Annales UMCS Informatica AI 7 (2007) 49-60

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Searching for efficient cellular automata based keys applied in symmetric key cryptography

Mirosław Szaban[1*], Franciszek Seredyński[1,2]

[1]*Computer Science Department, The University of Podlasie, Sienkiewicza 51, 08-110 Siedlce, Poland*
[2]*Polish-Japanese Institute of Information Technology, Koszykowa 86, 02-008 Warszawa, Poland*

Abstract

In this paper we consider a problem of generation by cellular automata of high quality pseudorandom sequences useful in cryptography. For this purpose one dimensional nonuniform cellular automata is used. The quality of pseudorandom sequences generated by cellular automata depends on collective behavior of rules assigned to the cellular automata cells. Genetic algorithm is used to find suitable rules from the earlier predefined set of rules. It has been shown that genetic algorithm eliminates bad subsets of rules and finds subsets of rules, which provide high quality pseudorandom sequences. These sequences are suitable for symmetric key cryptography and can be used in different cryptographic modules.

1. Introduction

An increasing need for safety and privacy of digital information in many areas can be observed today. Use of digital tools in communication, data exchange, fast development of electronic commerce transactions and increasing use of digital signatures contribute to creation of new generations of secure mechanisms. Cryptography techniques are an essential component of any secure communication. Nowadays two main cryptography systems are used: secret and public-key systems. An extensive overview of currently known or emerging cryptography techniques used in both types of systems can be found, e.g. in [1]. One of such a promising for cryptography technique is applying cellular automata (CA).

CA was proposed for public-key cryptosystems by Guan [2] and Kari [3]. Such systems require two types of keys: one key for encryption and the other one for decryption. One of them is held in private, the other rendered public. However, the main concerns of this paper are cryptosystems with a secret key. In

[*]Corresponding author: *e-mail address*: mszaban@ap.siedlce.pl

such systems the encryption and the decryption keys are the same. The encryption process is based, in particular, on generation of pseudorandom bit sequences (PBSs), and CA can be effectively used for this purpose. CA for the systems with a secrete key were first studied by Wolfram [4], and later by Habutsu et al. [5], Nandi et al. [6] and Gutowitz [7]. Recently this subject was studied by Tomassini and Perrenoud [8], Tomassini and Sipper [9], and Seredynski et al. [10] who considered one or two dimensional (2D) CA for encryption. This paper is an extension of these recent studies and concerns the application of one dimensional (1D) CA for the secret key cryptography.

In this paper we present new results concerning the application of CA for generation of high quality pseudorandom number sequences (PBSs), which can be used in the symmetric key cryptography. The next section presents the idea of an encryption process based on the Vernam cipher. The main concepts of CA are outlined in section 3. Section 4 presents earlier discoveries with the use of evolutionary technique called cellular programming (CP) and concerning CA rules suitable for cryptography. These rules are the subject of our study. Section 5 describes the statement of the problem. Section 6 gives details of genetic algorithm (GA) used to solve the problem. New solutions found with the use of GA are presented and discussed in section 7. The last section concludes the paper.

2. Symmetric key cryptography and Vernam Cipher

The main idea of cryptography with using a symmetric key is that both sides of cryptographic process apply the same key to encrypt and decrypt the message. The key is secret and most secure because only two persons can use it and other people can know only encrypted message. In our study we continue Vernam's approach to cryptography with the secret key.

Let P be a plain-text message consisting of m bits ($p_1p_2...p_m$) and ($k_1k_2...k_m$) is a bit stream of a key k. Let c_i be the *i-th* bit of a cipher-text obtained by applying *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 (bit of a cipher-text) using the same bit stream key k:

$$p_i = c_i XOR\ k_i$$

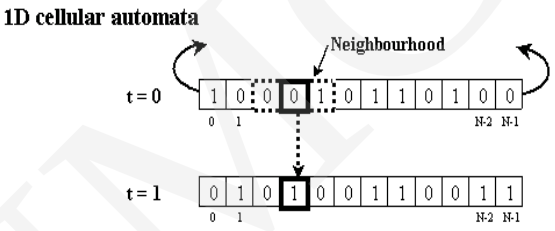
The enciphering algorithm called Vernam Cipher is known [1,11] as perfectly safe if the key stream is truly unpredictable and used only once. In this paper we give the answer to the questions: how to provide pure randomness of key bit stream, and how to obtain such a key of a length large enough to encrypt practical amounts of data. We can apply CA to generate high quality PBSs to

produce the safe secret key. We will show that by using 1D CA, the quality of PBSs for secret key cryptography and the safety of the key can be increased.

