Generating pseudo-random sequences from cellular automata and bent functions[1]

FRANCISCO J. GARCÍA[(a)] VERÓNICA REQUENA[(b)2], VIRTUDES TOMÁS[(b)3]
[(a)]Departament de Fonaments de l'Anàlisis Econòmica
[(b)]Departament de Ciència de la Computació i Intel·ligència Artificial
Universitat d'Alacant
Campus de Sant Vicent del Raspeig
Ap. correus 99, E-03080 Alacant
SPAIN

Abstract: In this paper we construct three different pseudo-random sequence generators based on cellular automata where their local transition functions are constructed from bent functions.

Key-words: Pseudo-random sequence, cellular automata, bent function, balanced function.

1 Introduction

Cellular automata (CA) were initiated in the early 1950s as a general framework for modeling complex structures capable of self-reproduction and self-repair [1]. The pseudo-random sequence generator based on CA has been extensively studied in the last decades. In 1986, Wolfram [7] first applied CA in pseudorandom number generation. After, other authors as Hortensius, Tsalides, Sipper and Perrenoud used the CA to generate the pseudorandom sequences used in cryptography [4, 5, 8]. But those methods are based on artificial methods to construct CA rules. The main disadvantage of those methods is that they are involved in large tedious work. Another valid method is introduced by Sipper and Tomassini [4], they used genetic algorithm to find the best CA randomizer rules automatically. A series of research work has been done to generate pseudo-random sequences.

In this paper, using a one-dimensional finite CA we generate pseudo-random sequences. The use of bent functions as local transition functions can generate more security due to their good cryptographic properties.

The article is organized as follows. In Section 2 we introduce some basic concepts about cellular automata, pseudo-random sequences generators, and bent functions and the notation we will use. In Section 3 we describe some pseudorandom sequence generators. In Section 4 we present some results. Finally, we provide some conclusions in Section 5.

2 Preliminaries

In this section we introduce the main concepts related to cellular automata, pseudo-random sequence generators and bent functions.

2.1 Cellular Automata

Cellular automata are discrete dynamical systems formed by a finite or infinite number of identical objects called cells [6]. These cells are endowed with a state which at a given time depends only on its own state one time step previously, and the states of its nearby neighbors at the previous time step. The states of the cells change at every discrete step of time according to a deterministic rule. The most used CA are finite and one-dimensional. More precisely, a **one-dimensional finite** CA can be defined as a 4-tuple $\mathcal{A} = (C, S, V, f)$. C is the cellular space formed by 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, the set of all possible values of the cells; usually, if the CA considered is finite with k states, then $S = \mathbb{Z}_k$. V is the set of cells which states in the instant t influence in state of the cell considered in the instant $t+1$; in this work and for the particular case of the one-dimension CA, we will consider for every cell $\langle i \rangle \in C$, its neighborhood V_r as the ordered set given by

$$V_r = \{\langle i-r \rangle, \ldots, \langle i \rangle, \ldots, \langle i+r \rangle\}.$$

Moreover, the **local transition function** $f : S^{(2r+1)} \to S$ is the function determining the evolution of the CA throughout the time, i.e., the changes

[1]This work was partially supported by Spanish grant MTM2005-05759.
[2]The work of this author was supported by a grant of the Vicerectorat d'Investigació, Desenvolupament i Innovació of the Universitat d'Alacant for PhD students.
[3]The work of this author was supported by a grant of the Vicerectorat d'Investigació, Desenvolupament i Innovació of the Universitat d'Alacant for PhD students.

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, and $V_i^{(t)}$ is the set of the states of the cell $\langle i \rangle$ at that time; the next state of the cell is given by

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

As the cellular space is finite, boundary conditions must be established in order to ensure that the evolution of the cellular automata is well-defined.

