Pseudorandom Number Generation Based on Controllable Cellular Automata

Sheng-Uei Guan and Shu Zhang

Department of Electrical & Computer Engineering
National University of Singapore
10 Kent Ridge Crescents, Singapore 119260
{eleguans, engp9594}@nus.edu.sg

Abstract A novel Cellular Automata (CA) — Controllable CA (CCA) is proposed in this paper. Further, CCA are applied in Pseudorandom Number Generation. Randomness test results on CCA Pseudorandom Number Generators (PRNGs) show that they are better than 1-d CA PRNGs and can be comparable to 2-d ones. But they do not lose the structure simplicity of 1-d CA. Further, we develop several different types of CCA PRNGs. Based on the comparison of the randomness of different CCA PRNGs, we find that their properties are decided by the actions of the controllable cells and their neighbors. These novel CCA may be applied in other applications where structure non-uniformity or asymmetry is desired.

Key words: cellular automata, randomness test, pseudorandom number generator, controllable, hybrid

1. Introduction

A central problem in any stream cipher scheme is to generate long, unpredictable key sequences. Since real random numbers can be obtained from some physical phenomena only, they are difficult to be employed in real applications. Hence, pseudorandom numbers generated by artificial design patterns have to be used instead. One commonly used type of Pseudorandom Number Generators is based on

Cellular Automata. Cellular Automata (CA) was initiated in the early 1950s as a general framework for modeling complex structures capable of self-reproduction and self-repair.

In 1986, Wolfram first applied CA in pseudorandom number generation [18]. The intensive interest in this field can be attributed to the phenomenal growth of the VLSI technology that permits cost-effective realization of the simple structure of local-neighborhood CA. Wolfram's work in [18] proved that the randomness of the patterns generated by maximum-length CA is significantly better than other widely used methods, such as Linear Feedback Shift Registers (LFSRs).

In the last decade, one-dimensional (1-d) CA Pseudorandom Number Generators (PRNGs) have been extensively studied [2,5,6,8,9,11,12,13,16,17]. Recent interest is more focused on two-dimensional (2-d) CA PRNGs [3,10] since it seems that their randomness is much better than that of 1-d CA PRNGs. But taking into account the design complexity and computation efficiency, it is quite difficult to conclude which one is better. Compared to 2-d CA PRNGs, 1-d PRNGs are easier to be implemented in a large scale. In this paper, we propose a family of novel CA PRNGs that obtain the same randomness as that of 2-d CA PRNGs without losing the structure simplicity of 1-d CA PRNGs.

In the following, we first give an overview on CA and CA PRNGs in section 2. Then in section 3, we present the definition and properties of a set of novel CA PRNGs — Controllable CA PRNGs. Section 4 compares Controllable CA PRNGs to 1-d and 2-d CA PRNGs. Section 5 delineates a set of Controllable CA PRNGs and compares their properties. Finally, we end the paper with a short discussion and conclusion in section 6 and 7.

2. Related Work

2.1 Introduction to Randomness Tests

Real random numbers can only be obtained from some physical phenomenon. The random numbers generated by PRNGs are not truly random, but only pseudo-random. Although this assertion is inevitable, we would still like to obtain sequences that behave as if they were random. Thus, the problem is how to decide whether the sequences are random enough? Statistical (empirical) test could be a good solution in this respect. If a sequence passed a number of quantitative tests, we could assert that the sequence is random. But we should also note that there are no guarantees, only predictions in numerical practice are possible. Following, we introduce two widely used randomness test suites—ENT test and DIEHARD. The former is designed according to the criteria set by Knuth [1]; the latter is devised by G. Marsaglia [4]. Detailed introduction to these two tests is given in appendix.

2.2 Cellular Automata Background

A cellular automaton is an array of cells where each cell is in any one of its permissible states. At each discrete time step (clock cycle) the evolution of a cell depends on its transition rule (the combinatorial logic), which is a function of the present states of its *k* neighbors for a *k*-neighborhood CA. The cellular array (grid) is n-dimensional, where n=1,2,3 is used in practice. We define the state of a CA at time t to be the n-tuple formed from the states of the individual cells, $x(t)=[x_1(t),...,x_n(t)]$. The next-state function of a 3-neighborhood (r=1) CA is computed as: $x(t+1)=f(x(t))=[f_1(0,x_1(t),x_2(t)),...,f_i(x_{i-1}(t),x_i(t),x_{i+1}(t)),...]$. When each f_i is a linear function, f is also a linear function, mapping n-tuples to n-tuples. The evolution of the i-th cell in a one-dimensional, 3-neighborhood CA can be represented as a function of the present states of the (i-1)-th, (i)-th, and (i+1)-th cells as: $x_i(t+1)=f_i(x_{i-1}(t),x_i(t),x_{i+1}(t))$, where f_i represents the transition rule for the (i)-th cell.

Some definitions used extensively to characterize the properties of CA are noted below.

Definition 1. If the rules of a CA cell involve only XOR logic, then it is called a *linear rule.* Rules involving XNOR logic are referred to as *complemented rules.* A CA with all the cells having linear rules is called a *linear CA*, whereas a CA having a combination of XOR and XNOR rules is called an *additive CA.*

Definition 2. If all the CA cells obey the same rule, then the CA is said to be a *uniform CA*; otherwise, it is a *non-uniform CA.*

Definition 3. A CA is said to be a *Periodic Boundary CA* (PBCA) if the extreme cells are adjacent to each other. A CA is said to be an *Intermediate Boundary* CA (IBCA) if the next state of the left-(right-) most cell depends on itself, its right (left) neighbor, and the one next to (before) it. A CA is said to be a *null-boundary* CA if the extreme cells are only connected to its left (right) cell.

Programmable CA (PCA) is initially mentioned in [17]. It is a non-uniform CA that allows different rules to be used at different time steps on the same cell. Compared to uniform CA, PCA allows several control lines per cell. Through these control lines, different functions (rules) can be applied to the same cell at different time steps according to the rule control signals. Fig. 1 is an additive PCA cell structure. Applying such a cell structure to each cell in an N-cell PCA, all the possible additive non-complemented rule combinations can be achieved to realize any non-uniform additive PCA. Such an N-cell PCA can implement 8^N different CA configurations.

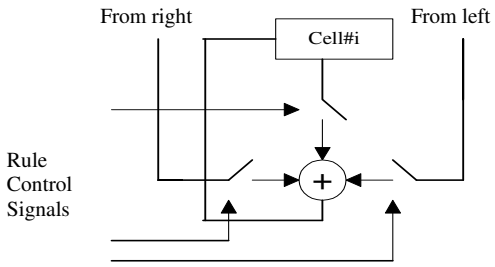


Fig. 1 An additive PCA cell structure

Generally, transition rule is one of the critical factors that decide the property of CA, whether it is uniform CA or PCA. Since there is a lot of work done to explore the properties of different rules, we only use those rules that have been proved to be good in random number generations in our work. Here we give the Boolean form of these rules and their numbers are given in accordance with Wolfram's convention.

