ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2009. No. 7(95)
ELEKTRONIKA IR ELEKTROTECHNIKA

AUTOMATION, ROBOTICS
T125
AUTOMATIZAVIMAS, ROBOTECHNIKA

Analysis of Unpredictable Cryptographic Pseudo-random Number Generator based on Non-linear Dynamic Chaotic System

A. Čitavičius, A. Jonavičius
Department of Electronics and Measurements Systems, Kaunas University of Technology,
Studentų str. 50, LT-51368 Kaunas, Lithuania, phone: +370 37 300539; e-mail: Algimantas.Citavicius@ktu.lt

Introduction

Random number generators (RNG) are gaining more and more interest due, in particular, to the increasing usage of cryptography, where they represent a critical point [1]. However, designing a good RNG is a difficult task. So usage of pseudo random number generators (PRNG) for cryptography seems to be perspective. However, it is important that the numbers used to generate cryptographic keys are not just seemingly random; they must be as much unpredictable as possible. In [2] we presented some original construction of cryptographic secure PRNG based on certain construction of non-linear dynamic chaotic system. The constructed generator has a complex structure and thus has a considerable good backward and forward unpredictability feature. This cryptographic secure PRNG consists of two main parts:

1. The auxiliary generator (AG), based on two linear congruential generators;
2. The non-linear dynamic chaos system (NLDCS).

As the NLDCS the generator of the formula

$$x_{t+1} = (1+\beta)(1+1/\beta)^{\beta} \cdot x_t(1-x_t)^{\beta} \tag{1}$$

was proposed. Here β is an integer in the range $1 \le \beta \le 4$, $x_t \in X = [0,1)$. The set X can be interpreted as a set of float numbers in computer presentation. The theoretical background for this PRNG construction is presented in [2]. To perform an analysis of the generator we used MATLAB/SIMULINK software. A functional model was developed using SIMULINK and results were processed using MATLAB.

Many test suites have been developed in the last years in order to check the quality of a RNG or PRNG. They are known as statistical tests for randomness, while, in fact, they are tests for non randomness. They analyze a sequence, assuming that it has been randomly generated, and try to refuse this hypothesis looking for some pattern. They also have to be interpreted in a statistical way, i.e., they are not pass/fail tests, but they say that the tested generator can be considered random or not, only with a certain probability.

For the research the SP 800-22 test suite from National Institute of Standard and Technology (NIST) [3] was chosen. This suite is composed of several different well known tests, each of them is applied to the same sequence of *n* bits (the NIST suggests $n = 10^6$) and gives a P-value i. e. the probability that the sequence under test is random. 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.

Zeros and ones bits generation

The probability density function of the pseudo random bits must be uniform, so it is necessary to establish a certain level in order to decide for which x_n a zero or a one is generated. If we have the distributions that have symmetrical probability density functions, the answer is clear: choosing the mean of the x_n values will assure the generation of the same numbers of bits according to the following formula [4]:

$$b_n = \begin{cases} 0, & x_n \le \bar{x}, \\ 1, & x_n > \bar{x}, \end{cases} \tag{2}$$

where $\bar{x}$ denotes the mean value, b_n – the bit generated by the *n*-th iteration of proposed PRNG.

The next way to get zeros and ones is to evaluate the median of the value set. Median should be the most suitable statistical characteristic that may split the bits domain into two equally filled sub-domains with equal number of zeros and ones bits. The criterion for generation of a one or a zero bit becomes in this case:

$$b_n = \begin{cases} 0, & x_n \le med, \\ 1, & x_n > med, \end{cases} \tag{3}$$

where *med* denotes the median of the values generated by the NLDCS.

After some research a different way for generation of ones and zeros have been chosen, because the quality of produced bits was not satisfied. PRNG based on NLDCS generates a set of float numbers. So according to IEEE 754 standard we used double precision floating point format to encode float numbers to binary format. The structure of this format is shown in Fig 1.

