



ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ/INFORMATICS AND INFORMATION PROCESSES

DOI: <https://doi.org/10.60797/IRJ.2026.172.88> EDN: HBXFFL

DEVELOPMENT OF A SOFTWARE MODULE FOR THE COMPREHENSIVE QUALITY ASSESSMENT OF PSEUDORANDOM SEQUENCES

Research article

Gibadullin R.F.^{1,*}, Shigabutdinov M.I.²¹ ORCID : 0000-0001-9359-911X;^{1,2} Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation

* Corresponding author (landwatersun[at]mail.ru)

Suggested: 08.07.2026; Accepted: 04.09.2026; Published: 16.10.2026

Abstract

The article presents the development of a software module for the comprehensive quality assessment of pseudorandom number generators (PRNGs) using an extended set of statistical tests and parallel data processing mechanisms. The relevance of the work stems from the growing requirements for the reliability of cryptographic systems amid digital transformation and the increasing computing power of potential attackers. The module implements 31 statistical tests, including 15 NIST STS tests, 12 Diehard tests, as well as 4 specialized checks, and supports three generators of different classes: Mersenne Twister, CryptMT and the cryptographically secure system RandomNumberGenerator. A key feature of the solution is the parallel execution of the test suites based on the Task Parallel Library and the ThreadLocal generator mechanism, which provides a 3–5-fold speedup of data processing on 8-core systems compared with the sequential mode while guaranteeing reproducible results for deterministic generators. The modular three-layer architecture on the C#/.NET 7 platform allows the list of supported generators and tests to be extended without modifying the system core. In the experiments, the Mersenne Twister generator demonstrated a generation speed of 148 million numbers per second while passing 98% of the tests, and the module's RAM consumption did not exceed 650 MB when processing samples of up to 10 GB. The practical significance of the solution is confirmed by its applicability to the verification of cryptographic systems, to scientific research, and to education.

Keywords: pseudorandom number generators, statistical testing, NIST STS, Diehard, parallel computing, Task Parallel Library, cryptographic strength, Mersenne Twister, p-value, multithreading.

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ КОМПЛЕКСНОЙ ОЦЕНКИ КАЧЕСТВА ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Научная статья

Гибадуллин Р.Ф.^{1,*}, Шигабутдинов М.И.²¹ ORCID : 0000-0001-9359-911X;^{1,2} Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация

* Корреспондирующий автор (landwatersun[at]mail.ru)

Предложена: 08.07.2026; Принята: 04.09.2026; Опубликовано: 16.10.2026

Аннотация

В статье представлена разработка программного модуля для комплексной оценки качества генераторов псевдослучайных чисел (PRNG) с использованием расширенного набора статистических тестов и механизмов параллельной обработки данных. Актуальность работы обусловлена растущими требованиями к надежности криптографических систем в условиях цифровой трансформации и увеличения вычислительной мощности потенциальных злоумышленников. Модуль реализует 31 статистический тест, в том числе 15 тестов NIST STS, 12 тестов Diehard, а также 4 специализированные проверки, и поддерживает три генератора разных классов: Вихрь Мерсенна, CryptMT и криптографически безопасную систему RandomNumberGenerator. Ключевой особенностью решения является параллельное выполнение наборов тестов на основе библиотеки параллелизма задачам и механизма генератора ThreadLocal, что обеспечивает 3–5-кратное ускорение обработки данных на 8-ядерных системах по сравнению с последовательным режимом, гарантируя при этом воспроизводимость результатов для детерминированных генераторов. Модульная трёхуровневая архитектура на платформе C#/.NET 7 позволяет расширять список поддерживаемых генераторов и тестов без изменения ядра системы. В ходе экспериментов генератор Вихрь Мерсенна продемонстрировал скорость генерации 148 миллионов чисел в секунду, пройдя при этом 98% тестов, а потребление оперативной памяти модулем не превышало 650 МБ при обработке выборок объемом до 10 ГБ. Практическая значимость данного решения подтверждается его применимостью к верификации криптографических систем, научным исследованиям и образованию.