Rule 30: $x_i(t+1) = x_{i-1}(t)$ XOR ($x_i(t)$ OR $x_{i+1}(t)$)

Rule 90: $x_i(t+1) = x_{i-1}(t)$ XOR $x_{i+1}(t)$

Rule 150: $x_i(t+1) = x_{i-1}(t)$ XOR $x_i(t)$ XOR $x_{i+1}(t)$

Rule 165: $x_i(t+1) = x_{i-1}(t)$ XNOR $x_{i+1}(t)$

Rule 105: $x_i(t+1) = x_i(t)$ XNOR ($x_{i-1}(t)$ XOR $x_{i+1}(t)$)

2.3 CA Pseudorandom Number Generators

In this section, we discuss several 1-d and 2-d CA PRNGs proposed since the last decade. Before we proceed to introduce the generators, we first investigate what properties of CA will affect the randomness of the sequences generated by CA PRNGs.

In general, there are four aspects of CA configuration affecting the randomness:

- Boundary conditions — null boundary or periodic boundary: generally periodic boundary condition is better than null boundary condition in random number generation.
- Length of a CA — the total number of cells in a CA: A CA formed from N cells with a single rule generally has a cycle length much shorter than 2^N-1. As the length of the CA increases the maximum possible cycle length of the pseudorandom sequence increases.

- Initial seed — the initial state configuration in CA: Generally, the effect of initial seed on randomness is trivial. To counteract its effect, in the following work, we apply the randomness test on a set of randomly generated initial seeds instead of only one.
- Transition rule — Obviously, the randomness of the sequences generated by different rules varies a lot.

2.3.1 1-d CA PRNGs

Rule-30 uniform CA has been extensively studied by Wolfram in 1986 [18]. It is the first time that scientists applied CA in random number generation. Wolfram's work on rule-30 CA demonstrated its ability to produce highly random, temporal bit sequences [19]. Wolfram also suggested that rule-30 CA can be efficiently implemented in parallel. A study in [13] shows that in rule-30 CA, the correlation dies out and a bit stream 3-4 cells away is no longer correlated. Therefore, the auto- and cross-correlation of a rule 30 CA is less than that of a parallel LFSR (Linear Feedback Shift Register) generator. Later, other rules are also applied in uniform CA PRNGs. Tomassini et al. concluded in [9] that according to the DIEHARD test results, rule 105 is the best RNG, followed by rules 165, 90 and 150, with rule 30 coming in last.

Following the idea of uniform CA PRNGs, more researchers focus their interest on non-uniform CA PRNGs since non-uniform CA PRNGs are better than uniform ones in general. The first non-uniform CA PRNG was proposed by P. D. Hortensius in 1989 [12]. This non-uniform CA uses rule 90 and 150. This CA PRNG is referred to hence as PCA 90-150. Surprisingly, the combination of these two simple linear rules yields maximum length cycles. Unlike uniform rule-30 CA, adjacent cells in non-uniform CA are not correlated in both time and space [12]. However, the binary sequences

produced by some cells in a non-uniform CA fail some random number tests because of distribution problems. Another non-uniform CA PRNG which uses the combination of rules 30 and 45 was also proposed by P. D. Hortensius [13]. This generator can evolve to a random pattern of outputs, but its bit sequence correlation is much higher than that of the PCA 90-150 [13].

Later in 1996, Sipper and Tomassini [11] evolved a 50-cell CA with a mélange of rules 90, 150 and 165. This CA is referred to henceforth as PCA 90-165. Based on their work, Tomassini et al. [9] evolved another 50-cell CA with the rule combination 90, 165,150 and 105 in 1999. This CA is referred to henceforth as PCA 90-105. These two CA are evolved using cellular programming evolution algorithm while those two CA proposed by P. D. Hortensius are handcrafted. The DIEHARD test results show that these two non-uniform CA PRNGs are better than those designed by P. D. Hortensius in [12,13]. But they still cannot pass some of the tests in DIEHARD and are inferior to the classical generators.

2.3.2 2-d CA PRNGs

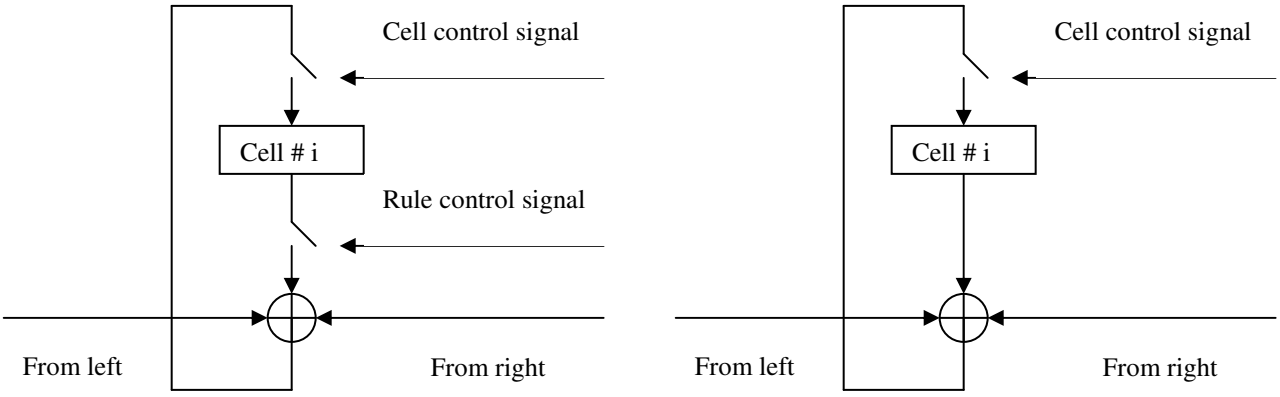
Work on 1-d CA PRNGs not only shows the suitability of CA in random number generation but also raises another question: is it possible to further improve the randomness of CA PRNGs? Chowdhury et al. [3] described a methodology for producing pseudorandom numbers by 2-d CA in 1994. Their results suggest that 2-d CA are superior to 1-d ones of the same size in pseudorandom number generation. Following his idea, Marco Tomassini et al. evolved several 8×8 2-d CA PRNGs with rule 15, 63, 31 and 47. DIEDARD test results show that some of the evolved CA PRNGs can pass all the tests in DIEHARD. And based on the observation of these evolved 2-d CA PRNGs, they can handcraft even better PRNGs.

