Evolving Collective Behavior of Cellular Automata for Cryptography

Miroslaw Szaban
The University of Podlasie
Computer Science Department
Sienkiewicza 51, 08-110 Siedlce, Poland
Email: mszaban@ap.siedlce.pl

Franciszek Seredynski
The University of Podlasie
Computer Science Department
Sienkiewicza 51, 08-110 Siedlce, Poland
Polish-Japanese Inst. of Inf. Technology
Koszykowa 86, 02-008 Warsaw, Poland
Email: sered@ipipan.waw.pl

Pascal Bouvry
Luxembourg University
Faculty of Sciences, Technology and Communication
6, rue Coudenhove Kalergi
L-1359 Luxembourg-Kirchberg
Luxembourg
Email: pascal.bouvry@uni.lu

***Abstract*— We consider 1D cellular automata (CA) and apply genetic algorithm (GA) to discover subsets of rules controlling CA cells, which collective behavior will result in a high quality of pseudorandom number sequences (PNSs) suitable for symmetric key cryptography. The search of subsets of rules is performed in a set of predefined rules. We discover new subsets of CA rules providing very high quality of PNSs, which can be used in cryptographic modules.**

I. Introduction

In the era of digital information computer resources are not safe from attacks. Public and widespread use of digital tools in communication quickly causes the creation of secure mechanisms. Cryptography techniques are one of them. 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 type of systems can be found in [9].

Cellular automata (CA) were proposed for public-key cryptosystems by Guan [2] and Kari [5]. In such systems two keys are required: one key for encryption and the other for decryption; one of them is held in private, the other rendered public. However the main concern of this paper are cryptosystems with a secret key. In such systems the encryption and the decryption key are the same. The encryption process is based on generation of pseudorandom bit sequences, and CA can be effectively used for this purpose. CA for systems with a secrete key were first studied by Wolfram [13], and later by Habutsu et al. [4], Nandi et al. [8] and Gutowitz [3]. Recently this subject was studied by the Tomassini - Perrenoud [11], Tomassini - Sipper [12], and Seredynski et al. [10] who's considered one or two dimensional (2D) CA for encryption schema. This paper is an extension of these recent studies and concerns on application of one dimensional (1D) CA for the secret key cryptography.

In this paper we present the new results concerning application of CA for symmetric key cryptography. The next section presents the idea of an encryption process based on Vernam cipher. The main concepts of CA are presented in section 3. Section 4 describes statement of problem. Section 5 presents our GA and its main stages. New solutions are described in section 6. Last section concludes the paper.

II. Symmetric Key Cryptography and Vernam Cipher

Cryptography with use of symmetric key characterizes that both sides apply this same key to encrypt and decrypt the message. This key is secret and most secure because only two person can use it and other people can known only encrypted message which is to difficult to break it. In our study we continue Vernam's approach to cryptography with 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: $c_i = p_iXORk_i$. The original bit p_i of a message can be recovered by applying the same operation XOR on c_i using the same bit stream key k: $p_i = c_iXORk_i$. The enciphering algorithm called Vernam Cipher is known [6], [9] as perfectly safe if the key stream is truly unpredictable and used only one time. We can apply CA to generate high quality pseudorandom number sequences (PNSs) and use them as the safe secret key. We will show that by using 1D CA, the quality of PNSs for secret key cryptography and the safety of the key can be increased.

III. 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 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. In this case a cyclic boundary condition is applied to a finite size of CA, what results is in a circle grid. Fig. 1a shows 1D CA in two subsequent moments of time t.

It is assumed that a state q_i^{t+1} of a cell i (see Fig. 1a) at the time $t+1$ depends only on states of its neighbourhood at the time t, i.e. $q_i^{t+1} = f(q_i^t, q_{i1}^t, q_{i2}^t, , q_{in}^t)$ and the transition function f, called a rule, which defines the rule of updating the cell i. Fig. 1b shows an example of a rule for CA with $r = 1$. This binary rule can be also named the rule 90, after

