THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 16, Special Issue 2015, pp. 367-374

AN EFFICIENT PRNG FOR STREAM CIPHERS BASED ON HYBRID CELLULAR AUTOMATA WITH NONLINEAR FEEDBACK

Radu DOGARU, Ioana DOGARU

University Politehnica of Bucharest, Dept. of Applied Electronics and Information Engineering, Natural Computing Laboratory, Romania
E-mail: radu_d@ieee.org

This work proposes a high efficiency PRNG suitable for use in stream ciphers. It is based on hybrid cellular automata with nonlinear feedback to improve its resistance to attacks. It is shown that it can be designed with an arbitrary number N of cells as long as $N>N_{th}$ with no need for designing a new polynomial like in traditional feedback shift registers (it is easily scalable). The space of the possible keys and IV is of the order $O(2^N)$. Statistical batteries of tests such as NIST and Rabbit are all passed when the number of cells N is larger than a threshold value $N_{th}>128$.

Key words: cellular automata, synchronous stream cipher, nonlinear feedback shift register, pseudo-random number generators

1. INTRODUCTION

The pseudo-random number generator (PRNG) is an important part of any synchronous stream stream-cipher [3] since it is responsible for the good cryptographic properties. Several features are of interest in designing a good PRNG for use in a stream cipher:

i) Good statistical properties. They are usually tested using standard batteries of tests like NIST [1] but also more recent battery of tests like TestU01 [2]. It implements a larger variety of tests than traditional test-suites like DIEHARD, NIST, etc. and is recently often used by many authors to assess cryptographic properties of their RNGs;

ii) Immunity to attacks – which can be achieved, in addition to good statistical properties, by the possibility to choose a large state space dimension (usually close to 2^N, where N is the number of cells in the state automata), large Key and IV (initialization vector) spaces. In order to ensure a good immunity to attacks, recently non-linear feedback are widely used [4, 5, 10], instead of the traditional but less secure linear feedback used in LFSR.

iii) High speed and low computational complexity. In many applications high stream rates up to the orders of Gbits/second are required. In order to achieve such rates, parallel platforms are a must, with field programmable gate arrays (FPGA) being a convenient option. Notably, among the many PRNG solutions proposed in the literature, those based on regular structures such as cellular automata are well suited for FPGA implementation achieving stream rates limited only by the maximal clock frequency of the FPGA chip.

Cellular automata (CA), particularly those with nonlinear feedback, are recently regarded as a convenient solution for efficient implementations [4,10]. Among nonlinear feedback automata in [6], authors report a cryptographically efficient (i.e. passing the DIEHARD and NIST tests) configuration (hybrid 37-bit LFSR and 16-bit CA), requiring about 1.37 LUTs/cell in FPGA implementations. Look-up tables = LUT are the basic building blocks in FPGA. Some companies define LE (logic element) as the basic building block, in fact an equivalent of LUT. A similar approach is reported in [7] and also in [8] targeting stream ciphers. Although they were reported as passing statistical test suites, they are difficult to scale-up for arbitrary numbers of cells N, and it is not very clear how they can ensure a large key space. In [9], a sophisticated one dimensional cellular automaton for pseudo-number generator and its FPGA hardware implementation is presented. The cells change the local rule based on the real-time clock of the system. Although their solution

passes both the NIST and DIEHARD tests, the allocated resources are rather high (about 37.8 LE per cell). Also one bit in the state variable is equivalent to one CA cell.

Here in we propose and test the properties of a PRNG (herein called N-HCA, where N stands from the number of cells and HCA stands from hybrid cellular automata first introduced in [11]) aiming to optimize all the above desired features and being applicable for synchronous stream ciphers. As detailed in Section 2, it has a regular lattice structure, and in this respect is somehow similar to Rabbit [12], but we are focusing on a low complexity design (each cell receives information from only 5 adjacent cells using a specified nonlinear operator or Boolean function having a suitable ID which may be considered as a public key. The space of all possible IDs is rather large (2^{32} in the case of 5-cells neighborhood), but we were able so far to identify a list of such IDs providing good PRNGS [13]. Compared with many other CA solutions (a detailed review is given in [14] our N-HCA PRNG ensures a high throughput (equal to the maximal clock rate when implemented in FPGA) and a low computational complexity). In [14, 15] a simpler version of the N-HCA was investigated (with only 3 cells neighborhood and up to N=32 cells). The main disadvantage of a small neighborhood is the reduced space for suitable public keys (there are only 256 possible IDs and among them only a few (four) meet the required PRNG properties (i.e. public keys). We were then interested in achieving good cryptographic properties with a very low number N of cells (up to 32) but in order to achieve it, chain structures or with down-sampling were needed, raising the complexity or lowering the speed when compared to the basic N-HCA model. Since [14] provides an efficient FPGA implementation solution of N-HCA with any desired number of cells, we tested the statistical properties of the sequence generated from an FPGA implementation with N=63 cells and all NIST tests were passed giving us the hint that the very low complexity N-HCA model (called PRNG-A in [14]) can achieve the cryptographic requirements if N is enough large.