Although 2-d CA PRNGs are better than 1-d CA PRNGs in random number generation, they lose the structure simplicity and computation efficiency of 1-d CA PRNGs. Therefore, how to find a set of CA PRNGs which not only can pass all the DIEHARD test but also have the merit of 1-d CA PRNGs becomes an important problem. Under this motivation, we propose a novel CA in the next section.

3. Controllable CA PRNGs

3.1 Controllable CA

Following the idea of PCA in which a rule control line is added into each cell to improve the randomness of CA PRNGs, we add more control lines on CA cells to control the neighborhood relation and updating of states to further improve the randomness of 1-d CA PRNGs. This results in a new type of CA — Controllable Cellular Automata. In this section, the structure and properties of CCA are presented. To explain the scheme explicitly, several new concepts are defined first to identify the CCA properties.



(a) Programmable controllable cell **(b) Non-programmable controllable cell**

Fig. 2 A controllable cell structure

Definition 4. A *Controllable CA* (CCA) is a CA in which some cells can be controlled via cell control signals.

Definition 5. If a cell is under the control of a cell control signal, it is a *controllable cell*; otherwise it is a *basic cell*. CCA is the combination of controllable cells and basic cells. Both controllable cells and basic cells could have rule control signals. Fig. 2 shows the non-programmable/programmable controllable cell structure. In this paper, we will discuss programmable controllable cells only.

The *status* of a controllable cell refers to how its state will be computed in the current time step. It is decided by its current cell control signal. The status of a controllable cell can be active or inactive. If the state of a controllable cell is computed as usual (only according to the current rule control signals and the states of its neighbors), it is an *active* cell and the states of its neighbors are computed as usual too. If the state of a controllable cell is not updated (its state will remain stable or be complemented) in the current time step, it is an *inactive* cell. The inactive controllable cell is referred to henceforth as an idle cell. How the states of its neighbors will be computed is decided by certain rules. And these rules decide the property of CCA.

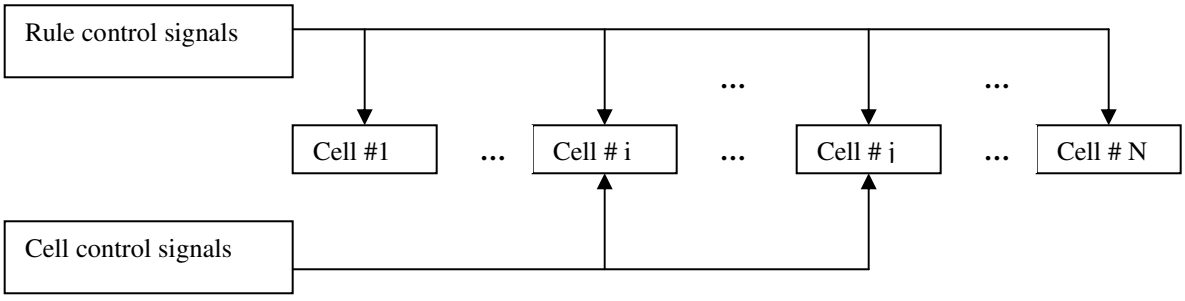


Fig. 3 The structure of a CCA

The structure of a CCA is shown in Fig. 3. All the cells in a CCA have rule control lines while only those controllable cells (i.e. cell #i and cell #j) have cell control lines. Compared to a PCA in the same size, CCA has several more cell control lines. The number of cell control lines is the same as that of

controllable cells in CCA. In our work, the rule (or cell) control signals are generated by a rule (or cell) control CA. In the next sub-section, we will present several basic CCA PRNGs and discuss their properties. Randomness tests show that CCA PRNGs are good in random number generation.

The usage of controllable cells in CCA differentiates itself from PCA in which only rule control signals exist. Once the actions of controllable cells and basic cells are specified, the setting of controllable cells will decide the performance of CCA. The common idea in PCA and CCA is that they both use some control lines on the CA cells to make the CA transition more unpredictable and flexible. The difference is that in PCA all the cells have uniform structures while obviously in CCA the structure of controllable cells are not the same as that of basic cells.

3.2 Basic Controllable CA PRNGs

The simplest example we can easily think of is that when a controllable cell is inactive, it keeps its latest state and the states of its neighbors are computed as usual. In this way, the controllable cell will affect the state computation of its neighbors during the current time step in the same way as the previous time step. This model is referred to as CCA0. Note that the one-bit memory of the inactive controllable cell is somehow wasted as we don't change its content, we can have a variation: when the controllable cell is inactive, its state is complemented. This model is referred to as CCA1. These two models are quite easy to be implemented in hardware and we even can find a PCA to simulate each. We may wonder that if so, what's the difference between a CCA0 (or CCA1) PRNG and PCA PRNG. To answer this question, we design the following randomness test on CA PRNGs. The ENT test is used to evaluate the randomness of CA PRNGs. The final result is a real number that is referred to

henceforth as randomness value. This value is calculated based on the test results of Entropy, SCC and Chi-square test using a function F:

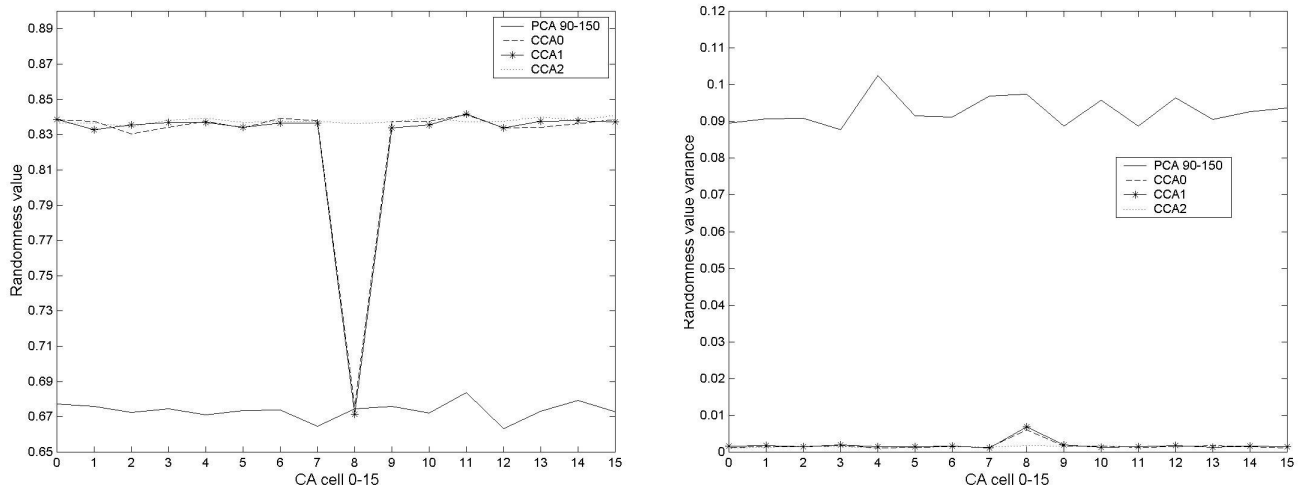
$$F = (\text{Entropy} - 6) * 20\% + (1 - |SCC|) * 20\% + f(\text{Chi-square}) * 60\%$$

$$f(\text{Chi-square}) = \begin{cases} 0; & \text{if Chi-square} > 90\% \text{ or } < 10\% \\ 1; & \text{if } 10\% < \text{Chi-square} < 90\% \end{cases}$$