Sign (1 bit)	Exponent (11 bit)	Mantissa (52 bit)

Fig. 1. IEEE 754 Double Floating Point Format

Using this encoding procedure we get some advantages. First, from one float number we can get 64 bits, therefore, we increase bit generation performance. Second, as we will show below, bits encoded by this schema pass NIST test suite. Actually, to get more randomness, from 64 bits we removed the highest 16 bits (2 bytes), because they represent sign, exponent and mantissa highest bits and they usually are not random. So we get 48 bits, which we use for testing purposes.

Statistical tests for randomness

The considered NIST Test Suite is a statistical package consisting of 15 tests that were developed to test the randomness of arbitrarily long binary sequences produced by either hardware or software based cryptographic RNGs or PRNGs. These tests focus on a variety of different types of non-randomness that could exist in a sequence. The 15 tests are [3]:

1. The Frequency (Monobit) Test,
2. Frequency Test within a Block,
3. The Runs Test,
4. Test for the Longest-Run-of-Ones in a Block,
5. The Binary Matrix Rank Test,
6. The Discrete Fourier Transform (Spectral) Test,
7. The Non-overlapping Template Matching Test,
8. The Overlapping Template Matching Test,
9. Maurer's “Universal Statistical” Test,
10. The Linear Complexity Test,
11. The Serial Test,
12. The Approximate Entropy Test,
13. The Cumulative Sums Test,
14. The Random Excursions Test, and
15. The Random Excursions Variant Test.

For each statistical test, a set of P-values corresponding to the set of sequences is produced. For a fixed significance level a certain percentage of P-values are expected to indicate failure. For example, if the significance level is chosen to be 0,01 (i.e., α = 0,01), then about 1% of the sequences are expected to fail. A sequence passes a statistical test whenever the P-value $\geq \alpha$ and fails otherwise. The parameter α denotes the significance level that determines the region of acceptance and rejection. We use α = 0,01 as suggested by NIST. A statistical test is formulated to test a specific null hypothesis (H_0). The null hypothesis under test is that the sequence being tested is random against the alternative hypothesis (H_1) for which the sequence is not random. We can commit two errors:

1. Reject H_0 when the sequence is generated by a perfect random generator (Type I error);
2. Accept H_0 when the sequence is generated by a generator that is non random (Type II error).

Testing NLDCS as a pseudo random bit generator

As mentioned above, to create a model of our generator we used MATLAB/SIMULINK software. The functional diagram of our generator is shown in Fig 2.

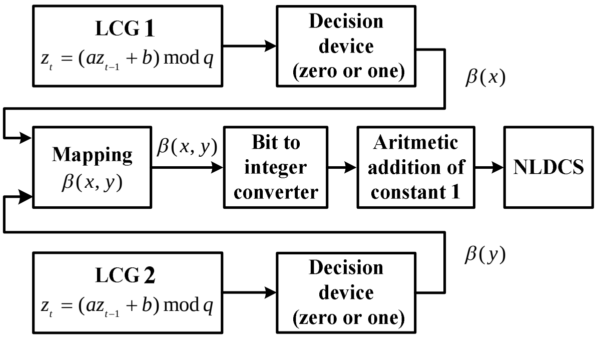
LCG 1
$z_t = (az_{t-1} + b) \bmod q$
Decision device (zero or one)
$\beta(x)$
Mapping $\beta(x, y)$
$\beta(x, y)$
Bit to integer converter
Aritmetic addition of constant 1
NLDCS
LCG 2
$z_t = (az_{t-1} + b) \bmod q$
Decision device (zero or one)
$\beta(y)$

Fig. 2. Functional diagram of our PRNG

Two linear congruential generators (LCG1 and LCG2) represent AG. Linear congruential generator generates random numbers, which are divided by modulo *m* and we get bits according to formula (2). After that, each *n* bit sequence is mapped to *n* bit of other generator. Because parameter β value must be integer value in the range $1 \leq \beta \leq 4$, mapped bits are then transformed to decimal code by using “Bit to integer converter”, and additionally the constant equal to 1 is added. Next, β value is transmitted to NLDCS. The generation function of formula (1) with different values of β is shown in Fig. 3 and the histogram of produced numbers is shown in Fig. 4.