3. Cellular automata

1D CA is in the simplest case a collection of two-state elementary cells arranged in a lattice of the length N, and locally interacting in a discrete time t. For each cell i called a central cell, a neighborhood of a radius r is defined. The neighborhood consists of $n = 2r + 1$ cells, including the cell i. A cyclic boundary condition is applied to a finite size of CA whose results are in a circle grid. Figure 1a shows the states of 1D CA in two subsequent moments of time t.

a)



b)

Rule of CA

Neighbourhood radius r=1, rule $01011010_2=90_{10}$

Number	7	6	5	4	3	2	1	0
Neighbourhood	111	110	101	100	011	010	001	000
Rule result	0	1	0	1	1	0	1	0

Fig. 1. 1D cellular automata: an initial configuration and the first time step a), an example of transition function – CA rule with the neighborhood radius r=1 b)

It is assumed that a state q_i^{t+1} of a cell i (see, Figure 1a) at the time $t + 1$ depends only on states of its neighborhood at the time t, i.e. $q_i^{t+1} = q_i^{t+1}=f(q_i^t, q_{i1}^t, q_{i2}^t,\ldots,q_{in}^t)$. The transition function f is called a rule, defining the rule of updating the cell i. Figure 1b shows an example of a binary CA rule with $r = 1$, also named as the rule 90 (after the conversion into a decimal system). We will call all rules for CA with $r = 1$ the short rules and those with $r = 2$ the long rules. The length L of a rule and the number of neighboring states for a binary CA is $L = 2^n$, where n is a number of cells of a given neighborhood. The number of such rules can be expressed as 2^L. For CA with e.g. $r = 2$ the length of the rule $L = 32$, and the number of such rules is 2^{32} and grows very fast with L.

CA can change states in time with the use of one rule assigned to CA cells and it is called a uniform CA. If two or more different rules are assigned to update cells, CA is called nonuniform.

The first who applied CA to generate PBSs was Wolfram [4]. He used uniform, 1D CA with $r = 1$, and rule 30. Hortensius et al. [12] and Nandi et al. [6] used nonuniform CA with two rules 90 and 150, and found better quality of the generated PBSs than that of the Wolfram system. Recently Tomassini and Perrenoud [8] proposed to use nonuniform, 1D CA with $r = 1$ and four rules 90, 105, 150 and 165, which provide high quality PBSs and huge space of possible secret keys which is difficult for cryptanalysis. They used CP to search these rules. In this study we continue this line of research.

4. Discovery of CA rules using cellular programming

The quality of PBSs largely depends on the set of applied CA rules. Such rules randomly assigned to CA cells produce, due to their collective behavior PBSs, which potentially can be used as keys in cryptography. In the attempt to enlarge key space it was proposed by Seredynski et al. [10] to consider CA neighborhoods with $r = 1$ and $r = 2$ and search rules suitable for generating high quality PBSs. CP was used to search such rules. CP is an evolutionary computation technique similar to the diffusion model of parallel genetic algorithms (see, Cantu-Paz [13]) and it was proposed by Sipper et al. [14] to discover rules for nonuniform CA.

During experiments (see, [10]) it was noticed that in a single run of CP the evolutionary algorithm produces a set of a few rules, with a very high value of the entropy, but the quality of a single rule depends on a neighborhood of the rule. As the result of experiments 8 short rules were selected: five short rules discovered previously [4], [12], [6], [8], i.e. 30, 90, 105, 150, 165 new short rules 86, 101 153 and additionally 39 long rules (see, Table 1).

