NANOSYSTEMS: PHYSICS, CHEMISTRY, MATHEMATICS, 2017, 8 (5), P. 600–605

Random number generator for cryptography

R. Soorat[1,2], K. Madhuri[1], A. Vudayagiri[1]

[1]R. C. Bose centre for cryptology and security, Indian Statistical Institute, Kolkata, India

[2]School of Physics, University of Hyderabad, Hyderabad 500046, India

rsoorat@gmail.com

PACS 07.05.Hd **DOI 10.17586/2220-8054-2017-8-5-600-605**

One key requirement for many cryptograhic schemes is the generation of random numbers. Sequences of random numbers are used at several stages of a standard cryptographic protocol. One simple example is a Vernam cipher, where a string of random numbers is added to message string to generate encrypted code. $C = M \oplus K$. It has been mathematically shown that this simple scheme is unbreakable if key K is as long as M and is used only once. The security of a cryptosystem shall not be based on keeping the algorithm secret but solely on keeping the key secret. The security of a random number generator (RNG) is related to the difficulty of predicting its future sequence values from past values. The quality and unpredictability of secret data is critical to securing communication by modern cryptographic techniques. The generation of such data for cryptographic purposes typically requires an unpredictable physical source of random data. We studied a chaotic circuit which consisted of an inductor, capacitance, diode and thus used for the BB84 protocol. We have studied both pseudo random and true random number generators and evaluated them through various tests like frequency, correlation, NIST etc.

Keywords: Hardware random number generator, cryptography, chaos, chaotic circuit.

Received: 18 September 2017

Revised: 25 September 2017

1. Introduction

A random process is a repeating process in which output is difficult to find a describable deterministic pattern. The term randomness is quite often used in statistics to signify well defined statistical properties, such as correlation. Saying that a variable is random means that the variable follows a given probability distribution; under these terms, random is different from arbitrary, because to say that a variable is arbitrary does not imply that there is such determinable probability distribution. A good RNG should work efficiently, which means it should be able to produce a large amount of random numbers in a short period of time. Random numbers are widely used in many applications, such as cryptography [1, 2], spread-spectrum communications [3], Monte Carlo numerical simulations [4], statistical analysis [5], information security, stochastic simulation, stream ciphers, ranging signal in a radar system, controlling signal in remote control, encryption codes or keys in digital communication, address codes and spread spectrum codes in code division multiple access (CDMA) and many others. So for simulations, the generation of large amounts of random numbers is crucial, and thus fast RNGs are required. RNGs are also used in the statistics to solve problems in many fields such as nuclear medicine, finance and computer graphics.

Since traditional random numbers are generated by algorithms and are essentially pseudo-random, they have potential danger in security-related fields like quantum key distribution. There are, in general, two types of generators for producing random sequences: true random number generators (TRNGs) and pseudo random number generators (PRNGs). PRNGs require some input called seeds, along with some deterministic algorithms to generate multiple pseudo random numbers. They are usually faster than TRNGs and are preferable when several random-like numbers are required. TRNGs make use of non-deterministic sources along with some post-processing functions for generating randomness. Such sources include physical phenomena such as thermal noise, atmospheric noise, radioactive decay and even coin tossing. such as electrical noises [6], frequency jitters in electrical oscillators [7] and chaotic circuits [8, 9], which can produce unpredictable random numbers of high quality yet at much lower rates than PRNGs because of the narrow bandwidth of these physical entropy sources. In addition, a number of documents exist which provide general advice on using and choosing random number sources [10–13]. Further discussions on the nature of randomness, pseudo random number generators (PRNGs), and cryptographic randomness are available from a number of sources [14–16].

Truly random numbers are the basis of many cryptographic applications like QKD, especially for the generation of keys that cannot be penetrated by hackers or other attackers it is important that the random numbers used be unpredictable. The BB84 protocol makes use of polarization states of single photons to map the bits 0, 1 of the encryption key, in two mutually unbiased bases. These bases should be random, which means we need to design a physical source for RNG to perform better than the available computer Pseudo random number generators (PRNG), but also should be compact and easy to integrate into the QKD device prototype. This requires a controller that generates four random states and its deterministic critically endangers the security of the entire protocol. For most applications, it is desirable to have fast random number generators (RNGs) that produce numbers which are as random as possible. There are different types of statistical tests that can be applied to a sequence to compare and evaluate the sequence to that of a truly random sequence [17–20].