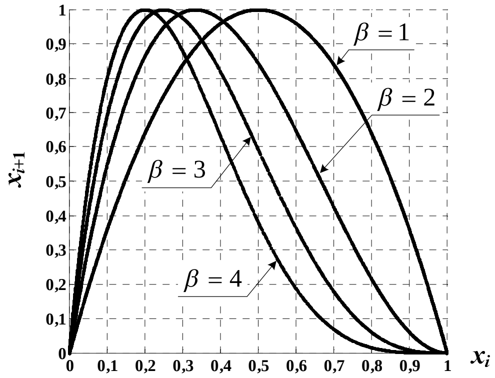

Fig. 3. The generation function of formula (1) with different values of β, $x_i \in (0; 1)$

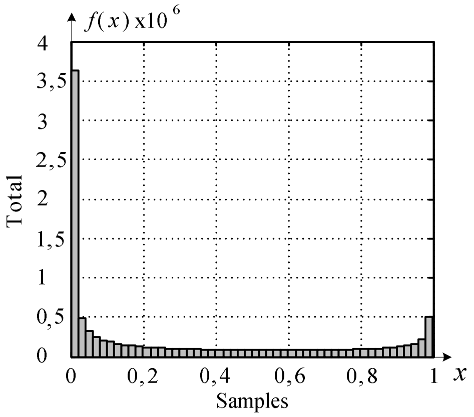

Fig. 4. The histogram of the produced numbers generated by our PRNG, $x_i \in (0; 1)$

One can see from the histogram (Fig. 4) that numbers produced by generator are not uniformly distributed. To flatten histogram we decided to choose narrower interval to generate float numbers. The interval [0,2; 0,8] was chosen and after the narrowing histogram domain, we obtained a more flattened histogram as shown in Fig. 5.

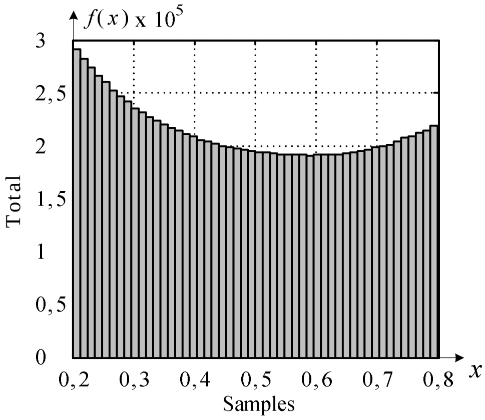

Fig. 5. The histogram of the produced numbers generated by our PRNG, $x_i \in$ [0,2; 0,8]

One way to increase randomness is to choose a narrower interval. However, the time needed to get the numbers is longer in this case. To check the quality of our PRNG and get a rough impression of generator's performance we created an appropriate the two-dimensional visualization of the numbers it produces [5]. This visualization is usually cold bit map and it is presented in Fig. 6. For chosen interval [0,56; 0,6] lattice structure of our generator is not visible. So, the key question now is to decide, if the PRNG with narrowed domain passes the tests for randomness.

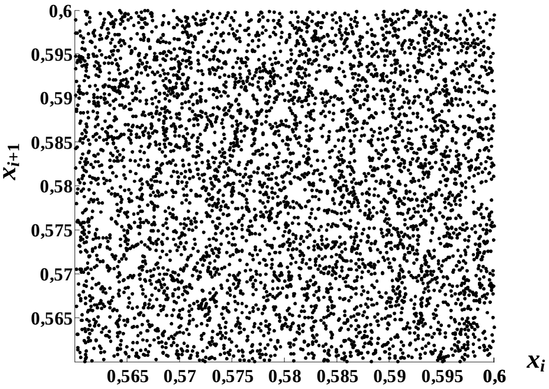

Fig. 6. The bit map of the float numbers generated by our PRNG, $x_i \in$ [0,56; 0,6]

An alternative way to flatten the histogram is to use some kind of approximation function. However, this way is more sophisticated. According to the results of our experiments, choosing the interval of our PRNG's generated numbers in the range [0,2; 0,8] and using described above float number to bit encoding schema we get suitably random bits that pass NIST statistical test suite.