In this paper a deeper investigation of the "standard" N-HCA structure with 5-cells neighborhood is considered, in a novel implementation (based on Python 2.7 with the JIT compiler from NUMBA)[1] allowing any N number of cells and investigated the statistical properties for various numbers N of cells. The main result of our work is that for any of the investigated ID (public keys) and with arbitrary secret keys (mask vectors of size N) all the statistical tests (both NIST and the Rabbit and Alphabit from TestU01) passed when $N>N_{th}$. For the worst case, N_{th} was found to be around 128. Section 3 presents detailed results of the PRNG statistical tests and Section 4 concluding remarks.

2. THE N-HCA CELLULAR AUTOMATA PRNG

The N-HCA model and its relation to a synchronous stream cipher is presented in Fig.1. In the following we will assume that our RNG is designed to be used in a stream cipher, consequently we will specify the secret key, the IV and the state space as well as the state bit used to generate the cipher-text by XOR-ing it with the plaintext. The rate of our cipher is limited only by the clock signal. In a sequential implementation (micro-controller, etc.) the rate is given by the clock frequency divided by N while in fully parallel implementation (e.g. FPGA) equals the CLK rate. With actual FPGAs one can easily obtain rates as high as 500 Mbits/s.

The HCA model is based on the following equation:

$$y = x_i\,(t+1) = m_i \oplus Cell\left(x_{i+2}(t), x_{i+1}(t), x_i\,(t), x_{i-1}(t), x_{i-2}(t), ID\right). \quad (1)$$

All HCA cells are updated simultaneously according to (1). Each cell has a position index i and a binary state $x_i(t)$. The index ranges from 1 to N while extreme cells are connected (ring topology). The ID is a decimal representation of the truth table defining the local Cell() Boolean function. As discussed later, it corresponds to an ANF (Algebraic Normal Form) representation of the cell in form of a linear or nonlinear polynomial (in direct relation to the physical implementation). Particular ID values have to be identified such that the dynamics of the N-HCA described by (1) is a complex one (chaotic), potentially useful for cryptographic applications. Further statistical tests should be applied to the generated streams to confirm

[1] https://store.continuum.io/cshop/academicanaconda

these properties. A review of the methods used to locate suitable IDs is presented in [13]. For cryptographic purposes one may consider ID as a public key.

The Initialization Vector (IV) is the initial state vector $X(0)=[x_{N-1}(0), x_{N-2}(0),..., x_0(0)]$. The PRNG keystream is represented by the dynamics of the $x_0(t)$ state bit. A secret key can be defined in the form of a binary mask vector $\mathbf{m}=[m_{N-1}, m_{N-2},.., m_i,.., m_0]$ and it can be randomly chosen from a space of 2^N possibilities. As shown in our previous works [11, 13] the mask can be optimized for maximal cycle length (period close to 2^N) when N is enough small (up to N=29 with actual computing capabilities). For larger N values such optimization is not necessary (in fact is also not possible) since, as results in Section 3 show, statistical properties are slightly influenced by the mask choice, the larger N (i.e. N>128) the smaller is this influence. For N>128 a random choice of the mask vector can be accepted with no problems.

