Cryptographic system security approaches by monitoring the random numbers generation

Svitlana Popereshnyak[1,†], Yuriy Novikov[2,†] and Yuliia Zhdanova[3,*,†]

[1] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37 Beresteiskyi ave., 03056 Kyiv, Ukraine

[2] Institute of Software Systems of the National Academy of Sciences of Ukraine, 40-5 Akademik Hlushkov ave., 03187 Kyiv, Ukraine

[3] Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudryavska str., 04053 Kyiv, Ukraine

Abstract

The paper examines one of the approaches to ensuring the security of cryptographic systems by monitoring the generation of random numbers. Random numbers play a key role in cryptography, in particular for generating keys, initialization vectors, and other important cryptographic parameters. Unreliable or predictable random numbers can lead to successful attacks on cryptographic protocols, making generation monitoring critical to the security of systems. The paper proposes an automated monitoring system that utilizes statistical tests to check randomness, entropy level, and the presence of correlations between generated numbers. Particular attention is paid to researching methods of detecting anomalies and reacting to them in real-time. Furthermore, the paper examines the effect of limited entropy in resource-constrained devices like those used in the Internet of Things (IoT) and explores the application of machine learning to enhance the monitoring of random number generation. The results demonstrate that implementing the monitoring system significantly enhances the resilience of cryptographic systems against attacks targeting random number generation.

Keywords

cryptography, random number generation, monitoring, entropy, statistical tests, anomalies, internet of things, security

1. Introduction

In today's conditions of rapid technological development, information protection is becoming one of the priority tasks in cyber security. Most cryptographic systems for data encryption, key generation, and user authentication are based on the use of random numbers. The quality of the random numbers used in these systems directly affects their resistance to cryptographic attacks. However, many random number generators are susceptible to attacks that reduce entropy or make their sequences predictable, creating a vulnerability for the entire cryptographic system.

The introduction of a random number generation monitoring system becomes an important element of cyber protection, as it allows for real-time detection of anomalies in the generation process and response to them, minimizing the risk of data compromise. The use of such systems increases the overall reliability of cryptographic protocols, especially in the face of entropy attacks or sequence prediction attempts.

The main problem is that cryptographic systems may be exposed to vulnerabilities when random number generators produce weak or insufficiently unpredictable sequences. This creates an opportunity for attacks on pseudorandom number generators, which can lead to the disclosure of keys or other sensitive information. Traditional approaches to random number generation do not always provide reliable control over the quality and randomness of sequences in real-time, which increases the risk of system compromise.

The implementation of a random number generation monitoring system addresses this issue by continuously overseeing the generation process through statistical tests and anomaly detection mechanisms. Such a system can automatically signal random violations and propose measures to eliminate them, which significantly increases the resistance of cryptographic systems to attacks.

The purpose of the research is to develop and implement a monitoring system for the generation of random numbers, which will allow us to automatically evaluate the quality and compliance of the generation with the criteria of randomness. This entails employing statistical methods to identify deviations from expected outcomes and ensure the reliability of random number generators across various systems, particularly in security-critical sectors like cryptography and the IoT [1, 2].

CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, October 26, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

spopereshnyak@gmail.com (S. Popereshnyak);
ynovikov@gmail.com (Y. Novikov);
y.zhdanova@kubg.edu.ua (Y. Zhdanova)

0000-0002-0531-9809 (S. Popereshnyak);
0009-0006-9800-8765 (Y. Novikov);
0000-0002-9277-4972 (Y. Zhdanova)

2. Review of literature and scientific publications

During the last decade, the issue of ensuring the security of cryptographic systems remains relevant, and many researchers pay attention to the generation of random numbers as one of the key aspects of this security. Random numbers are used to generate encryption keys, salt for hashing passwords, and other cryptographic processes. The poor quality of random numbers or their predictability can make cryptographic systems vulnerable to attack.