Table 1. Set of 47 rules discovered with CP and their values of entropy

No	Rule	Rule entropy
Rules known earlier		
1	30	3,98945
2	90	3,99320
3	105	3,99335
4	150	3,98983
5	165	3,63854
Rules with radius r = 1		
6	86	3,98981
7	101	3,99003
8	153	3,98873

Rules with radius r =2

No	Rule	Rule entropy	28	1457138022	3,98747
9	1824165059	3,97247	29	1470671190	3,98980
10	644466099	3,98885	30	1521202561	3,98822
11	702435689	3,98645	31	1537843542	3,98956
12	728094296	3,98830	32	1588175194	3,98903
13	869020563	3,98982	33	1704302169	3,98956
14	892695978	3,98918	34	1705400746	3,98986
15	898995801	3,99009	35	1720884581	3,98837
16	988725525	3,98805	36	1721277285	3,98850
17	1023152422	3,98708	37	1721277797	3,98889
18	1042531548	3,98814	38	1721325161	3,98987
19	1047380370	3,98953	39	1746646725	3,98853
20	1367311530	3,98873	40	1755030679	3,98928
21	1378666419	3,98817	41	1765883622	3,98714
22	1386720346	3,98835	42	1774427809	3,98757
23	1403823445	3,98851	43	1778009733	3,98730
24	1427564201	3,98899	44	1815843780	3,98983
25	1436194405	3,98871	45	2036803240	3,98826
26	1436965290	3,98973	46	2084275140	3,98968
27	1450086819	3,98649	47	2592765285	3,98974

Next, from the set of discovered 47 rules a subset of 8 rules ({*86, 90, 101, 105, 150, 153, 165, 1436194405*}) was selected in a semiautomatic way and a high cryptographic quality of this set was shown.

5. Statement of the problem

Closer analysis of selected subset of rules conducted by Bouvry et al. [15] has shown that some specific assignments of these rules to CA cells lead to bad statistical quality of PBSs generated by CA (CA cells generate stable or particularly stable sequences of 0s or 1s).

The purpose of this work was to eliminate from the set of discovered 47 rules bad rules and find subsets of rules which will be suitable for cryptographic purposes, for any assignments of them into CA cells. This search will be performed by GA.

6. Genetic algorithm searching useful subsets of CA rules

GA is a computational technique based on the principles of natural selection and genetics (see, Goldberg [16], Michalewicz [17]). Each individual of a GA population is coded in some way and represents a potential solution of a considered problem. A quality of individuals, in the sense of solving a problem, called value of a fitness function is evaluated. Genetic operators are applied to individuals in a single life cycle called a generation. The operators are the search engine of GA in each generation. The algorithm runs a predefined number of

generations and the best individual in the last generation is considered as a suboptimal or optimal solution of the problem.

6.1. Description of a problem in terms of genetic algorithm search

We consider a set of 47 rules as initial information for GA. We want to find in this set subsets of rules, whose collective behavior will provide generation by CA cells high quality PBSs, and will be free from the assignment problem presented earlier.

An individual in our population is not a single rule, but a subset of rules $ind_s^j = \{k_s^1, k_s^2, ..., k_s^j\}$, from the set of 47 rules, where $s \in \{1,...,S\}$, S is a number of all possible individuals, and j is a size of an individual. So, the length of individuals is not constant, but is defined by the random integer value.

6.2. Evaluation of individuals

An individual of GA population consists of a number of CA rules. To evaluate a quality of an individual, its rules should be assigned randomly to CA cells, and CA should start to evolve a number of time steps. Evolving CA which consists of N cells will produce N PBSs, quality of which will define a quality of an individual. To evaluate the statistical quality of each PBS, the entropy function E_h is used. We use the Shannon equation of even distribution as an entropy function.

To calculate value of the entropy each PBS is divided into subsequences of size $h = 4$. Let $k = 2$ be the number of values, which can take each element of a sequence and k_h a number of possible states 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_{hj} is a measured probability of occurrence of a sequence h_j in a PBS. The entropy achieves its maximum $E_h=h$ when the probabilities of the h_j possible sequences of the length h are equal to $1/k^h$. It is worth mentioning that the entropy is the only one of possible statistical measures of PBSs.

