$$

For a fixed t, the sequence of the values x_i for $1 \leq i \leq 2N+1$, is the *configuration* at time t. It is a mapping $c : [\![1, 2N+1]\!] \to \mathbb{F}_2$ which assigns a state of $\mathbb{F}_2$ to each cell of the cellular automaton. The initial configuration (at time 0) $x_1(0), \ldots, x_\ell(0)$ is the *seed* of the generator, the sequence $(x_N(t))_t$ is the *output sequence* and, when $r_1 = r_2 = r$, the number r denotes the radius of the rule.

For example, let us consider the following updating function f:

$$f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \text{ XOR } (x_i \text{ OR } x_{i+1}).$$

The truth table of the Boolean function f is

x_3	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_1	0	0	0	0	1	1	1	1
$f(x_1, x_2, x_3)$	0	1	1	1	1	0	0	0

The binary sequence 00011110 is the binary representation of the natural number 30 and allows to give a numbering of the rules (rule (30) in this case). In this paper, this notation is kept to describe a Boolean function. In the rest of this paper the additions of integers in $\mathbb{Z}$ will be denoted by $+$ and Σ and additions in $\mathbb{F}_2$ will be denoted by $\oplus$ and $\bigoplus$. For simplicity and when there will be no ambiguity, we shall denote by $+$ the addition of binary vectors. If x and b are two binary vectors, we denote by $x \cdot b$ their usual inner product $x \cdot b = \sum_{i=1}^{n} x_i b_i$. We also recall the definition of the *algebraic normal form* as an alternative representation of Boolean functions.

Definition 1.1. A Boolean function f with n variables is represented by a binary polynomial in n variables, called algebraic normal form (ANF): $f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u (\prod_{i=1}^{n} x_i^{u_i})$.

If we consider rule (30) again, its ANF is the polynomial $x_1 \oplus x_2 \oplus x_3 \oplus x_2 x_3$.

The *degree of the ANF* denoted by $d^\circ f$ is called the *algebraic degree* of the function. This makes sense thanks to the existence and uniqueness of the ANF. The *Hamming weight* $w_H(f)$ of f is the number of vectors x in $\mathbb{F}_2^n$ such that $f(x) = 1$. A function is *balanced* if $w_H(f) = w_H(1 \oplus f)$, i.e. $w_H(f) = 2^{n-1}$. The *Hamming distance* between f and g is $d_H(f, g) = w_H(f \oplus g)$.

1.2. Hybrid cellular automata

When the rules are not identical for all the cells, the automaton is called *non uniform*, (or *hybrid*) and is denoted HCA. Moreover if the rules are linear, the automaton is called *linear hybrid cellular automata* (LHCA).

In [5,25], Muzio et al. consider linear hybrid cellular automata using two different linear rules. They assume the CA null-boundary and the linear updating rules are (90) and (150). A similarity transform between LHCA and linear feedback shift register is proposed in [6] and was recently improved in [12].

1.3. Cellular programming approach

Tomassini and Sipper [27] proposed to use hybrid cellular automata for generating better pseudo-random sequences. In this model, the rules are obtained by an evolutionary approach (a genetic algorithm). They have designed a *cellular programming* algorithm for cellular automata to perform computations, and have applied it to the evolution of pseudo-random sequence generators. Their approach uses Koza's *entropy* $E_h = -\sum_{j=1}^{k^h} p_{h_j} \log_2 p_{h_j}$ where k denotes the number of possible values per sequence position, h a subsequence length and p_{h_j} is a measured probability of occurrence of a sequence h_j in a pseudo-random sequence.

Tomassini and Sipper have selected four rules of radius $r = 1$ for use in non-uniform cellular automata. The best rules selected by the genetic algorithm were rules 90, 105, 150 and 165 (all linear).

A series of tests (including χ^2 test, serial correlation coefficient, entropy and Monte Carlo, but no correlation-immunity analysis) were made with good results, showing that co-evolving generators are at least as good as the best available CA randomizer. The authors also use elementary rules which were proved to be not correlation-immune. This was further investigated in [24].