The reason why we adjust the original test values is to ensure that the final randomness value falls into [0,1] and a higher value represents better randomness. The objective of this test is to compare not only the randomness of CCA PRNGs and PCA PRNGs but also the randomness of controllable cells and basic cells. CA PRNGs generate random number sequence as following: each cell generates a random bit sequence. In each time step, the state of a cell is recorded. A cell's randomness value is the ENT test result on the sequence it generated. This test is repeated 100 times with 100 randomly generated initial seeds (for rule control CA and cell control CA, tested CA). The average randomness value is recorded as the final result and the variance of these 100 tests are recorded in the meantime. In all the figures presented in this paper, the randomness value is the average value. Fig. 4 shows the test result on CCA PRNGs and PCA 90-150 PRNGs. L is the length of CA; M is the number of cycles CA runs through.

The tested CCA0 and CCA1 both have only one controllable cell (cell 8). The rule combination and rule control signals for PCA, CCA0 and CCA1 are identical. As shown in Fig. 4 (a), all the PCA cells obtain a randomness value about 0.68. The basic cells in CCA0 have a randomness value about 0.84 while the controllable cell gets a much lower value which is nearly the same as that of PCA cells. Note that CCA0 and CCA1 get similar test result, we can derive that the 'complement' action of a controllable cell is not useful to improve its randomness in this case. Although the randomness of a

controllable cell is much worse than that of a basic cell, it improves the overall randomness of CCA0 and CCA1. Referring to Fig. 4 (b), we can see that the variance of CCA0 and CCA1 cells, whether they are basic cells or controllable cells, is much lower than that of PCA cells. It means that the controllable cell can also improve the overall performance stability of CCA0 and CCA1 PRNGs.



(a) Randomness value **(b) Randomness value variance**

Fig. 4 Comparison of PCA/CCA0/CCA1/CCA2 (L=16 bits, M=800 cycles)

Note: Definition of CCA2 is in the next page

The shortcoming of CCA0 and CCA1 is that the randomness of controllable cells is much worse than that of basic cells. To solve this problem, we design another CCA — where the neighbors of a controllable cell will bypass it when it is inactive. That means the inactive controllable cell won't be involved in the state computation of its neighbors. In this way, the neighborhood relationship is dynamically changed during the CA computation process. This type of CCA is referred to henceforth as CCA2 or Neighbor-changing CA (NCA). Note that CCA2 cannot be simulated by a PCA due to its neighbor-changing behavior. The randomness test result of CCA2 PRNG is also shown in Fig. 4. We can see that for CCA2, the randomness of a controllable cell is similar to that of a basic cell.

	Chi-square (pass rate)	Entropy (average value)	SCC (average value)
PCA 90-150	70%	6.101210	0.121479
CCA0	100%	6.241210	0.049906
CCA1	100%	6.268758	0.040313
CCA2	100%	6.423856	0.008055

Table 1 ENT test results of PCA/CCA0/CCA1/CCA2 PRNGs

Till now, all the random number sequences are sequentially generated by one cell of CA PRNGs. But generally CA PRNGs generate pseudorandom numbers in parallel. To evaluate the randomness of parallel CCA PRNGs, we design the following test: All the tested CA PRNGs have a length of 16. For each generator, an 8-bit integer is generated as the output random number during each time step. Each CA PRNG runs M cycles to generate a sequence of M random numbers. This sequence is evaluated with the ENT test. For each CA PRNG, this test is repeated 100 times with 100 randomly generated initial seeds. The pass rate in the Chi-square test is the percentage of tests whose Chi-square test result is between 10% and 90%. The entropy and SCC value are the arithmetic mean of all the 100 tests. Table 1 shows the test results on PCA, CCA0, CCA1 and CCA2 PRNGs. All the CCA PRNGs can pass the Chi-square test at 100% while PCA PRNGs get a much lower pass rate of 70%. CCA2 gets the highest entropy and SCC value among all the tested CA PRNGs. Therefore, we can say that CCA2 PRNG is the best among them. Since CCA0 and CCA1 get similar results in all the following tests, we will only discuss CCA0 in the following work to represent both of them. In the next section, we compare the randomness of CCA0/CCA2 PRNGs with that of 1-d PCA PRNGs and 2-d CA PRNGs.

4. Controllable CA PRNGs vs 1-d/2-d CA PRNGs

4.1 CCA0 PRNGs vs 1-d CA PRNGs

We have shown in the last section that according to the ENT test results, CCA PRNGs are better than PCA 90-150 PRNGs. To confirm this, we use a more complete test suite — DIEHARD in the following work. The tests in DIEHARD require a large binary file of random integers to be tested. In our work, all the random number sequences generated by CA PRNGs are 10M bytes. Table 2 shows the DIEHARD test results on PCA 90-150 and CCA PRNGs. The lengths of all the tested PRNGs are fixed as 64 cells. The rule combination, rule control signals and cell control signals are identical for all tested CA PRNGs. Rule combination is 90 and 150. Rule control CA uses rule 105. Cell control CA uses rule 165. In CCA PRNGs, the controllable cells are averagely distributed among basic cells. Site (cell) spacing and time spacing are used in PCA and CCA PRNGs to remove correlation. The site spacing parameter *ss* is the number of sites between outputs in CA. The time spacing parameter *ts* is the number of time steps between output numbers.

We first test the CA PRNGs without time spacing. To find a suitable site spacing parameter, we tested 3 different values: ss = 1, 2 or 3. Referring to Table 2, we can see that when *ss*=2, both CCA0 and PCA 90-150 PRNGs get the best results. But they still cannot pass all the testes in DIEHARD. M. Tomassini suggested in [9] that time spacing is crucial to generate a very high quality random number sequence. Our test results are in accord with his suggestion. With a time spacing of 1, CCA0 PRNGs can pass all the tests in DIEHARD. Since under all the circumstance, CCA0 PRNGs pass more tests than PCA 90-150 PRNGs, we can conclude with confidence that CCA0 is better than PCA 90-150 in random number generation.

