Proceedings of the Federated Conference on Computer Science and Information Systems pp. 381–385 DOI: 10.15439/2018F234 ISSN 2300-5963 ACSIS, Vol. 15

Improving pseudorandom generator on cellular automata with bent functions

Mukhamedjanov Daniyar*, Ryaskin Gleb†, Levina Alla‡ and Kaplun Dmitrii§
*ITMO University
49 Kronverksky pr., St. Petersburg, 197101, Russia Email: danmd.info@gmail.com
†ITMO University
Email: ryaskingleb20@gmail.com
‡ITMO University
Email: alla_levina@mail.ru
§Saint-Petersburg Electrotechnical University "LETI"
Email: dikaplun@etu.ru

***Abstract*—Nowadays the practice of researching pseudorandom number generators (PRNG) becomes more scalable because of its spreading in many spheres of computer science and, especially, cybersecurity. The problem is that existing generators are still have many disadvantages in terms of velocity, complexity or flexibility. Thus, the area of researching new algorithms of generating pseudorandom sequences is more than just applicable method, but the target for multiplying cybersecurity from the hardware to application level.**

This leads to make the set of available and useful PRNG larger and better by their features, like velocity, performance, simplicity in realization. These features match PRNG, based on cellular automata (CA), but not all rules, used in CA are appropriate for their transition functions. Bent functions are perfectly complement statistical weakness of some rules because of their non-linearity without loss of other features.

I. Introduction

MODERN society is vulnerable enough for hacking attacks, which are very large-spectered, from hardware attacks (on microchips) to web hacking (or application level hacking). Since, the first thing for computer science society and cybersecurity itself, is to protect personal data of users, which is the main aim for hackers. When we say "protect personal data", it comes cryptography methods in the first sight, like ciphers or lightweight cryptography, or coding theory. The basement of majority of cryptographical methods is generating random numbers, which can be both physically processed and mathematically. This paper is about second one, to be more exact, about pseudorandom number generator, based on homogenous structures using bent functions.

A. Homogenous structures overview

Lets take the set of k-dimensional vectors Z^k, set of 1 and $0 - E_n$, ordered set $V = (\alpha_1, \alpha_{(h-1)})$, where α_i is k dimensional vector from Z^k. Besides, lets determine a function $\phi = \phi(x_0, x_1, , x_{(h-1)})$, $\phi : (E_n)^h \to E_n$, additionally $(\phi(0, 0, ..., 0) = 0)$. As the result we'll get "four" $\sigma = (Z^k, E_n, V, \phi)$. That will be formal determinition of Homogenous structures (HS) [7], where E_n set of states of one cell in ϕ - local transition function.

Generally, HS are usually represented by ordered set of many Moore automata [13], which have states of other automata as input. To understand which of automata can influence on another one special scheme or neighborhood template can be used. There are several schemes, that are usually used in HS, like Moore's scheme (2 dimensional scheme, where target cell, surrounded by square of cells 3x3 if radius $r = 1$, or 5x5 if radius $r = 2$) or Neuman's one (2 dimensional scheme, where non diagonal cells surrounding the target one)

HS can be determined as dynamical discrete systems, where time and space are discrete. Changing of states of cells is conditioned by function ϕ, which is also included in rule. The rule is like a finite automaton with input, transition function and output.

Without loss of formalization and main idea, here and further term HS will be replaced by Cellular automata (CA), as this term is more widespread. Sharing CAs by complexity feature, it can be seen that there are two types of them: uniform CA (when all the grid have one rule and only one neighborhood template) or non-uniform CA (several neighborhood templates or(and) several rules). From this, perfomance features of both CA types are almost the same, besides memory (non-uniform needs more memory space for keeping rules and neighborhood templates). To save boards of the grid of CA, we will consider, that 2-dimensional grid is about toroidal form (without distortion in sizes).

