Turkish Journal of Science & Technology
20(1), 55-61, 2025

Research Paper
https://doi.org/10.55525/tjst.1528213

PCG-Generated Randomness: A NIST Analysis of 100-Million Bits

Zülfiye Beyza METİN[1*], Fatih ÖZKAYNAK[2]
[1,2] Department of Software Engineering, Faculty of Tecnology, Fırat University, Elazığ, Turkey
[*1] zbmetin@firat.edu.tr, [2] ozkaynak@firat.edu.tr

(Geliş/Received: 05/08/2024; Kabul/Accepted: 20/11/2024)

Abstract: The generation of random numbers is crucial for various applications, including cryptography, simulation, sampling, and statistical analysis. Cryptography utilizes random numbers to secure communication through the generation of encryption keys, thereby safeguarding sensitive information from unauthorized access. This study aims to evaluate the randomness and suitability of the Permuted Congruential Generator (PCG) algorithm for cryptography applications, through testing its generated random numbers using the National Institute of Standards and Technology (NIST) statistical tests. A novel method is proposed for generating 100 million bits using the PCG algorithm. The generated random numbers are then subjected to NIST testing. The results indicate that the PCG-generated random numbers pass most relevant statistical tests and comply with the standards of randomness necessary for cryptography. In conclusion, the PCG algorithm is demonstrated to be a robust, dependable, and appropriate random number generator for cryptography and other applications requiring random numbers.

Key words: Randomness, PCG, NIST, Security.

PCG Tarafından Oluşturulan Rastgelelik: 100 Milyon Bitlik Bir NIST Analizi

Öz: Rastgele sayıların üretilmesi kriptografi, simülasyon, örnekleme ve istatistiksel analiz gibi çeşitli uygulamalar için çok önemlidir. Kriptografi, şifreleme anahtarlarının oluşturulması yoluyla iletişimi güvence altına almak için rastgele sayıları kullanır ve böylece hassas bilgileri yetkisiz erişime karşı korur. Bu çalışma, Permuted Congruential Generator (PCG) algoritmasının kriptografi uygulamaları için rastgeleliğini ve uygunluğunu, üretilen rastgele sayıları Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) istatistiksel testlerini kullanarak test ederek değerlendirmeyi amaçlamaktadır. PCG algoritmasını kullanarak 100 milyon bit üretmek için yeni bir yöntem önerilmiştir. Üretilen rastgele sayılar daha sonra NIST testine tabi tutulmuştur. Sonuçlar, PCG tarafından üretilen rastgele sayıların ilgili istatistiksel testlerin çoğunu geçtiğini ve kriptografi için gerekli rastgelelik standartlarına uygun olduğunu göstermektedir. Sonuç olarak, PCG algoritmasının kriptografi ve rastgele sayı gerektiren diğer uygulamalar için sağlam, güvenilir ve uygun bir rastgele sayı üreteci olduğu gösterilmiştir.

Anahtar kelimeler: Rastgelelik, PCG, NIST, Güvenlik.

1. Introduction

Encryption refers to the process of converting plain text into a coded format that is unreadable to anyone except those who possess the decryption key [1]. This process utilizes mathematical algorithms to scramble the data, making it unreadable to anyone without the corresponding decryption key. Encryption is a crucial component of modern computer security and is employed in a variety of applications, such as online shopping, banking, and government communications [2]. Additionally, it is utilized to protect data in transit, such as when it is transmitted over the internet or through a network. In recent years, encryption has become increasingly important as an increasing amount of personal and sensitive information is stored and transmitted online. As a result, governments and organizations worldwide have implemented encryption policies to protect their data and communications [3].

There are various types of encryption, such as symmetric and asymmetric encryption. Symmetric encryption uses the same key for both encryption and decryption processes, while asymmetric encryption involves a public key for encryption and a private key for decryption. The effectiveness of an encryption system relies on the keys being genuinely random and distinct. Predictable or reused keys compromise the encryption's security [4].

