Cellular Programming and Symmetric Key Cryptography Systems

Franciszek Seredyński[1,2], Pascal Bouvry[3], and Albert Y. Zomaya[4]

[1] Polish-Japanese Institute of Information Technologies
Koszykowa 86, 02-008 Warsaw, Poland
[2] Institute of Computer Science of Polish Academy of Sciences
Ordona 21, 01-237 Warsaw, Poland
sered@ipipan.waw.pl
http://www.ipipan.waw.pl/~sered
[3] Luxembourg University of Applied Sciences
6, rue Coudenhove Kalergi, L-1359 Luxembourg-Kirchberg, Luxembourg
pascal.bouvry@ist.lu
http://www.ist.lu/users/pascal.bouvry
[4] School of Information Technologies, University of Sydney
Sydney, NSW 2006 Australia
zomaya@it.usyd.edu.au
http://www.cs.usyd.edu.au/~zomaya

Abstract. The problem of designing symmetric key cryptography algorithms based upon cellular automata (CAs) is considered. The reliability of the Vernam cipher used in the process of encryption highly depends on a quality of used random numbers. One dimensional, nonuniform CAs is considered as a generator of pseudorandom number sequences (PNSs). The quality of PNSs highly depends on a set of applied CA rules. To find such rules nonuniform CAs with two types of rules is considered. The search of rules is based on an evolutionary technique called cellular programming (CP). Resulting from the collective behavior of the discovered set of CA rules very high quality PNSs are generated. The quality of PNSs outperform the quality of known one dimensional CA-based PNS generators used in secret key cryptography. The extended set of CA rules which was found makes the cryptography system much more resistant on breaking a cryptography key.

1 Introduction

Confidentiality is mandatory for a majority of network applications including, for example, commercial uses of the Internet. Two classes of algorithms exist on the market for data encryption: secret key systems and public key systems. An extensive overview of currently known or emerging cryptography techniques used in both types of systems can be found in [14]. One of such a promising cryptography techniques are cellular automata (CAs). An overview on current state of research on CAs and their application can be found in [13].

CAs were proposed for public-key cryptosystems by Guan [1] and Kari [5]. In such systems two keys are required: one key is used for encryption and the other for decryption, and one of them is held in private, the other is published. However, the main concern of this paper are secret key cryptosystems. In such systems the same key is used for encryption and decryption. The encryption process is based on the generation of pseudorandom bit sequences, and CAs can be effectively used for this purpose. In the context of symmetric key systems, CAs were first studied by Wolfram [17], and later by Habutsu *et al.* [3], Nandi *et al.* [11] and Gutowitz [2]. Recently they were a subject of study by Tomassini and his colleagues (see, eg. [16]). This paper extends these recent studies and describes the application of one dimensional (1D) CAs for the secret key cryptography.

The paper is organized as follows. The following section presents the idea of an encryption process based on Vernam cipher and used in CA-based secret key cryptosystems. Section 3 outlines the main concepts of CAs, overviews current state of applications of CAs in secret key cryptography and states the problem considered in the paper. Section 4 outlines evolutionary technique called cellular programming and Sect. 5 shows how this technique is used to discover new CA rules suitable for encryption process. Section 6 contains the analysis of results and the last section concludes the paper.

2 Vernam Cipher and Secret Key Cryptography

Let P be a plain-text message consisting of m bits $p_1p_2...p_m$, and $k_1k_2...k_m$ be a bit stream of a key k. Let c_i be the $i-th$ bit of a cipher-text obtained by applying a XOR (exclusive-or) enciphering operation:

$$c_i = p_i \; XOR \; k_i.$$

The original bit p_i of a message can be recovered by applying the same operation XOR on c_i with use of the same bit stream key k:

$$p_i = c_i \; XOR \; k_i.$$

The enciphering algorithm called Vernam cipher is known to be [9,14] perfectly safe if the key stream is truly unpredictable and is used only one time. From practical point of view it means that one must find answers on the following questions: (a) how to provide a pure randomness of a key bit stream and unpredictability of random bits, (b) how to obtain such a key with a length enough to encrypt practical amounts of data, and (c) how to pass safely the key from the sender to receiver and protect the key.