2. Chaos based hardware random number generator

We are using a chaos based hardware random number generator which consists of a capacitor, a diode and an inductor. Fig. 1 is a simple LCR circuit built around a varactor diode. The voltage dependent capacitance acts as a nonlinear element, thus providing a chaotic oscillation. Generally, a hardware random number generator is based on sampling noise sources such as thermal noise or a reverse based diode. Different circuits have been described, but these methods are difficult, since their amplitudes are usually small and often masked by deterministic disturbances; as a result, another alternative is to use a chaotic oscillator for pseudo random generator due to its unpredictable behavior and relatively simplicity. We have built a chaotic circuit based RNG (CCRNG), based on an earlier design by T. Kuusela [22], which in turn, was built around the chaos generator of Matsumoto et al. [23]. We use chaotic circuits which are extremely simple, consisting of an inductor and a capacitor diode, the nonlinear element is the capacitor whose capacitance is varied as a function of voltage across it. The nonlinear capacitance of the diode is seen earlier in many papers. The capacitance is varied as $C(V) = C/(1 + V/\theta)\gamma$. Here, V is the voltage across the diode. If the circuit parameters and the external drive are suitably chosen, the system exhibits period doubling and chaotic behavior.

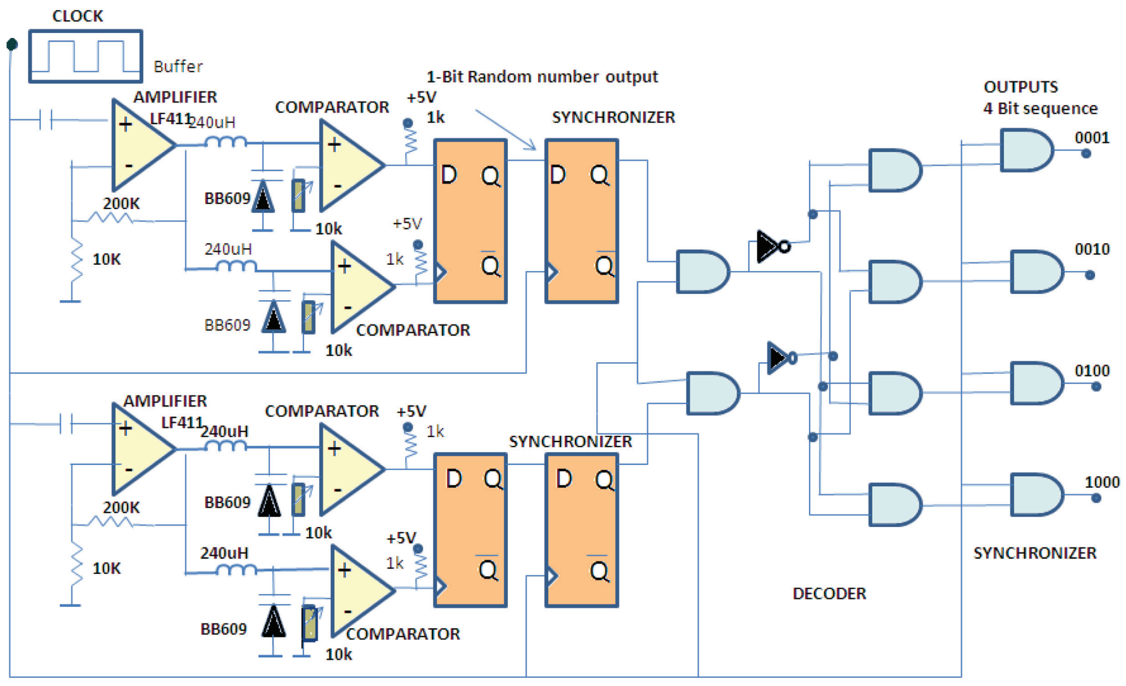

FIG. 1. Circuit diagram

The chaotic first-order differential equation and modified construction of chaotic clock Generator (CCG) by kuusela is as shown in Fig. 1. It has an external clock signal (a square wave of 500–600 KHz is used as a clock in this case). The CCG includes two identical chaotic oscillators; it consists of an inductor 240 μH and a varactor diode BB609 (nonlinear element). We are using two CCRNGs to generate a sequence of four bit structure in a random fashion, viz., 0001, 0010, 0100, and 1000. An amplifier is used to raise the signal level so that if the frequency and the amplitude of the driving clock are properly chosen, the circuit goes to a chaotic state and reliable operation is guaranteed, even the case of large tolerance. The fast operational amplifier with a large bandwidth is used as an amplifier (the LF 411 is used because of its high bandwidth and rapid response). The voltage across the capacitor diode is a random signal when the circuit exhibits period doubling and chaotic behavior. A comparator is used to convert analog signal into a digital signal (LM 311 response time is 200 ns). Even though the output of comparators is quite randomly generated, it does not generate all the possible bit sequences for all possible values. We use two comparators for each CCRNG; one of them takes a bit sample from the other. This is done by a simple D-flip flop (1st D-flip flop). To synchronize the output bit sequences with the driven clock, one more D-flip flop is used and an AND gate is used to avoid continues 1's and 0's. A decoder is used to generate four bits structures using two CCRNGs.