Ключевые слова: генераторы псевдослучайных чисел, статистическое тестирование, NIST STS, Diehard, параллельные вычисления, библиотека параллелизма задачам, криптографическая стойкость, Вихрь Мерсенна, р-значение, многопоточность.

Introduction

Amid digital transformation and the ubiquitous use of cryptographic systems, the requirements for the quality of pseudorandom number generators (PRNGs) are rising critically. Reliable PRNGs are the cornerstone of securing financial transactions, protecting personal data, and ensuring the correct operation of cryptographic protocols [1]. Random numbers are in demand across the entire spectrum of modern computing systems — from resource-constrained Internet of Things devices to heavily loaded server platforms — and an incorrect choice or implementation of a generator directly expands the system's attack surface [2]. The continuous growth of attackers' computing power and the anticipated advent of quantum computers [3] call for the development of new, more advanced methods for the comprehensive quality assessment of pseudorandom sequences.

For the statistical verification of PRNGs, the NIST STS [4], Diehard [5], and TestU01 [6] test batteries have become the de facto standard. However, a comparative analysis of the existing batteries shows that each of them has limitations: insufficient performance when processing large data volumes, the absence of built-in parallel processing, and weak integration into modern software development processes [7]. Moreover, empirical studies demonstrate that even widely used generators may fail rigorous statistical verification [8]. This creates the need for a specialized software module capable of providing a comprehensive and highly efficient assessment of PRNG quality in accordance with modern standards.

The aim of this work is to develop a software module for the comprehensive quality assessment of pseudorandom sequences that provides high testing accuracy, support for modern generation algorithms, and clear visualization of the results. To achieve this aim, the following tasks were accomplished:

1. an analysis of existing pseudorandom sequence generation algorithms and methods for their statistical testing was carried out;
2. a modular system architecture with support for parallel computing was designed;
3. a comprehensive set of 31 statistical tests was implemented, including tests from the NIST STS and Diehard suites, as well as specialized checks;
4. a user interface with advanced result visualization capabilities was created;
5. a comparative performance analysis of the implemented PRNGs was conducted, and the module's efficiency in the sequential and parallel modes was evaluated.

The object of the study is pseudorandom sequence generation algorithms and methods for evaluating their statistical properties. The subject of the study is software for the comprehensive quality testing of PRNGs. The scientific novelty of the work lies in the approach to the parallel execution of statistical tests based on a hybrid model of the Task Parallel Library and ThreadLocal generators, which provides a 3–5-fold speedup of data processing while preserving the reproducibility of deterministic generators' results, as well as in the modular architecture, which allows the set of supported tests and generators to be extended without modifying the module core. The practical significance lies in the possibility of applying the module in the creation and verification of cryptographic systems, in scientific research on new generation algorithms, and in education.

Research methods and principles

Pseudorandom sequences are numerical sequences generated by a deterministic algorithm whose statistical properties are as close as possible to those of truly random sequences. A key feature of such sequences is reproducibility: given the same initial value (seed), the generator always produces the same sequence. For a comprehensive quality assessment of a sequence, the uniformity of the distribution, the independence of the elements, the length of the repetition period, computational efficiency, and cryptographic strength are analyzed.

Three classes of generators are implemented in the developed module: the linear Mersenne Twister (MT19937) [9], the cryptographically secure CryptMT [10], and the system RandomNumberGenerator generator from the System.Security.Cryptography namespace of the .NET platform [11]. All generators implement a common IRandomGenerator programmatic interface with the NextUInt(), NextDouble(), and Initialize(seed) methods (the latter is applicable only to deterministic algorithms), which guarantees standardized testing and full interchangeability of the algorithms. Unlike the deterministic MT19937 and CryptMT, the RandomNumberGenerator generator is a wrapper over the operating system's cryptographic provider, does not support seeding, and serves in the module as the benchmark of unpredictability for comparative testing [11]. The main characteristics of the generators are presented in Table 1.