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

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

In particular, $C^{(0)}$ is the **initial configuration**. Hence, the **evolution** of $\mathcal{A}$ is the following sequence

$$\left(C^{(1)}, C^{(2)}, \ldots\right).$$

If we denote by $\mathcal{C}$ the set of all possible configurations of $\mathcal{A}$, then the global function of $\mathcal{A}$ is a linear transformation, $\Phi : \mathcal{C} \to \mathcal{C}$, that yields the configuration at the next time step during the evolution of the CA; that is, $C^{(t+1)} = \Phi(C^{(t)})$. If Φ is bijective then there exists another cellular automaton, $\mathcal{A}^{-1}$, called its **inverse**, whose global function is Φ^{-1}. When such inverse cellular automaton exists, $\mathcal{A}$ is called **reversible** and the evolution backwards is possible ([24]). In general, the evolution of a CA considers that the state of every cell at time $t+1$ depends on the state of its neighborhood at time t, $V_r^{(t)}$. Nevertheless, one can consider that this evolution also depends on the states of other cells at times $t+1, t+2$, etc. In this case, the transition function given in (1) can be represented in the following way

$$a_i^{(t+1)} = \sum_{h=0}^{k} f^{(t-h)} V_r^{(t-h)}$$

where each $f^{(t-h)}$ is a specific local transition function.

2.2 Pseudo-random sequence generator

A random bit generator is a device or algorithm, which outputs a sequence of statistically independent and unbiased binary digits. A pseudo-random bit generator (PRBG) is a deterministic algorithm which, given a truly random binary sequence of length m, outputs a binary sequence of length $k \gg m$ which "appears" to be random. The input to the PRBG is called the **seed** and the output is called a **pseudo-random bit sequence** or **pseudo-random sequence**. Good random properties of the generator are convenient to prevent statistical attacks; but moreover, it is necessary that the generator must be cryptographically secure.

In order to gain confidence in the "randomness" of a pseudo-random sequence, statistical tests are conducted on the produced sequences. In our study, we have considered the six main statistic tests which allow us to check, if a generated random or pseudorandom sequence inhibits certain statistical properties

1. **Frequency Test (Monobit Test)**: Are there equally many 1's like 0's?
2. **Serial Test (Two-bit Test)**: Are there equally many 00-, 01-, 10-, 11-pairs?
3. **Poker Test**: Are there equally many sequences of length q having the same value with q such that $\left\lfloor \frac{k}{q} \right\rfloor \geq 5 \times (2^q)$.
4. **Runs Test**: Are the numbers of runs (sequences containing only either 0's or 1's) of various lengths as expected for random numbers?
5. **Autocorrelation Test**: Are there correlations between the sequence and (non-cyclic) shifted versions of it?
6. **Maurer's Universal Test**: Can the sequence be compressed?

The above descriptions just give the basic ideas of the tests.

2.3 Bent functions

A Boolean function of n variables is a mapping $g : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$. We call the **truth table** of g the $(0,1)$-sequence of length 2^n given by

$$\boldsymbol{\xi}_g = (g(\boldsymbol{u}_0),\ g(\boldsymbol{u}_1),\ \ldots,\ g(\boldsymbol{u}_{2^n-1})).$$

The **Hamming distance** between two $(0,1)$-sequences $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, denoted by $d(\boldsymbol{\alpha}, \boldsymbol{\beta})$, is the number of postions where the two sequences differ. If $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are Boolean functions and $\boldsymbol{\xi}_g$ and $\boldsymbol{\xi}_h$ are the corresponding truth table, the **Hamming distance** between g and h, denoted by $d(g,h)$, is the Hamming distance between the $(0,1)$-sequences $\boldsymbol{\xi}_g$ and $\boldsymbol{\xi}_h$.

A $(0,1)$-sequence is **balanced** if it contains an equal number of 0s and 1s; so, a Boolean function is **balanced** if its truth table is balanced.

