Journal homepage: https://itce.com.ua/en

Information Technologies and Computer Engineering **Vol. 21, No. 3. 2024**

Article's History: Received:04.09.2024 Revised: 20.11.2024 Accepted: 26.12.2024

UDC 004.056 10.63341/vitce/3.2024.53

Comparative analysis of the results of pseudorandom number generators for digital noise generation

Oleksandr Isakov*

Postgraduate Student
Lviv Polytechnic National University
79013, 12 Stepan Bandera Str., Lviv, Ukraine
https://orcid.org/0009-0007-4632-9492

Stepan Voitusik

PhD, Associate Professor
Lviv Polytechnic National University
79013, 12 Stepan Bandera Str., Lviv, Ukraine
https://orcid.org/0000-0003-4234-3303

Abstract. The paper presents the results of a study of the characteristics of five different pseudorandom number generators for use in digital noise generation problems used to mask signals in cybersecurity. The relevance of the study was conditioned by the growing need for high-quality masking methods that provide both effective performance and reliability of randomness, which is important for protecting confidential information in modern digital systems. The purpose of the study was to compare the PCG, Xoshiro128++, WELL512a, Mersenne Twister, and KISS algorithms in terms of their performance, statistical randomness, and ability to effectively mask a useful signal with noise. The performance of the algorithms was evaluated using BenchmarkDotNet. Standard NIST, Dieharder, and TestU01 tests were used to check the quality of sequence randomness. For the generated noise, a spectral analysis was performed using the power spectral density value. The masking efficiency was calculated by the signal-to-noise ratio, the results of the autocorrelation function, and the noise spectrogram. The results of the study showed that PCG and KISS are the most productive in terms of speed, which makes them attractive for applications where fast random sequence generation is important. WELL512a and PCG demonstrated the highest randomness quality, consistently passing all statistical tests. Analysis of the spectral noise distribution showed that all generators provide a uniform power distribution before filtering, and after filtering, the noise is successfully limited in the high-frequency range. The signal-to-noise ratio for all algorithms was about -13.6 dB, which indicates similar efficiency in noise masking. Autocorrelation analysis confirmed a low correlation for all generators outside of zero lag, which is important for maintaining the quality of randomness in long sequences. The practical value of the study lies in the selection of the optimal pseudorandom number generator for noise reduction problems in cybersecurity. The results obtained provide recommendations for choosing algorithms based on their speed and randomness, which will ensure a high level of information protection in digital systems

Keywords: information security; noise characteristics; statistical randomness tests; spectral analysis; performance tests; signal noise

Introduction

Modern cybersecurity systems require continuous improvement of information security methods due to the growing threat of cyber-attacks, in particular, encryption, simulated protection, masking, and scrambling. One of the most effective approaches to preventing information leakage through speech channels is to use a pseudorandom number generator (PRNG) to generate digital noise, which ensures that the useful signal is jammed outside the controlled zone. However, the key issue remains the choice of the optimal algorithm for generating pseudorandom numbers, which will provide not only high speed, but also high noise quality in terms of statistical randomness, spectral characteristics, and autocorrelation level. Choosing the right PRNG for a specific task is important, since each

Suggested Citation:
Isakov, O., & Voitusik, S. (2024). Comparative analysis of the results of pseudorandom number generators for digital noise generation. *Information Technologies and Computer Engineering*, 21(3), 53-64. doi: 10.63341/vitce/3.2024.53

*Corresponding author

algorithm has its own characteristics that affect the performance and quality of digital noise.

Pseudorandom number generation is an important aspect of many cybersecurity systems, as it is these algorithms that ensure the reliability of cryptographic functions, digital noise generation, and other forms of information protection. There are a large number of studies devoted to the comparative analysis of various PRNG algorithms in terms of their performance (Syafalni *et al*., 2022) and the quality of randomness (Vennos *et al*., 2021), however, the choice of the optimal algorithm remains an open question. The need to create digital noise with appropriate characteristics that would meet modern cybersecurity requirements makes this research relevant and timely. The need to filter evenly distributed values leads to changes in PRNG results and requires additional noise verification in the context of digital signal processing.

Many papers focused on analysing the randomness of numbers generated by PRNG algorithms using well-known test packages such as NIST STS, Dieharder, and TestU01. For example, research by P. L'ecuyer (2017) has become one of the most influential, as it contains a comprehensive randomness analysis of various PRNGs using TestU01, which has become the basis for further comparative studies in this area. This approach helped to identify algorithms that provide high-quality number generation, and provided valuable recommendations for selecting PRNG for various applications.

However, it is not just the quality of randomness that is crucial for choosing a PRNG. For example, Z. Hu (2020) and K. Mandal (2022) examined cryptographic pseudorandom generators for specific applications such as encryption and simulated protection. The researchers emphasised the importance of PRNGs performance, and their ability to provide an appropriate level of security that meets the requirements of modern cryptographic protocols. These studies were an important step in understanding that it is important for cryptography to consider not only randomness, but also speed, especially in high-load environments.

O.V. Isakov & S.S. Voitusik (2023) performed a similar analysis, namely NIST statistical tests and frequency analysis of generated noise based on Additive Fibonacci Generators. Based on the results of the conducted studies, the importance of analysing the characteristics of noise after its filtering was determined. Additional statistical tests were performed in this study to obtain more accurate results. In addition, the analysis of the speed and overlap of noise with the real signal helped to draw conclusions about the effectiveness of noise, and not just about the results of frequency analysis of the generated noise.

F. Yu *et al*. (2021), M.S. Feali (2023) and S. Li *et al*. (2024) presented high-performance PRNGs implemented on FPGAs that successfully passed statistical tests. These studies demonstrated the effectiveness of hardware implementation for specific digital noise generation tasks where both performance and randomness are important. Such approaches confirm the feasibility of using specialised hardware solutions to improve the reliability and speed of PRNGs, which are used to create digital noise in conditions of large amounts of data.

Overall, previous research confirmed the importance of a comprehensive approach to selecting PRNGs for digital noise generation. While randomness testing is a prerequisite for determining the reliability of an algorithm, performance and noise spectral characteristics are also important factors to consider when creating efficient and secure systems. Unlike existing studies, which are usually limited to one aspect of PRNG estimation, this study is based on an integrated approach and considers not only speed, but also various aspects of noise quality, such as randomness, spectral characteristics, and resistance to filtration and de-noising processes.

The purpose of the study was to determine the optimal algorithm for generating pseudorandom numbers for creating digital noise, which is used in information security systems, by comparing the speed, quality of generation, and spectral characteristics of digital noise of various PRNGs. This approach will not only increase the level of security of cyber systems, but also ensure the stability of their operation in conditions of rapid growth in the volume of transmitted information. The conducted experiments provided conclusions about the optimal choice of PRNGs for generating digital noise, which makes this study relevant for improving information security systems in the context of modern cybersecurity challenges.

Materials and Methods

Five well-known algorithms were selected for the study: PCG, Xoshiro128 ++ , WELL512a, Mersenne Twister, and KISS. All algorithms were analysed using methods such as speed measurement, statistical randomness tests, spectral analysis, and SNR (Signal-to-Noise Ratio) estimation after noise was superimposed on the useful signal.