Table 1 - Comparison of the implemented pseudorandom sequence generators

DOI: <https://doi.org/10.60797/IRJ.2026.172.88.1>

Generator	Key feature	Repetition period
Mersenne Twister (MT19937)	624-element state array and the twisting operation (twist); excellent statistical properties and a high generation speed	$2^{19937} - 1$
CryptMT	A cryptographically secure modification of MT: post-processing of the output values with a 32-bit multiplicative accumulator eliminates linear artifacts	$2^{19937} - 1$ (inherits the MT19937 recurrence)
RandomNumberGenerator	A wrapper over the OS	Not defined (non-deterministic)

Generator	Key feature	Repetition period
(System.Security.Cryptography)	cryptographic provider; produces cryptographically secure non-deterministic values	source)

To verify the statistical properties, a suite of 31 tests is implemented in the module: 15 NIST STS (Special Publication 800-22) tests, 12 Diehard tests, and 4 specialized checks (including a bit-independence test and a monotonicity test), which are relevant for evaluating short sequences and detecting hidden correlations. All tests conform to a common *IRandomnessTest* interface. The main evaluation metric is the p-value, which, for a test to pass, must be no lower than the threshold significance level $\alpha = 0.01$ according to the NIST SP 800-22 standard [4]. The optimization methods listed in Table 2 were applied in implementing the tests.

Table 2 - Optimization and accuracy assurance methods for the statistical tests

DOI: <https://doi.org/10.60797/IRJ.2026.172.88.2>

Optimization method	Purpose	Technical implementation
64-bit floating-point arithmetic	Accuracy of p-value calculations	Relative error of the critical functions no greater than 10^{-6} (verified against reference values from Boost.Math)
SIMD vectorization	Acceleration of bit operations when processing large arrays	Vectorization of low-level bit transformations (up to a 4-fold speedup)
ArrayPool and Span<T>	Control of RAM consumption	Renting and reusing buffers; working with memory slices without copying
Memoization and lazy evaluation	Reduction of repeated computations	Caching the results of expensive statistical functions; deferred computation of intermediate results
Edge-case handling	Reliability on non-standard data	Special handling of short and degenerate sequences

The architecture of the software module is multilayered and strictly follows the principles of modularity, loose coupling, and separation of responsibilities (Separation of Concerns). The system core is structured into three logical layers: the generator layer (Data Layer), which serves as a unified data source; the testing layer (Business Logic Layer), which contains the computational logic of the statistical analysis; and the presentation layer (Presentation Layer), which is responsible for user interaction and results visualization. The core also contains a static *MathUtils* class with helper methods for computing statistical functions (*Erfc()*, *GammaIncomplete()*), which prevents code duplication.

Parallel computing support was implemented to address the problem of insufficient performance when processing the large data volumes (tens of billions of bits) required for comprehensive statistical analysis. The architecture uses a hybrid approach: the low-level threading model (*System.Threading.Thread*) ensures strict isolation of the generator state, while the high-level task-based model (*Task Parallel Library, TPL*) provides efficient distribution of the computational load across CPU cores. Parallelism is introduced at three structural levels: at the level of test suites (simultaneous execution of NIST STS and Diehard in independent threads), at the level of individual tests (parallelization of heavyweight checks such as the Linear Complexity Test and the Random Excursions Test), and within complex operations (matrix computations, Fourier transforms).

To ensure thread safety and reproducibility of the results, a mechanism of *ThreadLocal* generators is used: instead of a single shared PRNG object, an independent copy of the generator is created for each thread, initialized with a unique seed that incorporates the thread identifier; for the non-deterministic *RandomNumberGenerator*, which does not support seed initialization, an independent instance is created in each thread via the *Create()* factory method. The processing of large data arrays follows a MapReduce-like scheme [12]: at the Map stage, each worker thread independently processes its assigned block of data, performing a partial statistical analysis, while at the Reduce stage, the intermediate results are aggregated into the final statistical output using the thread-safe *ConcurrentBag* collection (Fig. 1).