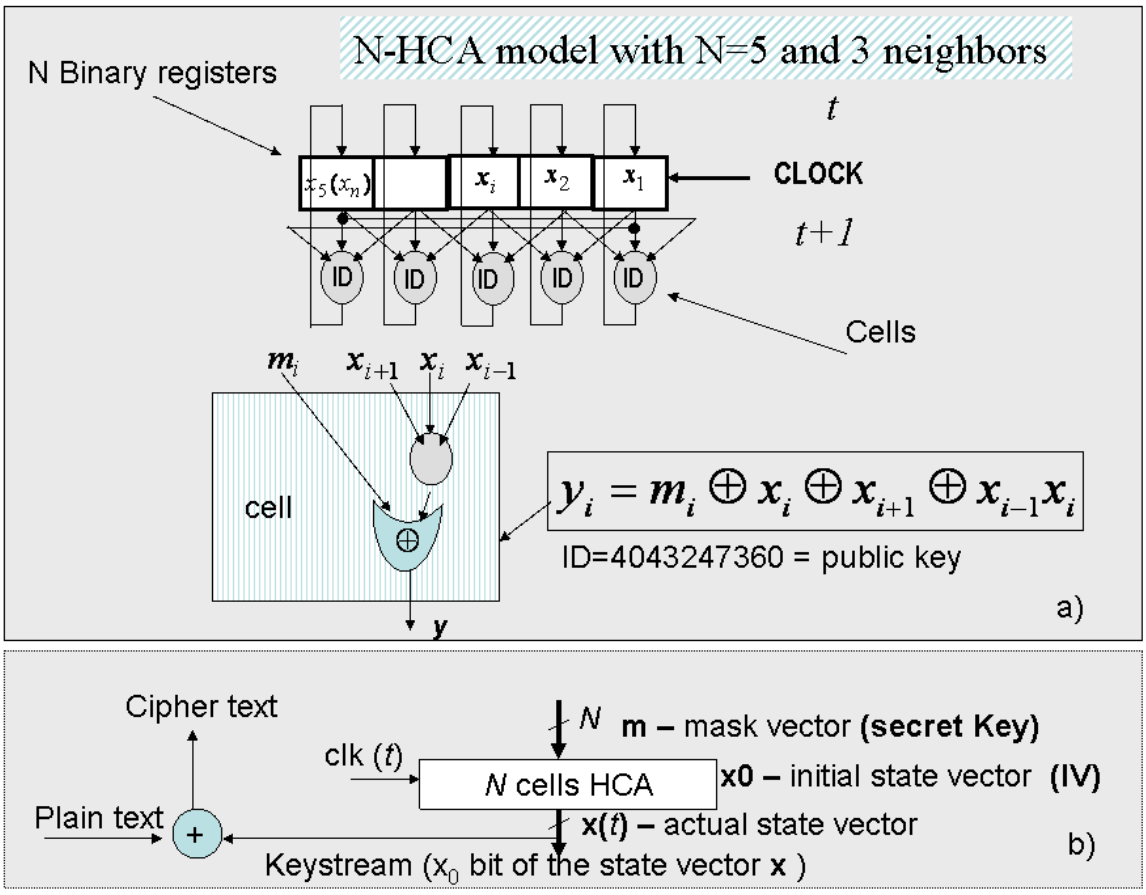

Fig. 1 – The structure of an N-HCA nonlinear cellular automaton: a) up – the "standard" N-HCA, here presented with 3 neighbours, easily expandable to 5-cells neighbourhood b) its role as PRNG in generating the keystream of a synchronous stream cipher. The receiver must act in synchrony and use the same IV and mask (secret key).

For any neighbourhood size (here m=5, leading to $M=2^m$ possible binary input vectors for the cell) the relationship between inputs and the output local CA rule can be characterized in two different ways and conversion functions are available [16]: a) ***Truth-Table*** *(TT)* representation: This is the most widely used representation. The rule is characterized by a binary vector $Y=[y_{M-1}, y_{M-2},..., y_0]$. Its representation in decimal basis is called a rule identifier (ID). The output y_k is a binary number assigned to the cell's output when its inputs ordered as a binary vector $[u_{m-1}, u_{m-2},.., u_0]$ are the binary representation of *k;* b) ***Algebraic Normal Form (ANF):*** This form is described by a binary vector $C=[c_0, c_1,...c_{M-1}]$ (using the method in [16] a unique conversion from **Y** to **C** and vice-versa exists) such that its coefficients are multipliers of an algebraic representation on the GF_2 exemplified next for the case of m=3 neighbourhood:

$$y = c_0 \oplus c_1 u_0 \oplus c_2 u_1 \oplus c_3 u_1 u_2 \oplus c_4 u_3 \oplus c_5 u_2 u_0 \oplus c_6 u_2 u_1 \oplus c_7 u_2 u_1 u_0 . \quad (2)$$