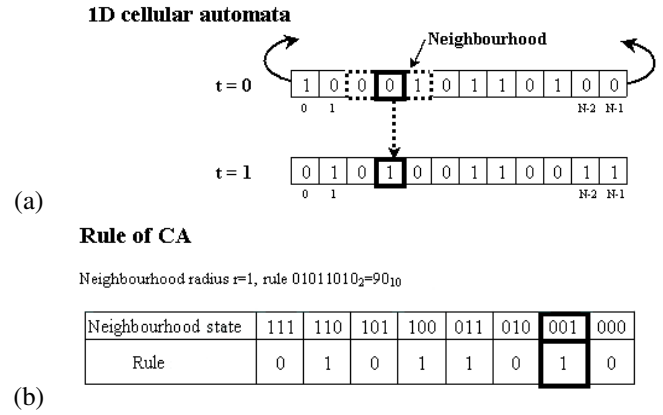


Fig. 1. 1D Cellular automata: (**a**) Initial configuration and first time step , (**b**) an example of transition function - CA rule with neighbourhood radius r=1.

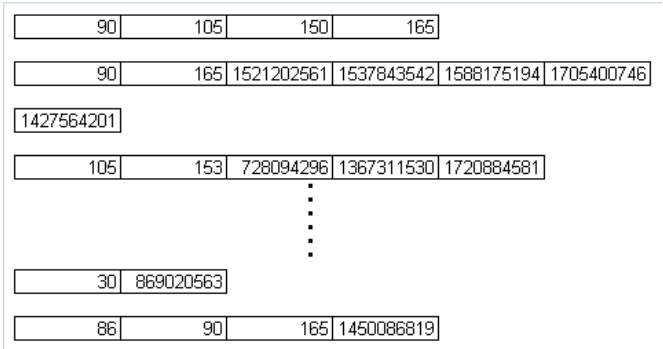


Fig. 2. An Example of population of individuals composed of rules.

conversion into a decimal system. All rules for CA with $r = 1$ we will call short rules and rules for CA with $r = 2$ we will call long rules. CA is called a uniform if one, the same rule is assigned to all cells. If two or more rules are assigned to cells, CA is called nonuniform.

IV. Statement of a problem

In [10] a set of 47 rules for 1D CA was discovered using cellular programming [11]. These rules (short and long) were characterized by high values of entropy and potentially were suitable for generating high quality PNSs, suitable for cryptography. Among these rules a subset of 8 rules was selected in a semi-automatic way and was shown a high cryptographic quality of this set. Next research [1] have shown however, that some assigning of these rules to CA cells leads to bad statistical quality of PNSs generated by CA.

The purpose of this work was to find in the set of discovered 47 rules, subset of rules which are suitable for cryptographic purposes, for any assigning them into CA cells. This search will be performed by GA.

V. Genetic Algorithm Searching Useful Subsets of CA Rules

GA [10] is a computational technique based on principles of natural selection and genetics.

Each of individuals of GA is a potential solution of a problem. In our population an individual is not a single rule, but a set of rules $ind_s^j = \{k_s^1, k_s^2, ..., k_s^j\}$, where $ind_s^j \in Ind_S$, $s \in \{1, ..., S\}$, S is a number of all possible individuals, and j is a size of individual(number of rules in it, chosen from the 47 discovered rules). So, the population P is composed of individuals from the set $Ind_S = \{ind_1, ..., ind_s, ..., ind_S\}$. For all of individuals we assign random value of a size $j = \sum k_s^i$, $k_s^i \in ind_s$, as a number of rules in it. Example of population composed of individuals with different lengths is presented in the Fig. 2. Rules of an individual of GA are assigned to cells of CA, and CA runs some number of time steps, producing PNSs.

All PNSs, which describe work done by CA's rules are evaluated when GA's finish its evolution. The entropy E_h is used to specify the statistical quality of each PNS. We used Shannon equation of even distribution as an entropy function. To calculate a value of the entropy each PNS is divided into subsequences of size h. In all experiments the $h = 4$ 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 of possible states of each sequence ($k^h = 16$). E_h can be calculated in the following way:

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

where p_{h_j} is a probability of occurrence of a sequence h_j in a PNS. The entropy achieves its maximum $E_h = h$ when the probabilities of the h_j possible sequences of the length h are equal to $\frac{1}{k^h}$. It is worth to mention that the entropy is only one of possible statistical measures of PNSs. Entropy was used as a fitness function of GA. The description of GA consists of the following steps:

```
  Alg. 1: Searching subsets of CA rules
coding an ind. in terms of the problem
gen=0
initial population P(gen)
REPEAT
  evaluate P(gen)
  soft tournament selection+elite strategy
  averaging crossover
  Gaussian mutation
  gen=gen+1
UNTIL termination condition NOT TRUE
Problem solution=the best ind. from P(gen)

  Alg. 2: Evaluation of ind. of P(gen)
FOR i=1 TO number_of_CA_tests DO
  set randomly initial states of CA cells
  assign rules from an ind. to CA cells
  run CA predefined number of steps
  evaluate the average entropy over all PNSs
END
```