Test name	P=32, ss=1, ts=0		P=16, ss=2, ts=0		P=8, ss=3, ts=0		P=16, ss=2, ts=1	
	PCA 90-150	CCA0 /CCA2 ccn=8	PCA 90-150	CCA0 /CCA2 ccn=8	PCA 90-150	CCA0 /CCA2 ccn=8	PCA 90-150	CCA0 /CCA2 ccn=8
1. Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
2. Runs up 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs up 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
3. 3D sphere	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
4. A parking lot	Fail	Pass	Fail	Pass	Fail	Pass	Fail	Pass
5. Birthday Spacing	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
6. Count the ones 1	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
7. Binary Rank 6*8	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
8. Binary Rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
9. Binary Rank 32*32	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
10. Count the ones 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
11. Bitstream test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
12. Craps wins	Fail	Pass	Fail	Pass	Pass	Pass	Pass	Pass
throws	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
13. Minimum distance	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass
14. Overlapping Perm.	Fail	Fail	Pass	Pass	Fail	Pass	Pass	Pass
15. Squeeze	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
16. OPSO test	Pass	Pass	Pass	Pass	Fail	Fail	Pass	Pass
17. OQSO test	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass
18. DNA test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Number of tests passed	***12***	***14***	***15***	***17***	***13***	***16***	***17***	***18***

Table 2 DIEHARD test result on PCA/CCA0 (L=64)
P=32/16/8-bit integer; ***p1***: averagely interleaving
ss: site spacing; ***ts***: time spacing; ***ccn***: number of controllable cells
count the ones 1: count the ones for specific bytes;
count the ones 2: count the ones for a stream of bytes

In [9], Marco Tomassini et al. showed the DIEHARD test results of PCA 90-165 and PCA 90-105 (in their work, these two CA are referred to as Sipper and Tomassini, and Tomassini et al., respectively). Table 3 shows the test results of their CA PRNGs and ours. Without time spacing, both PCA 90-165 and PCA 90-105 PRNGs fail at least 5 tests while CCA0/CCA2 PRNGs only fail one test. With a time spacing of 2, both PCA and CCA0/CCA2 PRNGs can pass 13 tests while they did not

provide the test results on the other 5 tests. Hence, we can conclude here that CCA0/CCA2 PRNGs are comparable to or better than PCA 90-105 and PCA 90-165 PRNGs.

Test name	L=50, ts=0, ss=2			L=50, ts=1, ss=2		
	CCA0/ CCA2	PCA 90-165	PCA 90-105	CCA0/ CCA2	PCA 90-165	PCA 90-105
1. Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass
2. Runs up 1	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 1	Pass	Pass	Pass	Pass	Pass	Pass
Runs up 2	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 2	Pass	Pass	Pass	Pass	Pass	Pass
3. 3D sphere	Pass	Pass	Pass	Pass	Pass	Pass
4. A parking lot	Pass	Pass	Pass	Pass	Pass	Pass
5. Birthday Spacing	Pass	Fail	Fail	Pass	Pass	Pass
6. Count the ones 1	Pass	Fail	Fail	Pass	Pass	Pass
7. Binary Rank 6*8	Pass	Pass	Pass	Pass	Pass	Pass
8. Binary Rank 31*31	Pass	Fail	Fail	Pass	Pass	Pass
9. Binary Rank 32*32	Pass	Fail	Fail	Pass	Pass	Pass
10. Count the ones 2	Pass			Pass		
11. Bitstream test	Pass			Pass		
12. Craps wins	Pass	Pass	Pass	Pass	Pass	Pass
throws	Pass	Pass	Pass	Pass	Pass	Pass
13. Minimum distance	Pass	Fail	Fail	Pass	Pass	Pass
14. Overlapping Permu	Fail	Pass	Pass	Pass	Pass	Pass
15. Squeeze	Pass	Pass	Pass	Pass	Pass	Pass
16. OPSO test	Pass			Pass		
17. OQSO test	Pass			Pass		
18. DNA test	Pass			Pass		
Number of tests passed	***17***	***8***	***8***	***18***	***13***	***13***

Table 3 DIEHARD test results of PCA 90-105/PCA 90-165/CCA0/CCA2 PRNGs (L=50 cells)

Considering that CCA use more control lines than PCA, we may suspect that whether we can further improve PCA's random quality with more control lines? 2-bit PCA PRNGs, which use two control lines per programmable cell, may be a good example to be compared with CCA PRNGs. In a 2-bit PCA, 4 rules are available for each cell during CA computation. Here, rules 90, 150, 165 and 105 are chosen. Table 4 presents the DIEHARD test results of PCA 90-150, CCA0, CCA2 and 2-bit PCA

with 50 cells. It shows that with a time spacing of 1 and site spacing of 2, both CCA0/CCA2 and 2-bit PCA PRNGs can pass all the tests in DIEHARD while PCA 90-150 PRNGs fail one test.

Test name	L=50, P=16, ss=2, ts=1			L=32, P=16, ss=1, ts=1			L=16, P=8, ss=1,ts=1		
	PCA 90-150	2-bit PCA	CCA0/ CCA2	PCA 90-150	2-bit PCA	CCA0/ CCA2	PCA 90-150	2-bit PCA	CCA0/ CCA2
1. Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
2. Runs up 1	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Pass	Pass
Runs Down 1	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
Runs up 2	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
Runs Down 2	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
3. 3D sphere	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
4. A parking lot	Pass	Pass	Pass	Fail	Fail	Pass	Fail	Pass	Pass
5. Birthday Spacing	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
6. Count the ones 1	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
7. Binary Rank 6*8	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
8. Binary Rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
9. Binary Rank 32*32	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
10. Count the ones 2	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
11. Bitstream test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
12. Craps wins	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Pass	Pass
throws	Pass	Pass	Pass	Fail	Pass	Pass	Fail	Fail	Fail
13. Minimum distance	Pass	Pass	Pass	Fail	Pass	Pass	Fail	Fail	Fail
14. Overlapping Permu	Fail	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
15. Squeeze	Pass	Pass	Pass	Fail	Pass	Pass	Fail	Fail	Fail
16. OPSO test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
17. OQSO test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
18. DNA test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
Number of tests passed	***17***	***18***	***18***	***9***	***12***	***13***	***0***	***3***	***3***