In this paper we address questions (a) and (b). We will apply CAs to generate high quality pseudorandom number sequences (PNSs) and a safe secret key.

3 Cellular Automata and Cryptography

One dimensional CA is in a simpliest case a collection of two-state elementary automata arranged in a lattice of the length N, and locally interacted in a discrete time t. For each cell i called a central cell, a neighborhood of a radius r is defined, consisting of $n_i = 2r+1$ cells, including the cell i. When considering a finite size of CAs a cyclic boundary condition is applied, resulting in a circle grid.

It is assumed that a state q_i^{t+1} of a cell i at the time $t+1$ depends only on states of its neighborhood at the time t, i.e. $q_i^{t+1} = f(q_i^t, q_{i1}^t, q_{i2}^t, ..., q_{ni}^t)$, and a transition function f, called a *rule*, which defines a rule of updating a cell i. A length L of a rule and a number of neighborhood states for a binary uniform CAs is $L = 2^n$, where $n = n_i$ is a number of cells of a given neighborhood, and a number of such rules can be expressed as 2^L. For CAs with e.g. $r = 2$ the length of a rule is equal to $L = 32$, and a number of such rules is 2^{32} and grows very fast with L. When the same rule is applied to update cells of CAs, such CAs are called uniform CAs, in contrast with nonuniform CAs when different rules are assigned to cells and used to update them.

S. Wolfram was the first to apply CAs to generate PNSs [17] . He used uniform, 1D CAs with $r = 1$, and rule 30. Hortensius *et al.* [4] and Nandi *et al.* [11] used nonuniform CAs with two rules 90 and 150, and it was found that the quality of generated PNSs was better than the quality of the Wolfram system. Recently Tomassini and Perrenoud [16] proposed to use nonuniform, 1D CAs with $r = 1$ and four rules 90, 105, 150 and 165, which provide high quality PNSs and a huge space of possible secret keys which is difficult for cryptanalysis. Instead to design rules for CAs they used evolutionary technique called cellular programming (CP) to search for them.

In this study we continue this line of research. We will use finite, 1D, nonuniform CAs. However, we extend the potential space of rules by consideration of two sizes of rule neighborhood, namely neighborhood of radius $r = 1$ and $r = 2$. To discover appropriate rules in this huge space of rules we will use CP.

4 Cellular Programming Environment

4.1 Cellular Programming

CP is an evolutionary computation technique similar to the diffusion model of parallel genetic algorithms and introduced [15] to discover rules for nonuniform CAs. Figure 1 shows a CP system implemented [10] to discover such rules. In contrast with the CP used in [16] the system has a possibility to evaluate nonuniform rules of two types. The system consists of a population of N rules (left) and each rule is assigned to a single cell of CAs (right). After initiating states of each cell, i.e. setting an initial configuration, the CAs start to evolve according to assigned rules during a predefined number of time steps. Each cell produces a stream of bits, creating this way a PNS.

After stopping evolving CAs all PNSs are evaluated. The entropy E_h is used to evaluate the statistical quality of each PNS. To calculate a value of the entropy each PNS is divided into subsequences of a size h. In all experiments the value $h = 4$ was used. Let k be the number of values which can take each element of a sequence (in our case of binary values of all elements $k = 2$) and k^h a number