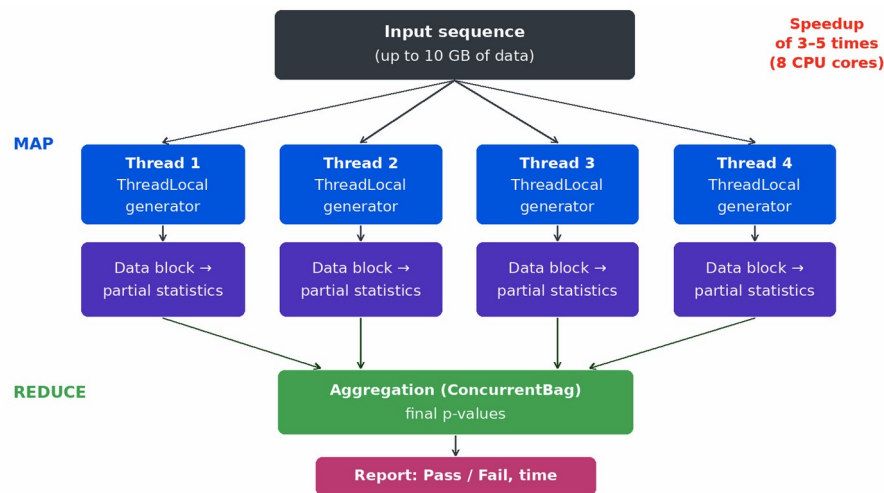


Figure 1 - Parallel data processing following a MapReduce-like scheme

DOI: <https://doi.org/10.60797/IRJ.2026.172.88.3>

The methodology for the experimental testing of the module is based on a comparative analysis of the three implemented PRNGs against two key metrics: statistical quality (the share of successfully passed test–sequence pairs, %) and performance (number generation speed per second). The statistical quality of each generator was evaluated on 100 independent sequences of one million bits each, which, for the suite of 31 tests, yields 3100 test–sequence pairs per generator; each performance measurement was repeated in 30 independent runs. All quantitative indicators are reported as mean values, and the spread of the results is characterized by 95% confidence intervals: for the mean generation speed and the parallel speedup these were computed using Student's t-distribution, and for the shares of passed pairs — by the Wilson method. To confirm the correctness of the statistical test implementations, the reference sequence set of the NIST Statistical Test Suite [4] was used. The experiments assessing the efficiency of the parallel mode were conducted on 8-core systems; on each run, the seed, the sample parameters, and the operating mode were recorded, which ensures full reproducibility of the experiments.

Main results

The developed module was subjected to a detailed comparative analysis against the most authoritative existing solutions: NIST STS [4], Dieharder [13] (an extension of the Diehard suite [5]), and TestU01 [6]. NIST STS is the de facto industry standard and a baseline requirement for the certification of cryptographically secure generators; however, it is strictly oriented toward sequential execution, and its output is a plain-text report. Dieharder contains 31 tests (a full run with parameter variation computes more than 110 test statistics) and can directly test hardware entropy sources, but it is also a command-line utility. TestU01 is known for an unprecedented volume of checks (106 tests in the BigCrush configuration, yielding about 160 test statistics) and is considered the gold standard for academic research; however, it has no built-in support for parallel computing and no graphical interface. The comparison results are given in Table 3.

Table 3 - Comparative analysis of the developed module and its key analogs

DOI: <https://doi.org/10.60797/IRJ.2026.172.88.4>

Characteristic	Developed module (C#/.NET 7)	NIST STS (C/C++)	Dieharder (C/C++)	TestU01 (C/C++)
Number of tests	31 (NIST, Diehard, 4 specialized)	15 (NIST only)	31 (more than 110 statistics)	106 (about 160 statistics)
Parallelism support	Yes (Task Parallel Library)	No (sequential)	Partial	No (sequential)
Results visualization	Interactive UI (charts, tables)	Text report	Text report	Text report
Automatic reproducibility	Yes (seed logging and saving)	Manual saving	Manual saving	Manual saving
Key advantage	Comprehensiveness, speed, clear UI	Compliance with the NIST standard	Extensive test set, physical entropy	In-depth linear complexity analysis

Comparative testing of the implemented generators confirmed the expected characteristics of each algorithm, allowing the module user to make an informed choice of tool for a specific task. The summary results are presented in Table 4. The estimates obtained are stable: the half-width of the 95% confidence interval does not exceed 1.4 percentage points for the share of passed test–sequence pairs and 4% of the mean value for the generation speed.

