d-Monomial Tests of Nonlinear Cellular Automata for Cryptographic Design

Sandip Karmakar, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury

Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur,
India

Abstract. Pseudorandom generation is a key to any cryptographic application. Linear Cellular Automata are known as good pseudorandom generators. However, for cryptographic applications nonlinearity is essential for its security. But, nonlinear Cellular Automaton shows high correlation between the input to the automaton and its generated sequence. Hence, for cryptography Cellular Automata rules need to be nonlinear as well as satisfy additional properties. With this motivation, in this paper, we analyze nonlinear Cellular Automata with a newly developed statistical measure called d-monomial test. Finally, we propose a process of *d-monomial characteristics addition* to get cryptographically suitable Cellular Automata.

1 Introduction

Cellular Automata are a self-evolving system of cells which updates itself per cycle following a rule embedded into it. Linear Cellular Automaton (CA) is known for its ability to generate pseudorandom sequences needed for various applications like VLSI testing and coding theory [11]. Several researchers have attempted to apply the pseudorandomness of CA to cryptography. The cryptanalysis of linear CA based cryptographic techniques [4] show that nonlinearity is needed for cryptographic applications. However, nonlinear CA shows high correlation. The 3-neighbourhood nonlinear rule 30 CA has long been considered a good pseudo-random generator and studied for cryptography [10]. It passed various statistical tests for pseudorandomness with good results, until Willi Meier and Othmar Staffelbach proposed an attack, exploiting its high correlation, on pseudorandom sequences generated by rule 30 CA [6], which would break any such system of 300 cells with a complexity of about 2^{19}. Another attack on rule 30 CA is also reported in [5]. These findings show that for cryptography, the data stream generated by CA needs to satisfy additional properties.

In this paper, we analyze the CA by modeling its rule as a Boolean function relating output bits with input bits. Parameters like nonlinearity, balancedness, resiliency and algebraic degree are known to be important for the cryptographic analysis of Boolean functions [7], [8]. Recently, d-monomial tests [3] on cryptographic Boolean functions have gained attention. An extension of d-monomial test proposed in [2] serves as another important tool in analyzing cryptographic

Boolean functions. In [7] and [8] some of the stated cryptographic properties of 3 and 4 neighbourhood CA rules are analyzed for a single iteration of the CA. However, multiple iterations of the CA are not considered.

In this work, we explore CA over multiple iterations. At each iteration of the CA, the relationship between the input and output of the CA is represented by a Boolean function. Subsequently, we perform d-monomial tests on such Boolean functions and investigate how the test performs with iterations. It can be mentioned that, uniform and hybrid CA have not been investigated in perspective of cryptographic suitability in this direction before. Following the experimental results, we derive few general conclusions about choice of rules in uniform or hybrid CA to expect certain cryptographic advantages. We expect the findings will also help in analysis of non-linear CA, in general.

This paper is organized as follows. Following the introduction, section 2 presents basic definitions and notations regarding Cellular Automata and the realted cryptographic terms. The list of hybrid CA rules and the reason to choose such rules are explained in section 3. In section 4, we briefly describe the model of our analysis. d-monomial test is introduced in section 5 and the main results of d-monomial tests are also presented in that section. We also draw certain observations from the experimental results in respect of constructing cryptographically suitable hybrid CA with respect to d-monomial tests in that section. Finally, the paper is concluded in section 6.

2 Prelimineries

In this section, we present the basic definitions of CA and also of the cryptographic properties.

2.1 Cellular Automata Related Definitions

Definition 1. *Cellular Automata: A cellular automaton is a finite array of cells. Each cell is a finite state machine $C = (Q, f)$ where Q is a finite set of states and f is a mapping $f : Q^n \rightarrow Q$. The mapping f, is called local transition function. n is the number of cells the local transition function depends on. On each iteration of the CA each cell updates itself with respective f.*

Adjacent cells of a cell are called the neighbourhood of CA. A 1-dimensional CA, whose rule depends on left and right neighbour and the cell itself is called a 3-*neighbourhood CA*. Similarly, if each cell depends on 2 left and 2 right neighbours and itself only, it is called 5-*neighbourhood CA*. A CA whose cells depend on 1 left and 2 right neighbouring cells is called a 4-*neighbourhood right skew CA*. A *left skewed* 4-*neighbourhood CA* can be defined similarly.