Random numbers are crucial for generating encryption keys used in both symmetric and asymmetric encryption [5]. In symmetric encryption, a random key encrypts and decrypts data [6]. Predictable or reused keys weaken encryption [7]. Asymmetric encryption uses mathematically related public and private keys. The private key decrypts what the public key encrypts, ensuring security [8]. Random numbers also support digital signature and key agreement protocols by creating unique values for authentication [9].

[*] Corresponding author: zbmetin@firat.edu.tr. ORCID Number of authors: [1]0000-0003-4376-7319, [2]0000-0003-1292-8490

Random numbers are generated through a process known as random number generation [10]. There are several methods for generating random numbers, including pseudorandom number generation, hardware random number generation, True Random Number Generation, and Permuted Congruential Generators (PCG).

PCG is a family of random number generators that employs a combination of a linear congruential generator (LCG) and a permutation function to produce a sequence of high-quality random numbers [11]. LCG is a simple and efficient method for generating pseudorandom numbers. It utilizes a mathematical formula to generate a sequence of numbers based on an initial value (seed) and a set of parameters (multiplier, increment, and modulus) [12]. However, LCGs are known to have certain weaknesses, such as poor statistical properties and a limited period of repetition. To overcome these limitations, PCG employs a permutation function to scramble the output of the LCG, creating a new sequence of numbers that possess superior statistical properties and an extended period of repetition. The permutation function can be a simple operation, such as a bitwise shift, or a more complex operation, such as a cryptographic hash function. PCG also incorporates a feature called "streams" which allows for the generation of multiple independent sequences of random numbers from the same seed without interference. This can be useful in applications requiring multiple random numbers, such as the generation of random keys for encryption or the simulation of random events in a computer game [13]. One of the key advantages of PCG over other random number generators is its ability to produce high-quality random numbers that pass various statistical tests and have a long period of repetition [14]. Additionally, it has a small memory footprint, making it suitable for embedded systems and other resource-constrained environments.

The NIST is applied to numbers produced by PCG as it is a widely accepted set of statistical tests for evaluating the randomness and quality of random numbers. The NIST is designed to assess the suitability of a random number generator for cryptographic purposes the statistical properties of the numbers generated by the generator [14]. The NIST includes a variety of tests that cover different aspects of random number generation, such as tests for uniformity, independence, and patterns in the numbers. These tests are designed to detect any biases or weaknesses in the random number generator that could potentially be exploited by an attacker [15]. By applying the NIST to numbers generated by PCG, it is possible to obtain a quantitative measure of the quality of the random numbers produced by the generator. The results of the tests can be used to determine if the numbers produced by PCG meet the standards for cryptographic use and to identify any potential weaknesses or biases that need to be addressed.

In this study, we analyze the random numbers generated by the PCG algorithm to determine how many of the NIST statistical tests they pass and to what extent they meet the randomness standards required for cryptographic applications. To achieve this, 100 random bit strings, each consisting of one million bits, were generated using the PCG method. In order to apply the NIST test to these generated bit strings, a method based on changing the stream values has been proposed.

The main cont are as follows:

- The examination of the application of the NIST to numbers generated by PCG, a widely-used random number generator.
- The proposal of a method for applying the NIST test to bit strings generated by the PCG method by changing the stream values.
- The assessment of the quality of random numbers generated by the PCG method through the application of the NIST and the identification of any potential biases or weaknesses in the generator.
- The demonstration of the suitability of PCG for cryptographic purposes based on the results of the NIST.

2. Materials and Methods

In this study, the seed and stream values of the PCG algorithm are defined as parameters and 32-bit coins and rolls are produced in the output. In the binary conversion process, ‘T’ variables are assigned a value of 0 and ‘H’ variables are assigned a value of 1. As a result of this binary conversion process, random numbers are obtained. The proposed method is illustrated in a block diagram representation in Figure 1.