We say that $g \in \mathcal{B}_n$ is an **affine function** if it takes the form

$$g(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle \oplus b$$

where $\boldsymbol{a} \in \mathbb{Z}_2^n$ and $b \in \mathbb{Z}_2$. If $b = 0$, we say that f is a **linear function**.

We define the **nonlinearity** of a function $g \in \mathcal{B}_n$ as

$$\mathrm{NL}(g) = \min\{d(g, \varphi) \mid \varphi \in \mathcal{A}_n\}$$

where $\mathcal{A}_n$ is the set of all affine functions. The nonlinearity of g is upper bounded (see [3]) by

$$\mathrm{NL}(f) \leq 2^{n-1} - 2^{\frac{n}{2}-1}.$$

We call **bent functions** the Boolean functions that achieve the maximum nonlinearity (see [3]). Consequently, bent functions only exist for n even.

The following result (see [2, 3]), that we quote for further references, gives us a characterization of a bent function.

Theorem 1: *Let $g(\boldsymbol{x})$ be a Boolean function of n variables. The following statements are equivalent:*

1. *$g(\boldsymbol{x})$ is a bent function.*
2. *For any $\boldsymbol{a} \in \mathbb{Z}_2^n \setminus \{\mathbf{0}\}$, the Boolean function $g(\boldsymbol{x}) \oplus g(\boldsymbol{a} \oplus \boldsymbol{x})$ is balanced.*
3. *$1 \oplus g(\boldsymbol{x})$ is also bent function; moreover, this is the complementary function of g.*

3 Models

In this section we present three different pseudo-random sequence generators based on cellular automata where the local transition function, f, is generate from bent functions.

3.1 Model 1

In the first model of pseudo-random sequence generator, we have considered the CA defined by the 4-tuple $\mathcal{A} = (C, S, V, f)$, where

1. C is the cellular space formed by a linear array of $m = 256$ cells. Initially, we fix a random initial configuration vector

$$C^{(0)} = \left(a_0^{(0)}, a_1^{(0)}, \ldots, a_{m-1}^{(0)}\right).$$

2. S is the finite set of states k, where $S = \mathbb{Z}_k$, with $k = 10000$.
3. For every cell $\langle i \rangle \in C$, we have taken V_2, i.e, the set given by

$$V_2 = \{\langle i-2 \rangle, \langle i-1 \rangle, \langle i \rangle, \langle i+1 \rangle, \langle i+2 \rangle\}$$

.

4. The **local transition function**, f, is based on bent functions in the following way. By Theorem 1 we know that the function defined by $g(\boldsymbol{x}) \oplus g(\boldsymbol{a} \oplus \boldsymbol{x})$, for any $\boldsymbol{a} \in \mathbb{Z}_2^n \setminus \{\mathbf{0}\}$ being g a bent function, is balanced. So, in this case, we take as f this balanced function defined by bent functions of 4 variables and in some specific cases we take the complementary function, $1 \oplus g(\boldsymbol{x})$.

   It is well know that in the case $n = 4$, the number of bent functions is 896, although we only consider the half, 448, so that, these are the main functions, being the rest the complementary functions.

   In each state t for $t > 0$, we need to take a bent function, $g(\boldsymbol{x})$, and a value $\boldsymbol{a}$ to construct the function $g(\boldsymbol{a} \oplus \boldsymbol{x})$. We fix the bent function $g(\boldsymbol{x})$ in the following way. Take the decimal representation of the configuration at time $t-1$, compute it module 448 and add 1; the resulting number will be in the range of the bent functions, and we consider the bent function associated to this number. We distinguish two cases to fix the function f of our CA.

   Let $\boldsymbol{a} = t \mod 2^n$.

   - If $\boldsymbol{a} \neq 0$, then $f(\boldsymbol{x}) = g(\boldsymbol{x}) \oplus g(\boldsymbol{a} \oplus \boldsymbol{x})$.
   - If $\boldsymbol{a} = 0$, then $f(\boldsymbol{x}) = 1 \oplus g(\boldsymbol{x})$.