The usual way to test a random or pseudo-random number generator is to generate a sequence of n bits and analyze it with the NIST test suite. As recommended by NIST we generated n=10^6 bits and analyzed them using NIST statistical test suite [3]. The results of one n=10^6 bits sequence is presented in Table 1. One can see that all P-values are greater than α value (we used α = 0,01 as suggested by NIST). So this bit sequence passes NIST statistical test suite.

Table 1. Results of NIST Statistical Tests. Number of sequences N=1, size of sequence n=10^6 bits

Nr.	Statistical Test	P-value
1	Frequency Monobit	0,875743
2	Block Frequency (M =128)	0,465315
3a	Cumulative Sums (Forward)	0,647463
3b	Cumulative Sums (Backward)	0,793273
4	Runs	0,788043
5	Longest Runs of Ones	0,672935
6	Binary Matrix Rank (M = 32)	0,908927
7	Spectral DFT	0,301862
8	Non-periodic Templates (m=1)	0,490959
9	Overlapping Templates	0,812349
10	Maurer's Universal(L=7, Q=1280)	0,718202
11	Approximate Entropy	0,046560
12	Random Excursions (x = +4)	0,242211
13	Random Excursions Variant ($x = -3$)	0,028403
14	Linear Complexity (M = 1024)	0,301757
15a	Serial (m = 16)	0,873099
15b	Serial (m = 16)	0,960443

Actually, some of the tests of the suite compute two (the *Cumulative Sum* and the *Serial* tests) or more (*Non-Overlapping Template Matching*, *Random Excursion* and *Random Excursion Variant*) P-values, however, it is very common considering only one of them.

However, we have always to remember about the possibility to commit a Type I or Type II error [6]. For example a periodic (and thus, *non random*) generator always passes the Frequency Test if the number of 1s and of 0s in the period is balanced. To overcome the impasse, a more intensive test is necessary, involving a number N of different sequences generated by the RNG under test. NIST suggests a strategy to check, if the P-values are uniformly distributed in the interval [0; 1] [3]. NIST recommends to conduct a chi-square test on the P-values, dividing the interval [0;1] into 10 sub-intervals. This tests the uniformity of the P-values. The degree of freedom is 9 in this case. If we define F_i as the number of occurrences that the P-value is in the i-th interval, then χ^2 statistic is

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - s/10)^2}{s/10} \quad , \qquad (4)$$

where s denotes the sample size. Following NIST recommendation and choosing significance level equal to 0,01% (i.e. 0,0001), the acceptance region of statistics will be $\chi^2 \le 33{,}72$.

To get more reliable results we tested N=100 different sequences with n=10^6 bits. Table 2 shows the results of a chi-square test on the P-values as described above.

Table 2. Results of sequences uniformity check

Nr.	**Statistical Test**	χ^2 **value**
1	Frequency Monobit	17,6
2	Block Frequency (M =128)	14,6
3a	Cumulative Sums (Forward)	10,8
3b	Cumulative Sums (Backward)	14,2
4	Runs	6,4
5	Longest Runs of Ones	5,8
6	Binary Matrix Rank (M = 32)	10,8
7	Spectral DFT	6,6
8	Non-periodic Templates (m=1)	14,2
9	Overlapping Templates	4,4
10	Maurer's Universal(L=7,Q =1280)	5,4
11	Approximate Entropy	11,8
12	Random Excursions (x = +4)	10,3
13	Random Excursions Variant (x=-3)	13,4
14	Linear Complexity (M = 1024)	7,2
15a	Serial (m = 16)	7
15b	Serial (m = 16)	10

Having looked at the table, we can see that evaluated chi-square values do not exceed limiting value of $\chi^2 \leq 33{,}72$.