Using this circuit we generate several sets of random sequence of 0's and 1's and tested its behavior. Initially, we tapped the voltage at the output of the chaotic circuit and at the edge of the varactor. This data was stored in the computer as a function of time and then, its variation was computed $\frac{dv}{dt}$. Plotting $\frac{dv}{dt}$ against V, giving a phase plot Fig. 2. This showed the behavior of the generated signal. The plot on left side shows the raw data of voltage v/s time, while those on right side shown the phase plot, $\frac{dV}{dt}$ v/s V. As the frequency is changed, the circuit becomes chaotic and we can see the bifurcation as shown in Fig. 2. The data is for different clock frequencies as output of chaos base random number generator before chaos up to (C); the output of chaos base random number generator at 500 kHz (D); the output of chaos base random number generator at 650 kHz (E). It can be seen from the phase plot in Fig. 2 (A) oscillation is constant as frequency is increased as in (B) the oscillation spreads. As frequency is further increased, in the phase plot, a clear frequency doubling behavior is shown, indicated by double loops. A further increase in frequency shows a multiple frequency regime in (D) and finally the chaotic oscillation in (E). The output of varactor is fed in to the comparator which maps the signal to 1 if the voltage is above a certain threshold and to 0 if it is a below the threshold. The sequence of 0's and 1's generated by the comparator, after being synchronized to the clock pulse, are recorded by the computer and analyzed. These data are obtained at the clock speed of about 650 kHz, since this is the region when chaos circuit is giving a proper chaotic output.

Figure 3 shows two different types of distribution. The total number of 1's and 0's are shown in (a) left side. For this run, it is asymmetric since there are more 1's than 0. This happens only for a few runs and can easily be corrected by changing the threshold. The graph on right (b) is more important, since it depicts the probability of getting zero following 1, getting 1 after 1 and simply of 0 after 0 and 1 after 0. The graph shows a slight higher frequency for the occurrence of 1–1, but this is due to fact that there is a higher occurrence of 1's as opposed to 0's. However, the graph clearly shows the same frequency for 1–0 and 0–1. This means that the system does not have any preference for 1 over 0.

In addition, we studied the bit correlation of random numbers from hardware module. Correlation $c_{ij}^{00} = \langle P_i(0)P_j(0)\rangle$, where $P_i(0)$ and $P_j(0)$ are the probability of finding 0 at i-th position and j-th position respectively. Extending this calculation, we can write $c_{ij}^{kl} = \langle P_i(k)P_j(l)\rangle$ for $k = 1, 0$ and $l = 1, 0$. For an ideal case, this should all be equal and have a value 0.25. The distance is the difference between i and j.

However, for Fig. 4, the closed square indicates correlation for 0–0, open square for correlation 0–1, solid line for 1–0 and closed circle for 1–1. Correlations for 0–1 and 1–0 are exactly identical at 0.25, indicating equal probability of getting pairs 0–1 and 1–0. The probability of getting 1–1 is higher, due to a systematic bias in the module for producing more 1's than 0's. This indicates the circuit is a near perfect coin toss system. It can be noticed from Fig. 4 bit correlation that the correlation has a small distribution around a mean value. We have passed this random sequence through the NIST test, and very few of the test pass through it.

3. Psedo random number generator using LabVIEW

LabVIEW produces a double precision, floating-point number between 0 and 1, exclusively which has a uniform distribution. We have generated 50,000 random points between 0 and 1 from LabVIEW software and studied their random properties. We have plotted $\frac{dn}{dt}$ vs n which is phase and number sequence between 0 and 1. Fig. 5 phase plot $\frac{dn}{dt}$ shows a highly non periodic behavior which is a clear indication of randomness.