Pseudorandom number generators (PRNG) were chosen in such a way that they included algorithms known for their simplicity, application, statistical characteristics, period, and ease of hardware implementation. The selected characteristics were met by the following set of generators:

- PCG (Permuted Congruential Generator) – high-speed and efficient generator designed to provide high-quality randomness;
- Xoshiro128 ++ – popular generator with fast execution, often used in scientific and technical applications;
- WELL512a (Well Equidistributed Long-period Linear) – generator with well-distributed random numbers that has a long period (Panneton *et al*., 2006);
- Mersenne Twister – generator with a very long service life and high randomness quality, widely used in numerous industries;
- KISS (Keep It Simple Stupid) – simple and fast generator.

BenchmarkDotNet – used to measure the performance of each PRNG algorithm in milliseconds, which allowed comparing the performance of different algorithms. Each

algorithm generated 268,435,456 values (2**33 bits) of the "uint" type, and the average generation time was written to the results file. This helped to evaluate the effectiveness of each PRNG for scenarios that require high speed.

Standardised NIST, Dieharder, and TestU01 tests were used to assess the quality of randomness of PRNG results. They helped to determine whether the generated numbers meet the requirements of randomness by evaluating various statistical properties of the sequences. The initial input coefficients were the same in all algorithms, except for those determined by the algorithm itself, and the array of initial values predicted in Mersenne Twister. Testing included evaluating frequency, block frequency, longest run length, matrix ranks, and pattern analysis. All tests used a p-value threshold from 0.01 to 0.99, where deviation from this range was considered unacceptable.

To evaluate the effectiveness of noise, a conversation recording from the Common Voice Delta Segment 19.0 (Sound data sets, n.d.), which was superimposed on the noise generated by each of the algorithms. This allowed estimating the level of the signal-to-noise ratio and the spectral properties of the superimposed noise.

Sequences of pseudorandom numbers with a sampling rate of 32kHz were generated for each PRNG. Each result was then filtered to limit the spectral range of noise within the specified frequency limits. Filtering helped to reduce excess power in high-frequency components and make noise more suitable for masking tasks.

The Fast Fourier Transform (FFT) was used to analyse the spectral characteristics of the generated noise. The power spectral density (PSD) was plotted for uniformly distributed white and filtered noise at specified frequencies (0-4,000 Hz), which allowed estimating the power distribution of noise over frequencies.

After superimposing the noise on the useful signal, the SNR for each PRNG was calculated. The equation was used to find the SNR:

$$SNR = 10 * Log_{10}\left(\frac{P_{signal}}{P_{noise}}\right)[dB], \quad (1)$$

where P_{signal} and P_{noise} – signal and noise power. This parameter shows how effectively noise masks the useful signal. The SNR value was calculated for each algorithm and compared to determine the difference in masking efficiency.

A spectrogram of superimposed noise on conversation recording was constructed, which allowed estimating spectral changes over time. This helped to visually test the noise efficiency of each algorithm, since high-quality masking should ensure an even distribution of noise in the spectrogram.

Results

Comparison of the performance of selected PRNG

BenchmarkDotNet is a popular .NET library that allows conducting detailed performance tests of methods and ensures high accuracy and repeatability of measurements. This is achieved through several important mechanisms:

- **automatic code isolation**: BenchmarkDotNet automatically allocates tested methods to ensure that they do not affect other processes;
- **warmup**: the tool performs several "warmup" iterations before starting basic testing, which helps to avoid the impact of initialisation processes;
- **repeatability**: BenchmarkDotNet runs the required number of tests to reduce the impact of random runtime deviations.

BenchmarkDotNet is often used in scientific research to evaluate the performance of algorithms and methods. For example, in the study by O. Balalaieva *et al*. (2023), the authors used this tool to compare the performance of various serialisation methods, proving its effectiveness in accurately measuring runtime. The study by N. Filho (2024) examined in detail examples of the use and effective operation of this tool. This confirmed the reliability of BenchmarkDotNet as a research tool where accurate performance measurement is important. The test description and configuration took place in accordance with the official documentation (BenchmarkDotNet official documentation, n.d.).

Test package and environment metadata:

- BenchmarkDotNet v0.14.0, Windows 10 (10.0.19045.4529/22H2/2022Update)
- Intel Core i5-7400 CPU 3.00GHz (Kaby Lake), 1 CPU, 4 logical and 4 physical cores .NET SDK 6.0.400

According to the results obtained (Table 1), the fastest algorithm was PCG with an average generation time of 740.8 ms, which indicates its efficiency and speed compared to others. The KISS (Keep It Simple Stupid) algorithm also showed good performance, with an average time of 867.0 ms, which makes it the second fastest. Other algorithms, such as Xoshiro128 ++ , WELL512a (Well Equidistributed Long-period Linear), and Mersenne Twister, had significantly longer runtimes – 1,381.2 ms, 2,116.7 ms, and 1,951.3 ms, respectively.

Table 1. Performance results

Method	Mean	Error	StdDev	Ratio
PCG	740.8 ms	2.81 ms	2.49 ms	1.00
Xoshiro128++	1,381.2 ms	7.68 ms	7.19 ms	1.86
KISS	867.0 ms	3.55 ms	3.32 ms	1.17
WELL512a	2,116.7 ms	14.92 ms	13.95 ms	2.86
Mersenne Twister	1,951.3 ms	20.81 ms	19.47 ms	2.63

Source: output of the BenchmarkDotNet library

These metrics suggest that the PCG and KISS algorithms are the most efficient among those considered in this study in terms of generation rate, which may be an important factor in applications where high performance is a critical requirement. The obtained ratios of Error and StdDev values to Mean indicate that the results of the performance test are reliable. The Ratio value shows the ratio of the performance of each algorithm to the fastest tested (PCG).

Results of statistical tests NIST, Dieharder and TestU01

NIST. The results of the NIST tests are presented in Table 2. The input data was divided into 100 bit streams according to the recommendations given in the paper by L.E. Basshham *et al*. (2010). All algorithms have passed the required limit (78 + of the RandomExcursion type and 96 + for other tests), and can be considered high-quality.

Table 2. Results of NIST statistical tests

Test name	PCG	WELL512a	Mersenne Twister	KISS	Xoshiro128++
AproximateEntropy	99/100	100/100	100/100	98/100	97/100
BlockFrequency	100/100	99/100	99/100	99/100	97/100
CumulativeSums	98/100	99/100	100/100	99/100	99/100
FFT	96/100	99/100	98/100	99/100	97/100
Frequency	98/100	99/100	100/100	99/100	100/100
LinearComplexity	99/100	99/100	99/100	99/100	100/100
LongestRun	99/100	98/100	100/100	100/100	98/100
NonOverlapingTemplate	99/100	99/100	99/100	98/100	99/100
OverlapingTemplate	99/100	99/100	97/100	98/100	100/100
RandomExcursion	87/89	91/92	80/82	91/93	79/80
RandomExcursionVariant	87/89	91/92	80/82	92/93	78/80
Rank	98/100	98/100	100/100	97/100	98/100
Runs	99/100	99/100	98/100	100/100	99/100
Serial	99/100	99/100	98/100	99/100	99/100
Universal	100/100	100/100	99/100	99/100	99/100

Source: developed by the authors based on NIST test results

PCG showed a high level of randomness in many tests, in particular, in the frequency and block frequency test, with a high proportion of passed tests (more than 98%). The algorithm did a good job of testing the length and rank of the longest run, which indicates its ability to generate long sequences of random values without obvious patterns. However, 4 out of 100 FFT tests failed. In this case, 96 completed tests are a sufficient condition for passing the test, but the remaining PRNGs showed better results.