Time spacing is a method of producing sequences with periodically interruptions in reading bits for one or several evolutions (iterations). That means, that only several iterations will influence on resulting bit sequence.

Site spacing is a method of producing sequences with periodically skipping of some bits in the grid in every iteration.

Widely spread so-called Wolfram's notation [17] for the rules. Firstly, lets call configuration the set of ones and zeros (two-state CA, where the sequence is considered as random number) at particular discrete moment of time. Further rule numbers are also suggested by Wolfram. Wolfram's rule 30 as an example:

How Wolfram's rules can be encoded goes further. For example, $f(111) = 0, f(110) = 0, f(101) = 0, f(100) = 1, f(011) = 1, f(010) = 1, f(001) = 1, f(000) = 0$, is denoted rule 30.

B. Pseudorandom number generators (PRNG) overview

Random numbers are needed in different applications, i.e. in the cryptography area and coding theory. A number of algorithms need repeatable random numbers, based on deterministic algorithms, it is more correct to call such numbers pseudorandom, since they differ from the true random sequences obtained as a result of natural physical process.

Random number generators should have a number of properties if they are to be successfully applied in long stochastic models, like those used in computational physics. The most important properties from this point of view are good results in standard statistical tests for randomness, computational efficiency, a long period (the minimum number between repetitions) and the reproducibility of the sequence, e.g. NIST Test Suite. Considering several examples of PRNG, let's pay attention to Linear Congruential method and LFSR. They are quite simple, but perfectly describe the main idea of pseudorandom numbers.

C. Bent function

Bent functions are boolean functions with an extreme value nonlinearity. The measure of nonlinearity is an important characteristic boolean functions in cryptography. Linearity and properties close to it testify to the simple structure of this function and, as a rule, represent a large source of information about many other of its properties.

The nonlinearity of a function f is the distance from f to a class of affine functions. We denote the nonlinearity of the function f in terms of N_f :

$$N_f = d(f, A(n)) = \min_{g \in A(n)} d(f, g)$$

where $A(n)$ is class of affine functions.

The formula for calculating N_f by the Walsh-Hadamard transform:

$$d(\langle a, x\rangle, f) = \hat{f}(a) = 2^{n-1} - \frac{1}{2}\max_{a \in Z_2^n} |\hat{f}(a)|$$

Let $f \in P_2(n)$, write for it the Parseval equality:

$$\Sigma_{a \in Z_2^n} \hat{f}^2(a) \geq 2^n$$

We have 2^n non-negative summands whose sum is 2^{2n}. Consequently $\max_{a \in Z_2^n} \hat{f}(a) \geq 2^n$, from which it follows that $\max_{a \in Z_2^n} |\hat{f}(a)| \geq 2^{\frac{n}{2}}$. Therefore

$$N_f = 2^{n-1} - \frac{1}{2}\max_{a \in Z_2^n} |\hat{f}(a)| \leq 2^{n-1} - 2^{\frac{n}{2}-1}$$

The function $f \in P_2(n)$ is called maximally nonlinear if $N_f = 2^{n-1} - 2^{\frac{n}{2}-1}$

Definition: A bent function is a Boolean function with an even number of variables for which the Hamming distance from the set of affine Boolean functions with the same number of variables is maximal.

The properties of bent functions:

1) bent functions exist only for even n;
2) bent functions depend statistically on all their arguments;
3) let f be a bent function, and h belong to the class of linear functions. Then $f \oplus h$ belongs to the class of bent functions;
4) Let $(f \in P_2(n), g \in P_2(m)$ - be functions of disjoint sets of variables. Then $f \oplus h$ is a bent function if and only if f and g are bent functions.