2.1. The Proposed Method

The proposed method generates a single-bit random number each time it is executed. In this study, we obtained a series of 100 one million-bit random numbers. The algorithm steps used to achieve this are as follows:

1. Define the PCG function with seed=42 and stream=21 values. (This choice was made because we obtained the most successful results as a result of trying all possible combinations.)

2. Generate one million random numbers with the specified parameters.
3. After each one million random numbers are generated, change the stream value.
4. Repeat Step 2 (100 times).

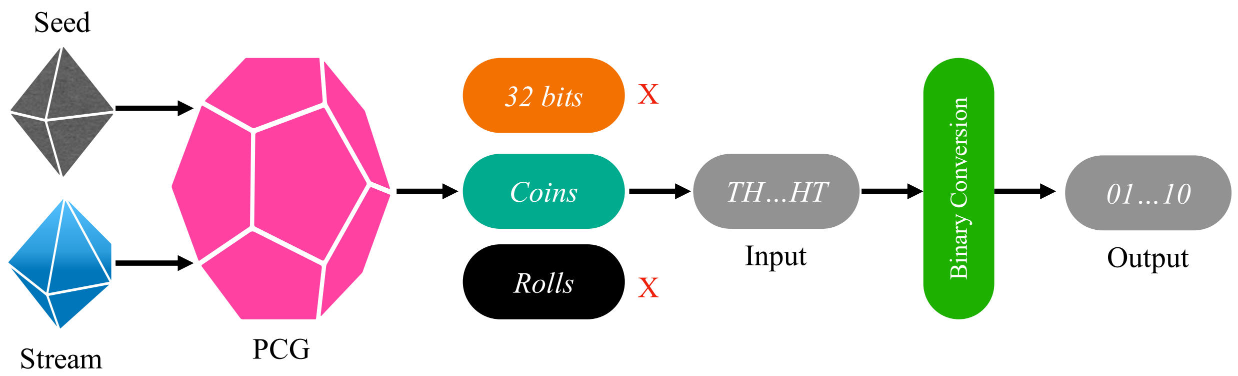

Figure 1. Block diagram of the recommended method.

2.2. Evaluating Randomness

Random and pseudo-random numbers are necessary for many cryptographic applications. Before using random numbers as inputs in various cryptographic protocols, it is important to verify that they conform to the definition of randomness. The NIST family of tests, developed by the US National Institute of Standards and Technology, is commonly used to evaluate the reliability of random number generators. NIST SP 800-22 includes 15 individual tests [16]. In this study, the NIST statistical test suite was used to assess the randomness of 100 one-million-bit random number sequences generated using the PCG algorithm. The NIST statistical test suite includes 15 different tests, as described below [17].

- The frequency test is a method for measuring whether the number of occurrences of the values 0 and 1 in a sequence are approximately equal, which is an expected characteristic in a real random number generator. The values 0 and 1 in the bit string should be relatively close to one another. If the value generated using the test equations, p, is greater than 0.01, the sequence is considered to be random.
- The block frequency test evaluates the proportion of 0s and 1s within blocks of m bits in a sequence. It focuses on observing the frequency of 1s in each block of m bits.
- The runs test analyzes the total number of runs in a sequence, where a run is a continuous series of the same bit. This test assesses the transitions between 0s and 1s to determine if the bit sequence changes states too slowly or too quickly.
- The longest runs of ones test focuses on observing the longest consecutive sequence of 1s in a string. The only parameter of the test is the block length (m). The array is divided into n blocks of m bits and the longest sequence of 1s in each block is checked. The frequencies of the obtained values are compared to expected values and a deviation is checked for. The block length and the number of blocks are decided by considering the length of the array.
- The rank test assesses the ranks of smaller submatrices extracted from the entire sequence. The objective of this test is to determine if there is any linear dependence among the subsequences of the original sequence.In the test, the sequence is split into M x M-bit matrices and the rank of each generated matrix is calculated. The frequencies of the ranks of the matrices formed in order are calculated, compared to the expected frequency and checked for a significant deviation.
- The discrete Fourier transform test is used to test whether there is any dominant harmonic in the sequence that would prevent randomness. It identifies periodic patterns in the sequence that differ from what would be expected in a truly random sequence.