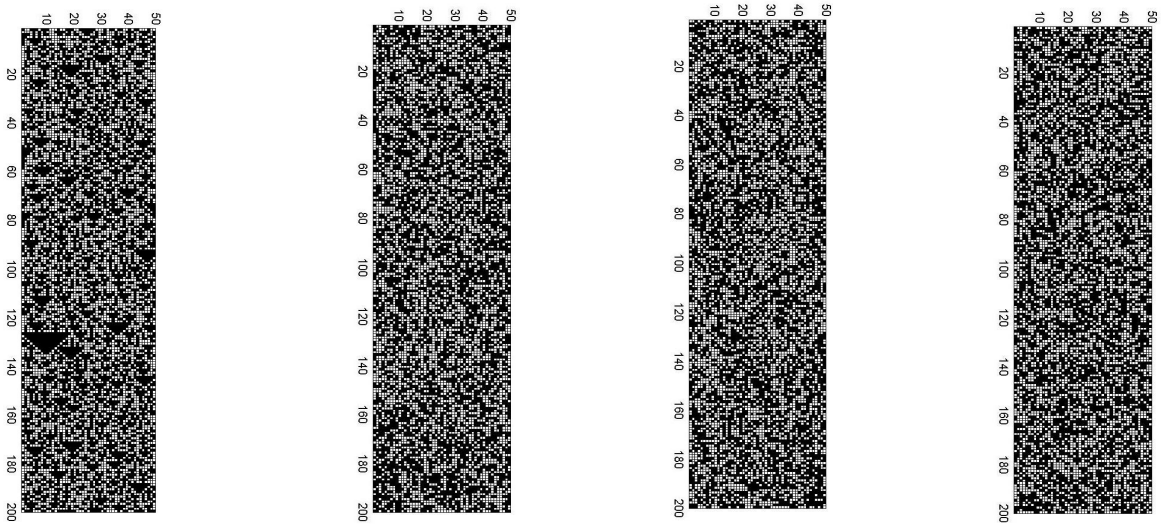
Table 4 DIEHARD test results of 2-bit PCA/CCA0/CCA2 PRNGs

Table 4 also presents the DIEHARD test results of CA PRNGs where L=16 and 32. Since the length of CA is critical to its randomness, we can say that a CA with less than 32 cells cannot generate good random number sequences. Under all the circumstance, CCA0/CCA2 PRNGs get better randomness than PCA 90-150 PRNGs. When the length of CA is 32, CCA0/CCA2 PRNGs can pass one more test than 2-bit PCA PRNGs. Further we apply the ENT test on CCA0, CCA2 and 2-bit PCA PRNGs. The test results are shown in Table 5. All of them can pass the Chi-square test at 100%. In the

SCC and entropy test, CCA2 PRNG gets better result than 2-bit PCA and CCA0 gets similar result as 2-bit PCA.

	Chi-square (pass rate)	Entropy (average value)	SCC (average value)
CCA0	100%	6.241210	0.049906
2-bit PCA	100%	6.308587	0.030309
CCA2	100%	6.423856	0.008055

Table 5 ENT test results of 2-bit PCA/CCA0/CCA2

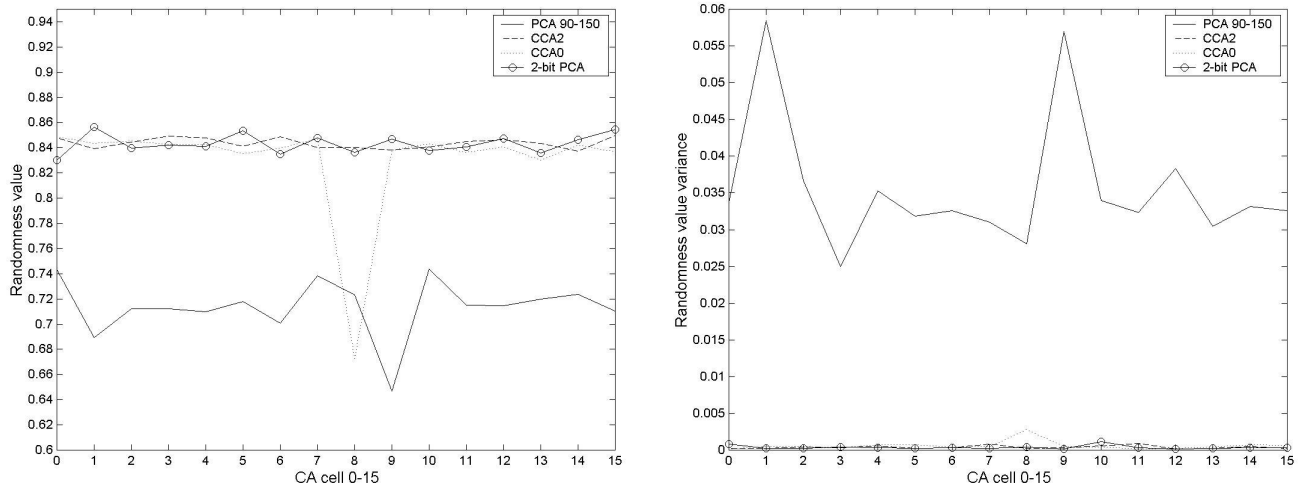


PCA 90-150 **2-bit PCA** **CCA0** **CCA2**

Fig. 5 Time-space diagram of PCA 90-150, CCA0 and 2-bit PCA

To examine the behavior of PCA 90-150, CCA0 and 2-bit PCA, we present the two-dimensional time-space diagram as shown in Fig. 5. Each generator has 50 cells and runs 200 time steps. The x-axis is from cell 1 to 50; the y-axis is from time step 0 to 200 (from top to bottom). Obviously, PCA 90-150 has some patterns inside. CCA0, CCA2 and 2-bit PCA have no such patterns in their diagrams.

Fig. 6 shows the randomness value of each cell in a 2-bit PCA PRNG where the randomness value function is defined as in section 3.2. The results show that a 2-bit PCA PRNG gets similar result as CCA2 PRNG and slightly better result than CCA0 PRNG. According to the DIEHARD/ENT/randomness value test results and the time-space diagram, we can conclude that CCA2 is better than 2-bit PCA and 2-bit PCA is comparable to CCA0 in random number generation. While considering the high wiring cost of 2-bit PCA, it is hard to say which one is better between CCA0 and 2-bit PCA PRNGs.