The WELL512a showed good randomness results, especially in frequency, block frequency, and rank tests. Most of the tests were passed with an indicator of 98-100%. The high efficiency of this algorithm in NonOverlappingTemplate tests was noted, which indicates its ability to generate sequences that are not adapted to patterns.

Mersenne Twister showed excellent results in many tests, in particular, the frequency test, which confirms its randomness. However, in the Runs test, there are fewer completed tests, which may indicate some frequency or less high randomness compared to other PRNGs.

The KISS algorithm passed most tests, including Block frequency, Runs test, and NonOverlappingTemplate, with high results (98-100% pass). However, the results of the Rank test are worse than those of other algorithms.

Xoshiro128 ++ showed slightly less stable results in some tests, especially in cumulative amounts, where its proportion of test completion was 97%, indicating possible periodic patterns in large samples.

Overall, the Mersenne Twister and WELL512a algorithms performed better in the context of randomness and compliance with NIST tests, while PCG and Xoshiro128 ++ had some deviations in specific tests. This analysis shows that the Mersenne Twister and WELL512a algorithms may be most suitable for generating digital noise in cybersecurity tasks due to their stability and compliance with a wide range of NIST tests.

Dieharder. Based on Dieharder tests for various pseudorandom number generators (PRNG), it is possible to evaluate their effectiveness and compliance with the criteria of randomness (Dieharder official documentation, n.d.). The results for the KISS, Mersenne Twister, PCG, WELL512a, and Xoshiro128 ++ algorithms are divided into three parts, which contain the sum of tests with the passed|week|-failed status (Table 3).

Table 3. Dieharder statistical test results

Test name	PCG	WELL512a	Mersenne Twister	KISS	Xoshiro128++
dab_bytedistrib	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
dab_dct	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
dab_filltree	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0
dab_filltree2	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0

Table 3. Continued

Test name	PCG	WELL512a	Mersenne Twister	KISS	Xoshiro128++
dab_monobit2	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_2dsphere	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_3dsphere	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_birthdays	1|0|0	1|0|0	0|1|0	1|0|0	1|0|0
diehard_bitstream	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_count_1s_byt	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_count_1s_str	1|0|0	1|0|0	1|0|0	0|1|0	1|0|0
diehard_craps	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0
diehard_dna	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_operm5	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_opso	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_oqso	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_parking_lot	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_rank_32x32	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_rank_6x8	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_runs	2|0|0	1|1|0	2|0|0	2|0|0	2|0|0
diehard_squeeze	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
diehard_sums	1|0|0	1|0|0	1|0|0	0|1|0	1|0|0
marsaglia_tsang_gcd	2|0|0	2|0|0	0|2|0	1|0|1	2|0|0
Preparin	0|0|0	0|0|0	0|0|0	0|0|0	0|0|0
rgb_bitdist	12|0|0	12|0|0	12|0|0	11|1|0	11|1|0
rgb_kstest_test	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
rgb_lagged_sum	30|2|1	31|2|0	32|1|0	27|6|0	30|3|0
rgb_minimum_distance	4|0|0	4|0|0	4|0|0	4|0|0	4|0|0
rgb_permutations	4|0|0	4|0|0	4|0|0	4|0|0	3|1|0
sts_monobit	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
sts_runs	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
sts_serial	30|0|0	30|0|0	28|2|0	30|0|0	29|1|0

Source: developed by the authors based on Dieharder test results

PCG showed excellent results by passing all the main tests with the PASSED rating. The algorithm performed particularly well in the diehard_operm5, diehard_rank_32x32, sts_monobit, and rgb_lagged_sum tests, which highlights its reliability and stability in most tests. Out of the entire list of tests, only rgb_lagged_sum had weak and one failed run, which makes PCG one of the best algorithms among the analysed ones according to the Dieharder test results.

WELL512a also showed stable results, passing most passed tests, including complex tests such as diehard_squeeze and rgb_minimum_distance. However, in the diehard_runs and rgb_lagged_sum tests, a WEAK score was obtained, which indicates some difficulties in maintaining a uniform distribution on large sequences.

Mersenne Twister also passed most tests, including diehard_bitstream, diehard_operm5, and sts_serial, with good p-value values, indicating its high randomness in most cases. However, some tests, in particular marsaglia_tsang_gcd and diehard_birthdays, showed a WEAK estimate indicating probable periodicity or less stable randomness for specific patterns.

KISS passed most tests with high p-value values, which indicates a high level of randomness, especially in template creation tests such as diehard_birthdays and diehard_rank_6x8. However, in the diehard_count_1s_str and diehard_sums tests, the algorithm showed weaknesses with a WEAK score, which may indicate problems in generating statistically uniform sequences for large amounts of data.

Xoshiro128 ++ performed well in the tests, passing them with high p-value values, especially in diehard_operm5 and rgb_bitdist. However, several tests, such as rgb_bitdist and rgb_permutations, gave WEAK scores, which may indicate some shortcomings in ensuring uniform randomness for specific structural patterns.

Analysis of Dieharder's results shows that PCG and WELL512a are the most stable and reliable generators in terms of randomness, successfully passing most tests without significant deviations. Mersenne Twister and Xoshiro128 ++ also showed a high level of randomness, but had weaknesses in some specific tests.

TestU01. Based on TestU01 tests for various PRNG algorithms, randomness efficiency was compared for the KISS, Mersenne Twister, PCG, WELL512a, and Xoshiro128 ++ algorithms. The main criterion for evaluating the success of passing the test is the p-value: to successfully pass the test, the value must be within $0.01 \leqslant p\text{–value} \leqslant 0.99$ (TestU01 official documentation (n.d.); Raza & Satpute, 2017). A p-value that goes beyond these limits usually indicates a deviation from statistical randomness, and a p-value close to 0 or 1 indicates the probability of a non-random nature of the sequence (Table 4).