Following the same kind of approach, Seredynski et al. in [24] have generalized the selection process to radius 2 rules. They use then both radius 1 and radius 2 rules in hybrid cellular automata. The rules selected by their genetic algorithm were 30, 86, 101 and 869020563, 1047380370, 1436194405, 1436965290, 1705400746, 1815843780, 2084275140 and 2592765285.

Their new set of rules was tested by a number of statistical tests required by the FIPS 140-2 standard [28] and the Marsaglia tests implemented in the Diehard program but no correlation-immunity analysis was made either.

Moreover, the entropy calculation used in cellular programming approach reduces the efficiency of the algorithm.

1.4. Cryptographic applications

In [30] Wolfram proposed to use cellular automata for a stream cipher application. In this case, it must be hard to find the current state (and the seed in particular) from the output sequence (known plaintext attack). An exhaustive search can be done over the 2^ℓ possible initial states, but for large ℓ, exhaustive search becomes impossible. In [30], the cellular automaton is a simple one-dimensional uniform cellular automaton with a 3-variable updating rule.

Next, cellular automata are used in many area in cryptography. In symmetric cryptography, a block cipher proposed in [13] has been broken in [3], using the linearity of the updating function and the block cipher of [23] using linear and non linear rules has been successfully cryptanalysed in [2]. Other propositions of block ciphers are [21] and [11]. Some hash functions are proposed in [10] (broken in [9]) and [19] and stream cipher in [18] [20].

For the design of cryptographic cellular automata, updating rules should have an adapted radius and desired properties. In all cases, the choice of the updating rules in the cellular automaton is a fundamental criterion for the security of the crypto-system.

2. Uniform cellular automata

2.1. 3-variable updating function and rule (30)

The investigation of S. Wolfram [30] is concentrated on updating functions f with 3 variables, i.e. $r_1 = r_2 = 1$. In particular, he considered rule (30) defined by

$$f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \text{ XOR } (x_i \text{ OR } x_{i+1}),$$

or equivalently

$$f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} + x_i + x_{i+1} + x_i x_{i+1} \bmod 2.$$

S. Wolfram claims the sequence $(x_i^t)_{t\geq 0}$ is pseudo-random for a given i. He extensively studied this particular rule, demonstrating its suitability as a high performance randomizer which can be efficiently implemented in parallel; this is one of the pseudo-random generators which was shipped with the connection machine CM2 and which is currently used in the Mathematica software.

2.2. Attack of Meier and Staffelbach

The first attack on this pseudo-random generator is proposed by W. Meier and O. Staffelbach in [17]. The output sequence $(x_N(t))_t$ is known for $t = 0, \ldots, N$. The sequence $(x_{N+1}(t))_t$ is called *right adjacent sequence*. The principle is that if the right adjacent sequence is known for $t = 0, \ldots, N-1$, then half of the seed $x_0(0), \ldots, x_{N-1}(0)$ is known, using the partial linearity of f :

$$x_{i-1}(t) = x_i(t+1) \text{ XOR } (x_i(t) \text{ OR } x_{i+1}(t)).$$

This attack is successfull if the right adjacent sequence is easily constructed. Statistical experiments show that there are only few possibilities for the right adjacent sequence. As noted in [15], this phenomenon comes from the correlation between the inputs and the outputs of rule (30). Indeed, we have

$$Pr\left(x_i(t+1) = 1 \oplus x_{i-1}(t)\right) = \frac{3}{4}.$$

In order to counter this attack, the updating function must be resilient. This concept is introduced by T. Siegenthaler, in relation with correlation attacks on stream ciphers [26].