- The overlapping template of all ones test is based on observing the frequency of occurrence of a predefined target sequence. An m-bit window is used to search for an m-bit sample. If the searched sample is not found, scanning continues by shifting the window one bit.
- Maurer's Universal Statistical Test checks whether the given array is sufficiently compressed. Excessive compression of the array indicates that the array is far from random.
- The non-overlapping template matching test is employed to detect non-periodic samples generated by a generator. In this test, as well as in the subsequent Overlapping Pattern Matching Test, an m-bit window is utilized to search for an m-bit sample. If the sample is not found, the window is shifted by one bit and the search continues. In the event that the sample is located, the window is repositioned to the first bit following the identified sample and the search continues.
- The serial test centers on the frequency of potential m-bit overlapping samples in the entire array.
- The approximate entropy test, similar to the Serial Test, focuses on the occurrence frequency of possible m-bit overlapping patterns within the sequence. This test aims to compare the observed frequency of overlapping blocks of two consecutive lengths with the expected frequency in a truly random sequence.
- The cumulative sum test aims to determine whether the cumulative sums of partial subsequences in the tested sequence are disproportionately small or large in comparison to the expected value of a known random sequence. This test can be considered as a cumulative total random walk, with random walk excursions in random sequences typically around zero.
- The random excursions test focuses on the number of precisely K visit cycles in the cumulative total random walk. The cumulative sum is obtained from the partial sums of the random walk, after arranging the sequence of (0,1) as (-1, +1). A random walk cycle consists of a sequence of steps of a certain length, beginning at a location that is considered random, until it becomes a complete cycle. The purpose of this test is to determine the number of visits to a particular state resulting from the expected bias in the random sequence during this cycle.
- The random excursions variant test evaluates the frequency of specific states being visited during a cumulative random walk. The goal is to identify deviations in the number of visits to these states from what is expected in a typical random walk.
- The linear complexity test examines the complexity of the bit string by examining the length of the Linear Feedback Shift Register (LFRS). The test assesses whether the array is sufficiently complex for randomness. Arrays are considered Linear Feedback Shift Register (LFSR) outputs, and the array is deemed not complex enough to be random if the smallest LFSR capable of forming the array is small.

Statistical tests of randomness are commonly known as hypothesis tests. The results of the tests are concluded as successful or unsuccessful based on the acceptance or rejection of the hypothesis. In NIST tests, the hypothesis is typically set to $\alpha=0.01$ [18]. It is desired that the calculated p-values for each statistical test are greater than the value of α.

3. Experimental Setup

In this study, 100 sequences of one million bits of random numbers were generated utilizing the PCG algorithm. Each of these sequences underwent NIST testing, with a significance level of $\alpha=0.01$. The methodology employed for testing the random number generation method based on PCG is depicted in Figure 2.

The NIST statistical test suite was chosen because it is a widely recognized standard for evaluating the randomness and quality of random number generators, especially in cryptographic contexts. The 15 tests it includes assess various statistical properties of randomness, such as uniformity, independence, and the absence of patterns. These properties are crucial for cryptographic applications, where predictable patterns could lead to vulnerabilities. By using the NIST suite, we ensure a comprehensive evaluation that addresses both short-term randomness and long-term unpredictability.