Note that in general (for any size *m* of the neighbourhood) c_k is the multiplier of a product (logical AND) of all input variables in a binary vector $[u_{m-1}, u_{m-2}, .., u_0]$ corresponding to 1 in the associated binary vector representing *k*. For example, in the case of *m*=3 (above equation) for $k = [u_2, u_1, u_0] = 5 = 101_2$ only the inputs corresponding to 1 in the input string $u_2 u_1 u_0$ are selected to be multiplied resulting in this case the term $c_5 u_2 u_0$. The ANF representation was found extremely useful for FPGA implementations since it allows a direct translation into a simple VHDL line describing the entire HCA structure [17]. ANF is also very useful to reveal whether the automata network is a linear or a non-linear (better immunity to algebraic attacks) one. A nonlinear automaton has at least one term with more than 2 inputs in the ANF equation (2).

3. RESULTS IN APPLYING STATISTICAL TESTS

The host computer used to report the results herein is an average PC from Dell with Pentium Dual Core CPU, E7500 @ 2.93GHz, 2 GB of RAM, running Windows. With this implementation a 100 million length key-stream is generated by an *N*-HCA with *N*=512 cells in 1800 seconds. Time of the same order is needed to run all NIST tests on the same computational platform. Masks (secret keys) were selected randomly as *N*-sized binary vectors. In all cases the initial state (IV) was a CA state with all bits 0 except the middle one which was set on 1. Public keys were various IDs (among the good PRNG ones, with 5-cell neighborhood), specified in the next for each experiment.

NIST tests: In this case, if not specified otherwise a sequence of 100 million bits was generated using various *N*, IDs (public keys) and private keys (masks). Let us consider some examples:

ID= 3432828060. It corresponds to the following third-degree ***nonlinear*** cell polynomial: $y_i = x_{i-2} \oplus x_i \oplus x_{i+1} \oplus x_{i-1} x_i \oplus x_{i+2} x_i \oplus x_{i-1} x_i x_{i+2}$ revealed by the ANF of the specified ID [13]. For reasonable low number of bits *N* (e.g. *N*<29) when a systematic study of the attractors is possible, it was revealed that this a very particular ID where very long sequences (periods close to 2^N) can be obtained for either odd or even values of *N*. This property makes it useful to build a "chaotic counter" for square image sensors in [18] covering almost all pixels in one single long and chaotic loop. All the other IDs discussed herein are also of the conservative type (no transients, i.e. all states are in loops) but only when *N* is odd. Consequently, for even values of *N* shorter PRNG sequences (so, less convenient cryptographically) are obtained. The results of the statistical tests are reported in Table 1 showing that when $N > N_{th} \geq 128$ all 188 tests are passed successfully.

Table 1

Statistical tests for *ID*= 3432828060

N	32	64	128	256	512
Failed tests (in 188)	3	1	0	0	0
Name of failed test:	BlockFrequency FFT Universal	NonOverlappingTemplate	-	-	-

Next, some more detailed analysis was performed for N-HCA with *ID*=4043247360, corresponding to a second-degree nonlinear cell polynomial $y_i = x_i \oplus x_{i+1} \oplus x_{i-1} x_i$ (one among the 4 *ID*s for good PRNG among all with 3-cell neighborhood). Results are summarized in Table 2 revealing $N_{th} = 31$ for odd numbers of cells (predicted by previous research results as giving the better PRNG sequences).

Table 2

Statistical tests for *ID*= 4043247360 (10 million bits in 100 sequences of 0.1 million each)

N	17	19	21	23	27	31	33	65	511
Failures	152	118	20	3	1	0	0	0	0

The above results indicates that for each particular ID there may be a different N_{th}, the smaller its value the more effective is the PRNG implementation (i.e. passing statistical tests with less resources). For the next *ID*, it is shown that choosing an odd *N* makes a difference in passing the statistical tests (which are not passed for *N*=64 but passed for *N*=65). However, for enough large number of cells (128, 256, etc.) tests are passed for either odd or even *N* (now the state space is so high that the sequence length is extremely large in any case).