Definition 2.1 (Resilient functions)**.** A Boolean function f mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$ is t-resilient if for any coordinates $i_1, \ldots i_t$, for any binary constant $c_1, \ldots, c_t$ and for all $y \in \mathbb{F}_2$,

$$Pr(f(x) = y \mid x_{i_1} = c_1, \ldots, x_{i_t} = c_t) = \frac{1}{2},$$

where x_i with $i \notin \{i_1, \ldots, i_t\}$ verifies $Pr(x_i{=}1) = Pr(x_i{=}0){=}0.5$.

For a Boolean function f with n variables, the Walsh transform $\widehat{f}$ of f is defined over $\mathbb{F}_2^n$ by

$$\widehat{f}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus u \cdot x} .$$

The Walsh transform of f is related to the Hamming weight of the function $f \oplus l_b$ (where $l_b(x) = b \cdot x$) via the relation: $\widehat{f}(b) = 2^n - 2w_H(f \oplus l_b)$. It satisfies Parseval's relation:

$$\sum_{b \in \mathbb{F}_2^n} \widehat{f}^2(b) = 2^{2n} . \tag{1}$$

G.Z. Xiao and J.L. Massey give a spectral characterization of resilient functions [33]

Theorem 2.2. *A Boolean function in n variables is t-resilient iff for all u whose Hamming weight verifies $0 \leq w_H(u) \leq t$, we have $\widehat{f}(u) = 0$.*

Theorem 2.3. *For a t-resilient ($0 \leq t < n-1$) Boolean function with n variables, there is an upper bound for its algebraic degree d: $d \leq n-t-1$ if $t < n-1$ and $d = 1$ if $t = n-1$.*

Theorem 2.3 shows that the algebraic degree of a 1-resilient function with 3 variables is at most 1, i.e. the Boolean function is linear. B. Martin has explored all the Boolean functions in 3 variables and concludes that there is no non linear function in 3 variables which is 1-resilient [15] which is also a consequence of the Siegenthaler bound [26].

2.3. **Attack of Koç and Apohan**

The second attack against rule (30) is proposed by C.K. Koç and A.M. Apohan [1].

The algorithm is based on the linear approximation of the updating function. The resistance to this attack depends on the distance between the updating function and the set of all linear functions. Let l be the best linear approximation of the updating function f with n variables and A the number of points such that $f(x) = l(x)$. Let $\alpha = A/2^n$ be the probability than $f(x) = l(x)$. Then the algorithm requires at most $2^{(n-1)(1-\alpha)l}$ trials for arbitrary nonlinear functions and seed. Under these conditions, the number α must be small.

Definition 2.4 (Nonlinearity)**.** Let f be a Boolean function mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$. The nonlinearity N_f of f is defined by

$$N_f = \min_{(a_0,\ldots,a_n)\in\{0,1\}^{n+1}} d_H(f(x), a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n) .$$

The relation between Walsh transform and nonlinearity comes from W. Meier and O. Staffelbach [16].

Theorem 2.5. *Let f be a function mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Then*

$$N_f = 2^{n-1} - \frac{1}{2}\max_{u\in\mathbb{F}_2^n} |\widehat{f}(u)| ,$$

and

$$N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{2}$$

A cellular automaton resists the attack of [1] if the nonlinearity of its updating function is high. Boolean functions where nonlinearity meet the bound (2) are called *bent* functions. Bent functions in an even number of variables exist but are never balanced. Moreover, there is a tradeoff between resilience order and nonlinearity [22] :

Theorem 2.6. *Let f be a Boolean function t-resilient over $\mathbb{F}_2^n$. Then there is an upper bound on the nonlinearity of f: $N_f \leq 2^{n-1} - 2^{t+1}$.*

By [29] this bound is tight for $t \geq 0.6n$.

2.4. **4-variable updating functions**

In this section, we investigate the $2^{16} = 65536$ elementary CA rules with 4 variables and give a complete classification of the functions which have good resilience and nonlinearity properties. These functions are used to get cellular automata for generating cryptographic pseudorandom sequences.