Table 4. Results of TestU01 statistical tests

Test name	PCG	WELL512a	Mersenne Twister	KISS	Xoshiro128++
Results of CollisionOver	8|0|0	7|1|0	8|0|0	8|0|0	8|0|0
scomp_LempelZiv	5|0|0	5|0|0	5|0|0	5|0|0	5|0|0
scomp_LinearComp	4|0|0	2|0|2	2|0|2	4|0|0	4|0|0
sknuth_CouponCollector	0|1|3	2|1|1	2|1|1	2|0|2	1|0|3
sknuth_Gap	0|0|4	0|0|4	0|0|4	0|0|4	0|0|4
sknuth_MaxOft	17|0|1	17|0|1	17|0|1	17|0|1	17|0|1
sknuth_Run	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0
sknuth_SimpPoker	2|0|2	1|1|2	0|2|2	1|1|2	2|1|1
smarsa_BirthdaySpacings	6|1|0	7|0|0	7|0|0	7|0|0	7|0|0
smarsa_GCD	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0
smarsa_MatrixRank	6|0|0	4|0|2	5|0|1	6|0|0	6|0|0
smarsa_Savir2	0|1|0	1|0|0	1|0|0	1|0|0	1|0|0
smultin_Multinomial	4|0|0	4|0|0	4|0|0	4|0|0	4|0|0
smultin_MultinomialOver	0|0|2	0|0|2	0|0|2	0|0|2	0|0|2
snpair_ClosePairs	17|0|0	17|0|0	17|0|0	17|0|0	17|0|0
snpair_ClosePairsBitMatch	2|0|0	2|0|0	2|0|0	2|0|0	1|1|0
sspectral_Fourier3	6|0|0	6|0|0	6|0|0	6|0|0	6|0|0
sstring_AutoCor	15|4|1	18|2|0	20|0|0	20|0|0	20|0|0
sstring_HammingCorr	3|0|0	3|0|0	3|0|0	3|0|0	2|0|1
sstring_HammingIndep	5|0|1	5|0|1	5|0|1	5|0|1	5|0|1
sstring_HammingWeight2	8|0|0	8|0|0	8|0|0	8|0|0	8|0|0
sstring_LongestHeadRun	4|0|0	4|0|0	4|0|0	4|0|0	4|0|0
sstring_PeriodsInStrings	2|0|0	2|0|0	1|1|0	2|0|0	2|0|0
sstring_Run	3|0|1	4|0|0	3|0|1	3|1|0	3|1|0
svaria_AppearanceSpacings	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0
svaria_SampleCorr	1|0|0	1|0|0	1|0|0	1|0|0	1|0|0
svaria_SampleMean	3|0|0	3|0|0	3|0|0	3|0|0	3|0|0
svaria_SampleProd	2|0|0	2|0|0	2|0|0	2|0|0	2|0|0
svaria_SumCollector	1|0|0	0|1|0	1|0|0	1|0|0	0|1|0
svaria_WeightDistrib	0|1|3	0|0|4	1|0|3	0|0|4	0|1|3
swalk_RandomWalk1	29|1|0	29|1|0	29|1|0	30|0|0	29|0|1

Source: developed by the authors based on TestU01 test results

PCG shows excellent results by passing most TestU01 tests with a p-value that is in the range of 0.01 and 0.99, for example, the value of 0.61 in the CollisionOver test. Thus, PCG can be considered one of the most stable PRNGs, which provides high-quality randomness without significant deviations in tests.

The WELL512a also successfully passed most tests with a p-value within the acceptable range, such as a value of 0.54 in the CollisionOver test. However, in some tests, WELL512a showed a p-value close to the limit, which may indicate possible instability under certain conditions.

Mersenne Twister also showed stable results with p-value within the acceptable range in most tests. For example, the CollisionOver test for Mersenne Twister showed a p-value of 0.90, which indicates its good randomness. However, there were tests where the p-value was close to 0 or 1, which can be an indicator of periodic patterns under certain conditions.

The KISS algorithm successfully passed most of the tests. In some tests, such as CollisionOver, p-value values were within acceptable limits (for example, p-value 0.70), which indicates acceptable randomness. However, in some cases, the algorithm showed a p-value close to the acceptability limit, which may indicate potential problems with certain sequence structures. However, the KISS algorithm has the least non-completed tests compared to other algorithms.

Xoshiro128 ++ showed good randomness results in most tests, getting p-value values in the range of 0.01 and 0.99, for example, a value of 0.32 in CollisionOver. The algorithm had small deviations in some tests, but generally showed stable randomness, which makes it acceptable for use in noise generation problems.

PCG and KISS are the most stable generators according to TestU01 results, providing reliable randomness. Mersenne Twister and Xoshiro128 ++ also showed high results, but with some deviations under certain conditions.

Results of generated noise

1. On autocorrelation graphs (Fig. 1) for each algorithm (KISS, PCG, WELL512a, Xoshiro128 ++, Mersenne Twister), it can be seen that all PRNGs have a sharp peak at zero latency, and then autocorrelation quickly decreases to zero for other delays. This suggests that the generators do not have significant internal correlations, which confirms the randomness of their sequences. The rapid reduction of the autocorrelation coefficient to zero is an important indicator of qualitative randomness for digital noise, which is used to mask signals in cybersecurity.

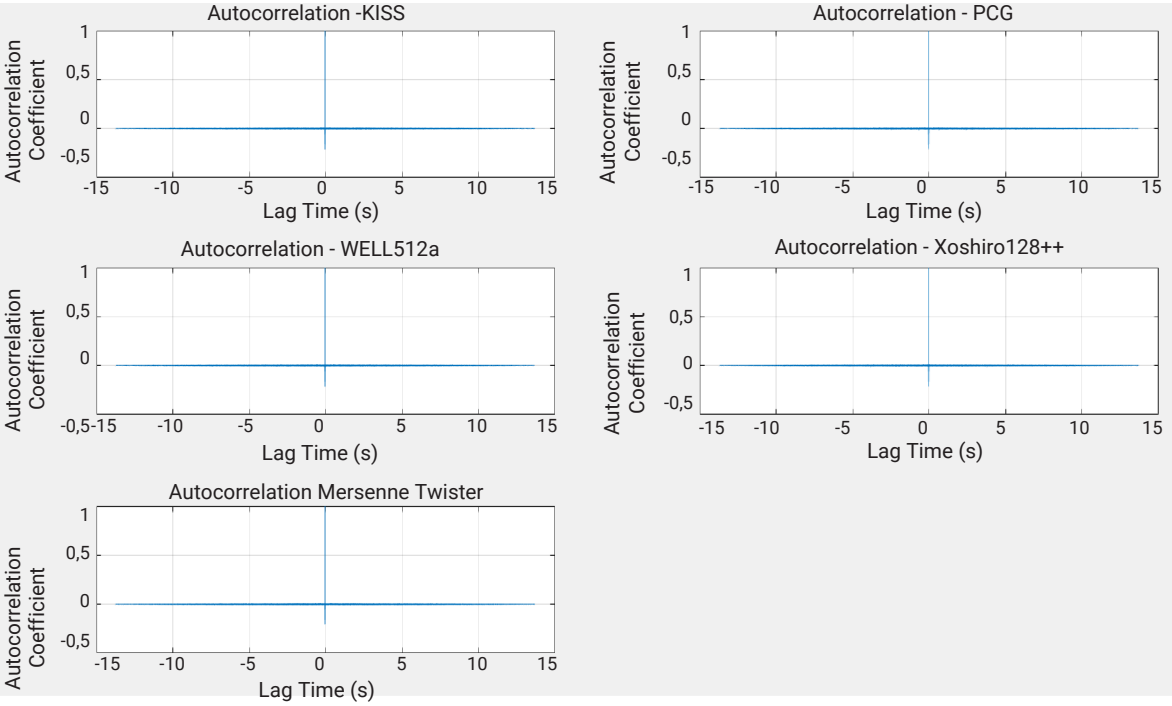

Figure 1. Results of autocorrelation functions
Source: developed by the authors in the Matlab environment

The results show the same Zero Crossing Lag value for all PRNGs at -13.6994 seconds. This may indicate the stability and similarity of the behaviour of each generator in creating a sequence of random numbers. This result is important for determining the absence of long-term correlations, which indicates a good quality of randomness.

2. Spectral power density before filtering (Fig. 2). Before filtering, the PSD results for all PRNGs show that the power is evenly distributed over a wide frequency range, without a drop in the high frequency range. This is typical for well random noise signals, where power is evenly distributed over frequencies. This uniformity before filtering is a confirmation of the high quality of randomness of all algorithms.