Figure 6 (a) is a histogram graph which is a specific visual representation of data that measures the number of incidents of 0 and 1 for a sample set. The left side shows the distribution of 1's and 0's. We observe the same frequency for 1's and 0's, which is truly random behavior. The graph on right shows the probability of 1–1 is higher than 0–0 while that of 0–1 and 1–0 are the same.

Figure 7 bit correlation test shows the same correlation value of 0.25. This is a clear indication of a very good random number generator. It can be observed from Fig. 7 that the correlation has a small distribution around a mean value. In order to investigate this, we plotted the histogram of the correlation values. This distribution is very narrow about the mean value. In addition to the correlation test, we collected a bit stream of 0's and 1's from the circuit and processed them through the NIST test suite, with most of the tests being passed.

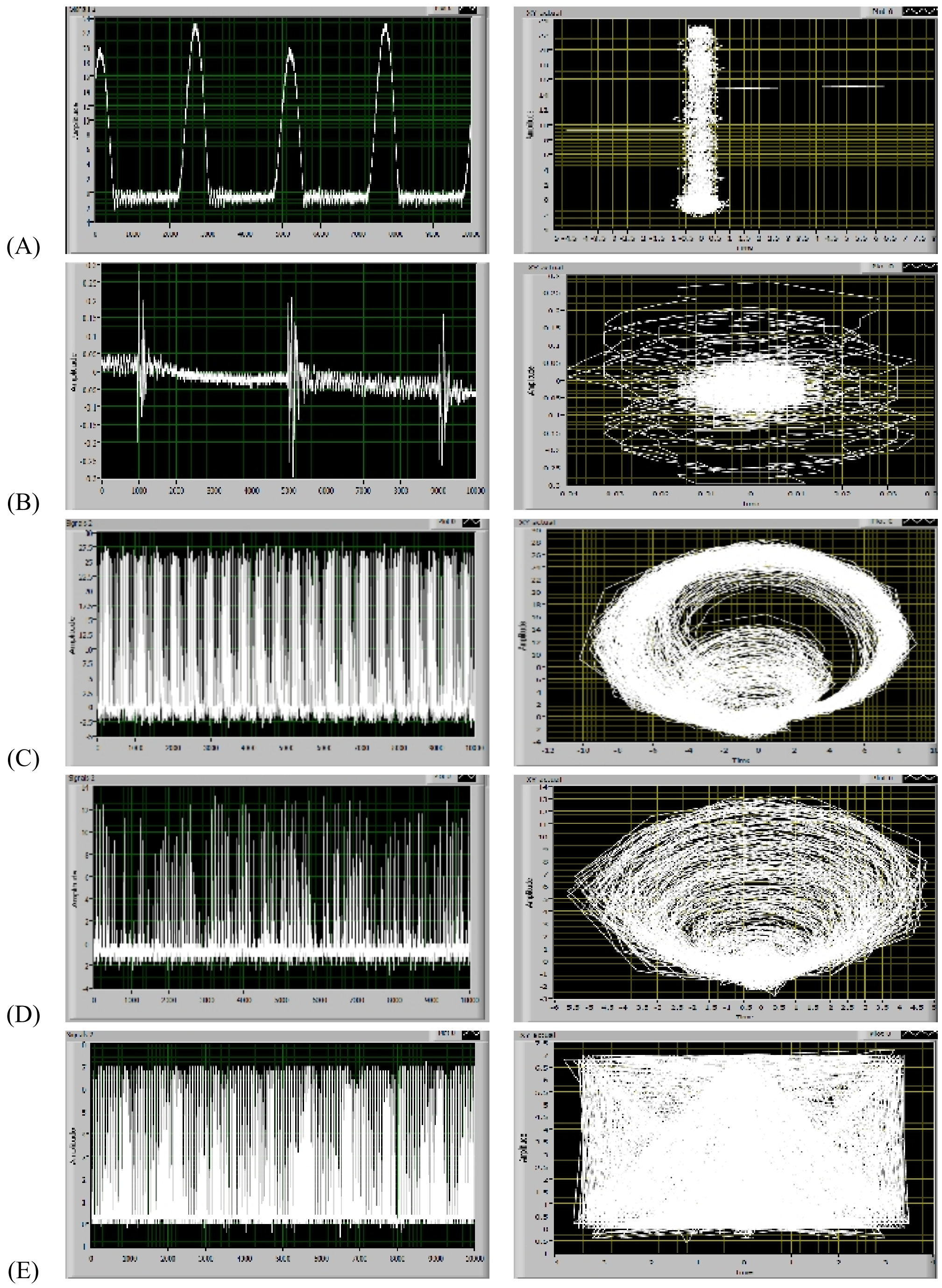

FIG. 2. Analog signal output of the chaos clock generator at different frequencies and corresponding phase plots