Definition 2. *Rule: The local transition function for a* 3*-neighbourhood CA cell can be expressed as follows:*

$$q_i(t+1) = f[q_i(t), q_{i+1}(t), q_{i-1}(t)]$$

where, f denotes the local transition function realized with a combinational logic, and is known as a rule of CA [9]. Here, $q_i(t)$ represents the value of the i^{th} cell after t iterations. The decimal value of the truth table of the local transition function is defined as the rule number *of the cellular automaton.*

For one dimensional 3-neighbourhood CA the definitions of some rules are given below:
Rule 30: $f = q_{i-1}(t) \oplus (q_{i+1}(t) + q_i(t))$, where $+$ is the Boolean 'OR' operator and $\oplus$ is the Boolean 'XOR' operator.
Rule 60: $f = q_{i-1}(t) \oplus q_i(t)$.
Rule 90: $f = q_{i-1}(t) \oplus q_{i+1}(t)$.

A CA whose local transition function is same accross the cells is called *uniform CA*. A CA whose local transition function is *not* same for all the cells is a *hybrid CA*. Hybrid CA may be constructed by choosing different linear rules or by choosing different linear and nonlinear rules over the automaton. A CA whose first and last cells are connected to 0 is called *null-boundary* CA.

A CA whose local transition function consists of only 'XOR' operator is called a *linear CA*. A CA whose at least one local transition function consists of 'AND'/'OR' in addition to 'XOR' is *nonlinear CA*. For example, rule, $f = q_{i-1}(t) \oplus q_{i+1}(t)$ employed in each cell is a linear CA and $f = q_{i-1}(t).q_{i+1}(t)$ employed in each cell is a nonlinear CA, where, $q_{i-1}(t)$ and $q_{i+1}(t)$ denotes left and right neighbours of the i^{th} cell at t^{th} instance of time. A uniform CA each of whose transition function is, $f = q_{i-1}(t) \oplus (q_{i+1}(t) + q_i(t))$ is a *rule* 30 *uniform CA*.

Any CA can be utilized to generate sequences by first selecting a *seed* and then updating each cell according to its transition function. State values from the middle cell of the cell array are output to represent generation of sequences. This sequence can be tested for pseudorandomness.

2.2 Cryptographic Terms and Primitives

We now present definitions of related cryptographic terms and properties used in this paper.

Definition 3. *Pseudorandom Sequence: An algorithmic sequence is pseudorandom if it cannot be distinguished from a truly random sequence by any efficient (polynomial time) probabilistic procedure or circuit.*

A variable or its negation (x or $\bar{x}$) is called a literal. Any number of 'AND'-ed literals is called a *conjunction*. For example, $x.y.\bar{z}$ is a *conjunction.*

Definition 4. *Algebraic Normal Form: Any* Boolean function *can be expressed as* XOR *of* conjunctions *and a* Boolean constant, True *or* False*. This form of the* Boolean function *is called its* Algebraic Normal Form (ANF)*.*

Every *Boolean function* can be expressed in *ANF*. As an example, $f(x_1, x_2, x_3) = (x_1 \oplus x_2).(x_2 \oplus x_3)$ is not in *ANF*. Its *ANF* representation is, $f(x_1, x_2, x_3) = x_1.x_2 \oplus x_1.x_3 \oplus x_2 \oplus x_2.x_3$.

Definition 5. *Algebraic Degree: The maximum number of literals in any* conjunction *of* ANF *of a* Boolean function *is called its* degree*. Ciphers expressible or conceivable as a* Boolean function *have* algebraic degree *which is the same as the* degree *of the* ANF *of the* Boolean function*.*

Thus, $f(x_1, x_2) = x_1 \oplus x_2 \oplus x_1.x_2$ has algebraic degree 2.