One of the key areas of research is the study of attacks on Random Number Generators (RNGs) and their impact on the security of cryptographic systems. Various types of PRNGs and their vulnerabilities are considered in [3–5]. Research indicates that predictable or insufficiently random sequences can compromise cryptographic keys. Additionally, various attacks on cryptographic systems have highlighted the necessity of real-time monitoring of random number generation quality.

Recent research indicates that merely employing cryptographically secure random number generators (CSPRNGs) is not always adequate for ensuring a high level of security. The works [6, 7] propose the development of a system for real-time monitoring of random number generation to identify anomalies and deviations from randomness. These systems use statistical tests to assess the level of entropy and the presence of correlations in sequences of random numbers.

Some of the popular monitoring methods include the Chi-square test for distribution uniformity, Pearson's test for correlations, and entropy analysis for measuring unpredictability. Such systems allow the detection of anomalies before they lead to real problems in cryptographic processes.

With the development of the IoT, there is a need to use lightweight and energy-efficient random number generation methods. Publications [8, 9] analyze the impact of insufficient entropy in IoT devices on the cryptographic stability of these systems. The researchers particularly highlight the significance of monitoring random number generators, especially given the limited resources of IoT devices. Insufficient entropy sources can lead to duplication of keys and other cryptographic data, which poses a security threat.

Recent studies, such as [10–13], have proposed the application of machine learning techniques for monitoring random number generation. Machine learning algorithms can analyze large volumes of data, and identify hidden patterns and anomalies that may go unnoticed using traditional statistical methods. These studies show that hybrid approaches combining statistical tests and machine learning can significantly improve the reliability of cryptographic systems.

With the development of cloud computing, cryptographic systems increasingly rely on random number generation in cloud environments. Research [14, 15] emphasizes the need to monitor the generation of random numbers in the conditions of scalable cloud environments [16, 17]. The publications describe the use of distributed monitoring systems that can monitor the performance of RNGs in different virtual environments and detect anomalies related to the computational load.

The literature review shows the importance of monitoring the generation of random numbers as a critical component of the security of cryptographic systems. Most modern studies point to the need to implement automated monitoring systems to detect anomalies and maintain a high level of entropy. This applies to both classic cryptographic systems and modern platforms such as IoT and cloud computing [18].

Employing advanced techniques, such as machine learning and statistical analysis, can significantly enhance the resilience of random number generators against attacks, thereby ensuring the reliability and unpredictability of cryptographic operations.

3. Analysis of stability of generators

Analysis of the stability of pseudorandom number generators (PRNGs) in real conditions consists in determining how well they can withstand external factors that can affect the quality of random number generation (Table 1). Such factors include noise, limited computing resources, changes in the execution environment, and other technical or physical influences.

The research conducted yielded the following results:

- A decrease in the quality of random numbers can be observed in conditions of unstable power supply or increased loads on the system, which leads to a decrease in entropy or an increase in the predictability of sequences.
- Reliable generators exhibit consistent random number generation even in the face of significant fluctuations in available resources or external conditions, ensuring high levels of randomness and speed.