(a) Randomness value **(b) Randomness value variance**

Fig. 6 Comparison of PCA/CCA0/CCA2/2-bit PCA (L=16, M=800)

4.2 CCA PRNG vs 2-d CA PRNG

We have introduced in section 2.3 that Marco Tomassini et al. evolved several 2-d CA PRNGs. In [10], they showed that some of the evolved 8×8 2-d CA PRNGs could pass all the tests in DIEHARD. Referring to Table 2, we can see that CCA0 PRNG with a time spacing of 2 can pass all the tests in DIEHARD too. Thus, we may say that according to the DIEHARD test results, CCA0 PRNGs can compete with 2-d CA PRNGs. Except statistical tests, cycle length (the length of a CA's state cycle) is

also important to determine whether a CA is suitable for random number generation. Table 6 shows the cycle length of PCA 90-150, 2-bit PCA, CCA0, CCA1 and 2-d CA. The results are calculated as average values over 20 random initial seeds. The results show that the average length of PCA 90-150 is the smallest. The average cycle length of CCA0 is slightly greater than 2-bit PCA, but less than CCA2. This matches the conclusion we have derived from the ENT test that CCA2 is better than CCA0 in random number generation. The average cycle length of 2-d CA is greater than CCA0 but less than CCA2. It means that CCA PRNGs can be better than or comparable to 2-d CA PRNGs and their performance depends on their configurations.

Type(No. of cells)	Avg. cycle length	Max cycle length	Max/Avg.	Log_2 (Avg. cycle)
PCA 90-150 (16)	2521	65536	26.0	11.3
2-bit PCA (16)	2943	65536	25.8	11.4
CCA0 (16)	3179	65536	20.6	11.6
CCA2 (16)	15411	65536	4.25	13.96
2-d CA (4×4)	4778	65536	13.72	12.22

Table 6 Average and maximum cycle length of CA PRNGs

5. Discussions

We have introduced CCA0, CCA1 and CCA2 in the last two sections. In this section, we discuss various issues in CCA configurations. We first discuss the usage of controllable cells in CCA0/CCA2 PRNGs.

5.1 Number of Controllable Cells in CCA0/CCA2 PRNGs

From the view of hardware design, the shortcoming of a controllable cell is that it needs one more control line than a basic cell. Too many controllable cells will increase design complexity and lose the merit of structure simplicity. On the other hand, if too few controllable cells are used, it may take a

long time for controllable cells to affect the other basic cells in the CA computation process. Thus, the problem lies in the choice of a suitable ratio of controllable cells to basic cells in CCA.

Test name	CCA0 L=64, P=16, ss=2, ts=1			CCA2 L=64, P=16, ss=2, ts=1		
	ccn=1	ccn=8	ccn=64	ccn=1	ccn=8	ccn=64
1. Overlapping sum	Pass	Pass	Fail	Pass	Pass	Pass
2. Runs up 1	Pass	Pass	Fail	Pass	Pass	Pass
Runs Down 1	Pass	Pass	Fail	Pass	Pass	Pass
Runs up 2	Pass	Pass	Fail	Pass	Pass	Pass
Runs Down 2	Pass	Pass	Fail	Pass	Pass	Pass
3. 3D sphere	Pass	Pass	Fail	Pass	Pass	Pass
4. A parking lot	Pass	Pass	Fail	Pass	Pass	Pass
5. Birthday Spacing	Pass	Pass	Fail	Pass	Pass	Pass
6. Count the ones 1	Pass	Pass	Fail	Pass	Pass	Pass
7. Binary Rank 6*8	Pass	Pass	Fail	Pass	Pass	Fail
8. Binary Rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass
9. Binary Rank 32*32	Pass	Pass	Fail	Pass	Pass	Pass
10. Count the ones 2	Pass	Pass	Fail	Pass	Pass	Fail
11. Bitstream test	Pass	Pass	Fail	Pass	Pass	Pass
12. Craps wins	Pass	Pass	Fail	Pass	Pass	Pass
throws	Pass	Pass	Fail	Pass	Pass	Pass
13. Minimum distance	Pass	Pass	Fail	Pass	Pass	Pass
14. Overlapping Permu	Pass	Pass	Fail	Pass	Pass	Pass
15. Squeeze	Pass	Pass	Fail	Pass	Pass	Pass
16. OPSO test	Pass	Pass	Fail	Pass	Pass	Pass
17. OQSO test	Pass	Pass	Fail	Pass	Pass	Fail
18. DNA test	Pass	Pass	Fail	Pass	Pass	Fail
Number of tests passed	***18***	***18***	***1***	***18***	***18***	***14***

Table 7 DIEHARD test results of CCA0/CCA2 with different number of controllable cells

To compare the randomness of parallel CCA0/CCA2 PRNGs in various configurations, we apply the DIEHARD test on three CCA0/CCA2 configurations: all CCA0/CCA2 PRNGs have 64-cells; each configuration may have one controllable cell, 8 or 64 controllable cells. The test condition is the same as that in Table 2. Table 7 shows the test results. In CCA0, the first two configurations exhibit similar randomness quality and pass all the tests in DIEHARD; the third one with 64 controllable cells can only pass one test in DIEHARD. It shows that in CCA0 too many controllable cells will degrade the overall randomness of CCA0 PRNGs. In CCA2, the first two configurations can pass all the tests in

DIEHARD too; the third one can pass 14 tests. It shows that too many controllable cells in CCA2 will also degrade the overall randomness quality. Comparing to the test results of CCA0 in which the third configuration can only pass one test, obviously CCA2 gets better test results than CCA0 under this configuration. It means that the randomness of CCA2 has less dependency on their controllable cell configurations than CCA0. In other words, we can say that CCA2 is more stable than CCA0 in random number generation.

5.2 Choice of Output Cells in CCA

We have shown in the last subsection that a suitable ratio of controllable cells in CCA is important to generate good random number sequences. Assuming that we have already found a good ratio of controllable cells, we would face another problem — which cells should be chosen to generate the output? A cell that generates output bits is called an output cell. Generally, not all the cells in a CCA will generate output bits in each time step because the correlation of connected cells is larger than that of those unconnected ones. Thus, how to choose the output cells is a critical issue in CCA PRNGs. We have pointed out that for CCA0, the randomness of a controllable cell is much worse than that of a basic cell. Therefore, in this paper, we only use the basic cells as output cells in CCA0. In CCA2, the controllable cells get similar randomness value as basic cells. Thus, all the cells in CCA2 are eligible to generate output bits. To decrease the correlation of connected cells, we use site spacing when choosing output cells. Referring to Table 2, we can see that a suitable site spacing parameter improves the randomness of CCA PRNGs.