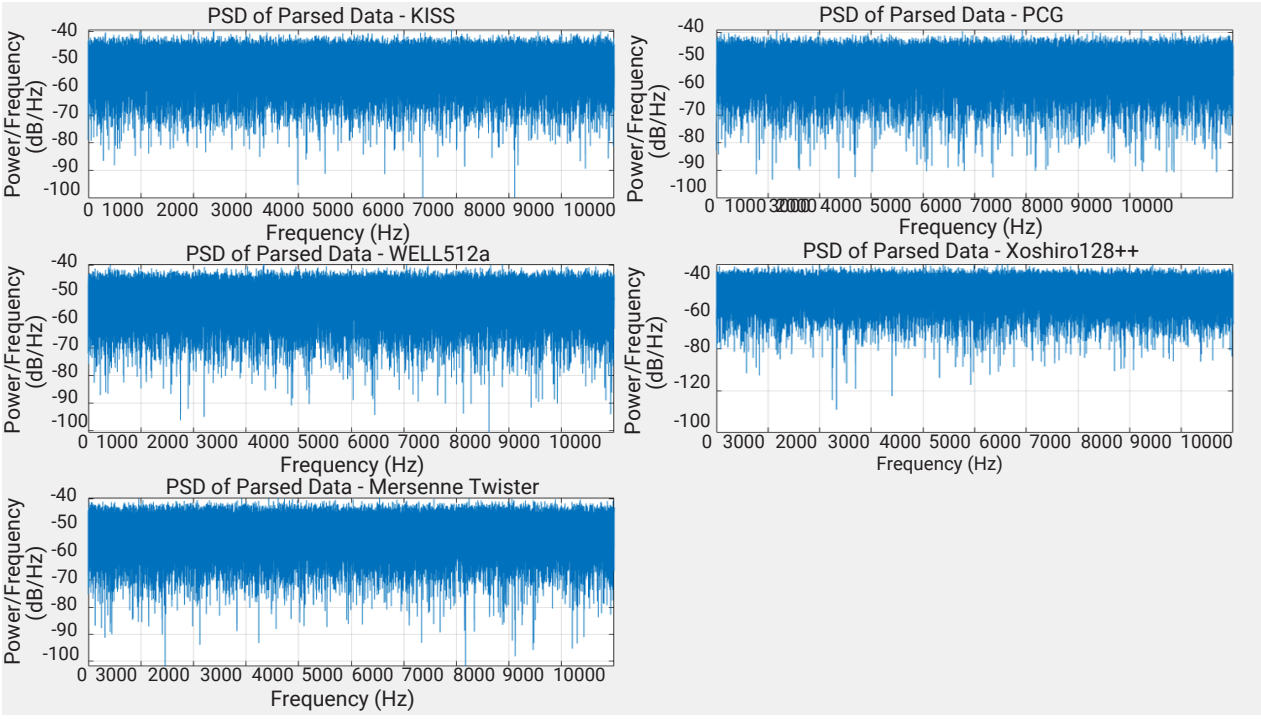

Figure 2. PSD of PRNG results, white noise results
Source: developed by the authors in the Matlab environment

3. Power spectral density (Fig. 3) and frequency noise analysis (Fig. 4) in the specified frequencies (0-4,000 Hz). Filtered PRNG results show a decrease at high frequencies, which is consistent with the expected filtering effect. In each case, a decrease in power at high frequencies is visible, which confirms the successful use of a filter to limit the frequency range of noise. This allows using filtered noise as corresponding to the specified frequency limits to add to the useful signal in masking problems.