Let's outline GA operators. Selection is based on generally

known tournament selection. We used soft form of tournament and extend it by elite strategy [7]. Crossover adapted to our problem is the averaging crossover [7]. In this operation participates two parents, with predefined probability p_k to be a parent. Selected pair of individuals ind_p^m and ind_q^n become the parents. After crossover operation it gives a child, new individual ind_s^j for next generation of population. From selected parents with sizes m and n, we calculate size $j = m + E(R_{(0,1)}(n-m))$ of the child, where $R(0,1)$ is a randomly chosen number in the bracket (0,1), and E is an integer value of the number in bracket. Child will be created in the following way:

$$ind_s^j = (ind_p^m \cap ind_q^n) \cup A^i, \tag{2}$$

and will be composed of those rules, which are in both parents, and $i = j - \overline{(ind_p^m \cap ind_q^n)}$ rules, which were randomly chosen from set A^i, set of other rules from parents, selected in the following way:

$$A^i = (ind_p^m \setminus ind_q^n) \cup (ind_q^n \setminus ind_p^m). \tag{3}$$

In mutation process in our GA, selected rule of an individual is replaced by a rule from the whole set of rules, using Gaussian distribution $N(m, \sigma)$. Rules (x value in distribution) are arguments, putted in order according to increasing value of rule's name on x axis. Expected value in this point is $m = No(k_i) + 1 - No(k)div2$, where $No(k_i)$ is a number of rule k_i selected to mutation and $No(k)$ is number of rules in all set. Mutation is a change of selected gene (selected with predefined mutation's probability p_m), which is the rule from the individual into the rule $k_j \Leftrightarrow No(k_i) + 1 - No(k)div2 \in (x - 0.5, x + 0.5]$, where the randomly selected $x \in \{m - \sigma\sqrt{2\ln(\sigma\sqrt{2\pi}R_{(0,1)})}, m + \sigma\sqrt{2\ln(\sigma\sqrt{2\pi}R_{(0,1)})}\}$ according to linear distribution $N(No(k_i) + 1 - No(k)div2, \sigma)$. Finally, we replace old rule k_i by a new rule k_j.

GA is executed predefined number of times, i.e. particular number of generations gen are generated.

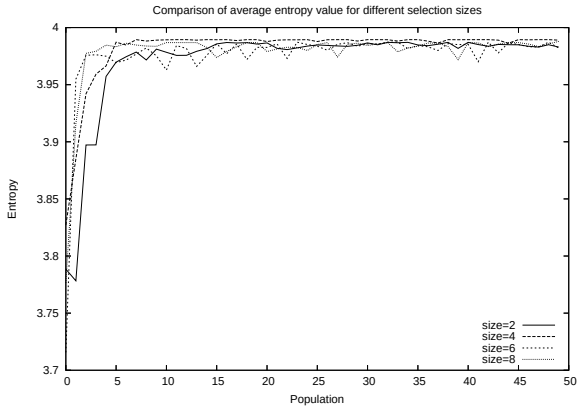
VI. Experimental Results

A number of experiments have been conducted. The population of GA consists of 50 individuals. Individuals contain a number of rules ranging between 2 and 10. The algorithm was running 50 generations. CA controlled GA worked by 4096 time steps and fitness function was computed from sequence of 4096 bits. The value of a fitness function of a given individual is the average of entropy values of all PNSs generated by CA rules of the individual.

The purpose of the first set experiments was to tune setting parameters of GA. We found that the best results in entropy values the algorithm generates with tournament size equal to 4 (see Fig. 3a), with probability of winner acceptation then the range 0.7 to 0.9 (see Fig. 3b). Tournament selection was supported by elite strategy with elite size equal to 1. The probability of crossover was equal to 0.7. The probability of mutation based on Gaussian distribution was equal to 0.001.