4. Conclusion

Since software methods only offer pseudo-random number codes, the need for other sources is important. We have therefore analyzed hardware and software based random number generators. We have compared and analyzed the randomness behavior between software and hardware random number generators. Although the correlation test gives very good results, the NIST test showed that some of them test pass through it. We have implemented this method for quantum key distribution based on BB84 protocol.

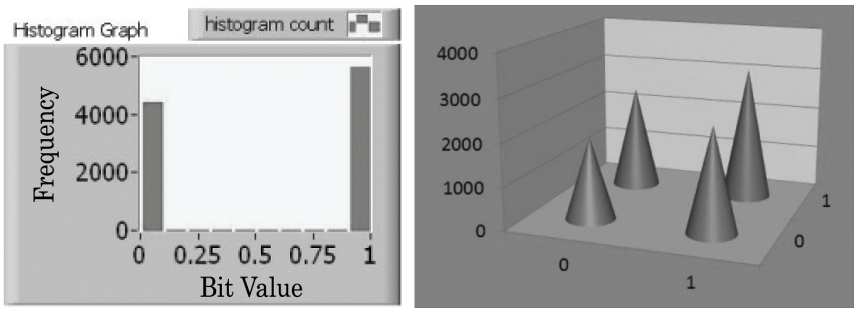

FIG. 3. (a) Histogram distribution of 0 and 1 (b) bit correlation

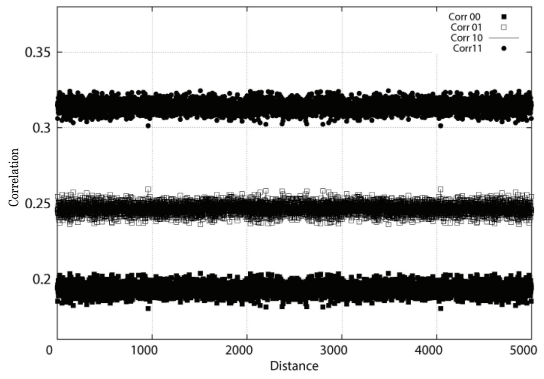

FIG. 4. Successive bit correlation of hardware module

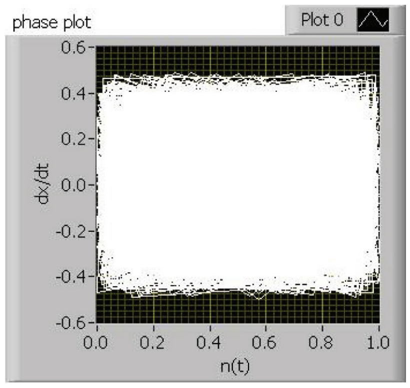

FIG. 5. Phase plots

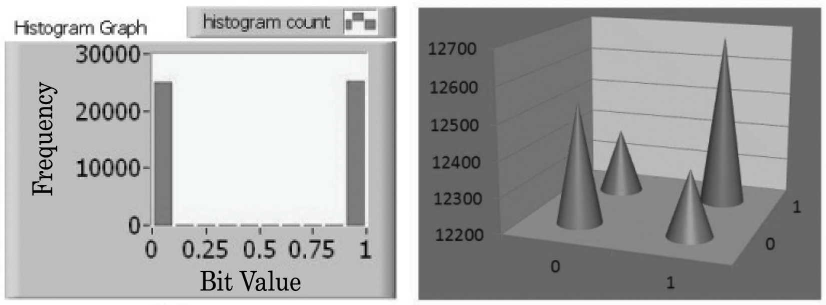

FIG. 6. Histogram distribution of 0 and 1 (b) bit correlation

Acknowledgements

This work was supported by department of information technology, Government of India. We thanks to Sunitha for her support.