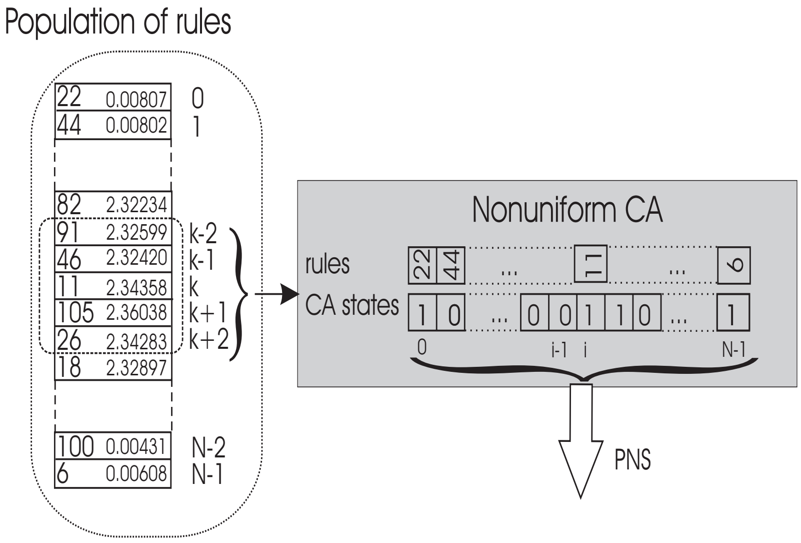


Fig. 1. CP environment for evolution of rules of nonuniform CAs.

of possible states of of each sequence ($k^h = 16$). E_h can be calculated in the following way:

$$E_h = -\sum_{j=1}^{k^h} p_{h_j} log_2\, p_{h_j},$$

where p_{h_j} is a measured probability of occurrence of a sequence h_j in a PNS. The entropy achieves its maximal value $E_h = h$ when the probabilities of the k_h possible sequences of the length h are equal to $1/k^h$. The entropy will be used as a fitness function of CP.

A single PNS is produced by a CA cell according to assigned rules and depends on a configuration c_i of states of CAs. To evaluate statistically reliable value of the entropy, CAs run with the same set of rules C times for different configurations c_i, and finally the average value of entropy is calculated and serves as a fitness function of each rule from the population of rules.

After evaluation of a fitness function of all rules of the population genetic operators of selection, crossover and mutation are locally performed on rules. The evolutionary algorithm stops after some predefined number of generations of CP. The algorithm can be summarized in the following way:

1: initiate randomly *population* of N rules of type 1 ($r = 1$) or type 2 ($r = 2$), or both types, and create CAs consisting of N cells

2: assign $k - th$ rule from the CP population to $k - th$ cell of CAs

```
3: for i=1 .. C do
      {  create randomly configuration $c_i$ of CAs
         evolve CAs during $M$ time steps
         evaluate entropy of each $PNS$ }

4: Evaluate fitness function of each rule

5: Apply locally to rules in a specified sequence genetic operators of
     selection, crossover and mutation

6: if STOP condition is not satisfied return to 2.
```

4.2 Genetic Operators

In contrast with the standard genetic algorithm population, rules – individuals of CP population occupy specific place in the population and have strictly defined neighborhood. For example, the rule 11 (see Fig. 1) (also indexed by k) corresponds to $k - th$ cell of CAs, and rules 46 and 105 are its immediate neighbors. All rules shown in this figure belong to the first type of rules with $r = 1$, i.e. a transition function of the rule depends on 3 cells, a given cell and two cell-neighbors. However, in more general case considered in the paper, we assume that rules are either of type 1 ($r = 1$, short rules) or of type 2 ($r = 2$, long rules).

Additionally to a neighborhood associated with two types of rules we use also an evolutionary neighborhood, i.e. the neighborhood of rules which are considered for mating when genetic operators are locally applied to a given rule. The size and pattern of this neighborhood may differ from the neighborhood associated with types of rules. Figure 1 shows an example of the evolutionary neighborhood for the rule k which is created by rules $k-2, k-1, k, k+1, k+2$. It is assumed that the pattern of such a neighborhood is the same for all rules and is a predefined parameter of an experiment.