An exhaustive search by the Walsh transform of all Boolean functions with 4 variables is realized, to find a list of 1-resilient functions, with high nonlinearity.

There are exactly 200 non linear balanced functions which are 1-resilient and all are quadratics (cubics balanced functions are not resilient by Siegenthaler bound (Theorem 2.3)).

A Boolean function in 4 variables is defined by an integer between 0 and 65536, keeping Wolfram's notation for CA rules with 3 variables. For example, the function (34680), with binary representation 100001110111000 corresponds to the following truth table:

x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$f(x_1, x_2, x_3, x_4)$	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	1

For classifying the functions, we use their algebraic normal form (ANF). For instance the ANF of rule (280) (=100011000 in binary) corresponds to the polynomial

$$f(x_1, x_2, x_3, x_4) = x_1x_2 \oplus x_3 \oplus x_4 \ . \tag{3}$$

For the classification of these functions, let σ denote a 4 by 4 permutation matrix. Two Boolean functions f and g are 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 representant of each class f, its corresponding ANF and the cardinal of each class:

f	ANF	ANF	card.
34680	280	$x_1x_2 \oplus x_3 \oplus x_4$	12
6120	360	$x_4 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3$	8
7140	300	$x_2 \oplus x_4 \oplus x_1x_2 \oplus x_1x_3$	48
11730	282	$x_1 \oplus x_3 \oplus x_4 \oplus x_1x_2$	24
34740	1308	$x_2 \oplus x_3 \oplus x_4 \oplus x_1x_2 \oplus x_4x_2$	48
39318	4374	$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_3x_4$	12
7128	5432	$x_3 \oplus x_4 \oplus x_1x_2 \oplus x_3x_1 \oplus x_4x_2 \oplus x_4x_3$	24
11220	380	$x_2 \oplus x_3 \oplus x_1x_2 \oplus x_3x_1 \oplus x_4x_2$	24

The nonlinearity 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 nonlinearity equal to 4. There exist some Boolean function with 4 variables with nonlinearity 6, but they are not resilient, according Theorem 2.6.

Two classes of functions are more interesting because these functions can be inplemented with only 3 logical gates. These functions are (34680) and (39318) or equivalently

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus (x_3 \text{ OR } x_4\ )$$

In this case, the two CA are described by the updating function

$$x_i(t+1) = x_{i-1}(t) \text{ XOR } x_i(t) \text{ XOR } (x_{i+1}(t) \text{ AND } x_{i+2}(t))\ ,$$

and

$$x_i(t+1) = x_{i-1}(t) \text{ XOR } x_i(t) \text{ XOR } (x_{i+1}(t) \text{ OR } x_{i+2}(t))\ .$$

If the resilience order of the updating function must be greater than 1 (for example with a generalization of [17]), then Boolean functions with 5 variables should be used.

3. Hybrid Cellular automata

3.1. Linear hybrid cellular automata

In the context of sequence generation, several authors consider the case where different cells of the CA use different rules. This model is called *hybrid cellular automata*. In [5, 25], Muzio et al. consider linear hybrid cellular automata using two different linear rules. The automata are null-boundary and the linear updating rules are (90) and (150). In [6], LHCA are proved equivalent to linear feedback shift registers.

Let $D = [d_0, \dots, d_{\ell-1}]$ be a binary vector with the following properties : if $d_i = 0$ then cell i uses rule (90), else cell i uses rule (150). In other terms,

$$x_i(t+1) = x_{i-1}(t) + d_i.x_i(t) + x_{i+1}(t) \bmod 2 \ .$$

Using this rule, it is possible to give a matrix representation of the transition between two consecutive states. Let the current state be $s(t) = (x_0(t), \dots, x_{\ell-1}(t))$, then

$$s(t+1) = A.s(t) \ , \tag{4}$$

where the binary matrix A, called *transition matrix*, is defined by

$$\begin{pmatrix} d_0 & 1 & 0 & \dots & 0 \\ 1 & d_1 & 1 & \ddots & \vdots \\ 0 & 1 & d_2 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & d_{l-1} \end{pmatrix}$$