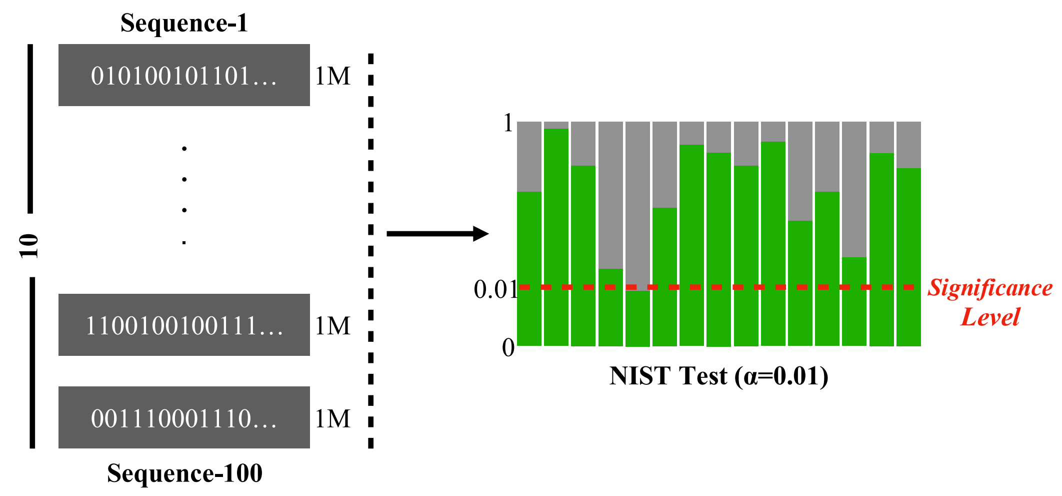

Figure 2. The testing structure utilized in the study.

3.1. NIST Results

In the NIST test, the p-value is used to express the likelihood of a hypothesis being true. In the context of hypothesis testing, the p-value is a numerical value between 0 and 1, and its magnitude is crucial in determining the validity of the hypothesis [19]. In this study, 100 random number sequences of 1 million bits were generated using the PCG algorithm, and each of these sequences underwent NIST testing. The significance level was set at α=0.01. The p-values obtained from each test are presented in Table 1, for samples randomly selected from the 1-million-bit random number sequences.

Table 1. Results from NIST tests on random numbers generated.

Test	Sequence-1 (P)	Sequence-2 (P)	Sequence-3 (P)	Sequence-4 (P)	Sequence-5 (P)
Frequency	0.525815	0.025124	0.487777	0.158261	0.618781
Block Frequency	0.378385	0.848552	0.132612	0.748528	0.237045
Runs	0.278555	0.572323	0.653277	0.234722	0.372731
Longest Runs Of Ones	0.311016	0.575607	0.532123	0.658764	0.142112
Rank	0.348308	0.402712	0.548777	0.570032	0.6401218
Dft	0.281435	0.445888	0.225511	0.488345	0.220654
Overlapping Template Of All Ones	0.126013	0.606571	0.165242	0.524711	0.207075
Universal Statistical	0.727131	0.167532	0.473012	0.513728	0.725812
Non Overlapping Template Matching	0.286852	0.005787	0.121352	0.116552	0.003462
Serial	0.464433	0.224327	0.351381	0.575431	0.721515
Approximate Entropy	0.6165204	0.232114	0.833664	0.428243	0.161452
Cumulative Sums	0.8202184	0.018425	0.862662	0.077256	0.583116
Random Excursions	0.226583	0.122675	0.113715	0.124321	0.235313
Random Excursions Variant	0.446532	0.382112	0.275228	0.105312	0.318621
Linear Complexity	0.286852	0.135210	0.253642	0.751383	0.613712

The NIST test results of 100 1 million-bit random numbers generated with the PCG algorithm indicate that the numbers are highly random and suitable for cryptographic use. The results demonstrate that the numbers pass most of the statistical tests in the NIST suite, which include tests for randomness, uniformity, and independence. The p-values for many tests were well above the commonly used significance level of 0.01, indicating that the numbers are highly unlikely to be non-random. Overall, these results indicate that the PCG algorithm is a reliable and efficient method for generating high-quality random numbers for cryptographic applications. The performance of the generated 100 random number sequences of 1 million for each test in the NIST suite is presented in Figure 3. In Figure 3, those indicated in gray represent the random number sequences that passed the test, while those in red represent the sequences of numbers that failed the test.