We give examples of bent functions of a different number of variables.

for $n = 4$:

$f(x_0, x_1, x_2, x_3) = x_0x_1 + x_2x_3$
$f(x_0, x_1, x_2, x_3) = x_0x_1 + x_2x_3 + x_0 + x_1$

for $n = 6$:

$f(x_0, x_1, x_2, x_3, x_4, x_5) = x_0x_1 + x_2x_3 + x_4x_5$
$f(x_0, x_1, x_2, x_3, x_4, x_5) = x_0x_1x_2 + x_1x_3x_4 + x_0x_1 + x_0x_3 + x_1x_5 + x_2x_4 + x_3x_4$

for $n = 8$:

$f(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_0x_1x_2 + x_1x_3x_4 + x_2x_3x_5 + x_0x_3x_6 + x_2x_4 + x_1x_6 + x_0x_4 + x_0x_5 + x_3x_7$

From the point of view of cryptography, the important criteria that a Boolean function f of n variables must satisfy are the following :

- equilibrium - the function f takes values 0 and 1 equally often;
- the propagation criterion $PC(k)$ of order k - for any nonzero vector $y \in Z_2^n$ weight at most k, the function $f(x + y) + f(x)$ is balanced;
- the maximum nonlinearity - the function f is such that the value of its nonlinearity N_f is maximal;

II. Algorithm of generating pseudorandom numbers on CA using Bent functions

The main idea of algorithm is in using CA Rules and bent functions to generate pseudorandom sequences of bits by usage of simple XOR operation to improve statistical features of CA sequences. Unique features of bent functions, like non-linearity and simplicity allows to generate quite random sequences in the grid of CA. Also, in terms of hardware or software implementation bent functions are simple enough to realize. As a result, we will get deterministic bent function output with n inputs of CA cells with Wolfram's notation rules.

A. Grid

From the point of view of geometry grid in our algorithm is represented by 2-dimensional parallelogram with sizes p and q, divided by equal cells, which contains only one of two possible states 0 or 1. Actually, the grid can be chosen with random sizes, but it is strongly recommended to create the grid, where p and q are prime numbers. It can improve periodical feature of output sequence. Let's divide this grid into two blocks, where one block is for CA rule and another one is for bent function sequence. Every block b_i is 2 dimensional parallelogram with sizes l_{b_i} and w_{b_i}. Obviously, each block consists of $l_{b_i} * w_{b_i}$ cells. But result output sequence will be only from block with CA rule.

The matter is the fact, that the grid should be filled with initial states to produce next evolutions. For this reason there are a number of ways to do it. As we need deterministic algorithm and minimum of memory usage, it must be simple way to fill the grid. The method, describing below is called NESW (NESW- North, East, South, West). Formally, we take a result of size multiplication $l_{b_i} * w_{b_i}$ of each block b_i and start filling it cyclically from the South-West corner of block with the bits of received number, moving to the North till the edge of block or almost filled cell, then moving to the East, South and West, repeating conditions. This method also reminds spiral moving to the center of the grid of the block. The whole process is demonstrated on the Fig. 1

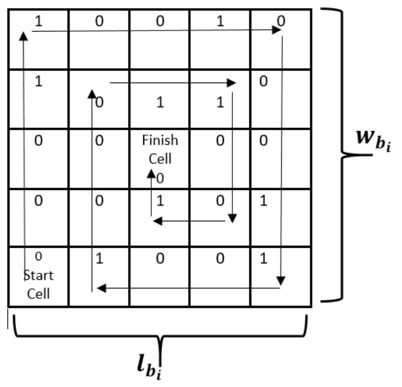


Fig. 1: b_i block grid view with NESW scheme of filling

B. Bent functions

As it mentioned before, traditional pseudorandom generators on CA use rules by Wolfram's notation, e.g. $Rule30$ or $Rule90$ and $Rule150$ together for better results. Another option is using some Boolean functions for changing states of cells in the grid, but these functions were linear. In this paper we research possiblity of using non-linear functions (bent functions as denoted before). But bent functions exist only for even number of arguments. And standard templates of neighborhood don't match for this condition. Fortunately, we can choose neighborhood template for the CA. Actually, we can choose as many templates as a number of blocks is and use different bent functions with only one restriction - possible lack of memory. Other sides of this method is flexible enough for realization.