Table 1. ANF of used 3 and 4-neighbourhood Rules

Type	Rule #	Nbd	ANF
Linear	15	3	$\bar{x_1}$
	60	3	$x_1 \oplus x_2$
	90	3	$x_1 \oplus x_3$
	150	3	$x_1 \oplus x_2 \oplus x_3$
	240	3	x_1
Non-linear	30	3	$(x_2.x_3) \oplus x_1 \oplus x_2 \oplus x_3$
	37	3	$x_1 \oplus x_2 \oplus (x_1.x_2.x_3) \oplus 1$
	45	3	$x_1 \oplus x_3 \oplus (x_2.x_3) \oplus 1$
	75	4	$x_1 \oplus x_2 \oplus x_2 \oplus (x_1.x_2) \oplus (x_1.x_3) \oplus (x_3.x_4) \oplus (x_1.x_3.x_4)$
	86	4	$x_2 \oplus x_3 \oplus x_4 \oplus (x_1.x_2) \oplus (x_1.x_3) \oplus (x_1.x_4) \oplus (x_2.x_3) \oplus (x_1.x_2.x_3)$
	91	3	$x_2 \oplus (x_1.x_2) \oplus (x_1.x_3) \oplus (x_2.x_3) \oplus (x_1.x_2.x_3) \oplus 1$
	120	3	$x_1 \oplus (x_2.x_3)$
	180	3	$x_1 \oplus x_2 \oplus (x_2.x_3)$
	210	3	$x_1 \oplus x_3 \oplus (x_2.x_3)$

3 Choice of CA Rules for Cryptographic Applications

Existing literature shows that only linear rules 90 and 150 and nonlinear rule 30 have been explored for cryptographic applications. Till date, uniform nonlinear and hybrid nonlinear rules have not been studied much.

We have considered 3 and 4-neighbourhood uniform nonlinear CA configurations. Combination of rule 30 and some other linear and nonlinear 3-neighbourhood rules have also been explored. The hybrid CA configurations are constructed by using the rules given in table 1.

The objective of the proposed construction is that the linear rules help to reduce the correlation, while the nonlinear rule provides required nonlinearity. The effect becomes prominent after a few initial iterations required to *mix* both these types of rules. The reason rule 30 is taken for the hybrid rulesets is that, it is nonlinear and it has good pseudorandom characteristics [10]. Though, rule 30 has good pseudorandom characteristics and it is a balanced rule, it has a strong correlation, namely, the probability, $Pr[x_i(t+1) = 1 \oplus x_{i-1}(t)] = \frac{3}{4}$, where $x_i(t)$ is the state of the i^{th} cell of the CA at t^{th} instance of time. The above property led to its cryptanalysis described by Willi Meier and Othmar Staffelbach [6].

We provide an analytical argument in favour of the fact that the introduction of linear rules reduce the correlation.

Consider two CA configurations, (i) Uniform rule 30 CA, C_1; (ii) Hybrid CA having alternate rules 30 and 60, C_2. The input to both the CA are denoted by an array, $x_i, 0 \leq i \leq n$, where n is the length of the CA. Now,

1. In case of C_1, $Pr[x_i(t+1) = 1 \oplus x_{i-1}(t)] = \frac{3}{4}$, for all time instances t. Hence, $Pr[x_i(t+2) = x_{i-2}(t)] = \frac{9}{16}$.
2. In case of C_2, due to introduction of rule 60, i.e., $x_i(t+1) = x_{i-1}(t) \oplus x_i(t)$, we have, $Pr[x_i(t+1) = x_{i-1}(t)] = \frac{1}{2}$. Hence, there is no bias in predicting the output of the corresponding cell. This effect of unpredictibility propagates to the nonlinear rule 30 in subsequent iterations. Therefore, $Pr[x_i(t+2) = x_{i-2}(t)] = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8}$. Thus, the correlation gets reduced with each iteration. This justifies the construction of hybrid CA by alternating nonlinear and linear rules.