5.3 Randomness Value Function Variations

We have introduced a function F in section 3.2 to calculate the randomness value of cells based on the ENT test results. In that function, 60%, 20% and 20% have been chosen as the weightage of the Chi-Square, SCC and Entropy test results to derive the final randomness value. The reason for having this ratio is that the Chi-Square test is the most important one among the three tests. A higher ratio of the Chi-square test result in the randomness value may help us to better in differentiating good random sequence from bad ones. Later test results show that the randomness value of cells is generally higher than 0.6, which means that most of the tested sequences get 1 in the original Chi-Square test results. Under this situation, the 0.6 value seems to be a waste since almost all sequences can get it. We have tried different weightages of the Chi-Square, SCC and entropy test results like 4:3:3 or even 3.3: 3.3.5:3.3.5, however, the results remain the same.

6. Conclusion

In this paper, we have discussed several CCA PRNGs and compare them with PCA PRNGs and 2-d CA PRNGs. We find that CCA PRNGs are better in random number generation than PCA PRNGs. They can compete with 2-d CA PRNGs while their design complexity and wiring cost will be lower than that of the 2-d ones. We have also discussed several different types of CCA PRNGs. CCA0 and CCA1 have the simplest configurations. CCA2 gets the best randomness value among all the tested generators. All of them can pass the entire tests in DIEHARD. Further, these CCA PRNGs may be applied in applications other than random number generation where structure non-uniformity or asymmetry is desired.

REFERENCES

[1] D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd ed. Reading, Mass.: Addison-Wesley, 1998.

[2] De la Guaz-Martinez and Amparo Fuster-Sabater, Cryptographic design based on cellular automata, In Proceedings of 1997 IEEE International Symposium on Information Theory, 1997, pp. 180.

[3] D. R. Chowdhury, I. S. Gupta and P. Pal Chaudhuri, A class of two-dimensional cellular automata and applications in random pattern testing, Journal of Electrical Testing: Theory and Applications, Vol. 5, 1994, pp. 65-80.

[4] G. Marsaglia, "Diehard", http://stat.fsu.edu/~geo/diehard.html, 1998.

[5] I. Kokolakis, I. Andreadis and Ph. Tsalids, Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators, Microprocessors and Microsystems, Vol. 20, 1997, pp. 643-658.

[6] M. Matsumoto, Simple cellular automata as pseudorandom m-sequence generators for built-in self-test, ACM Trans. on Modeling and Computer Simulation, Vol. 8, No. 1, 1998, pp. 31-42.

[7] M. Mihaljevic, Security examination of a cellular automata based pseudorandom bit generator using an algebraic replica approach, In Proceedings of Applied Algebra, Algorithms and Error Correcting Codes, Lecture notes in Computer Science, Vol. 1255, 1997, pp. 250-262.

[8] M. Mihaljevic and Hideki IMAI, A family of fast keystream generators based on programmable linear cellular automata over GF(q) and time-variant table, IEICE Trans. Fundamentals, Vol. E82-A, No. 1, 1999, pp. 32-39.

[9] Marco Tomassini, Moshe Sipper, Mose Zolla and Mathieu Perrenoud, Generating high-quality random numbers in parallel by cellular automata, Future Generation Computer Systems, Vol. 16, 1999, pp. 291-305.

[10] Marco Tomassini, Moshe Sipper and Mathieu Perrenoud, On the generation of high-quality random numbers by two-dimensional cellular automata, IEEE Trans. Comput., Vol. 49, 2000, pp. 1146-1151.

[11] Moshe Sipper and Marco Tomassini, Generating parallel random number generators by cellular programming, International Journal. Modern Physics, Vol. 7, No.2, 1996, pp.181-190.

[12] P. D. Hortensius, R.D. Mcleod, and H.C. Card, Parallel random number generation for VLSI system using cellular automata, IEEE Trans. Comput., Vol. 38, 1989, pp. 1466-1473.

[13] P. D. Hortensius, R. D. Mcleod, Werner Pries, D. Michael Miller and H. C. Card, Cellular automata-based pseudorandom number generators for built-in self-test, IEEE Transactions on Computer-Aided Design, Vol. 8, No. 8, 1989, pp. 842-859.

[14] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chattopadhyay, Additive cellular automata: theory and applications, Vol. 1, Los Alamitos, Calif.: IEEE CS Press, 1997.

[15] Palash Sarkar, A brief history of cellular automata, ACM Computing Surveys, Vol. 32, No. 1, 2000, pp. 80-107.

[16] Paul H. Bardell, Analysis of cellular automata used as pseudorandom pattern generators, In Proceedings of International Test Conference (1990), 1990, pp. 762-768.

[17] S. Nandi, B. K. Kar, and P. Pal Chaudhuri, Theory and applications of cellular automata in cryptography, IEEE Trans. Comput. 43, 1994, pp. 1346-1357.

[18] S. Wolfram, Cryptography with cellular automata, Advances in Cryptography: Proceedings of CRTPTO 85, Lecture Notes in Computer Science, Vol. 218, 1985, pp. 429-432.

[19] S. Wolfram, Theory and Applications of Cellular Automata: Including Selected Papers 1983-1986, World Scientific publishing Co., Inc., River Edge, NJ. 1986.

[20] W. Pries, A. Thanailakis, and H.C. Card. Group properties of cellular automata and VLSI applications, IEEE Trans. Comput., Vol. C-35, 1986, pp. 1013-1024.

[21] ENT test suite, http://www.fourmilab.ch/random

APPENDIX

1. ENT Test

ENT is a Pseudorandom Number Sequence Test Program, which applies various tests to sequences of bytes stored in files and reports the results of those tests [21]. This program is useful for evaluating pseudorandom number generators for encryption. ENT performs a variety of tests on the input stream of bytes in *in_file* and produces output as follows on the standard output stream:

- Entropy: the information density of the contents of the file, expressed as a number of bits of character. The optimal value is 8. Larger value means better randomness.
- Chi-square Test: the most commonly used test for the randomness of data, sensitive to errors in pseudorandom sequence generators. This test is calculated for the stream of bytes in the file and expressed an absolute number and a percentage that indicates how frequently a truly random sequence would exceed the value calculated. "Good" results are between 10%-90%, with extremities on both sides representing non-satisfactory random sequences.
- Serial Correlation Coefficient (SCC): measures the extent to which each byte in the file depends upon the previous byte. For random sequences, this value should be close to 0. Whether the value is positive or negative does not affect the randomness and smaller absolute value means better randomness.

2. DIEHARD

DIEHARD seems to be the most powerfully general test for randomness. Generally, a PRNG which can pass DIEHARD can be considered good. The DIEHARD battery of test consists of 18 different, independent statistical tests. Results of tests are so called "P-value" which is a real number between 0 and 1. For any given test, smaller P-value means better test result with the exception that a P value less than 0.025 or larger than 0.975 means that the PRNG has failed the test at the .05 level. A complete description of all the tests in DIEHARD is available in [4].