During the process of evolving subsets of rules by GA we observed creation of bad subsets of rules. Fig. 4 shows Time

(a)



(b)

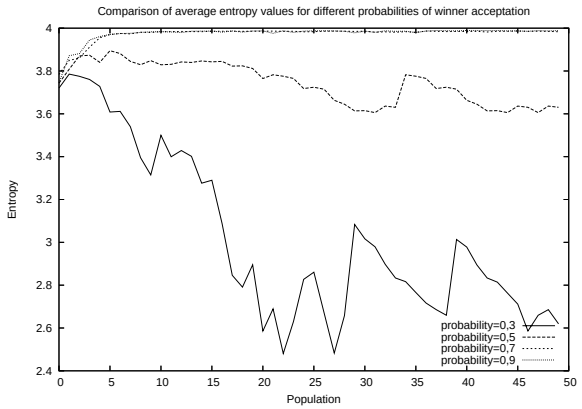
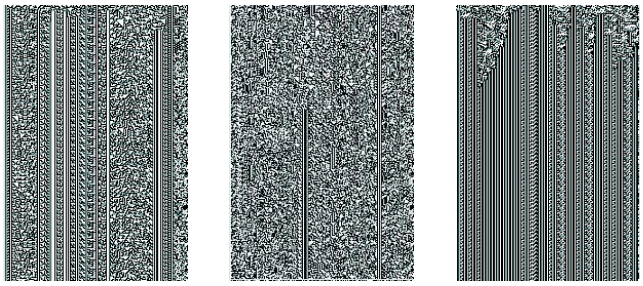


Fig. 3. Comparison of parameters for genetic operators: (**a**) tournament sizes equal to 2, 4, 6, 8, (**b**) probabilities of winner acceptation equal to 0.3, 0.5, 0.7, 0.9.

Space Diagrams of several subsets discovered in initial stage of running GA. Distribution of rules in CA resulted in some bit streams which do not change in time (see Fig. 4). In our algorithm Entropy test eliminates bad sets of rules during the work, because entropy value of bad sets is lower than in others sets.

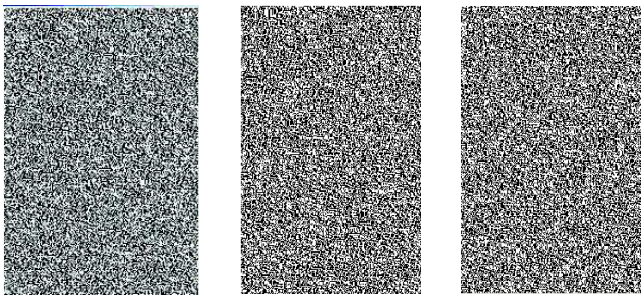
Finally, from 47 rules GA selected 10 subsets of rules (see Table I). All of these 10 sets are composed of rules from the set of 5 rules: $1436194405, 1436965290, 1721325161, 1704302169, 1705400746$. Each set from the new sets of rules, is characterized by the high value of entropy, so CA with use those sets of rules gives high quality distribution in generated PNSs (see Table I). Values of entropy fluctuate up to value 3.989. Ideal value of entropy is equal to 4, when the distribution in PNS is ideal. Results of discovered sets of rules are close to maximal value, so obtained sets give good results. In Fig. 5 Time Space Diagrams are presented for some of new sets. There is no constant sequences like in Fig. 4, where are presented low quality PNSs.

The next step of the study is series of cryptographic tests called FIPS 140-2 tests. That set of tests is composed of four tests: Monobit, Poker, Runs and Long Runs tests. These tests evaluate a module creating number sequences and if the test result is positive then module can be called PNSG - generator



(a) (b) (c)

Fig. 4. Time Space Diagram of CA run with use of bad sets of rules: (**a**) set {105, 86, 30} , (**b**) set {86, 150, 1755030679, 1778009733, 869020563} and (**c**) set {105, 1720884581, 2036803240, 150, 1778009733}.



(a) (b) (c)

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

of PNSs. In experiments was shown that each set of new rules passed all test from tests set in 100%.

If we compare sets discovered earlier and the best set from new discovered sets, we get that quality of new set of rules is better than previously defined sets (see Table II).

Values of entropy in the new generator is the highest (average value) than in others generators, comparison is presented in Table II. Applied cryptographic module use presented above generators and gives different space of key bit stream. Discovered module has the lowest key space only for our old set of rules, but it has only 5 rules and determine simple generator. Small number of rules in the set determines low quantity of data to send by safe communication channel. Small amount of rules in the new set leads to the simpler generation of PNSs with the higher quality from that we conclude that the new generator gives better results by lowest costs of time, memory of application and quantity of secured data.