Evidently introducing more linear rules with nonlinear rules reduce correlations faster. For this reason we have considered a series of rule-30 based hybrid CA. In other words, the combination of rules is made in expectation of obtaining pseudorandom characteristics of rule 30 without the weakness of correlation.

For the experiment we have taken the following hybrid CA rulesets:

1. *Ruleset 1*: Rules 30 and 60 spaced alternately over a 3-neighbourhood CA.
2. *Ruleset 2*: Rules 30, 60 and 90 spaced alternately over a 3-neighbourhood CA.
3. *Ruleset 3*: Rules 30, 60, 90 and 120 spaced alternatively over a 3-neighbourhood CA.
4. *Ruleset 4*: Rules 30, 60, 90, 120 and 150 spaced alternatively over a 3-neighbourhood CA.
5. *Ruleset 5*: Rules 30, 60, 90, 120, 150, 180, 210, 240 spaced alternatively over a 3-neighbourhood CA.
6. *Ruleset 6*: Rules 30, 60, 90, 120, 150, 180, 210, 240, 15, 45 spaced alternatively over a 3-neighbourhood CA.

4 Functional Model of CA for Testing Crypto-Properties

For the experiment, we have taken an $n+1$-cell *null-boundary* CA(*figure* 1). Here, without loss of generality, n is assumed to be odd. Each cell of the CA is assumed to have an unknown value, $x_i, 0 \leq i \leq n$ at the beginning. Boolean rules are set into the CA cells according to the CA configuration needed. Thus, each cell's output is determined by a corresponding local transition function f_i. Collectively, the functions are represented as F. The output bits of the CA are denoted by, $y_0, y_1, \ldots y_n$. The middle cell's output $(y_{\frac{n+1}{2}})$ is analyzed. Here, $f_{\frac{n+1}{2}}$ is the local transition function of the $\frac{n+1}{2}^{th}$ cell and $f^t_{\frac{n+1}{2}}$ is defined recursively as follows:

$$f^{t+1}_{\frac{n+1}{2}} = f_{\frac{n+1}{2}}(f^t_{\frac{n+1}{2}})$$

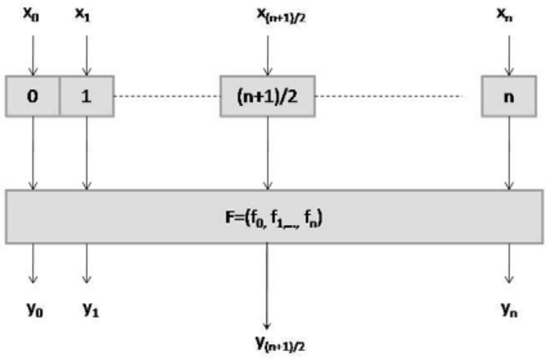

Fig. 1. Configurations of CA experimented

For that, we express the $\frac{n+1}{2}^{th}$ cell's output as a function of initial input unknowns, $x_i, 0 \leq i \leq n$. For a 3-neighbourhood CA,

$$y_{\frac{n+1}{2}} = f_{\frac{n+1}{2}}(x_{\frac{n+1}{2}-1}, x_{\frac{n+1}{2}}, x_{\frac{n+1}{2}+1})$$

and for a 4-neighbourhood right-skewed CA,

$$y_{\frac{n+1}{2}} = f_{\frac{n+1}{2}}(x_{\frac{n+1}{2}-1}, x_{\frac{n+1}{2}}, x_{\frac{n+1}{2}+1}, x_{\frac{n+1}{2}+2})$$

This process is iterated for multiple clock cycles, for 3 and 4-neighbourhood CA,

$$y^t_{\frac{n+1}{2}} = f^t_{\frac{n+1}{2}}(x_{\frac{n+1}{2}-t}, \ldots, x_{\frac{n+1}{2}}, \ldots, x_{\frac{n+1}{2}+t})$$

$$y^t_{\frac{n+1}{2}} = f^t_{\frac{n+1}{2}}(x_{\frac{n+1}{2}-t}, \ldots, x_{\frac{n+1}{2}}, \ldots, x_{\frac{n+1}{2}+2t})$$