A single PBS is produced by a CA cell according to the assigned rules from individual and depends on a configuration of cells c_i all states of CA in time. To evaluate statistically reliable value of the entropy, CA with the rules assigned from the same individual ind_s (subset of rules) runs during T time steps. CA works in a predefined number of runs and starts with different initial configurations c_i for one individual. Entropy value is calculated for each PBS generated by CA cells with the assigned CA rules. Finally, the values of all entropies are averaged and serve as a fitness function *fi(ind)* of each individual

from the population. The values of fitness function for all individuals were mainly used to perform the selection operator in GA.

The evaluation of fitness of each GA individual according to the quality of generated PBSs can be described by the algorithm presented below:

Algorithm 1. Evaluation of individuals in P(gen)

```
FOR i=1 TO number_of_CA_tests DO
   set randomly initial states of CA cells
   assign rules from individual to CA cells
   run CA predefined number of steps
   evaluate the average entropy over all PBSs
END FOR
   evaluate the average entropy over the number_of_CA_tests
```

6.3. Genetic operators

Let us outline the GA operators used in the evolutionary process. Selection is based on the generally known tournament selection. We used a soft form of tournament and extend it by the elitist model (see, [16,17]). Crossover adapted to our problem consists in the averaging crossover (see, [17]), but it is modified to fit the individual environment. In this operator two parents are selected with a predefined probability p_k to be a parent. A selected pair of individuals ind_p^m and ind_q^n becomes parents. After the crossover operation a new individual ind_s^j is created and added to the next generation. From the selected parents of sizes m and n, we calculate the size $j = m + \mathrm{INT}(R_{(0,1)}(n - m))$ of the child, where $R(0,1)$ is a randomly chosen number from the range (0,1), and INT is a nearest integer value received after the conversion of the random number. The child ind_s^j will be created as $ind_s^j = \left(ind_p^m \cap ind_q^n\right) \cup A^i$, i.e. it will be composed of these rules, which are in both parents, and with $i = j - \overline{\overline{(ind_p^m \cap ind_q^n)}}$ rules, which are randomly selected from the set $A^i \in (ind_p^m \setminus ind_q^n) \cup (ind_q^n \setminus ind_p^m)$, the set of other rules from parents.

In the mutation process, a selected rule of an individual is replaced by a rule from the whole set of rules with the use of the Gaussian distribution $N(m;\sigma)$ expressed by the function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right).$$

Rules corresponding to an argument of the Gaussian function (x values in equation) are placed in order according to the increasing value of rule name on

the x axis. Mutation is a change of selected gene k_i (selected with the predefined mutation probability p_m), which is the rule from the individual into the rule k_j placed in $(x-0{,}5;\, x+0{,}5]$, where the selected $x \in \left\{ m-\sigma\sqrt{-2\ln(R_{(0,1)}\sigma\sqrt{2\pi})},\ m+\sigma\sqrt{-2\ln(R_{(0,1)}\sigma\sqrt{2\pi})} \right\}$ is an argument corresponding to the randomly selected value of Gaussian function. Finally, we replace the old rule k_i by a new rule k_j.

GA used to discover subsets of CA rules can be summarized in the following way:

Algorithm 2. Searching by GA subsets of CA rules

```
coding an individual in terms of the problem
gen=0
create initial population P(gen)
REPEAT
   evaluate P(gen)
   soft tournament selection + elitist model
   averaging crossover
   Gaussian mutation
   gen=gen+1
UNTIL termination_condition NOT TRUE
Solution=the best individual from P(gen)
```

7. Experimental results

A number of experiments has been conducted with the system. The population of GA consisting of 50 individuals, contained a number of rules ranging between 2 and 10. The algorithm was run for 50 generations. CA with *N=200* cells controlled by rules corresponding to a given individual of GA population was run 4096 time steps and the fitness function was computed from the sequence of 4096 bits. The value of the fitness function of a given individual was averaged over the entropy values of all PBSs corresponding to all CA cells, and over the number of CA tests (a number of initial allocations of rules).