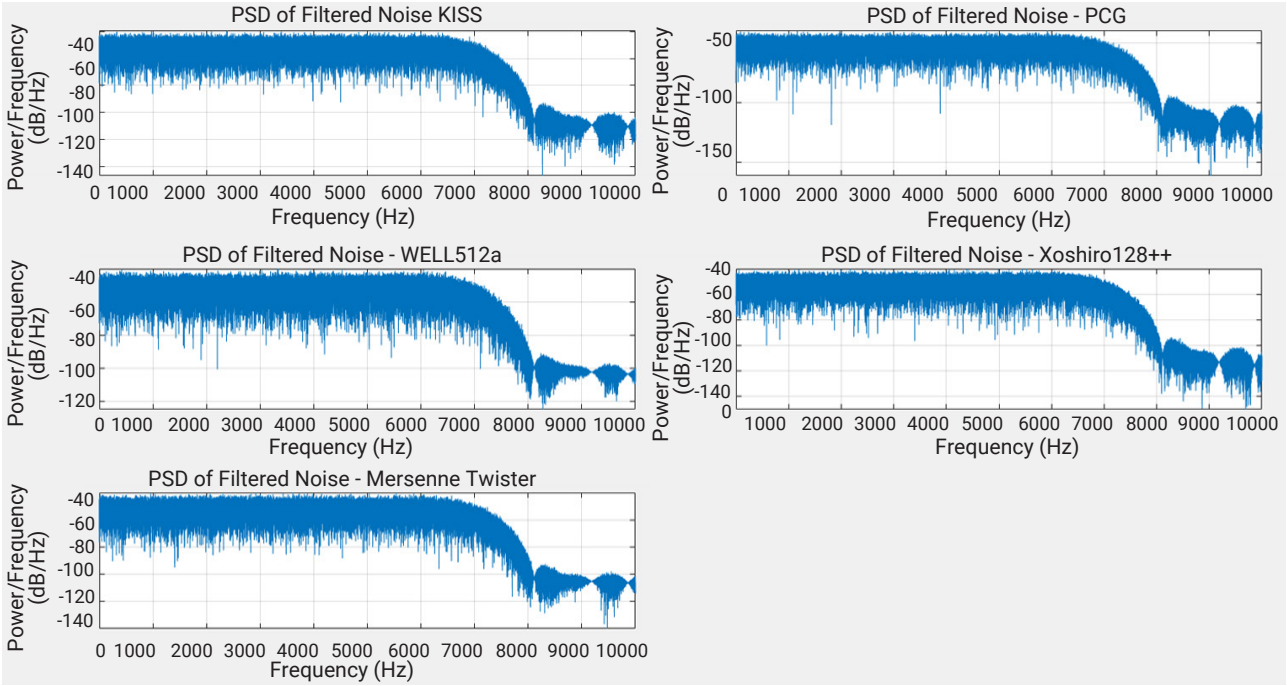

Figure 3. PSD of filtered noise

Source: developed by the authors in the Matlab environment

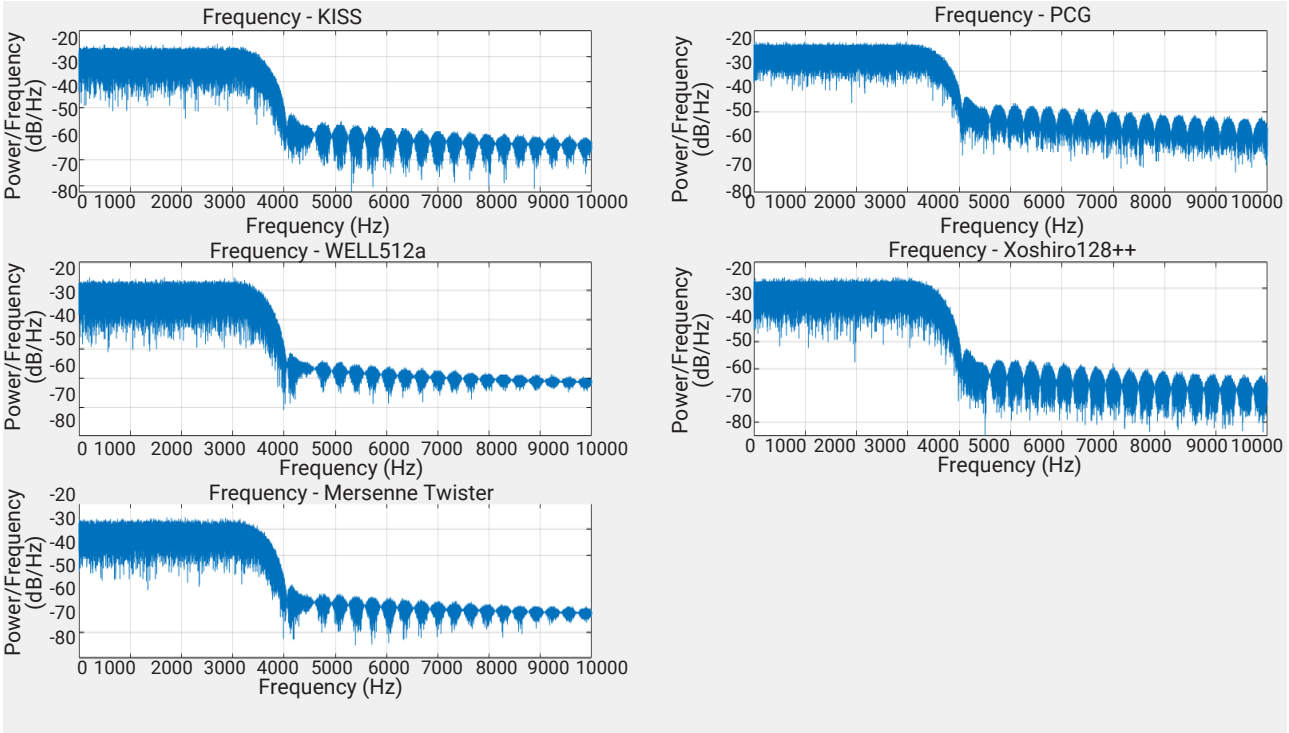

Figure 4. Frequency analysis of filtered noise

Source: developed by the authors in the Matlab environment

4. Signal-to-noise ratio was calculated by equation (1). The results show a close SNR value for each PRNG, with an average value of about -13.6 dB. This indicates the same efficiency of each generator for superimposing noise on the useful signal. A low SNR value indicates strong masking, which is important for added security, as the useful signal becomes less legible under the noise layer.

5. Spectrogram of noise superimposed on a conversation. The spectrograms of each PRNG superimposed on the conversation show that the generated noise completely overlaps the most intense frequencies (Fig. 5). The noise from each generator provides uniform masking, although small differences in the intensity and uniformity of noise distribution between the generators can be seen on the spectrogram. This indicates that all PRNGs provide reliable conversation masking, which makes it difficult to identify the original signal.

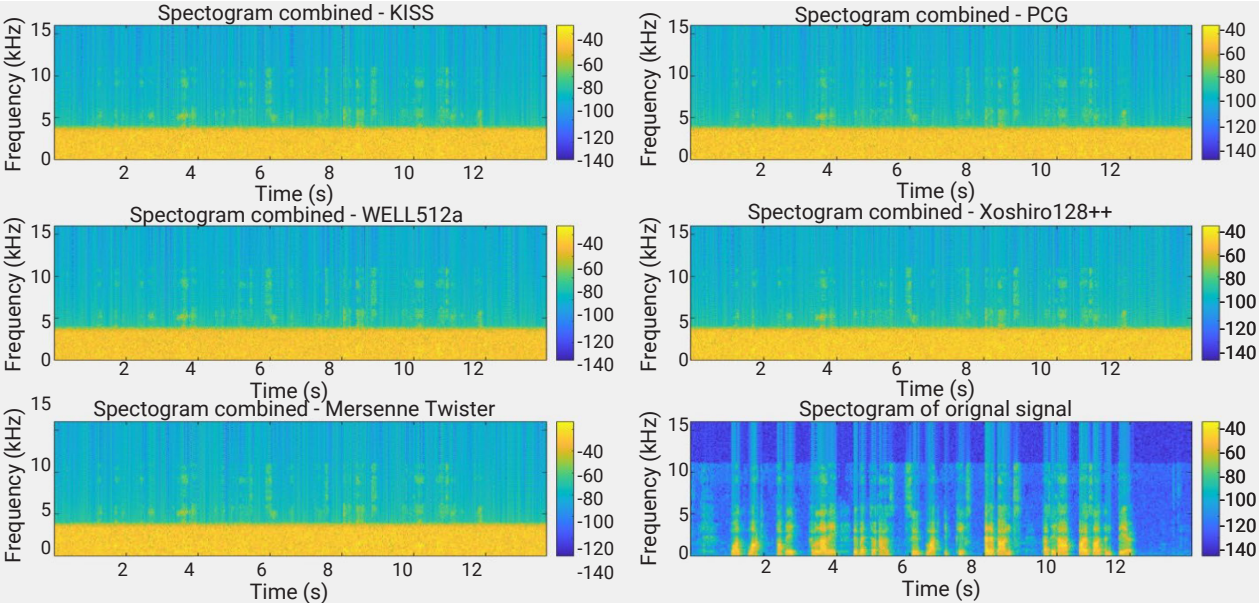

Figure 5. Frequency analysis of filtered noise

Source: developed by the authors in the Matlab environment

The results showed that all the considered PRNG algorithms (KISS, PCG, WELL512a, Xoshiro128 ++, Mersenne Twister) demonstrate a high level of randomness and provide reliable masking of the useful signal, which is confirmed by their autocorrelation characteristics, frequency distribution, and SNR. Minor differences in power spectral densities between algorithms can be considered when selecting a specific PRNG for specific tasks, but in general, all generators demonstrated an appropriate level of quality for signal masking purposes in cybersecurity.

Discussion

This study performed a comparative analysis of five pseudorandom number generators (PCG, Xoshiro128 ++, WELL512a, Mersenne Twister, and KISS) in terms of performance, randomness, spectral characteristics, and noise masking efficiency in cybersecurity tasks. Based on the conducted tests – statistical (NIST, Dieharder, TestU01), spectral analysis, Snr calculations and autocorrelation function analysis – comprehensive data were obtained that allow us to evaluate the quality and effectiveness of these algorithms in generating digital noise.

The results of performance tests showed that the PCG algorithm showed the highest efficiency with an average generation time of 740.8 ms for 268,435,456 values, which makes it an attractive choice for tasks where high performance is important. KISS, with a score of 867.0 Ms, also showed good speed indicators. Other generators, such as the Mersenne Twister and WELL512a, had a longer runtime, which may limit their use in systems where the generation rate is a critical parameter. Comparison with literature sources showed that previous studies have also noted the high performance of PCG and KISS in problems that require rapid random number generation (L'ecuyer, 2017).