Thus, it is clear that at t^{th} iteration, for a 3-neighbourhood CA, the output bit is a function of $2t+1$ bits and for a 4-neighbourhood CA it is a function of $3t+1$ bits. Beyond 3^{rd} iteration, the Boolean function acts upon 10 more variables and hence becomes unwieldy to analyze. In this paper, we have listed results of first 3 iterations only. We have chosen the $\frac{n+1}{2}^{th}$ cell for our analysis because it will be least effected by the boundary null values and more affected by the neighbouring cells and thus better charaterize the rule of the CA. However, in case of hybrid configurations, we have analyzed output of all nonuniform middle cells and have selected the best rule as the output.

Historically, researchers have studied balancedness, nonlinearity, resiliency and algebraic degree [7], [8] to explore CA as a crypto-primitive. Our emphasis is on a new cryptographic test called d-monomial test.

5 d-Monomial Test

d-Monomial test is a statistical test for pseudorandomness introduced independently in [1] and [3]. It investigates the Boolean function representation of each output bit in terms of input bits. If a Boolean function of n Boolean variables is a good pseudorandom sequence generator, then it will have $\frac{1}{2}\binom{n}{d}$ d-degree monomials. The distribution is *binomial.* A χ^2 test with one degree of freedom is applied to count to measure how *unbiased* the count is. A deviation will indicate non-randomness. For example, consider the function $f(x_1, x_2, x_3) = x_1 \oplus x_2$, it has 2, 1-degree monomials and 0, 2 degree monomial. The ideal number of 1, 2 and 3 degree monomials would be $\frac{1}{2}\binom{3}{1} = 1.5$, $\frac{1}{2}\binom{3}{2} = 1.5$ and $\frac{1}{2}\binom{3}{3} = 0.5$. It turns out that it has 2, 1-degree monomials more and 1 2-degree monomial less, hence it is expected to be non-pseudorandom. On the other hand, $f(x_1, x_2, x_3) = x_1 \oplus x_2.x_3$ is expected to be a good pseudorandom generator.

In spite of its simplicity, this test gained huge appreciation in cryptography community. It proved to be a good tool in analyzing the degree of pseudorandomness of cryptographic systems. To the best of our knowledge, d-monomial test has not been applied to CA configurations previously. We explore different CA configurations under this test.

The d-monomial test can be considered a stronger form of pseudorandomness test than is captured by the cryptographic properties, balancedness, nonlinearity, resiliency and algebraic degree. Not much work has been done on constructing Boolean functions which satisfy d-monomial test. On the other hand, lot of work exists for making good balanced, nonlinear, resilient and high algebraic degree Boolean functions.

Note that, since d-monomial test does not output a single value, it is difficult to compare d-monomial characteristics of two Boolean functions. We have given preference to Boolean functions having ideal values in higher degree over ideal values in lower degree. This is justified as cryptanalysis of Boolean functions having higher degree terms is harder than Boolean functions having lower degree terms.

Example of Calculation of Ideal d-Monomial Value: Let, number of variables in the Boolean function be, $n = 5$ and let $d = 4$; then, ideal number of 4-degree terms will be, $\frac{1}{2}\binom{n}{d} = \frac{1}{2}\binom{5}{4} = 2.5$. We will approximate it to 2.

5.1 d-Monomial Test of Uniform CA

The experiment is done using *Mathematica 7.0 Student Edition.* Each i^{th} cell of the CA is assumed to be initialized with an unknown Boolean value, x_i. The Boolean rules of the CA are simulated and value of each cell is updated per iteration. For example, if the CA operates itself uniformly with rule 30, the i^{th} cell will have value $x_{i-1} \oplus (x_{i+1} + x_i)$ after the first iteration of operation, where, $\oplus$ stands for Boolean 'XOR' operator and $+$ stands for 'OR' operator.