A sequence of genetic operators performed locally on a given rule depends on values of fitness function of rules (numbers on the right side of rule names, see Fig. 1) from the evolutionary neighborhood of this rule. Genetic operators are applied in the following way:

1. if the $k - th$ rule is the best (the highest value of the fitness function) in its evolutionary neighborhood then the rule survives (selection) and remains unchanged for the next generation; no other genetic operators are performed
2. if in the evolutionary neighborhood of the rule k only one rule exists which is better than considered rule then the rule k is replaced by the better rule (selection) only if both rules are of the same type, and next mutation on this rule is performed; the rule remains unchanged if the better rule is of the other type
3. if two rules better than the rule k exist in the neighborhood then a crossover on the pair of better rules is performed; a randomly selected child from a pair of children replaces rule k, and additionally a mutation is performed

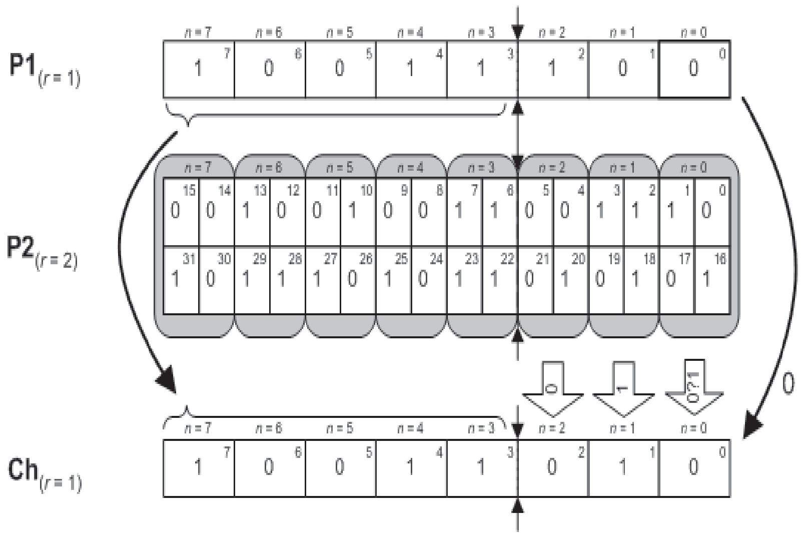


Fig. 2. Example of crossover resulting in a short child rule.

4. if more than two rules better than the rule k exist in the neighborhood then two randomly selected better rules create (crossover) a pair of children; on a randomly selected child a mutation is performed, and the child replaces the rule k.

Two types of rules existing in a CP population can be considered as two species of a coevolutionary algorithm. Therefore to perform a crossover between rules special regulations are required. It is assumed that two parental rules of the same species create a single child rule of the same species, which can replace either the first type of a rule or the second type of the rule. If rules of different types take part in the mating then a species of a child depends on species of a replaced rule, and is the same as a species of a rule to be replaced. Figure 2 shows a crossover between a short rule 156 ($r = 1$) and a long rule 617528021 ($r = 2$), and the result of crossover – a short rule 154.

The short rule $P1$ taking part in crossover consists of 8 genes ($n = 0, ..., 7$) which values correspond to values of transition function defined on 8 neighborhood states $\{000, 001, ..., 111\}$ existing for $r = 1$. The long rule $P2$ consists of 32 genes, each corresponding to values of transition function defined on 32 neighborhood states existing for $r = 2$. The long rule is folded because there is a strict relation between a state order number which corresponds to $j - th$ gene of $P1$ and states' order numbers corresponding to genes $2j, 2j+1$ and $2j+16, 2j+17$ of $P2$. These order numbers of states of $P2$ are just an extension of corresponding order number of a gene from $P1$. For example, the gene $n = 7$ of $P1$ corresponds to the neighborhood state $\{\mathbf{111}\}$, and genes 15, 14 and 31, 30 of $P2$ correspond to states respectively $\{0\mathbf{111}1, 0\mathbf{111}0\}$ and $\{1\mathbf{111}1, 1\mathbf{111}0\}$ containing the state of $P1$ (marked in bold).