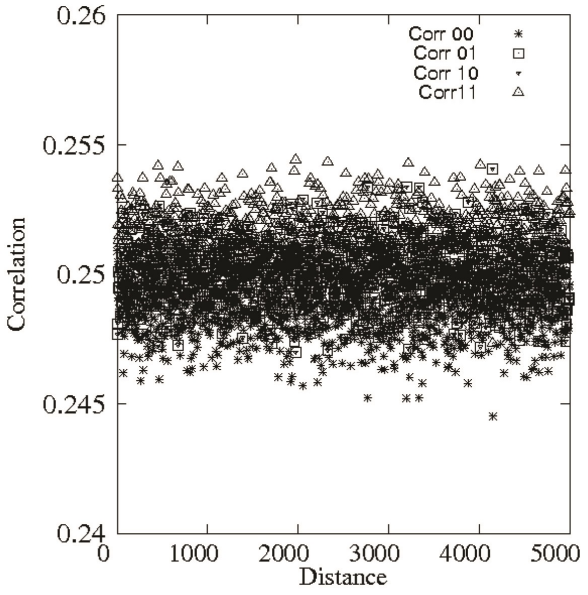

FIG. 7. Successive bit correlation of software module

References

[1] Bennett C.H., Brassard G. Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, New York, 1984, P. 8.

[2] Burrows J.H. *Security requirements for cryptographic modules*. FIPS, Gaithersburg, 2001, 56 p.

[3] Pickholtz R.L., Schilling D.L., Milstein L.B. Theory of spread-spectrum communications-a tutorial. *IEEE Trans. Commun.*, 1982, **30** (5), P. 855–884.

[4] Metropolis N., Ulam S. The Monte Carlo method. *J. Am. Stat. Assoc.*, 1949, **44** (247), P. 335–341.

[5] Nazarathy M., Newton S.A., et al. Realtime long range complementary correlation optical time domain reflectometer. *J. Lightwave Technol.*, 1989, **7** (1), P. 24–38.

[6] Petrie C., Connelly J. A noise-based IC random number generator for applications in cryptography. *IEEE Trans. Circ. Syst. I Fundam. Theory Appl.*, 2000, **47** (5), P. 615–621.

[7] Bucci M., Germani L., et al. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC. *IEEE Trans. Comput.*, 2003, **52** (4), P. 403–409.

[8] Stojanovski T., Kocarev L. Chaos-based random number generators-Part I: analysis. *IEEE Trans. Circ. Syst. I Fundam. Theory Appl.*, 2001, **48** (3), P. 281–288.

[9] Stojanovski T., Pihl J., Kocarev L. Chaos-based random number generators-Part II: Practical realization. *IEEE Trans. Circ. Syst. I Fundam. Theory Appl.*, 2001, **48** (3), P. 382–385.

[10] Krhovjak J. *Cryptographic random and pseudorandom data generators*. Brno, 2009.

[11] Matthews T. Suggestions for random number generation in software. *RSA Laboratories' Bulletin*, 1996, **1**.

[12] Ellison C. Cryptographic Random Numbers. IEEE P1363 Working Draft, 1997, 11 p.

[13] Buchovecká S. Analysis of a True Random Number Generator. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2012.

[14] Knuth D. *The Art of Computer Programming: Volume 2, Seminumerical Algorithms*, Addison-Wesley, 1981.

[15] Menezes A., van Oorschot P., Vanstone S. *Handbook of Applied Cryptography*, CRC Press, 1996.

[16] Goldreich O. *Foundations of Cryptography Fragments of a Book*, 1995, 290 p.

[17] Rukhin A. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. *U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2000*, 152 p.

[18] Marsaglia G. *Random Number CDROM, with The Diehard Battery of Tests of Randomness*. Florida State University, 1985.

[19] Good I.J. The serial test for sampling numbers and other tests for randomness. *Proc. Cambridge Philos. Soc.*, 1953, **47**, P. 276–284.

[20] Baron M., Rukhin A.L. Distribution of the Number of Visits For a Random Walk. *Communications in Statistics: Stochastic Models*, 1999, **15**, P. 593–597.

[21] Spitzer F. *Principles of Random Walk*. Princeton: Van Nostrand, especially, 1964, 269 p.

[22] Kuusela T. Random Number Generation Using a Chaotic Circuit. *J. Nonlinear Sci.*, 1993, **3**, P. 445–458.

[23] Matsumoto T., Chua L.O., Tanaka S. Simmplext chaotic nonautonomous circuit. *Phys. Rev. A*, 1984, **30**, 1155.