Statistical tests by NIST, Dieharder, and TestU01 showed that all generators pass most tests with high p-value values, which indicates a high level of randomness of the generated sequences. However, some tests indicate minor deviations for Xoshiro128 ++ and Mersenne Twister in Dieharder tests, in particular, in "marsaglia_tsang_gcd", which may indicate a certain structure in the generated sequences. Differences in the results for the WELL512a and PCG, which passed all high-performance tests, indicate their high randomness quality for digital noise. These results are consistent with data from previous studies (Bhattacharjee & Das, 2022), which confirm the stability of PCG and WELL512a in random generation. In the mentioned study, among the list of 29 PRNG algorithms, PCG received the second, and WELL512a – the fifth place according to the results of statistical characteristics. The uniform

distribution of values in graphical tests is confirmed by the spectrogram in this paper, where the uniform distribution of frequencies within the specified limits allowed masking the speech signal.

PSD analysis before and after filtering showed that all generators provide uniform power distribution over a wide frequency range before filtering, which is typical for random noise. After applying filtration in the specified frequency ranges, a significant power reduction was achieved in the high-frequency range, which confirms the filtration efficiency. This makes the generated noise more suitable for masking tasks, limiting the effect on the high-frequency components of the useful signal. It is known that a uniform distribution of noise power is a prerequisite for its use in masking problems, which is confirmed by other studies in this field (Kajikawa *et al*., 2012; Deza & Ihshaish, 2021).

The SNR results showed slight differences between the generators, with an average value of about -13.6 dB for all algorithms. This indicated a similar efficiency of each generator in masking the useful signal, since a low SNR level indicates effective masking. However, the difference in SNR of less than 0.1 dB is not statistically significant, which indicates a similar level of noise influence on the useful signal for each of the generators. S.M. Kuo *et al*. (2010) confirmed that SNR in the range from -10 to -20 dB are acceptable for masking problems, which provides sufficient masking without significant loss of useful signal quality.

Analysis of the autocorrelation function showed that all generators have a significant peak at zero latency and a rapid decrease in correlation values for other delays, which is a sign of high randomness and the absence of internal patterns in the sequences. This is an important factor for problems where low correlation in long noise sequences is required to avoid the appearance of periodicity or patterns that can affect the quality of masking. The study by F. Yu *et al*. (2021) also indicated that the presence of a sharp peak at zero latency and low correlations for other delays are characteristic features of high-quality random number generators.

The corator spectrogram superimposed on the conversation recording showed a uniform frequency distribution of noise and effective masking of the low-frequency components of the useful signal. This indicates that all PRNGs provide reliable masking, making the useful signal less legible against the background of noise. Minor differences in power distribution on the spectrogram may be related to the characteristics of each generator, but do not significantly affect the overall quality of masking (Mandal, 2022). The mentioned study describes the implementation of a cryptographic PRNG, the purpose of which is to hide data by superimposing noise. The results showed that the noise level may differ depending on the data that needs to be hidden. However, it was enough to make it impossible to reproduce the original information.

The results of the current study confirmed the data obtained by other authors regarding the randomness and efficiency of various pseudorandom number generators. In particular, J.D. Cook (2017) showed in his paper that the PCG generator shows high performance and stable randomness, which is consistent with the data of this study, where PCG showed the best result in speed (740.8 ms for a large number of values) while maintaining the quality of randomness in statistical tests.

The study by P. L'ecuyer (2017), devoted to the analysis of the WELL512a generator, also showed that WELL512a provides reliable results in problems where high randomness is important for long generation periods. Current results confirm this, as the WELL512a has passed all major randomness tests, providing a stable spectral distribution over the frequency range, which makes it attractive for problems of masking a useful signal with noise.

According to the study by M. Saito & M. Matsumoto (2008**)**, Mersenne Twister has excellent randomness in large samples, but can exhibit certain periodic patterns in specific tests, such as marsaglia_tsang_gcd. The results of the current study confirmed these results, since Mersenne Twister successfully passed most of the tests, but found weaknesses in some specialised Dieharder tests, the results are consistent with the above study.

Conclusions

This study performed a comparative analysis of five pseudorandom number generators (PCG, Xoshiro128 ++, WELL512a, Mersenne Twister, and KISS) for digital noise generation tasks focused on signal masking in cybersecurity. The purpose of the study was to determine the optimal PRNG in terms of speed, randomness, and masking efficiency. The PCG and KISS generators showed the highest performance in performance tests, making them suitable for applications where fast generation of large amounts of random data is critical. PCG, with an average generation time of 740.8 ms for 268,435,456 values, proved to be the most effective.

All the generators under consideration passed the key statistical tests of NIST, Dieharder, and TestU01, which indicates their ability to generate qualitative random sequences. However, Xoshiro128 ++ and Mersenne Twister showed minor deviations in some specialised Dieharder tests, which may indicate the presence of certain structural features in their sequences that may be noticeable under certain conditions. Spectral analysis before and after filtration showed that all generators provide an even distribution of pre-filtration power, and filtering reduces power in the high-frequency range. This confirms the suitability of the generated noise to mask signals in a given frequency range.

The SNR values for all generators were close to -13.6 dB, which indicates the effectiveness of noise signal masking for each algorithm. The small difference in SNR between the generators indicates a similar ability of each algorithm to mask signals in environments with similar conditions. Analysis of the autocorrelation function revealed a high level of randomness with low correlation values for all generators outside the zero lag. This indicates the absence of noticeable internal correlations in the generated sequences, which is important for

creating qualitative noise that does not contain periodic or template components. The spectrogram showed a uniform frequency distribution of noise for each of the generators, which provides effective masking of the low-frequency components of the useful signal. Minor differences in power distribution between generators did not significantly affect the overall masking efficiency.

Considering the results of all experiments, it can be concluded that the best option for generating digital noise is the PCG algorithm. If the preference between performance and statistical tests is given to tests, then the WELL512a algorithm is the best option, but it is the slowest among the tested algorithms and, according to the results of the test, 2.86 times slower than PCG.

Further research may be aimed at analysing other, less common PRNG algorithms to evaluate their effectiveness in masking tasks and at developing new noise filtering methods that will allow adapting its spectral characteristics to specific conditions of use. In addition, it is promising to investigate dynamic filtering parameters to improve SNR depending on the type of useful signal, and to study the effect of different frequency ranges of noise on the effectiveness of masking in various cybersecurity applications. It is also worth noting the trend of trying to develop a new framework that will facilitate and automate the process of further research to find new PRNGs.

Acknowledgements

None.

Conflict of Interest

None.

References

[1] Balalaieva, O., Marchenko, I., Korotenko, G., Beshta, D., & Pikuz, A. (2023). Performance research of C# programming language data serializers using the developed software product for testing. *Reporter of the Priazovskyi State Technical University. Section: Technical Sciences,* 47, 8-24. doi: 10.31498/2225-6733.47.2023.299923.

[2] Bassham, L.E. *et al*. (2010). *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. Gaithersburg: National Institute of Standards and Technology. doi: 10.6028/nist.sp.800-22r1a.

[3] BenchmarkDotNet official documentation. (n.d.). Retrieved from https://surl.li/yobqul.

[4] Bhattacharjee, K., & Das, S. (2022). A search for good pseudo-random number generators: Survey and empirical studies. *Computer Science Review*, 45, article number 100471. doi: 10.1016/j.cosrev.2022.100471.