Table 4 - Summary of the generator test results

DOI: <https://doi.org/10.60797/IRJ.2026.172.88.5>

Generator	Statistical quality (share of passed test–sequence pairs), %	Performance, million numbers/s	Primary purpose
Mersenne Twister	98	148	High-speed simulations and scientific computations
CryptMT	82	89	Cryptographic strength, balance of speed and quality
RandomNumberGenerator	99	42	Cryptographic applications, moderate speed

Testing of the parallel mode on 8-core systems demonstrated the high efficiency of the chosen approach: parallel execution of the test suites consistently shows a 3–5-fold speedup compared with sequential execution: in all 30 repeated runs, the speedup factor remained within the indicated range, and the half-width of the 95% confidence interval for its mean value did not exceed 0.2. Achieving ideal linear speedup is impossible due to the constraints imposed by Amdahl's law [14]: parallelization efficiency is reduced by mandatory sequential stages — the initial data loading, the preparation of input parameters, and the final aggregation of the results.

The module's resource consumption was optimized through the use of the ArrayPool and Span<T> structures: RAM consumption does not exceed 650 MB even when running the most resource-intensive tests, such as Overlapping Template Matching, when processing samples of up to 10 GB. The accuracy of the computations was verified by comparison with reference values from the Boost.Math library: the relative error of the high-precision approximations of the mathematical functions (the incomplete gamma function and the complementary error function Erfc) does not exceed 10^{-6} . In addition, an automated verification system was introduced that compares the obtained statistical metrics against known reference values.

System reliability is ensured at the architectural and software levels: the reproducibility of the deterministic generators' results is guaranteed by identical output sequences for the same seed; the aggregation of intermediate results in parallel computations is performed via the thread-safe ConcurrentBag collection; and memory-control and buffer-overflow protection mechanisms are built into the resource-intensive tests. For the system RandomNumberGenerator generator, which implements the IDisposable interface, the module's wrapper calls the Dispose method, which releases the cryptographic context of the operating system provider. The user interface, implemented with Windows Forms, provides selection of the generator and testing parameters, switching between the sequential and parallel modes, a results table with the columns "Test", "P-value", "Status", and "Time (s)", interactive plots of the distribution of p-values, as well as export of the results to JSON and CSV.

Discussion

The obtained results show that combining generally recognized test suites with a parallel execution model eliminates the main practical drawback of the classical batteries — the low speed of processing large samples under sequential execution, noted, among other places, in survey works on statistical test batteries [7]. The visual analysis of the distribution of p-values implemented in the module enables a researcher to quickly detect statistical anomalies, whereas the analogs require additional processing of text reports. Automatic logging of the seeds and sample parameters eliminates the manual recording of experimental conditions, which is especially important for certification and scientific work, where reproducibility is a mandatory requirement.

The generator test results are consistent with assessments known in the literature: Mersenne Twister demonstrates excellent statistical properties and the highest speed but is not cryptographically secure, whereas cryptographic generators naturally lag behind in performance [8], [15]. Note that the share of passed checks obtained for CryptMT is lower than expected for a cryptographically secure generator; this result requires further analysis and, according to a preliminary assessment, is related to the implementation parameters of individual tests rather than to the properties of the algorithm itself. This confirms the appropriateness of supporting several classes of generators in the module: the choice of algorithm should be determined by the task — simulations and scientific computations permit the use of MT19937, while cryptographic applications require CryptMT or RandomNumberGenerator.

The limitations of the current version of the module should also be noted. In the depth of its linear complexity analysis, it is inferior to the TestU01 library, which is widely used in modern randomness research, including for the evaluation of ciphers [16]. The module lacks a mode for directly testing hardware (physical) entropy sources, which is implemented in Dieharder [13]. In addition, recent studies show that the tests of the NIST SP 800-22 suite are not fully statistically independent [17], so the interpretation of the aggregated percentage of passed tests requires caution. The security limitations include the lack of temporary-file encryption and a simplified error-handling model in the UI layer; for industrial use, it is recommended to protect the handling of the seed in RAM, to add checksums for the executable code, and to support hardware acceleration of cryptographic operations (HSM).