So, we have m blocks in the grid and can choose $t_n \leq m$ templates of neighborhood and $t_f = t_n$ number of bent functions. We define target cell as a cell, which next state would be defined by bent function output.

Defining conditions for templates and bent functions:

1) number of cells in each $i-$th template. including target cell, must be even;
2) obviously number of cells is equal to number of arguments of appropriate bent function in the same block;
3) each b_i block has abstract toroidal form without size distortions;
4) target cell could be chosen anywhere in neighborhood template.

Now, using bent function $f(x_0, x_1, x_2, x_3) = x_0x_1 + x_2x_3 + x_0 + x_1$ lets see how our target cell will change on the next iteration (other cells should be changed also, but we want to check the output of function).

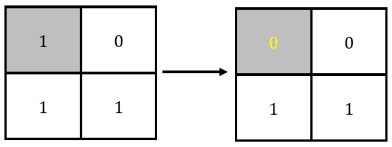


Fig. 2: How bent function works

C. Result

As we defined before the first block is our clasical CA with its own rule, but not all the rules produce statistically strong sequences. To improbe this scenario, we propose the second block in the grid, which would be producing by bent function. After needed number of evolutions block $b_1 XOR$ing with block b_2 bit by bit, with appropriate size of sequence. So far, it can be concluded, that resulting sequence in block b_1 would be much more statistically strong. Thus, we can increase the set of rules, which produces strong, random and flexible sequences of bits.

D. Repeating

As our algorithm is deterministic, that leads to repeating evolution of CA with the same input and initial states. For this aim it can be generated abstract key to pass it through the channel for checking identity. So, the key K may be interpreted like the following sequence of bits. We use symbol | to mark concatenation.

$$K = p|q|l_{b_1}|w_{b_1}|...|l_{b_m}|w_{b_m}|t_{v_1^1}|t_{v_1^2}|...|t_{v_m^1}|t_{v_m^2}|V_1|...|V_m|T$$

where $t_{v_i^j}$ is the size of template parallelogram, including all possible cells in the template. And V_i are sent like the sequence of bits, where each bit, if it is 1 than cell is in the template, and when it is 0, than out of template.

Here are numbers of cells, instead of which should placed 1 or 0. T is bit sequence for determining all of bent functions, using in algorithm and other needed meta information. Thus, in spite of the fact that we must send such a long sized key, we can put some data in the boundary cells, placed around our main grid in order to save correct transition of state in case of software realization.

III. Test results

Our algorithm was tested with the help of NIST Test Suite, which developed for testing RNG and PRNG. The process of test involved: generation sequences of bits by our algorithm (1000000 bits), testing .txt file with sequence with NIST Test Suite. We tested various numbers of bent functions up to 16 arguments with different neighborhood templates appropriately, NESW method of filling the grid with initial states. Results of tests are averaged for all the experiments.

Head parameter of NIST Test Suite, showing the quality of sequence is $P-value$, which shows the difference between testing sequence and random sequence. The test statistic is used to calculate a $P-value$ that sumarizes the strength of the evidence against the null hypothesis. For these tests, each $P-value$ is the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested, given the kind of non-randomness assessed by the test. If a $P-value$ for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. A $P-value$ of zero indicates that the sequence appears to be completely non-random. A significance level (α) can be chosen for the tests. If $P-value \geq \alpha$, then the null hypothesis is accepted; e.g., the sequence appears to be random. If $P-value < \alpha$, then the null hypothesis is rejected; e.g., the sequence appears to be non-random. The parameter α denotes the probability of the $TypeI$ error (if the data is, in truth, random, then a conclusion to reject the null hypothesis (e.g., conclude that the data is non-random) will occur a small percentage of the time) [10]. α is equal to 0.01.[10]

Here is the table (Fig. 3), which shows the difference on the NIST Test Suite between "clear" usage of $Rule30$ - $rule30$ column, with bent function of 4 arguments - $bent4$ column and bent function of 6 arguments - $bent6$ column. All bent functions were balanced: $f(x) + f(x + y)$ form, where x denotes vector of arguments, and y - random vector.