VII. Conclusions

We have presented a searching mechanism based on GA that allowed to select a small set of rules, which are more effective than the initial set. We selected 10 sets giving high quality PNSs. CA with these rules was used as a PNSs generator. Rules from the new selected set was proposed as a seed to produce key stream of bits, which are applied in Vernam Cipher. CA with rules from the new set was found as very high quality PNS generators. The best sets were tested by standard tests of randomness. One selected set improves time of work CA, by decreasing number of CA rules. A high quality of generated PNSs saves quite large key space, and decreased its own protected quantities of cryptosystems data. Our future work will be devoted to increase a set of rules to full set of rules ($r = 1$ and $r = 2$), and also consider higher dimension CA.

TABLE I

Discovered 10 subset of rules and theirs performance.

No	Subset of rules	Entropy test (value) Min.	Ave.	Max.	FIPS Mono.	Poker	140-2 test Runs	L.runs	(%)
1	1436194405, 1721325161	3.98787	3.98923	3.99022	100	100	100	100	
2	1436194405, 1704302169	3.98827	3.98940	3.99072	100	100	100	100	
3	1704302169, 1721325161	3.98812	3.98936	3.99048	100	100	100	100	
4	1436194405, 1721325161, 1704302169	3.98846	3.98951	3.99082	100	100	100	100	
5	1436965290, 1705400746, 1704302169, 1721325161	3.98835	3.98940	3.99041	100	100	100	100	
6	1436194405, 1436965290, 1704302169, 1721325161	3.98878	3.98941	3.99032	100	100	100	100	
7	**1436194405, 1436965290, 1704302169, 1721325161, 1705400746**	**3.98882**	**3.98947**	**3.99031**	**100**	**100**	**100**	**100**	
8	1436965290, 1705400746	3.98881	3.98947	3.99017	100	100	100	100	
9	1436194405, 1436965290, 1704302169	3.98834	3.98941	3.99013	100	100	100	100	
10	1436965290, 1704302169, 1721325161	3.98862	3.98928	3.99049	100	100	100	100	

TABLE II

Comparison of the new set of rules, and earlier proposals.

Test	Wolfram rule: 30	Nandi rules: 90, 150	Tomassini and Perrenoud rules: 90, 105, 150, 165	Our old set: rules 86, 90, 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	100	100	100	100	100
Poker test	100	100	100	100	100
Runs test	100	100	100	100	100
Long runs test	100	100	100	100	100
Key space	N*2N*X	2N*2N*X	4N*2N*X	8N*2N*X	5N*2N*X

X - module settings

References

[1] Bouvry P., Klein G. and Seredynski F. *Weak Key Analysis and Microcontroller Implementation of CA Stream Ciphers*, LNAI 3684, Springer, 2005, pp. 910-915

[2] Guan P. *Cellular Automaton Public-Key Cryptosystem*, Complex Systems 1, 1987, pp. 51-56

[3] Gutowitz H. *Cryptography with Dynamical Systems*, in E. Goles and N. Boccara (Eds.) Cellular Automata and Cooperative Phenomena, Kluwer Academic Press, 1993

[4] Habutsu T. et al. *A Secret Key Cryptosystem by Iterating a Chaotic Map*, Proc. of Eurocrypt'91, 1991, pp. 127-140

[5] Kari J. *Cryptosystems based on reversible cellular automata*, Personal Communication, 1992

[6] Menezes A. et al. *Handbook of Applied Cryptography*, CRC Press, 1996

[7] Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 1994
[8] Nandi S. et al. *Theory and Applications of Cellular Automata in Cryptography*, IEEE Trans. on Computers, v. 43, 1994, pp. 1346-1357
[9] Schneier B. *Applied Cryptography*, Wiley, New York, 1996
[10] Seredynski F., Bouvry P. and Zomaya A. *Cellular Automata Computation and Secret Key Cryptography*, Parallel Computation 30, 2004, pp. 753-766
[11] Tomassini M. and 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, pp. 722-731
[12] Tomassini M. and Sipper M. *On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata*, IEEE Trans. on Computers, 2000, v. 49, No. 10, pp. 1140-1151
[13] Wolfram S. *Cryptography with Cellular Automata*, in Advances in Cryptology: Crypto '85 Proceedings, LNCS 218, Springer, 1986, pp. 429-432