Table 3

Statistical tests for *ID*= 2779097770 (100 million bits in 100 sequences)

N	64	65	128	256
Failures	27	0	0	0

A similar behavior is observed for the *ID* considered in Table 4.

Table 4

Statistical tests for *ID*= 2947502160 (100 million bits in 100 sequences)

N	64	128	256	512
Failures	58	0	0	0

TestU01 statistical tests*:* For *ID*=3432828060 (the only one guaranteeing good cryptographic properties for either odd or even *N)*, with small number of cells (*N*=32) the following 3 among 38 tests failed (Fig.2a) for a certain **key1** selected as mask vector.

```
       Test                                   p-value
 ----------------------------------------------------
  2   ClosePairsBitMatch,     t = 2          4.9e-04
  8   Fourier3                               1.1e-46
 10   PeriodsInStrings                       eps
 ----------------------------------------------------
All other tests were passed
```

a)

```
 Test                                        p-value
 ----------------------------------------------------
  1  MultinomialBitsOver                     1 -  7.8e-05
  2  ClosePairsBitMatch, t = 2               7.6e-06
  3  ClosePairsBitMatch, t = 4               1.9e-06
  8  Fourier3                                7.1e-107
 10  PeriodsInStrings                        eps
 25  RandomWalk1 R (L = 1024)                2.5e-04
 ----------------------------------------------------
All other tests were passed
```

b)

Fig. 2 – ID=3432828060, with small number of cells (*N*=32); 1 million bits: a) Rabbit tests with key1; b) Rabbit tests with key2.

Increasing the number of cells *N*=64 the result is "All tests were passed" is the outcome of both Rabbit and Alphabit batteries. Note that for the above *N* values other batteries of tests in TestU01 (e.g. AlphabitFile with 1048576 bits) gave "all passed" results.

Sensitivity to keys: Next we considered another randomly assigned key (or mask vector) key2 and performed the same tests as above for *N*=32, 64, 128, 256, 512. Only for the case *N*=32 the Rabbit failed a number of 6 among 38 tests (now a bit larger than in the case of the previous key) as seen in Fig.2b. Alphabit tests passed in this case as for key1. All the other cases gave "all passed" results. Thus, with small number of cells *some sensitivity to the key is observed*, but when increasing *N* properly such sensitivity disappears. This property stands for any other of the tested IDs as well.

The influence of the tested stream size: All the above results were obtained with a stream of only 1 million bits (giving a high speed of the test battery). Let us examine the influence of the number of bits on the result of the Rabbit test. In the next experiments streams of 100 million bits were used (the same as used in the NIST tests). Now much more tests are failing for low *N*. The case of *N*=32 is given bellow. There are 11 tests among all 40 failing now. Also, the Alphabit battery of tests gives 9 failures in this case. As

expected, the larger the number of bits provided to the statistical test, the more relevant and safer is the result. Of course, providing large sequences implies long times for both their generation and testing.

```
      Test                                  p-value
----------------------------------------------------
 1  MultinomialBitsOver                    1 -  1.1e-05
 2  ClosePairsBitMatch, t = 2             1.1e-06
 3  ClosePairsBitMatch, t = 4             1.0e-09
 8  Fourier3                              eps
10  PeriodsInStrings                      eps
12  HammingCorr, L = 32                   1 - eps1
13  HammingCorr, L = 64                   1 -  3.5e-10
16  HammingIndep, L = 32                  eps
17  HammingIndep, L = 64                  eps
24  RandomWalk1 H                         eps
24  RandomWalk1 R                         1.6e-04
----------------------------------------------------
All other tests were passed
```

Results obtained for various numbers of cells *N* are summarized in the next table (Table 5):

Table 5

Statistical tests for *ID*=3432828060

N	32	64	128	256	512
R Failed tests (in 40)	11	2	0	0	0
A Failed tests (in 17)	9	0	0	0	0

In reports on TestU01 tests "A" refers to "Alphabit" and "R" refers to the "Rabbit" battery of tests. The results in Table 5 are in concordance with results from NIST tests applied in the same conditions, showing that all tests are passed if $N_{th} \geq 128$. Similar studies were performed for other public keys (IDs) and summarized in the next tables 6 and 7:

Table 6

Statistical tests for *ID*= 2779097770

N	64	65	128	256
R Failed tests (in 40)	14	0	0	0
A Failed tests (in 17)	11	0	0	0

As discussed previously and revealed by the NIST tests as well, for these 2 IDs there is a completely different result when *N*=65 versus *N*=64: The explanation was already given in the case of NIST tests (they give good periods only for odd *N*). Still, when N is large enough even values can be considered as well. The reports in Tables 6, 7 show that for these particular IDs the threshold N_{th} is a bit higher than for the other IDs (now *N*=64 gives failures) confirming also a similar finding from NIST tests.