The purpose of the first set of experiments was to tune the setting parameters of GA. We found that the best entropy values of individuals are generated by GA with the tournament size equal to 4 (see, Figure 2), and with the probability of winner acceptation from the range 0.7 to 0.9. Tournament selection was supported by the elite strategy with the elite size equal to 1. The probability of crossover was equal to 0.7. The probability of mutation based on the Gaussian distribution was equal to 0.001.

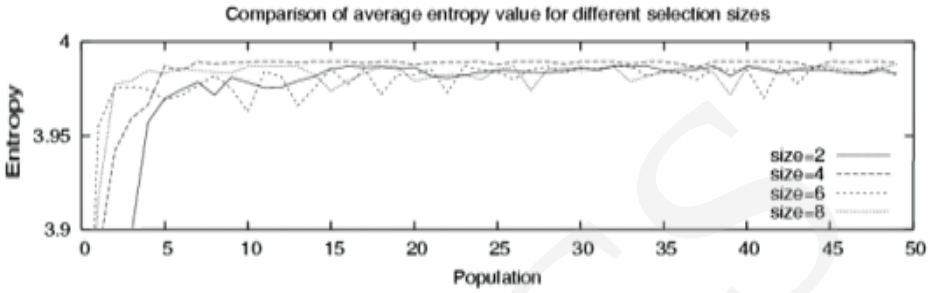


Fig. 2. Comparison of parameters for the genetic operators: tournament sizes equal to 2, 4, 6, 8

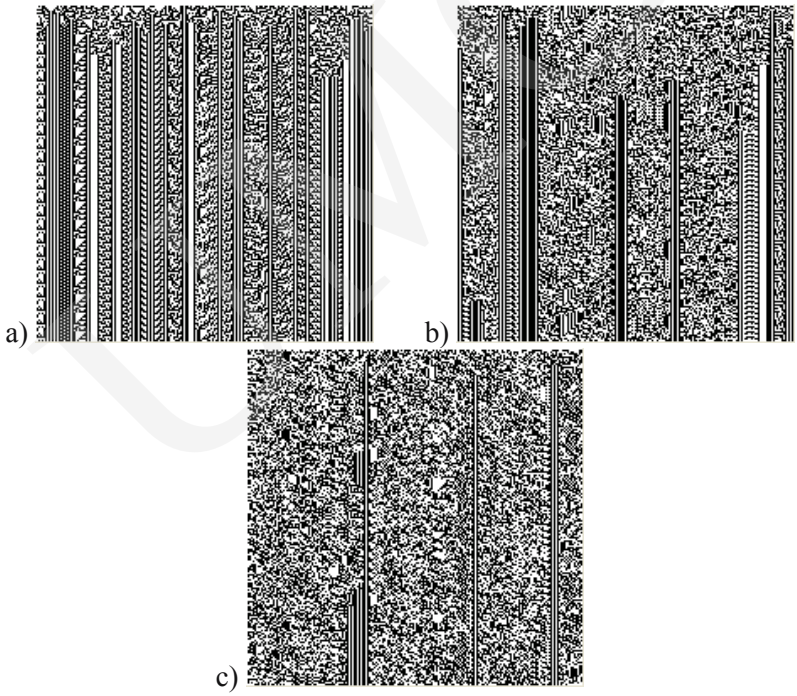


Fig. 3. Time Space Diagrams of CA run with the use of bad sets of rules: set {105, 86, 30} (a), set {86, 150, 1755030679, 1778009733, 869020563} (b) and set {105, 1720884581, 2036803240, 150, 1778009733} (c)

During the process of evolving subsets of rules by GA we observed, in particular, creation of bad subsets of rules. Figure 3 shows Time Space Diagrams of several subsets discovered in the initial stage of running GA. Distribution of rules in CA resulted in some cyclic or constant (unchanged in time) bit streams (see, Figure 3). In our algorithm, genetic operators and entropy test eliminate bad sets of rules during the work, because entropy value of bad

sets is much lower than in other sets (pure stable sequence has the entropy value equal to 0).