The prospects for the project's development are determined by current trends in cryptography: the plans include integration with the TestU01 library to deepen the linear complexity analysis, extended functionality for testing physical entropy sources [18], and the addition of support for elliptic-curve-based generators [19] and post-quantum cryptographic algorithms [3], [20].



Conclusion

The developed software module for assessing the quality of pseudorandom sequences is a complete, comprehensive solution that combines modern statistical analysis methods with an efficient software implementation on the .NET 7 platform. The goal set at the outset of the work has been achieved: generators of three classes (Mersenne Twister, CryptMT, RandomNumberGenerator) have been implemented and optimized; a suite of 31 statistical tests based on NIST STS and Diehard, supplemented with the authors' specialized checks, has been created and integrated; and a user interface with advanced visualization capabilities has been developed.

Experimental verification confirmed the effectiveness of the adopted solutions: the leader in generation speed is Mersenne Twister (148 million numbers per second with 98% of the checks passed); the best statistical quality was, as expected, demonstrated by the system RandomNumberGenerator (99%); the hybrid parallel model based on the Task Parallel Library and ThreadLocal reduces the analysis time by a factor of 3–5 relative to the sequential mode; and RAM usage when processing samples of up to 10 GB remains within 650 MB. The modular architecture ensures flexibility and ease of extension, while support for a deterministic mode guarantees the reproducibility of research.

The practical significance of the module is defined by three main areas of application: the verification of generators in scientific research and computer modeling, the certification of cryptographic modules in accordance with international standards, and use in education. The priority directions for further development are integration with TestU01, support for testing physical entropy sources, and the addition of post-quantum cryptographic algorithms, which will allow the module to remain relevant amid rapidly changing cryptographic security requirements.

Финансирование

Данная работа/публикация была финансирована грантом Академии наук Республики Татарстан, выделенным высшим учебным заведениям, научным и другим организациям в целях содействия реализации планов по развитию кадрового потенциала, направленных на стимулирование научно-преподавательского состава к защите докторских диссертаций и ведению научно-исследовательской деятельности. (Договор № 15/2025-PD-KAI от 22 декабря 2025 г.).

Конфликт интересов

Не указан.

Рецензия

Рудой Е.М., ООО «ГК «Иннотех», Москва Российская Федерация
DOI: <https://doi.org/10.60797/IRJ.2026.172.88.6>

Funding

This work/publication was funded by a grant from the Academy of Sciences of the Republic of Tatarstan provided to higher education institutions, scientific and other organizations to support human resource development plans in terms of encouraging their research and academic staff to defend doctoral dissertations and conduct research activities. (Agreement № 15/2025-PD-KAI dated December 22, 2025).

Conflict of Interest

None declared.

Review

Rudoi E.M., Innotech Group LLC, Moscow Russian Federation
DOI: <https://doi.org/10.60797/IRJ.2026.172.88.6>