Let Δ be the characteristic polynomial of A, that is $\Delta = |xId - A|$. A polynomial is said to be a *HCA polynomial* if it is the characteristic polynomial of some HCA. The following theorem gives a property on HCA polynomial [5, 8] :

Theorem 3.1. *Let $p \in \mathbb{F}_2[X]$ of degree n and p' its formal derivative. Then p is a HCA polynomial if and only if for some solution q for y of the congruence*

$$y^2 + (x^2 + x)p'y + 1 \equiv 0 \bmod p \ , \tag{5}$$

Euclid's greatest division algorithm on p and q results in n degree one quotients.

If the polynomial p is irreducible, then equation (5) has exactly two solutions, both of which result in n degree one quotient :

Theorem 3.2. *Let $p \in \mathbb{F}_2[X]$ be an irreducible polynomial. Then p has exactly two HCA realizations with one being the reversal of the other.*

Given a polynomial, the construction of the corresponding HCA is called the *synthesis approach.* A detailed method is described in [5] using this theorem to find HCA polynomial.

Moreover, given p and q two irreducible polynomials with corresponding transition matrix P and Q, the transition matrix corresponding to $p.q$ is given by $\left(\begin{smallmatrix} P & 0 \\ 0 & Q \end{smallmatrix}\right)$. This operation corresponds to concatenation of two LHCA [7].

3.2. Synthesis of classical sequences by LHCA

A *semi-bent functions* is a Boolean functions in an odd number n of variables, the Walsh transform of which takes only three values $0, \pm 2^{(n+1)/2}$. Some of these are traces of AB vectorial functions [4] in the form $f(x) = Tr(ax + bx^s)$, where a, b are scalars of the extension field $\mathbb{F}_{2^n}$ and Tr the trace function from $\mathbb{F}_{2^n}$ down to $\mathbb{F}_2$. For monomial AB functions, the exponents s are in the list of Gold, Kasami, Welch, Niho (see Table 1).

Name	s	Conditions
Gold	$2^i + 1$	$i \wedge n = 1, 1 \leq i \leq n/2$
Kasami	$2^{2i} - 2^i + 1$	$i \wedge n = 1, 1 \leq i \leq n/2$
Welch	$2^{(n-1)/2} + 3$	
Niho	$2^{2r} + 2^r - 1$	$r = t/2$ for t even $r = (3t+1)/2$ for t odd with $1 \leq r \leq n = 2t + 1$

TABLE 1. Exponents of Almost Bent monomials

In all cases, for α a primitive element on $\mathbb{F}_{2^n}$ and m_α the minimal polynomial of α, the polynomial $m_\alpha m_{\alpha^s}$, is used to synthesize HCA. Using these polynomials, sequences with good correlation properties can be synthesized by linear hybrid cellular automata.

3.3. Cryptography with LHCA

Linear HCA cannot serve directly as pseudo-random sequence generators for cryptography. Equation (4) is used to describe any current state of the CA in terms of the initial state by the relation

$$s(t) = A^t.s(0) \ . \tag{6}$$

Let N be the index of the cell which gives the output sequence $(s_N(t))_t$ of the CA. Then

$$s_N(t) = a_N(t).s(0) \ , \tag{7}$$

where $a_N(t)$ denotes the ℓ-bit vector corresponding to the row N of the matrix A^t.
In this case, the initial state is known by the inversion of the binary linear system according to (7).

In these conditions, the binary vector $D = [d_0, \dots, d_{\ell-1}]$ must be non constant. If the vector D changes in function of $s(t)$, e.g.

$$d_i(t) = s_{i+2}(t),$$

then

$$s_i(t+1) = s_{i-1}(t) \oplus s_i(t) \oplus d_i(t).s_{i+1}(t) \ . \tag{8}$$

In this case, we get the non linear Boolean function (280) of section 2 again.
In fact, if $d_i(t)$ is a linear combination of bits of the current state $s(t)$, then the updating function corresponds to a quadratic Boolean function.