Finally, of 47 rules GA selected some subsets of rules. All of these sets are composed of rules from the set of 5 rules: *1436194405; 1436965290; 1721325161; 1704302169; 1705400746*. Each set from the new sets of rules is characterized by the high value of entropy, so CA with the use of these sets of rules generate high quality PBSs. The values of entropy fluctuate around the value 3.989. The ideal value of entropy is equal to 4, when the distribution in PBS is ideal. In our research, values of entropy for the discovered subsets of rules are close to the maximal value, so the obtained sets are probably suitable for the use it as a seed of generator of cryptographic keys. In Figure 4 Time Space Diagrams are presented for some of new sets. One can see that there are no constant sequences like in Figure 3, where bad assignments of rules generated low quality PBSs.

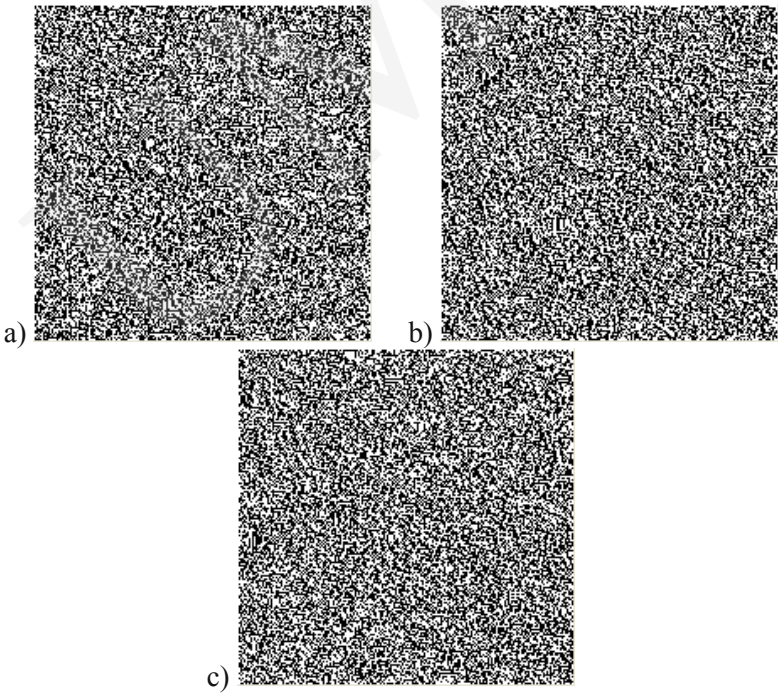


Fig. 4. Time Space Diagrams of CA with discovered good subsets of rules: subset of 5 rules {1436194405, 1436965290, 1721325161, 1704302169, 1705400746} (a), subset of 4 rules {1436194405, 1436965290, 1704302169, 1721325161} (b) and subset of 3 rules {1436965290, 1704302169, 1721325161} (c)

Before we use the proposed rules to generate key streams, we need to put them under a series of cryptographic tests called FIPS PUB 140-2. This set of

tests is composed of four tests: Monobit, Poker, Runs and Long Runs test. These tests evaluate a module which generates a number of sequences and if the test result is positive. After passing these tests the module can be called a pseudorandom number sequence generator (PNSG). In experiments it was shown that each set of new rules passed all tests in almost 100%. In particular, we can notice the largest subset consisting of 5 rules. This subset provides not only a high quality of PBSs, but also the largest key space.

Table 2. Comparison of the new set of rules with the earlier proposals

Test	Wolfram rule: 30	Nandi rules: 90, 150	Tomassini and Perrenoud rules: 90, 105, 150, 165	Our old rules set: 86, 90, 101, 105, 150, 153, 165, 1436194405	New set: 1436194405, 1436965290, 1704302169, 1721325161, 1705400746
Entropy min	3.988271	3.988626	3.988546	3.332641	3.988825
Entropy ave.	3.989431	3.989424	3.989402	3.938360	3.989474
Entropy max.	3.990477	3.990330	3.990253	3.990003	3.990315
Monobit test	99.92%	99.88%	99.84%	96.07%	99.81%
Poker test	99.98%	99.97%	99.98%	96.5%	99.93%
Runs test	99.8%	99.92%	99.89%	95.44%	99.85%
Long runs test	100%	100%	100%	99.46%	100%
Key space	2^N*X	2^N*2^N*X	4^N*2^N*X	8^N*2^N*X	5^N*2^N*X

X – module settings

When we compare sets of rules discovered earlier with the best set from the new discovered sets, we can see that the quality of the new set of rules is better than that of previously found sets (see, Table 2).