Список литературы на английском языке / References in English

1. Barker E. Recommendation for random number generation using deterministic random bit generators (NIST Special Publication 800-90A, Rev. 1) / E. Barker, J. Kelsey. — National Institute of Standards and Technology, 2015. — DOI: 10.6028/NIST.SP.800-90Ar1.
2. Kietzmann P. A guideline on pseudorandom number generation (PRNG) in the IoT / P. Kietzmann, T.C. Schmidt, M. Wählisch // ACM Computing Surveys. — 2021. — Vol. 54. — № 6. — DOI: 10.1145/3453159.
3. Bernstein D.J. Post-quantum cryptography / D.J. Bernstein, T. Lange // Nature. — 2017. — Vol. 549. — № 7671. — P. 188–194. — DOI: 10.1038/nature23461.
4. Rukhin A. A statistical test suite for random and pseudorandom number generators for cryptographic applications (NIST Special Publication 800-22, Rev. 1a) / A. Rukhin, J. Soto, J. Nechvatal [et al.]. — National Institute of Standards and Technology, 2010. — DOI: 10.6028/NIST.SP.800-22r1a.
5. Marsaglia G. The Marsaglia random number CDROM including the Diehard battery of tests of randomness / G. Marsaglia. — Florida State University, 1995. — URL: <https://web.archive.org/web/2016/http://stat.fsu.edu/pub/diehard/> (accessed: 15.10.2026).
6. L'Ecuyer P. TestU01: A C library for empirical testing of random number generators / P. L'Ecuyer, R. Simard // ACM Transactions on Mathematical Software. — 2007. — Vol. 33. — № 4. — DOI: 10.1145/1268776.1268777.
7. Almaraz Luengo E. Recommendations on statistical randomness test batteries for cryptographic purposes / E. Almaraz Luengo, L.J. García Villalba // ACM Computing Surveys. — 2021. — Vol. 54. — № 4. — DOI: 10.1145/3447773.
8. Bhattacharjee K. A search for good pseudo-random number generators: Survey and empirical studies / K. Bhattacharjee, S. Das // Computer Science Review. — 2022. — Vol. 45. — DOI: 10.1016/j.cosrev.2022.100471.
9. Matsumoto M. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator / M. Matsumoto, T. Nishimura // ACM Transactions on Modeling and Computer Simulation. — 1998. — Vol. 8. — № 1. — P. 3–30. — DOI: 10.1145/272991.272995.
10. Matsumoto M. Cryptographic Mersenne Twister and Fubuki stream/block cipher (Report No. 2005/165) / M. Matsumoto, T. Nishimura, M. Hagita, M. Saito. — Cryptology ePrint Archive, 2005. — URL: <https://eprint.iacr.org/2005/165> (accessed: 15.08.2026).



11. RandomNumberGenerator class (System.Security.Cryptography). — Microsoft Learn, 2025. — URL: <https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.randomnumbergenerator> (accessed: 15.08.2026).
12. Dean J. MapReduce: Simplified data processing on large clusters / J. Dean, S. Ghemawat // Communications of the ACM. — 2008. — Vol. 51. — № 1. — P. 107–113. — DOI: 10.1145/1327452.1327492.
13. Brown R.G. Dieharder: A random number test suite (Version 3.31.1) / R.G. Brown. — Duke University, 2021. — URL: <https://webhome.phy.duke.edu/~rgb/General/dieharder.php> (accessed: 15.10.2026).
14. Amdahl G.M. Validity of the single processor approach to achieving large scale computing capabilities / G.M. Amdahl // Proceedings of the April 18–20, 1967, Spring Joint Computer Conference (AFIPS '67). — ACM, 1967. — P. 483–485. — DOI: 10.1145/1465482.1465560.
15. Almaraz Luengo E. A brief and understandable guide to pseudo-random number generators and specific models for security / E. Almaraz Luengo // Statistics Surveys. — 2022. — Vol. 16. — P. 137–181. — DOI: 10.1214/22-SS136.
16. Sleem L. TestU01 and Practrand: Tools for a randomness evaluation for famous multimedia ciphers / L. Sleem, R. Couturier // Multimedia Tools and Applications. — 2020. — Vol. 79. — № 33–34. — P. 24075–24088. — DOI: 10.1007/s11042-020-09108-w.
17. Almaraz Luengo E. Further analysis of the statistical independence of the NIST SP 800-22 randomness tests / E. Almaraz Luengo, B. Alaña Olivares, L.J. García Villalba [et al.] // Applied Mathematics and Computation. — 2023. — Vol. 459. — DOI: 10.1016/j.amc.2023.128222.
18. Herrero-Collantes M. Quantum random number generators / M. Herrero-Collantes, J.C. Garcia-Escartin // Reviews of Modern Physics. — 2017. — Vol. 89. — № 1. — DOI: 10.1103/RevModPhys.89.015004.
19. Brown D.R.L. A security analysis of the NIST SP 800-90 elliptic curve random number generator / D.R.L. Brown, K. Gjøsteen; edited by A. Menezes // Advances in Cryptology – CRYPTO 2007 (Lecture Notes in Computer Science, Vol. 4622). — Springer, 2007. — P. 466–481. — DOI: 10.1007/978-3-540-74143-5_26.
20. National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard (FIPS 203). — U.S. Department of Commerce, 2024. — DOI: 10.6028/NIST.FIPS.203.