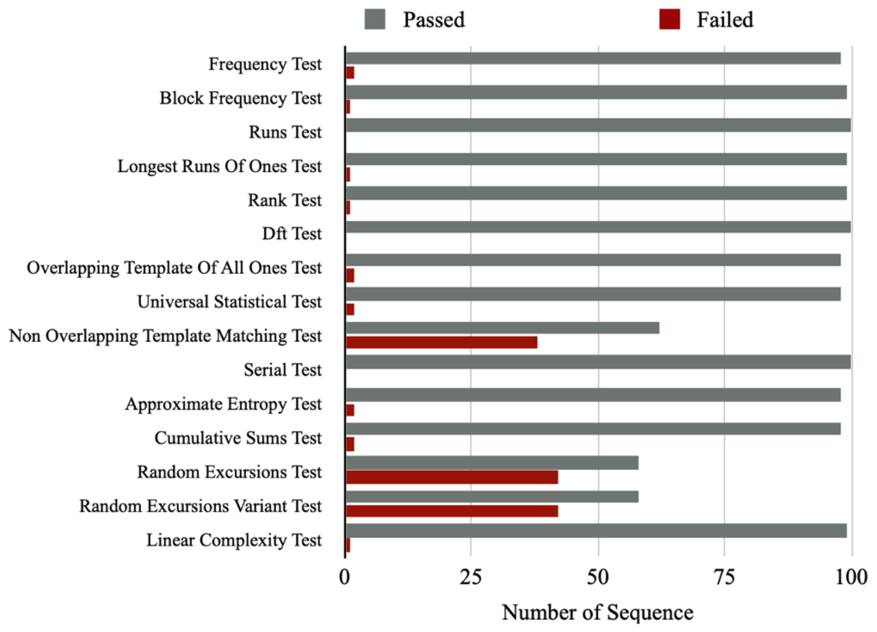

Figure 3. Performance of the nist tests on 100 million random numbers.

4. Discussion

The evaluation of random number sequences generated by the PCG algorithm using the NIST test suite revealed an overall success rate of 90.94%. However, three tests—the Non-Overlapping Template Matching Test, Random Excursions Test, and Random Excursions Variant Test—showed noticeably weaker performance. These failures could be linked to specific characteristics of the PCG algorithm, such as its structure or periodicity, which may not align well with the statistical patterns these tests are designed to detect.

The Non-Overlapping Template Matching Test, which looks for specific bit patterns within the sequence, highlighted potential limitations in the algorithm's bit generation process. This failure could indicate that the PCG algorithm occasionally produces predictable patterns, which may be detrimental in applications requiring high levels of randomness, such as cryptography. Similarly, the Random Excursions and Random Excursions Variant Tests, which analyze the number of state visits within a random walk, exposed possible deficiencies in long-range dependencies or correlations in the sequences generated by PCG. This suggests that while the algorithm performs well in the short term, it may exhibit periodic behavior or correlations over longer sequences, which is undesirable for cryptographic applications where long-term unpredictability is essential.

While 30 out of the 100 sequences passed all 15 NIST tests, the failures in these specific tests raise important questions about the algorithm's suitability for cryptographic purposes. For cryptography, the randomness quality must be exceptionally high to avoid vulnerabilities. These results suggest that the PCG algorithm, in its current form, may not yet provide the level of randomness required for secure cryptographic applications without further refinement. The PCG algorithm demonstrates several advantages, such as efficiency and good performance in the majority of other statistical tests. These benefits indicate that with targeted improvements, it may still be a viable candidate for cryptographic use. Future work will focus on addressing the identified weaknesses, particularly in relation to the failed tests, and further investigating the security implications. Additionally, we plan to support the analysis with alternative test suites, such as TestU01, to provide a broader evaluation of the algorithm's performance and ensure its robustness for cryptographic applications.