Then we keep the values $C^{(k)}(n/2)$ which generate the pseudo-random sequence.

3.2 Model 2

In this model and the next, we have taken the same C, S, V, but change the way to choose the local transition function f.

Initially we fix a bent function of 4 variables, $g(\boldsymbol{x})$, and a value of $\boldsymbol{a}$ inside the range $[1, 2^n - 1]$. Here, we do not distinguish different cases according to the state to choose f; however we always take as local transition function the balanced function defined by the bent function fixed; that is, $f(\boldsymbol{x}) = g(\boldsymbol{x}) \oplus g(\boldsymbol{a} \oplus \boldsymbol{x})$.

With these parameters we generate a different pseudo-random sequence for each bent function and for each value of $\boldsymbol{a}$. As in the previous model, we keep the values $C^{(k)}(n/2)$ which generate our pseudo-random sequence.

3.3 Model 3

Finally, this model perform a mixed model using the previous ones.

First, we fix a bent function of 4 variables, $g(\boldsymbol{x})$ and for each function we generate a different pseudo-random sequence.

We have a different local transition function f according to the value $\boldsymbol{a}$ chosen. We take $\boldsymbol{a}$ in the following way. For $l = 0, 1, \ldots, \lfloor \frac{k}{2^n} \rfloor$

$$a = t \mod l(2^n - 1)$$

At the end, we obtain of the same way the pseudo-random sequence generator.

4 Results

In this section we report the main results obtained from the three models of pseudo-random sequence generators explained in the previous section. We have considered random seed vectors for the different models. As a minimum security requirement the length k of the seed to a PRBG should be large enough to make brute-force search over all seeds infeasible for an attacker. The length of our seeds is $m = 256$.

In the first model, we use all bent functions and all values of $\boldsymbol{a}$ for each seed. In this case, all the obtained sequences for the different seeds pass all our tests.

In the second model, for each bent function we fix a value for $\boldsymbol{a}$ and we check some random seeds for these. In this model and the next, we have taken as valid functions those that at least six of the ten seeds pass all the tests. For the different values of $\boldsymbol{a}$, we have obtained that around 63% of the bent function pass all the tests.

In the last model, for each bent function we check some seeds and the results obtained are that around 35% of the bent function pass all the tests.

5 Conclusions

If we observe the previous results it seems to be that the best model to generate pseudo-random sequences based on cellular automata using bent functions is the first model, where we use all bent functions and the different values for $\boldsymbol{a}$ for each seed taken.

Now, our research is based on the study of some particular bent functions that have good behaviour in all the models explained and to observe their results.

References:

[1] P. SAKAR. A brief history of cellular automata. *ACM Comput. Surv*, 32 (1), 2000.

[2] J. SEBERRY and X.-M. ZHANG. Constructions of bent functions from two known bent functions. *Australasian Journal of Combinatorics*, 9: 21–35 (1994).

[3] J. SEBERRY, X.-M. ZHANG and Y. ZHENG. Nonlinearity and propagation characteristics of balanced Boolean functions. *Information and Computation*, 119: 1–13 (1995).

[4] M. SIPPER and M. TOMASSINI. Computation in artificially evolved, non–uniform cellular automaton. *Theoretical Computer Science*, 217(1): 81–98 (1999).

[5] M. TOMASSINI and M. PERRENOUD. Crytography with cellular automata. *Applied Soft Computing Journal*, 1(2): 151–160 (2001).

[6] S. WOLFRAM. Cellular automata. *Los Alamos Science*, 9, 1983.

[7] S. WOLFRAM. Crytography with cellular automata. In *Advances in Cryptology — Crypto'85*, Vol. 218 of *Lecture Notes in Computer Science*, pages 429–432. Springer, Heidelberg, 1986.

[8] S. WOLFRAM. A New Kind of Science. *Wolfram Media, Inc.*, Illinois, 2002.