Conclusions

The aim of this paper is to demonstrate the performance of our PRNG based on NLDCS. The investigation showed that to get pseudo random bits, additional requirements have to be met. These are:

1. To generate float numbers in interval [0,2; 0,8];

2. To use double precision floating point format for encoding float numbers to binary format and remove the highest 16 bits.

The described PRNG construction passed NIST statistical test suite. It can be used as a good source for random cryptographic key generation.

References

1. **Menezes A., Oorschot van P., Vanstone S.** Handbook of Applied Cryptography // CRC Press. – 1996. – 780 p.
2. **A. Čitavičius, A. Jonavičius, S. Japertas.** Unpredictable cryptographic pseudo-random number generator based on non-linear dynamic chaotic system // Electronics and Electrical Engineering. – 2007. – No. 7(79). – P. 29–32.
3. **National Institute of Standards and Technology.** A statistical test suite for random and pseudorandom number generators for cryptographic applications // Special publication 800-22. – Revision 1. – August 2008. – 131 p.
4. **R. Ursulean.** Reconsidering the Generalized Logistic Map as a Pseudo Random Bit Generator // Electronics and Electrical Engineering. – 2004. – No. 7(56). – P. 10–13.
5. **Janke W.** Pseudo Random Numbers: Generation and Quality Checks // In Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms. – John von Neumann Institute for Computing. – Jülich. – 2002. – NIC Series. – Vol. 10. – P. 447–458.
6. **Pareschi F., Rovatti R., Setti G.** Second-level NIST Randomness Tests for Improving Test Reliability // IEEE International Symposium on Circuits and Systems. – New Orleans, 2007. – P. 1437–1440.

Received 2009 04 09

A. Čitavičius, A. Jonavičius. Analysis of Unpredictable Cryptographic Pseudo-random Number Generator based on Non-linear Dynamic Chaotic System // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. – No. 7(95). – P. 25–28.

Most of pseudo-random number generators are unsuitable for cryptographic applications despite their statistical and correlation characteristics due to their backward and forward predictability. In this paper we presented an analysis of unpredictable cryptographic pseudo-random number generator based on non-linear dynamic chaotic system. We showed that we can get 48 bits from one float point and these bits generally passed NIST statistical test suite, therefore, we recommend to use this generator for cryptographic keys generation. Ill. 6, bibl. 6 (in English; summaries in English, Russian and Lithuanian).

А. Читавичюс, А. Йонавичюс. Анализ непредсказуемого криптографического генератора псевдослучайных чисел, основанного на нелинейной динамической хаотической системе // Электроника и электротехника. – Каунас: Технология, 2009. – № 7(95). – С. 25–28.

Большенство псевдослучайных генераторов чисел являются неподходящими для криптографии, несмотря на их статистические и корреляционные особенности из-за их прямой и возвратной предсказуемости. Представлен анализ генератора псевдослучайных чисел, основанного на нелинейной динамической хаотической системе. Было показано, как с одного реального числа можно получить 48 бит. Этот генератор прошёл NIST статистические тесты, поэтому рекомендуется использовать его для генерации криптографических ключей. Ил. 6, библ. 6 (на английском языке; рефераты на английском, русском и литовском яз.).

A. Čitavičius, A. Jonavičius. Nenuspėjamas kriptografinis pseudoatsitiktinių skaičių generatorius netiesinės dinaminės chaotinės sistemos pagrindu // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2009. – Nr. 7(95). – P. 25–28.

Dėl galimo tiesioginio ir atgalinio sekos narių numatymo dauguma pseudoatsitiktinių skaičių generatorių nėra tinkami naudoti kriptografijoje, nepaisant jų statistinių ir koreliacinių savybių. Atlikta pasiūlyto saugaus atsitiktinių skaičių generatoriaus, sukurto dinaminės chaotinės sistemos pagrindu, analizė. Parodyta, kaip iš generuojamo vieno realaus skaičiaus galima gauti 48 bitus, kurie apskritai imant tenkina NIST statistinius testus. Siūloma šį generatorių naudoti kriptografiniams raktams generuoti. Il. 6, bibl. 6 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).