The values of entropy provided by the new set of rules are higher (the average value) than in other sets (see, Table 2). Corresponding cryptographic modules (PNSGs) are characterized by different key spaces. The best discovered module based on 5 rules has the key space larger than the previous modules, except the one with our old set of rules. It means that the new module is characterized by the key space large enough, but at the same time lower amount of data needs to be sent by a safe communication channel. The high quality of generated PBSs is still preserved. So, we conclude that the new cryptographic module is more effective.

8. Conclusions

We have presented a searching mechanism based on GA that allowed selecting small sets of rules, which are more effective than the initial set. We

selected 10 sets of rules, which provide high quality PBSs. The sets were tested by the standard tests of randomness. One selected from them improves the quality of generated PBSs, preserving at the same time huge key space. CA with these rules can be used as a PBSs generator. Random assignments of these rules to CA cells, together with an initial CA cells configuration can be used as a seed to produce keys (stream of bits), which are applied in Vernam Cipher. Our future work will focus on study of a full set of rules ($r = 1$ and $r = 2$), and also consider taking into acount higher dimension CA.

References

[1] Schneier B., *Applied Cryptography*, Wiley, New York, (1996).
[2] Guan P., *Cellular Automaton Public-Key Cryptosystem*. Complex Systems, 1 (1987) 51.
[3] Kari J., *Cryptosystems based on reversible cellular automata*. Personal Communication, (1992).
[4] Wolfram S., *Cryptography with Cellular Automata*. Crypto '85 Proceedings, LNCS 218, Springer, (1986) 429.
[5] Habutsu T., Nishio Y., Sasae I., Mori, S. A Secret Key Cryptosystem by Iterating a Chaotic Map. *Proc. of Eurocrypt'91*, (1991) 127.
[6] Nandi S., Kar B.K., Chaudhuri P.P., Theory and Applications of Cellular Automata in Cryptography. *IEEE Trans. on Computers*, 43 (1994) 1346.
[7] Gutowitz H., *Cryptography with Dynamical Systems*. in *E. Goles and N. Boccara (Eds.) Cellular Automata and Cooperative Phenomena*, Kluwer Academic Press, (1993).
[8] Tomassini M., Perrenoud M., *Stream Ciphers with One- and Two-Dimensional Cellular Automata*. in *M. Schoenauer et al. (Eds.) Parallel Problem Solving from Nature – PPSN VI*,, LNCS 1917, Springer, (2000) 722.
[9] Tomassini M., Sipper M., *On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata*. IEEE Trans. on Comp., 49(10) (2000) 1140.
[10] Seredynski F., Bouvry P., Zomaya A., *Cellular Automata Computation and Secret Key Cryptography*. Parallel Computing, 30 (2004) 753.
[11] Menezes A., Oorschot P., Vanstone S., *Handbook of Applied Cryptography*, CRC Press, (1996).
[12] Hortensius R.D., McLeod R.D., Card, H.C., *Parallel random number generation for VLSI systems using cellular automata*. IEEE Trans. on Computers, LNCS 218, Springer, 38 (1989) 1466.
[13] Cantu-Paz E., *Parallel Genetic Algorithms*, in *GECCO 2003: 2003 Genetic and Evolutionery Computation Conference*, Tutorial Program, Chicago, Illinois, (2003) 241.
[14] Sipper M., Tomassini M., *Generating parallel random number generators by cellular programming*. Int. Journal of Modern Physics C, 7(2) (1996) 181.
[15] Bouvry P., Klein G., Seredynski F., *Weak Key Analysis and Microcontroller Implementation of CA Stream Ciphers*. LNAI 3684, Springer, (2005) 910.
[16] Goldberg D.E., *Genetic Algorithms in Search, Optimization and MachineLearning*. Addisn-Wesley Pub, (1989).
[17] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, (1994).