3-neighbourhood CA: Among uniform rules, rule 37 and rule 91 have the best d-monomial characteristics. Table 2 lists values of number of n^{th} degree

Table 2. Comparison of d-monomial characteristics of rule 30, 37 and 91

	Number of n^{th} degree terms						
Rules	*1*	*2*	*3*	*4*	5	6	*7*
Ideal	*1,2,2*	*1,5,10*	*0,5,52*	*0,2,52*	*0,0,10*	*0,0,3*	*0,0,0*
30	3,3,6	3,5,11	1,2,8	0,0,4	0,0,1	0,0,0	0,0,0
37	2,3,5	0,8,13	1,4,21	0,0,26	0,1,8	0,0,1	0,0,1
91	1,2,3	3,7,8	1,6,14	0,5,12	0,1,11	0,0,6	0,0,1

Table 3. Comparison of d-monomial Characteristics of 4-neighbourhood CA

	Number of n^{th} degree terms								
Rules	*1*	*2*	*3*	*4*	5	6	*7*	*8*	*9*
Ideal	*2,3,5*	*3,10,22*	*2,17,30*	*0,17,210*	*0,10,126*	*0,3,210*	*0,0,30*	*0,0,22*	*0,0,5*
75	0,0,5	0,0,13	2,6,21	1,3,26	0,2,10	0,2,8	0,0,8	0,0,2	0,0,6
86	0,0,0	0,0,0	2,0,0	1,5,0	0,6,3	0,3,14	0,4,15	0,0,8	0,0,2

terms in the generated Boolean functions over the three iterations for rules 30, 37 and 91. According to the table, rule 30 performs worse than rule 37 and 91. It fails in generating some higher degree terms (degree 6 and 7 for example). It also generates less number of 3, 4 and 5-degree terms than rule 37 and 91. It generates close to ideal number of 2-degree terms compared to rule 37 and 91. But as already mentioned, ideal number of lower degree terms is not as important as ideal number of higher degree terms in view of cryptanalysis. Rule 91 performs better than rule 37 in higher degree terms and should be given preference. The growth rate of number of n^{th} degree terms is quite fast for both rules 37 and 91. Note that, even this two CA are far distant from the ideal d-monomial characteristics.

4-neighbourhood CA: We look at their d-monomial test results of rule 75 and rule 86 in table 3. This two rules outperform the other rules exmerimented. Both rules, 75 and 86 have many higher degree terms and few or no lower degree terms. However, the number of higher degree terms is not ideal too. Between rules 75 and 86 we should choose rule 75 as it generates closer to ideal number of terms of each degree than rule 86. Again note that, even the best rules, rule 75 and rule 86 are far distant from the ideal d-monomial characteristics.

5.2 *d*-Monomial Test of Hybrid CA

d-monomial test results of the 6 hybrid CA rulesets are given in the table 4. The table above shows that, rulesets 5 and 6 are better rulesets than all other hybrid configurations, as they generate closer to ideal number of terms of almost all n^{th} degree terms than the other rulesets. Rules 37 and 91 have better d-monomial characteristics but the generated functions have low resiliency and are unbalanced compared to the hybrid counterparts.

Table 4. *d*-Monomial Characteristics of Hybrid CA

	Number of n^{th} degree terms			
Rules	*1*	*2*	*3*	*4*
Ideal	*1,2,3*	*1,5,10*	*0,5,52*	*0,2,52*
Ruleset 1	3,3,5	1,3,3	0,0,2	0,0,0
Ruleset 2	3,3,2	1,3,3	0,0,1	0,0,0
Ruleset 3	3,2,4	1,3,5	0,1,3	0,0,0
Ruleset 4	3,2,4	1,3,7	0,1,7	0,0,2
Ruleset 5	3,2,4	1,3,5	0,2,6	0,0,3
Ruleset 6	3,2,4	1,3,5	0,2,6	0,0,3

d-monomial characteristic is an important metric in finding which rules should be combined in a hybrid CA. As an example, let us form a CA consisting of rules 30 and 37 spaced alternatively. The *d*-monomial characteristics of the CA is given in table 5 along with characteristics of rule 30 and 37. We have seen that, rule 30 and rule 37 have good *d*-monomial characteristics. But, note that, the new CA performs even better than both the rules in higher degree terms (degree 5 onwards). At higher degree terms its *d*-monomial values are very close to ideal. In the middle and lower degree terms also the *d*-monomial values are good, though rule 37 has better values in this region.