For $bent4$ variant we used the following function:

$$f(x_0, x_1, x_2, x_3) = x_0x_1 + x_1x_2 + x_2x_3$$

and as random vector was $y = 1011$

For $bent6$ variant we used the following function:

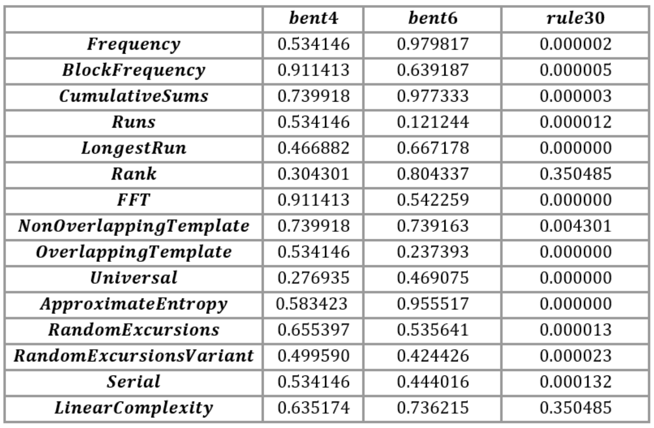
$$f(x_0, x_1, x_2, x_3, x_4, x_5) = x_0x_1 + x_2x_3 + x_4x_5$$

and as random vector was $y = 101110$

There will be represented results of average tests on about 100 rules without bent functions and with them of 4 and 6 arguments of Proportion criteria (Fig. 4) and P-value criteria (Fig. 5)

IV. Conclusion

In spite of the fact, that not all the rules in classical CA can't be strongly recommended for generating pseudorandom sequences and numbers, we can find a way to increase statistical features up to hundred times with only using bent functions without loss of velocity and simplicity. Thus, the main advantage of this idea is that the set of using rules enlarges and can be used in a different way with results near needed distribution.



	bent4	***bent6***	***rule30***
Frequency	0.534146	0.979817	0.000002
BlockFrequency	0.911413	0.639187	0.000005
CumulativeSums	0.739918	0.977333	0.000003
Runs	0.534146	0.121244	0.000012
LongestRun	0.466882	0.667178	0.000000
Rank	0.304301	0.804337	0.350485
FFT	0.911413	0.542259	0.000000
NonOverlappingTemplate	0.739918	0.739163	0.004301
OverlappingTemplate	0.534146	0.237393	0.000000
Universal	0.276935	0.469075	0.000000
ApproximateEntropy	0.583423	0.955517	0.000000
RandomExcursions	0.655397	0.535641	0.000013
RandomExcursionsVariant	0.499590	0.424426	0.000023
Serial	0.534146	0.444016	0.000132
LinearComplexity	0.635174	0.736215	0.350485

Fig. 3: Statistical results of P-value for 3 variants of $Rule30$

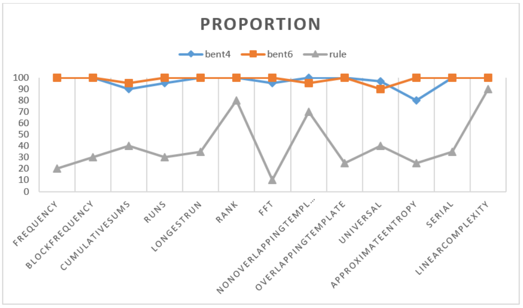


Fig. 4: Graph of difference between average test results (Proportion)