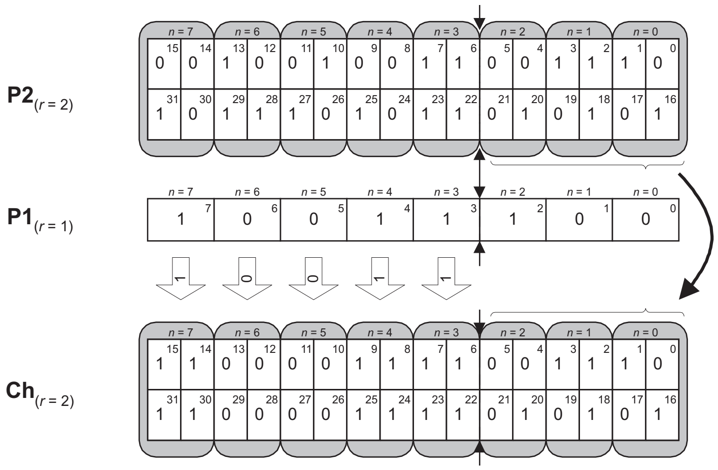


Fig. 3. Example of crossover resulting in a long child rule.

As Fig. 2 shows both rules $P1$ and $P2$ are crossed between genes 2 and 3 and a child Ch corresponding to a short rule ($r = 1$) is created. For this purpose the left part of the short rule is copied to the left part of the child. The right part of Ch is created according to the right part of $P2$ on the basis of majority of 0's or 1's in the corresponding genes. Figure 3 shows the crossover of two rules resulting in a long child rule. Last genetic operator is a flip-bit mutation performed with the probability $p_m - 0.001$.

5 Discovery of Rules in 1D, Nonuniform CAs

In all conducted experiments a population of CP and the size of nonuniform CAs were equal to 50 and the population was processing during 50 generations. The CAs with initial random configuration of states and a set of assigned rules evolved during $M = 4096$ time steps. Running CAs with a given set of rules was repeated for $C = 300$ initial configurations. Figure 4 shows an example of running CP for the evolutionary neighborhood $i-3, i-2, i, i+2, i+3$. One can see that whole CAs is able to produce very good PNSs after about 40 generations (see, the average value avg of the entropy close to 4).

A typical result of a single run of an evolutionary process starting with a random rules assigned to cells of CAs is discovering by CP a small set of good rules which divide the cellular space of CAs into domains – areas where the same rules, short ($r = 1$) or long ($r = 2$), live together. Evolutionary process is continued on borders of domains where different rules live. This process may result in increasing domains of rules which are only slightly better than neighboring rules, which domains will decrease and finally disappear. This happens in particular

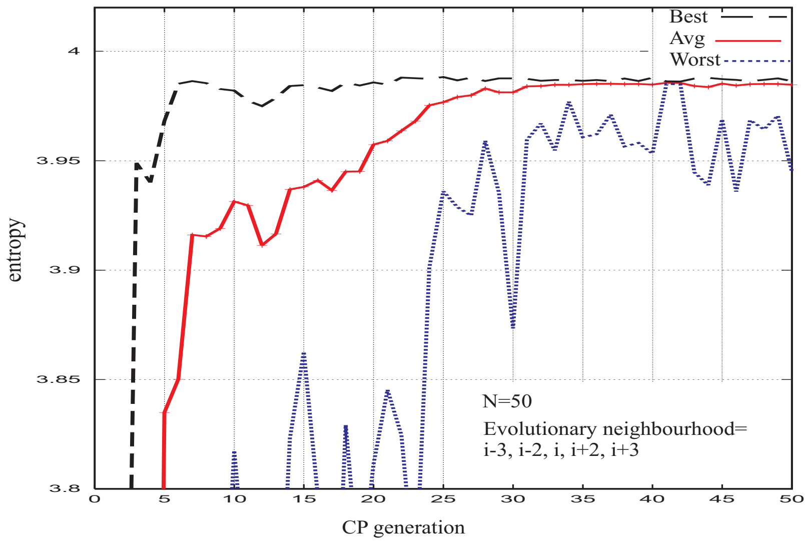


Fig. 4. A single run of CP evolutionary process.