An other possibility is to consider the vector $[d_0, ..d_{n-1}] = [y_0(t), .., y_{n-1}(t)]$, where the bits $y_i(t)$ are the internal state of an other CA. Generalizing this system, a cascade of n CA is constructed, where the state of the nth CA corresponds to the vector D of the $n+1$th CA. This cascade provides high quality pseudo random sequence and a possibility for larger cycles.

We illustrate our previous proposition by an example: let CA_1 be a uniform CA using rule (39318) of length ℓ. The internal state of CA_1 corresponds to the binary vector $D = [d_0, \dots, d_{\ell-1}]$ and the rule of the second automaton CA_2 is described by equation (8).

The output sequence of this generator is given by the sequence $(s_N(t))_t$ where s is the internal state of CA_2 and $\ell = 2N + 1$.

In this case, the cryptanalysis of the pseudo random generator of [13] proposed in [3] is not possible because rules are non linear and the output of the generator consists in just one bit.

As proposed in [18], it is suitable that the output sequence is not directly one (or several) bit of the current state. In this case the output sequence is filtered by a Boolean function. This scheme is a variant of the classical filtered linear shift register. The Boolean function must have the same properties as high nonlinearity or resilience order.

Conclusion

We have used the synthesis approach to give an effective CA-realization of classical sequences with good correlation properties. We have also presented good and bad ways to construct pseudo-random binary sequences for cryptographic applications with cellular automata.

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).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.

These results can be extended in many directions. If the number of variables of the Boolean function must be increased, the research of good updating rules is a classical problem in symmetric cryptography. Other applications of CA than pseudo random generation (e.g. hash functions) can also be investigated.

Acknowledgements

The authors thank Sihem Mesnager for helpful comments.

References

[1] A.M. Apohan and C.K. Koc. Inversion of cellular automata iterations. In *Computer and Digital Techniques*, volume 144, pages 279–284, 1997.

[2] F. Bao. Cryptanalysis of a new cellular automata cryptosystem. In *ACISP*, pages 416–427, 2003.

[3] S.R. Blackburn, S. Murphy, and K.G. Paterson. Comments on "theory and applications of cellular automata in cryptography". *IEEETC: IEEE Transactions on Computers*, 46, 1997.

[4] C. Carlet, P. Charpin, and V. Zinoviev. Codes, bent functions and permutations suitable for DES-like cryptosystems. *Designs Codes and Cryptography*, 15:125–156, 1998.

[5] K Cattell and J.C Muzio. Synthesis of one-dimensional linear hybrid cellular automata. *IEEE Trans. on Computer-aided design of integrated circuits and systems*, 15(3):325–335, 1996.

[6] K Cattell and J.C Muzio. An explicit similarity transform between CA and LFSR matrices. *Finite fields and their applications*, 4:239–251, 1998.

[7] K Cattell, S Zhang, X Sun, M Serra, J.C Muzio, and D.M Miller. One-dimensional LHCA: their synthesis, properties and applications in VLSI testing. Report, University of Victoria, B.C., Canada, 1998.

[8] S.J Cho, U.S Choi, H.D Kim, Y.H Hwang, J.G Kim, and S.H Heo. New synthesis of one-dimensional 90/150 linear hybrid group CA. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 26(9):1720–1724, 2007.

[9] J. Daemen, R. Govaerts, and J. Vandewalle. A framework for the design of one-way hash functions including cryptanalysis of damgard's one-way function based on a cellular automaton. In *ASIACRYPT: Advances in Cryptology*. LNCS, Springer-Verlag, 1991.

[10] I.B. Damgard. Design principle for hash functions. In *Advances in cryptology CRYPTO 89*, volume 435, pages 416–427, 1990.