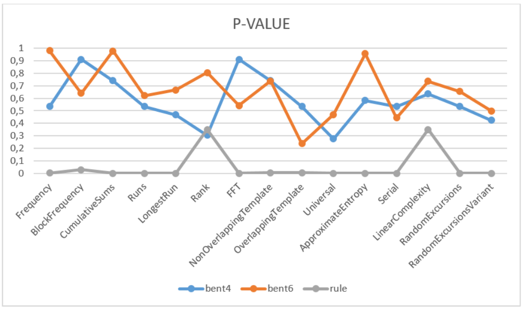


Fig. 5: Graph of difference between average test results (P-value)

V. References

Acknowledgment

The research is supported by the grant of the Russian Science Foundation (Project 17-71 -20077)

References

[1] Stephen Wolfram "Random sequence generation by cellular automata", *Advances in Applied Mathematics*, 1986

[2] Andrew Ilachinski "Cellular Automata: A Discrete Universe", *World Scientific*, 2001.

[3] M. Tomassini, M. Sipper, and M. Perrenoud "On the generation of high-quality random numbers by two-dimensional cellular automata" *IEEE Transactions on Computers*, 49(10):1146 - 1151, Oct 2000

[4] M. Tomassini, M. Perrenoud "Cryptography with cellular automata", *Appl. Soft Comput.*, 1(2):151 - 160, 2001

[5] Mads Haahr "True random number service" www.random.org, 1998

[6] M. Tomassini, M. Sipper, M. Zolla, M. Perrenoud, "Generating high-quality random numbers in parallel by cellular automata", *Future Gener.Comput. Syst., 16(2 - 3):291 - 305*, December 1999.

[7] V. B.Kudryavtsev, A.S.Podkolzin, "Cellular automata", *Intellectual systems* 1 - 4(10):657 - 692, 2006

[8] Ferguson, Niels; Schneier, Bruce; Kohno, Tadayoshi, "Chapter 9: Generating Randomness", *Cryptography Engineering: Design Principles and Practical Applications.*, Wiley Publishing, Inc. 2010

[9] Richard Brent "Uniform random number generators for supercomputers", 1992

[10] A. L. Rukhin, "A statistical test suite for random and pseudorandom number generators for cryptographic applications", *U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology*, rev. edition, 2010

[11] M. Tomassini, "Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time", *Springer*, 2005

[12] Toru Sasaki, Hiroyuki Togo, Jun Tanidaa and Yoshiki Ichiokab "Stream cipher based on pseudo-random number generation using optical affine transformation", *Applied Optics*, 39(14):2340 - 6 Âů June 2000

[13] Moore, Edward F "Gedanken-experiments on Sequential Machines", *Automata Studies, Annals of Math. Studies. Princeton, N.J.: Princeton University Press*, (34): 129 - 153, 1956

[14] Weisstein, Eric W. "von Neumann Neighborhood", *MathWorld–A Wolfram Web Resource*, http://mathworld.wolfram.com/vonNeumannNeighborhood.html

[15] Weisstein, Eric W. "Moore Neighborhood" *MathWorld–A Wolfram Web Resource*, http://mathworld.wolfram.com/MooreNeighborhood.html

[16] Alberto Dennunzio, Enrico Formenti, Julien Provillard Non-uniform cellular automata:Classes, dynamics, and decidability *Journal of Information and Computation*, Elsevier, 2012

[17] Wolfram S. "Cellular Automat" *Los Alamos Science* 9: 2 - 21, 1983

[18] Dobbertin H., Leander G. , "A survey of some recent results on bent functions" *Sequences and their applications.* , SETA, 2004.

[19] N. N. Tokareva, "Bent functions and their generalizations", *Prikl. Diskr. Mat.*, 2009, supplement 2, 5 - 17

[20] Carlet C., "On the higher order nonlinearities of Boolean functions and S-boxes, and their generalizations " *The Fifth Int. Conf. on Sequences and Their Applications* , SETA, 2008 P. 345 - 367 (Lecture Notes in *Comput. Sci.* V. 5203).