when two neighboring domains are occupied respectively by the same short rules and the same long rules. The search space of short rules is much smaller than the search space of the long rules. Therefore better short rules are discovered faster than better long rules, and for this reason long rules are gradually replaced by short rules. To limit this premature convergence of short rules, the short and long rules are initially randomly assigned to cells in the proportion of 1:3 in all subsequent experiments.

The purpose of the experiments which followed was to discover an enlarged set of rules (to enlarge the key space of cryptography system) which working collectively would produce very high quality PNSs. It was noticed that in a single run of CP the evolutionary algorithm produces typically a small set of rules with a very high value of the entropy. In the result of evolutionary searching process a set of 8 short rules (including 5 rules found by [16]) and a set of 39 long rules was found.

6 Analysis and Comparison of Results

The entropy used as the fitness function for evolution CA rules producing high quality PNSs is only one of existing statistical tests of PNSs. None of them is enough strong to claim statistical randomness of a PNS in the case of passing a given test. For this purpose uniform CAs consisting of 50 cells evolved during 65536 time steps with each single discovered rule. Each PNS produced by CAs was divided into 4-bit words and tested on general statistical tests such as the entropy, χ^2 test, serial correlation test [6] (some weaker rules after this testing were removed), and next on a number of statistical tests required by the FIPS

140-2 standard [12], such as monobit test, poker test, runs test, and long runs test.

Figure 5 shows results of testing rules against the FIPS 140-2 standard. The best scores were achieved by rules 30, 86, 101, 153 and by 8 long rules. Rules 90, 105, 150 and 65 [16] working separately in uniform CAs obtained good results in test of entropy and long runs test, quite good results in serial correlation test and monobit test but were weak in χ^2 test, poker test and runs test. However this set of rules working collectively in nonuniform CAs achieves good results (see Fig. 6).

For this reason only 10 rules were removed from discovered set of rules which have passed the FIPS 140-2 standard testing. These rules were worse than Tomassini & Perrenoud rules. However passing all statistical tests does not exclude a possibility that the PNS is not suitable for cryptographic purposes. Before a PNS is accepted it should pass special cryptographic tests.

Therefore rules which passed tests were next submitted to a set of Marsaglia tests [7] – a set of 23 very strong tests of randomness implemented in the Diehard program. Only 11 rules passed all 23 Marsaglia tests. These are short rules 30, 86, 101, and long rules 869020563, 1047380370, 1436194405, 1436965290, 1705400746, 1815843780, 2084275140 and 2592765285.

The purpose of the last set of experiments was a selection of a small set of short and long rules for nonuniform CAs which working collectively would provide a generation of very high quality PNSs suitable for the secret key cryptography. Simple combination of different rules which passed all Marsaglia tests in nonuniform CAs have shown that resulting PNSs may have worse statistical characteristic than PNSs obtained using uniform CAs. On the other hand, experiments with Tomassini & Perrenoud rules show that rules that separately are working worse can provide better quality working collectively. For these reasons rules 153 and some long rules which obtained very good results in general tests but not passed all Marsaglia tests were also accepted for the set of rules to search a final set of rules.