5. Conclusion

This study presents an approach for generating 100 random numbers of 1 million bits using the PCG algorithm. The numbers were tested using the NIST, and the results indicate that the numbers with high p-values. While some test results were quite low, like Non-Overlapping Template Matching Test, Random Excursions Test and Random Excursions Variant Tests. However, 90.94% success rate was obtained on the NIST test of random numbers generated by the proposed method. This approach has several advantages over existing methods for generating random numbers. The PCG algorithm is relatively simple and easy to implement, yet it produces high-quality random numbers that are suitable for cryptographic applications. Furthermore, the PCG algorithm has a small memory footprint and can generate a large number of random numbers quickly. In conclusion, this approach can be a valuable tool for researchers and practitioners in fields that require high-quality random numbers, such as cryptography and computer simulations. The results of this study support the use of the PCG algorithm for generating random numbers and provide a solid foundation for further research in this area.

Acknowledgments

This work was supported in part by Fırat University Scientific Research Projects Coordination Unit with project number TEKF.24.27 and. The Scientific and Technological Research Council of Turkey under Grant 121E600 and 122E337.

References

[1] Akashi N, Nakajima K, Shibayama M, Kuniyoshi Y. A mechanical true random number generator. New J Phys 2022; 24(1): 249-252.
[2] Basharat I, Azam F, Muzaffar AW. Database security and encryption: A survey study. Int J Comput Appl 2012; 47(12): 28-34.
[3] Bouillaguet C, Martinez F, Sauvage J. Practical seed-recovery for the PCG pseudo-random number generator. IACR Trans Symmetric Cryptol 2020; 2020(3): 175-196.
[4] Dong L, Chen K. Cryptographic protocol. Security analysis based on trusted freshness. Springer, 2012.
[5] Johnston D. Random number generators—principles and practices. In Random Number Generators—Principles and Practices. De Gruyter Press, 2018.
[6] Kohlbrenner P, Gaj K. An embedded true random number generator for FPGAs. Proc ACM SIGDA Int Symp Field Program Gate Arrays 2004; 71–78.
[7] Naik RB, Singh U. A review on applications of chaotic maps in pseudo-random number generators and encryption. Ann Data Sci 2022; 1-26.
[8] O'Neill ME. PCG: A family of simple fast space-efficient statistically good algorithms for random number generation. ACM Trans Math Softw, 2014.
[9] Paar C, Pelzl J. Understanding cryptography: a textbook for students and practitioners. Springer Sci Bus Media, 2009.
[10] Panda M. Performance analysis of encryption algorithms for security. In Proc Int Conf Signal Process Commun Power Embed Syst (SCOPES). IEEE, 2016; 278-284.
[11] Pareschi F, Rovatti R, Setti G. Second-level NIST randomness tests for improving test reliability. In Proc IEEE Int Symp Circuits Syst (ISCAS) 2007; 1437-1440.
[12] Patnala TR, Jayanthi D, Majji S, Valleti M, Kothapalli S, Karanam SR. A modernistic way for key generation for highly secure data transfer in ASIC design flow. In Proc Int Conf Adv Comput Commun Syst (ICACCS). IEEE, 2020; 892-897.
[13] Ponuma R, Amutha R. Compressive sensing based image compression-encryption using novel 1D-chaotic map. Multimed Tools Appl 2018; 77: 19209-19234.
[14] Sharma S, Gupta Y. Study on cryptography and techniques. Int J Sci Res Comput Sci Eng Inf Technol 2017; 2(1): 249–252.
[15] Soto J. Statistical testing of random number generators. In Proc Natl Inf Syst Secur Conf 1999; 12.
[16] Thambiraja E, Ramesh G, Umarani DR. A survey on various most common encryption techniques. Int J Adv Res Comput Sci Softw Eng 2012; 2(7).
[17] Zaman JKMS, Ghosh R. Review on fifteen statistical tests proposed by NIST. J Theor Phys Cryptogr 2012; 1: 18-31.
[18] Zaru A, Khan M. General summary of cryptography. J Eng Res Appl 2018; 8(02): 68-71.
[19] Zhu S, Ma Y, Lin J, Zhuang J, Jing J. More powerful and reliable second-level statistical randomness tests for NIST SP 800-22. In Adv Cryptol – ASIACRYPT. Springer 2016; 307-329.