Table 7

Statistical tests for *ID*= 2947502160

N	64	128	256
R Failed tests (in 40)	16	1	0
A Failed tests (in 17)	12	0	0

Finally, the same ID and in the same conditions as in Table 2 (NIST) is now investigated and results are reported in Table 8 for TestU01 battery. Observe that now the threshold is a bit higher (33 instead of 31) revealing that TestU01 may be in general considered a bit more severe than NIST (noted recently by other authors as well).

Table 8

Statistical tests for *ID*= 4043247360 (10 million bits stream in 100 sequences of 0.1 million each)

N	17	19	21	23	27	31	33	65	511	1024
R Failed tests (in 39)	33	23	13	5	1	2	0	0	0	0
A Failed tests (in 17)	17	15	4	2	1	1	0	0	0	0

For those interested in further statistical or security tests, a sequence of 100 million bits (in binary format) is provided at the following address *http://atm.neuro.pub.ro/radu_d/seq_512_3432828060_refm.dat* It was generated using *N*-HCA with *N*=512 cells.

4. CONCLUSIONS

A Nonlinear Feedback Cellular Automata (N-HCA) was investigated from the perspective of using it as a cryptographically good PRNG for building synchronous stream ciphers. It has a very low complexity and can be implemented as a fast throughput solution (particularly in fully parallel implementations e.g. FPGA) [14, 19]), its complexity being limited only by the number *m* of neighbour cells. While *m*=3 neighbour cells ensures the lowest implementation complexity (1 LUT per cell in FPGA [14]), with a slight sacrifice in complexity increasing it to *m*=5 ensures a larger space of possible HCA architectures with multiple choices for the public key. Also, in order to increase complexity and immunity to algebraic and other kind of attacks chained architectures based on the principle exposed in [15] can be easily derived from the model discussed herein.

Both NIST and TestU01 battery of statistical tests were applied to some representative IDs with different numbers of cells *N* confirming that the proposed PRNG is cryptographically secure when the number of cells *N* is sufficiently large. Given the N-HCA structure the key space is also increasing when *N* grows, giving more immunity to attacks. In addition, the nonlinearity of the feedback makes the resulting key-stream less vulnerable to algebraic attacks. Further research will consider applying a wider battery of tests and considering various kind of attacks (e.g. algebraic attacks [20]) when the proposed PRNG is integrated in a synchronous stream cipher. Methods such as [21] an others used in [22] to assess the security of nonlinear feedback shift register (NLFSR) ciphers will be also considered, since our HCA it is actually a NLFSR.

REFERENCES

1. National Institute of Standards and Technology, *Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules*, US Government Printing Office, Washington, 1999.
2. P. L'ECUYER, R. J. SIMARD, *Testu01: A C library for empirical testing of random number generators*, ACM Trans. Math. Softw., **33**, *4*, paper 22, 2007.
3. E.P. DAWSON, L.SIMPSON, *Analysis and Design Issues for Synchronous Stream Ciphers*, Niederreiter, Harald (Ed.) Coding Theory and Cryptology, Singapore University Press World Scientific, Singapore, pp. 49-90, 2002.
4. S. GHOSH, A. SENGUPTA, D. SAHA, D. R. CHOWDHURY, *A Scalable Method for Constructing Non-linear Cellular Automata with Period 2n − 1*, J. Was, G.C. Sirakoulis, and S. Bandini (Eds.): ACRI 2014, LNCS, **8751**, pp. 65–74, 2014.
5. E. DUBROVA, *A Scalable Method for Constructing Galois NLFSRs With Period 2^n-1 Using Cross-Join Pairs*, IEEE Transactions on Information Theory, **59**, *1*, pp. 703-709, 2013.
6. J. C. CERDA, et al., *An Efficient FPGA Random Number Generator using LFSRs and Cellular Automata*, IEEE 55th International Midwest Symposium Circuits and Systems (MWSCAS), pp.912-915, 5-8 Aug 2012.
7. J.M. COMER, et al., *Random Number Generators using Cellular Automata Implemented on FPGAs*, 44th IEEE Southeastern Symposium on System Theory University of North Florida, Jacksonville, FL, pp.67-72, March 11-13, 2012.
8. L. RAUT, D. H. K. HOE., *Stream Cipher Design using Cellular Automata Implemented on FPGAs*, 45th Southeastern Symposium on System Theory Baylor University, Waco, TX, USA, pp.146-149, March 11, 2013.
9. L. KOTOULAS, et al., *1-D Cellular Automaton for PseudoRandom Number Generation and its Reconfigurable Hardware Implementation*, IEEE International Symposium on Circuits and Systems, ISCAS, 21-24 May, 2006.
10. M.A.B.SHEMAILI, C.Y. YEUN, M.J. ZEMERLY, K. MUBARAK, *A Novel Hybrid Cellular Automata Based Cipher System for Internet of Things*, Future Information Technology, Lecture Notes in Electrical Engineering, **276**, James J. (Jong Hyuk) Park et al. (eds.), Springer-Verlag Berlin Heidelberg, pp. 269-276, 2014.