In the result of combining rules into sets of rules and testing collective behavior of these sets working in nonuniform CAs the following set of rules has been selected: 86, 90, 101, 105, 150, 153, 165 ($r = 1$), and 1436194405 ($r = 2$). Among the rules are 4 rules discovered in [16]. The set of found rules have been tested again on statistical and cryptographic tests using nonuniform CAs with random assignment of rules to CA cells. Figure 6 shows results of testing this new set of rules and compares the results with ones obtained for Tomassini & Perrenoud rules. One can see that results of testing both sets on general tests and FIPS 140-2 tests are similar. However, the main difference between these results can be observed in passing Marsaglia tests: while the new discovered set of rules passes all 23 Marsaglia tests, the Tomassini & Perrenoud set of rules passes only 11 tests. Figure 7 shows a space-time diagram of both set of rules working collectively.

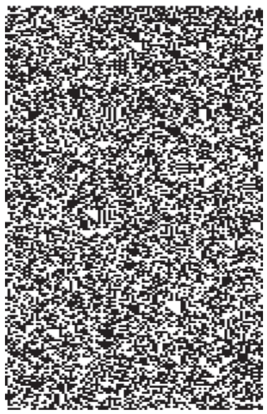
The secret key K which should be exchanged between two users of considered CA-based cryptosystem consists of a pair of randomly created vectors: the vector R_i informing about assigning 8 rules to N cells of CAs and the vector $C(0)$

No	Rules	Monobit Test	Poker Test	Run Test	Long Run Test
Rules found by Tomassini & Perrenoud					
1	30	50	50	50	50
2	90	46	0	23	50
3	105	41	0	4	50
4	150	45	0	12	50
5	165	46	0	14	50
Rules of radius r = 1 (No 6, 7, 8) and r = 2					
6	86	50	50	50	50
7	101	50	50	50	50
8	153	50	50	50	50
11	728094296	50	5	17	50
12	869020563	50	50	49	50
13	892695978	50	2	9	50
14	898995801	50	0	4	50
15	988725525	50	11	16	50
17	1042531548	38	0	12	50
18	1047380370	50	50	47	50
19	1367311530	50	5	5	50
20	1378666419	42	3	16	50
21	1386720346	50	20	32	50
22	1403823445	50	19	32	50
23	1427564201	46	1	27	50
24	1436194405	50	50	50	50
25	1436965290	50	50	50	50
27	1457138022	50	0	0	50
28	1470671190	50	50	49	50
29	1521202561	40	0	3	50
30	1537843542	50	48	37	50
31	1588175194	50	21	27	50
32	1704302169	50	50	50	50
33	1705400746	50	50	50	50
35	1721277285	49	1	4	50
37	1721325161	50	50	50	50
38	1746646725	49	6	2	50
39	1755030679	50	49	34	50
43	1815843780	50	50	50	50
45	2036803240	50	0	0	50
46	2084275140	50	50	50	50
47	2592765285	50	50	49	50

Fig. 5. Testing rules against the FIPS 140-2 standard

Test	Tomassini & Perrenoud Rules (90, 105, 150, 165)	Discovered Rules (86, 90, 101, 105, 150, 153, 165, 1436194405)
Min entropy	3,9988	3,9987
Max entropy	3,9998	3,9997
Min χ^2	5,0254	6,998
Max χ^2	26,396	30,805
Min correlation	0,00007	-0,00006
Max correlation	0,02553	0,01675
Monobit test	50	50
Poker test	50	50
Run test	50	50
Long run test	50	50
Number of passed Marsaglia tests	11	23

Fig. 6. Comparison of rules found by Tomassini & Perrenoud [16] and new set of discovered rules.


a)

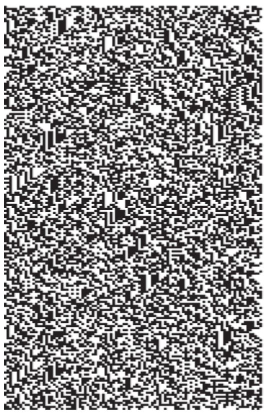

b)

Fig. 7. Space-time diagram of CAs with $N = 100$ and $M = 200$ time steps working collectively with (a) randomly assigned Tomassini & Perrenoud [16] rules, and (b) with new set of discovered rules.

describing an initial binary state of CA cells. The whole key space has therefore the size $8^N * 2^N$. The key space is much larger than the key space $(4^N * 2^N)$ of 1D CA-based system [16]. Therefore the proposed system is much more resistant for cryptographic attacks.