[11] P. Joshi, D Mukhopadhyay, and D. RoyChowdhury. Design and analysis of a robust and efficient block cipher using cellular automata. In *Advanced Information Networking and Applications*. IEEE Computer Society, pages 67–71, 2006.

[12] D Kagaris. A similarity transform for linear finite state machines. *Discrete Applied Mathematics*, 154:1570–1577, 2006.

[13] B. Kar, S. Nandi, and PalChaudhury P. Theory and applications of cellular automata in cryptography. *IEEE Transactions on Computer*, 43(12):1346–1357, 1994.

[14] F.J. MacWilliams and N.J.A. Sloane. *The theory of error correcting codes.* North-Holland, 1977.

[15] B. Martin. A Walsh exploration of elementary CA rules. *Journal of Cellular Automata*, 2008. To appear.

[16] W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *EUROCRYPT '89*, pages 549–562, Springer-Verlag, 1990.

[17] W. Meier and O. Staffelbach. Analysis of pseudo random sequences generated by cellular automata. In *EUROCRYPT '91*, Lecture Notes in Computer Science. Springer Verlag, 1991.

[18] M. J. Mihaljevic. An improved key stream generator based on the programmable cellular automata. In *ICICS '97: Proceedings of the First International Conference on Information and Communication Security*, pages 181–191, 1997. Springer-Verlag.

[19] M.J. Mihaljevic, Y. Zheng, and H. Imai. A cellular automaton based fast one-way hash function suitable for hardware implementation. *Lecture Notes in Computer Science*, 1431:217–??, 1998.

[20] M.J. Mihaljevic, Y. Zheng, and H. Imai. A fast and secure stream cipher based on cellular automata over GF(q). In *Global Telecommunications Conference, 1998. GLOBECOM 98*, volume 6, pages 3250–3255, 1998.

[21] D. Mukhopadhyay and D. RoyChowdhury. Cellular automata : an ideal candidate for a block cipher. In *First International Conference on Distribued Computing and Internet Technology ICDCIT 2004*, volume 3347, 2004.

[22] P. Sarkar and S. Maitra. Nonlinearity bounds and constructions of resilient boolean functions. In *CRYPTO '00*, pages 515–532, 2000. Springer-Verlag.

[23] S. Sen, C. Shaw, D RoyChowdhury, N. Ganguly, and P. PalChaudhuri. Cellular automata based cryptosystem (CAC). In *ICICS*, volume 2513 of *Lecture Notes in Computer Science*, pages 303–314. Springer Verlag, 2002.

[24] F. Seredynski, P. Bouvry, and A. Y. Zomaya. Cellular automata computations and secret key cryptography. *Parallel Comput.*, 30(5-6):753–766, 2004.

[25] M. Serra, T. Slater, J.C. Muzio, and D.M. Miller. The analysis of one dimensional cellular automata and their aliasing properties. *IEEE Trans. on Computer-aided design*, 9:767–778, 1990.

[26] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–, 1984.

[27] M. Sipper and M. Tomassini. Co-evolving parallel random number generators. In *Parallel Problem Solving from Nature – PPSN IV*, pages 950–959, 1996. Springer Verlag.

[28] National Institute Of Standards Technology. FIPS publication 140-2, Security requirements for cryptographic modules. US Gov. Printing Office, Washington, 1997.

[29] Yuriy V. Tarannikov: *On resilient Boolean functions with maximal possible nonlinearity,* Springer LNCS 1997 (2000) 19–30.

[30] S. Wolfram. Cryptography with cellular automata. In *CRYPTO 85*, Lecture Notes in Computer Science. Springer Verlag, 1985.

[31] S. Wolfram. Random sequence generation by cellular automata. *Advances in applied mathematics*, 7:123–169, 1986.

[32] S. Wolfram. *A new kind of science*. Wolfram Media Inc., Champaign, Ilinois, US, United States, 2002.

[33] G-Z. Xiao and J. L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Trans. on Information Theory*, 34(3):569–, 1988.