11. R. DOGARU, *Hybrid Cellular Automata as Pseudo-Random Number Generators with Binary Synchronization Property*, Proceedings of the International Symposium on Signals Circuits and Systems (ISSCS'09), Iasi Romania, pp. 389-392, July 2009.
12. M. BOESGAARD, M. VESTERAGER, T. PEDERSEN, J. CHRISTIANSEN, O. SCAVENIUS, *Rabbit: a new high-performance stream cipher*, Johansson, T. (ed.), FSE, LNCS, **2887**, pp. 325–344. Springer, Heidelberg, 2003.
13. R. DOGARU, I. DOGARU, *Applications of Natural Computing in Cryptology: NLFSR based on Hybrid Cellular Automata with 5-cell Neighborhood*, Proceedings of the Romanian Academy, Series A, **14**, pp. 365–372, 2013.
14. I. DOGARU, R. DOGARU, *FPGA implementation and evaluation of two cryptographically secure hybrid cellular automata*, Communications (COMM), 10th International Conference on, pp.1-4, 29–31 May 2014.
15. R. DOGARU, I. DOGARU, *Efficient and cryptographically secure pseudorandom number generators based on chains of hybrid cellular automata maps*, Communications (COMM), 10th International Conference on , pp.1-4, 29–31 May 2014.
16. S. RONJOM, M. ABDELRAHEEM, L. E. DANIELSEN, *TT and ANF Representations of Boolean functions*, Online Database of Boolean Functions, 2007. Available: *http://www.selmer.uib.no/odbf/help/ttanf.pdf* (last time accessed May 2015).
17. I. DOGARU, R. DOGARU, *Algebraic Normal Form for Rapid Prototyping of Elementary Hybrid Cellular Automata in FPGA*, Proceedings ISEEE 2010 (September 2010, Galati, Romania), pp. 273-276, 2010.
18. R. DOGARU, I. DOGARU, H. KIM, *Chaotic Scan: A Low Complexity Video Transmission System for Efficiently Sending Relevant Image Features*, IEEE Trans. on Circuits and Systems for Video Technology, **20**, *2*, pp. 317 – 321, 2010.
19. I. DOGARU, R. DOGARU, *A comparative study of several 2D cellular automata implementations in FPGA*, Fundamentals of Electrical Engineering (ISFEE), International Symposium on, pp.1–4, 28–29 Nov. 2014.
20. W. MEIER, E. PASALIC, C. CARLET, *Algebraic Attacks and Decomposition of Boolean Functions*, Advances in Cryptology – EUROCRYPT-2004, Lecture Notes in Computer Science, Berlin, Germany, Springer-Verlag, **3027**, pp. 474–491, 2004.
21. P. STANKOVSKI, *Greedy distinguishers and nonrandomness detectors*, Progress in Cryptology – INDOCRYPT 2010 (G. Gong and K. C. Gupta, eds.), Lecture Notes in Computer Science, **6498**, pp. 210–226, Springer, 2010.
22. E. DUBROVA, M. HELL, *Espresso: A Stream Cipher for 5G Wireless Communication Systems*, Cryptology ePrint Archive, 2015.