7 Conclusions

In the paper we have reported results of the study on applying CAs to the secret key cryptography. The purpose of the study was to discover a set of CA rules which produce PNSs of a very high statistical quality for a CA-based cryptosystem which is resistant on breaking a cryptography key. The main assumption of our approach was to consider nonuniform 1D CAs operating with two types of rules. Evolutionary approach called CP was used to discover suitable rules. After discovery of a set of rules they were carefully selected using a number of strong statistical and cryptographic tests. Finally, the set consisting of 8 rules has been selected. Results of experiments have shown that discovered rules working collectively are able to produce PNSs of a very high quality outperforming the quality of known 1D CA-based secret key cryptosystems, which also are much more resistant for breaking cryptography keys that known systems.

References

1. Guan, P.: Cellular Automaton Public-Key Cryptosystem, *Complex Systems* 1, (1987) 51–56
2. Gutowitz, H.: Cryptography with Dynamical Systems, in E. Goles and N. Boccara (Eds.) *Cellular Automata and Cooperative Phenomena*, Kluwer Academic Press, (1993)
3. Habutsu, T., Nishio, Y., Sasae, I., Mori, S.: A Secret Key Cryptosystem by Iterating a Chaotic Map, *Proc. of Eurocrypt'91*, (1991) 127–140
4. Hortensius, P.D., McLeod, R.D., Card H.C.: Parallel random number generation for VLSI systems using cellular automata, *IEEE Trans. on Computers* 38, (1989) 1466–1473
5. Kari, J.:, Cryptosystems based on reversible cellular automata, personal communication, (1992)
6. Knuth, D.E.:, *The Art of Computer Programming*, vol. 1 & 2, *Seminumerical Algorithms*, Addison-Wesley, (1981)
7. Marsaglia, G.: Diehard `http://stat.fsu.edu/~geo/diehard.html`, (1998)
8. M. Mitchell, An Introduction to Genetic Algorithms (Complex Adaptive Systems), Publisher: *MIT Press*, ISBN: 0262133164
9. Menezes, A., van Oorschot, P., Vanstone, S.: *Handbook of Applied Cryptography*, CRC Press, (1996)
10. Mroczkowski,A.: Application of Cellular Automata in Cryptography, Master Thesis (in Polish), Warsaw University of Technology, (2002)
11. Nandi, S., Kar, B.K., Chaudhuri, P.P.: Theory and Applications of Cellular Automata in Cryptography, *IEEE Trans. on Computers*, v. 43, (1994) 1346–1357
12. National Institute of Standards and Technology, Federal Information Processing Standards Publication 140-2: *Security Requirements for Cryptographic Modules*, U.S. Government Printing Office, Washington (1999)
13. Sarkar, P.: A Brief History of Cellular Automata, *ACM Computing Surveys*, vol. 32, No. 1, (2000) 80–107
14. Schneier, B.: *Applied Cryptography*, Wiley, New York, (1996)
15. Sipper, M., Tomassini, M.: Generating parallel random number generators by cellular programming, *Int. Journal of Modern Physics C*, 7(2), (1996) 181–190

16. Tomassini, M., Perrenoud, M.: Stream Ciphers with One- and Two-Dimensional Cellular Automata, in M. Schoenauer at al. (Eds.) *Parallel Problem Solving from Nature – PPSN VI*, LNCS 1917, Springer, (2000) 722–731
17. Wolfram, S.: Cryptography with Cellular Automata, in *Advances in Cryptology: Crypto '85 Proceedings*, LNCS 218, Springer, (1986) 429–432