Both rule 30 CA and rule 37 CA are heavy on higher degree terms. Their combination is expected to have more number of higher degree terms. Likewise, linear rules can not add terms of more than degree 1 to the generated Boolean function. But combining linear rules with higher algebraic degree rules, we will be able to add missing degree 1, 2 and other lower degree terms in the generated Boolean functions, which will perform better in *d*-monomial test. The above observation is important in hybrid CA constructions. This may be a way of reaching ideal *d*-monomial characteristics. We refer to this process as *d-monomial characteristics addition*. However, no direct relationship in *d*-monomial values of component rules and the hybrid rule can be inferred from the result (table 5). But, understanding the behaviour of this process is crucial in design of cryptographically suitable CA or good Boolean function generator.

Table 5. *d*-Monomial Characteristics Addition of Hybrid CA

	Number of n^{th} degree terms						
Rules	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Ideal	*1,2,2*	*1,5,10*	*0,5,52*	*0,2,52*	*0,0,10*	*0,0,3*	*0,0,0*
30	3,3,6	3,5,11	1,2,8	0,0,4	0,0,1	0,0,0	0,0,0
37	2,3,5	0,8,13	1,4,21	0,0,26	0,1,8	0,0,1	0,0,1
(30, 37)	2,4,2	0,6,9	3,7,13	0,2,19	0,0,10	0,0,4	0,0,1

6 Conclusion

We have presented experimental results of d-monomial test of different configurations of uniform and hybrid CA. We have seen that it is possible to construct hybrid CA that can provide fair d-monomial characteristics and at the same time can be resilient, balanced and nonlinear. It is possible to improve d-monomial characteristics of CA rules by combining rules into a hybrid CA. We referred to this process as *d-monomial charateristics addition*. This property can be employed to form cryptographically suitable CA.

References

1. Filiol, E.: A new statistical testing for symmetric ciphers and hash functions. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 342–353. Springer, Heidelberg (2002)
2. Johansson, T., Englund, H., Turan, M.S.: A framework for chosen iv statistical analysis of stream ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
3. Juhani, M., Saarinen, O.: Chosen-iv statistical attacks on e-stream stream ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, pp. 5–19 (2006)
4. Murphy, S., Paterson, K.G., Blackburn, S.R.: Theory and applications of cellular automata in cryptography. IEEE Transactions on Computers, 46(5) (1997)
5. Koc, C.K., Apohan, A.M.: Inversion of cellular automata iterations. IEE Proceedings of Computers and Digital Techniques 144(5), 279–284 (1997)
6. Meier, W., Staffelbach, O.: Analysis of pseudo random sequences generated by cellular automata. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 186–199. Springer, Heidelberg (1991)
7. Martin, B., Sole, P.: Pseudo-random sequences generated by cellular automata. In: International Conference on Relations, Orders and Graphs: Interactions with Computer Science (2008)
8. Martin, B., Sole, P., Lacharme, P.: Pseudo-random sequences, boolean functions and cellular automata. Boolean Functions: Cryptography and Applications (2007)
9. Roy Chowdhury, D., Nandi, S., Chattopadhyay, S., Pal Chaudhuri, P.: Ca and its applications: A brief survey. Additive Cellular Automata - Theory and Applications (1997)
10. Wolfram, S.: Cryptography with cellular automata. CRYPTO: Proceedings of Crypto (1985)
11. Wolfram, S.: Random sequence generation by cellular automata. Advances in Applied Mathematics 7, 123–169 (1986)