[5] Cook, J.D. (2017). *Testing the PCG random number generator*. Retrieved from https://surl.li/vjlfia.

[6] Deza, J.I., & Ihshaish, H. (2021). Qnoise: A generator of non-gaussian colored noise. *SSRN Electronic Journal*. doi: 10.2139/SSRN.3975571.

[7] Dieharder official documentation with test suit. (n.d.) Retrieved from https://surl.li/gruuvy.

[8] Feali, M.S. (2023). Realization of a pseudo-random number generator utilizing two coupled Izhikevich neurons on an FPGA platform. *Analog Integrated Circuits and Signal Processing,* 119(1), 57-68. doi: 10.1007/s10470-023-02223-2.

[9] Filho, N. (2024). Performance analysis in Csharp with BenchmarkDotNet: Report and evaluation. *ZENODO,* 1(12). doi: 10.5281/ZENODO.13826811.

[10] Hu, Z. (2020). High-speed and secure PRNG for cryptographic applications. *International Journal of Computer Network and Information Security (IJCNIS)*, 12(3), 1-10. doi: 10.5815/ijcnis.2020.03.01.

[11] Isakov, O.V., & Voitusik, S.S. (2023). Comparative analysis of digital noise generated by additive Fibonacci generators. *Ukrainian Journal of Information Technology*, 5(1), 67-76. doi: 10.23939/ujit2023.01.067.

[12] Kajikawa, Y., Gan, W.-S., & Kuo, S.M. (2012). Recent advances on active noise control: Open issues and innovative applications. *APSIPA Transactions on Signal and Information Processing*, 1, article number e3. doi: 10.1017/ATSIP.2012.4.

[13] Kuo, S.M., Kuo, K., & Gan, W.S. (2010). Active noise control: Open problems and challenges. In *The 2010 International conference on green circuits and systems* (pp. 164-169). Shanghai: IEEE. doi: 10.1109/icgcs.2010.5543076.

[14] L'Ecuyer, P. (2017). History of uniform random number generation. In *2017 Winter simulation conference (WSC)* (pp. 202-230). Las Vegas: IEEE. doi: 10.1109/wsc.2017.8247790.

[15] Li, S., Lin, Z., Yang, Y., & Ning, R. (2024). A high-performance FPGA PRNG based on multiple deep-dynamic transformations. *Entropy,* 26(8), article number 671. doi: 10.3390/e26080671.

[16] Mandal, K. (2022). Cryptographic pseudorandom noise generators for lattice-based cryptography and differential privacy. In *2022 10th international workshop on signal design and its applications in communications (IWSDA)* (pp. 1-4). Colchester: IEEE. doi: 10.1109/iwsda50346.2022.9870587.

[17] Panneton, F., L'Ecuyer, P., & Matsumoto, M. (2006). Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software*, 32(1), 1-16. doi: 10.1145/1132973.1132974.

[18] Raza S.F., & Satpute V.R. (2018). PRaCto: Pseudo random bit generator for cryptographic application. *KSII Transactions on Internet and Information Systems*, 12(12). doi: 10.3837/TIIS.2018.12.029.

[19] Saito, M., & Matsumoto, M. (2008). SIMD-oriented Fast Mersenne Twister: A 128-bit pseudorandom number generator. In A. Keller, S. Heinrich & H. Niederreiter (Eds.), *Monte Carlo and Quasi-Monte Carlo methods 2006* (pp. 607-622). Berlin-Heidelberg: Springer. doi: 10.1007/978-3-540-74496-2_36.

[20] Sound data sets. (n.d.). Retrieved from https://commonvoice.mozilla.org/en/datasets.
[21] Syafalni, I., Jonatan, G., Sutisna, N., Mulyawan, R., & Adiono, T. (2022). Efficient homomorphic encryption accelerator with integrated PRNG using low-cost FPGA. *IEEE Access*, 10, 7753-7771. doi: 10.1109/access.2022.3143804.
[22] TestU01 official documentation. (n.d.). Retrieved from https://surl.li/nemayh.
[23] Vennos, A., George, K., & Michaels, A. (2021). Attacks and defenses for single-stage residue number system PRNGs. *IoT*, 2(3), 375-400. doi: 10.3390/iot2030020.
[24] Yu, F., Zhang, Z., Shen, H., Huang, Y., Cai, S., Jin, J., & Du, S. (2021). Design and FPGA implementation of a pseudo-random number generator based on a hopfield neural network under electromagnetic radiation. *Frontiers in Physics*, 9. doi: 10.3389/fphy.2021.690651.

Порівняльний аналіз результатів генераторів псевдовипадкових чисел для генерації цифрового шуму

Олександр Ісаков

Аспірант
Національний університет «Львівська політехніка»
79013, вул. Степана Бандери, 12, м. Львів, Україна
https://orcid.org/0009-0007-4632-9492

Степан Войтусік

Кандидат фізико-математичних наук, доцент
Національний університет «Львівська політехніка»
79013, вул. Степана Бандери, 12, м. Львів, Україна
https://orcid.org/0000-0003-4234-3303

Анотація. У статті викладено результати дослідження характеристик п'яти різних генераторів псевдовипадкових чисел для застосування в задачах генерації цифрового шуму, який використовується для маскування сигналів у кібербезпеці. Актуальність роботи зумовлена зростаючою потребою у високоякісних методах маскування, які забезпечують як ефективну продуктивність, так і надійність випадковості, що важливо для захисту конфіденційної інформації у сучасних цифрових системах. Метою дослідження було порівняння алгоритмів PCG, Xoshiro128++, WELL512a, Mersenne Twister та KISS за показниками їхньої швидкодії, статистичної випадковості та здатності ефективно маскувати корисний сигнал шумом. Швидкодія алгоритмів оцінювалася за допомогою BenchmarkDotNet. Для перевірки якості випадковості послідовностей використовувалися стандартні тести NIST, Dieharder та TestU01. Для згенерованого шуму проведено спектральний аналіз за допомогою значення спектральної щільності потужності. Ефективність маскування було розраховано співвідношенням сигнал/шум, результатами автокореляційної функції і спектрограми шуму. Результати дослідження показали, що PCG та KISS є найбільш продуктивними з точки зору швидкодії, що робить їх привабливими для застосувань, де важлива швидка генерація випадкових послідовностей. WELL512a та PCG продемонстрували найвищу якість випадковості, стабільно проходячи всі статистичні тести. Аналіз спектрального розподілу шуму показав, що всі генератори забезпечують рівномірний розподіл потужності до фільтрації, а після фільтрації шум успішно обмежується у високочастотному діапазоні. Співвідношення значення сигналу до шуму для всіх алгоритмів становили близько -13.6 dB, що вказує на подібну ефективність при маскуванні шумом. Автокореляційний аналіз підтвердив низьку кореляцію для всіх генераторів за межами нульового лагу, що є важливим для збереження якості випадковості в довгих послідовностях. Практична цінність дослідження полягає у виборі оптимального генератора псевдовипадкових чисел для задач зашумлення в кібербезпеці. Отримані результати надають рекомендації щодо вибору алгоритмів з урахуванням їхньої швидкодії та випадковості, що дозволить забезпечити високий рівень захисту інформації у цифрових системах

Ключові слова: захист інформації; шумові характеристики; статистичні тести випадковості; спектральний аналіз; тести продуктивності; зашумлення сигналу