Table 1
Types of generators testing for resistance to external factors

Type	Problem	Testing
Noise immunity testingм	Noise attacks. Generators can be subject to noise attacks, where the input data is distorted by exposure to external noise. For example, for hardware generators, it could be electromagnetic radiation, while for software generators it could be a malfunction of the hardware or operating system.	During the testing, experiments are carried out with the addition of artificial noise to the system to check the resistance of the HPC to such influences. This can be done by emulating an unstable environment, such as generating random numbers under varying power levels or network failures.
Testing in conditions of limited resources	Limitation of computing resources. IoT devices and other low-power systems often have limits on computing power, RAM, and energy. Generators must remain reliable even with minimal resources.	Experiments are being carried out with the limitation of available resources during the execution of HPC. For example, artificially reducing the amount of available RAM or increasing delays in processor cycles allows you to assess how this will affect the performance and quality of random numbers.
Resistance to entropy attacks	Entropy reduction. One important factor is the level of entropy from which random numbers are generated. If entropy decreases due to external influences or a lack of sufficient sources of entropy, this can lead to predictable generation results.	Entropy sources are analyzed during testing. For example, there may be limited input data (noise from physical sensors or random sources from the OS) to test whether the HPC can generate sufficiently random numbers.
Analysis under conditions of high loads	High load on the system. Real-world conditions often include HPC operation under high load, for example, when several processes simultaneously use generator resources.	Conducting stress tests, which include increasing the number of requests to the generator or performing other computational tasks at the same time, allows you to evaluate how this affects the speed and randomness of the generated numbers.
The influence of the reliability of hardware components	Hardware failures. Hardware oscillators can be susceptible to problems with the components themselves, such as aging or defects in the chips.	Simulating hardware component failures or conducting tests on various devices with differing levels of wear and tear enables the evaluation of their resistance to such factors.
Analysis using statistical tests	Some statistical tests (eg, Chi-squared test, Pearson test, autocorrelation analysis) are used to detect outliers or non-random patterns during testing.	Testing using multivariate statistical methods allows you to assess the quality of randomness under variable external conditions [19, 20].

Testing pseudorandom number generators in real conditions allows you to determine their resistance to various external influences, such as noise, limited resources, and high loads. The analysis results contribute to enhancing generators for use in critical systems like IoT and cryptographic algorithms, thereby ensuring reliable random number generation even in challenging conditions.

4. Study of the effectiveness of PRNGs

Investigating the performance of PRNGs for IoT infrastructure applications is an important step in determining their suitability in terms of resources and performance. The primary criteria for assessing efficiency include computational complexity, speed, energy consumption, and memory utilization. Let's examine the key steps along with examples of research and evaluations regarding the effectiveness of various PRNGs. (Table 2).

Performance evaluation criteria:

Computational complexity. An estimate of the number of operations required to generate one random number. Algorithms of different complexity are studied (linear complexity O(n), logarithmic complexity O(log n), constant complexity O(1)).

Example: A simple algorithm of congruent HPC has linear complexity since at each step a simple operation of multiplication, addition, and subtraction is performed modulo.

Speed action. It quantifies the number of random numbers a generator can produce within a given time frame (such as numbers generated per second). Algorithms on different processor architectures are studied: ARM for IoT devices, which often have limited computing power.

Example: Comparing a classic LCG (Linear Congruent Generator) and a more complex algorithm such as Mersenne Twister can show that LCG has a speed advantage on simple IoT device processors.

Energy consumption. The total power consumption for random number generation over a certain time or number of operations is measured. Important for battery-powered IoT devices where energy savings are critical.

Example: Simple algorithms with minimal computing load will be less energy-consuming compared to more complex generators that require a lot of resources for their work.

Memory usage. The amount of RAM required for the operation of the generator is estimated. In many IoT devices, memory is limited, so memory efficiency is a key factor.

Example: Algorithms such as LCG require less memory compared to algorithms based on complex tables, such as the Mersenne Twister, which requires large buffers for its operation.

Table 2
Evaluating the effectiveness of various PRNGs

PRNG	Complexity	Speed (numb./sec)	Energy consumption (mW)	Memory usage (kB)
LCG	$O(1)$	10^6	50	2
Mersenne Twister	$O(n)$	10^4	150	10
XORShift	$O(1)$	10^5	70	3
CSPRNG	$O(n^2)$	10^3	200	20

For IoT devices, where speed and energy efficiency are crucial, simple generators such as LKG or XORShift demonstrate superior performance in both speed and power consumption. However, in cases where cryptographic

robustness is required, CSPRNG, despite the higher resource costs, is a necessary choice.

5. Description of the random number generation monitoring system

A random number generation monitoring system should automate data collection, analysis, and visualization processes to ensure real-time control of generation quality and stability. This will effectively detect any deviations from randomness or other anomalies in the operation of PRNGs and hardware generators.

Let's consider the main components of the monitoring system (Fig. 1):

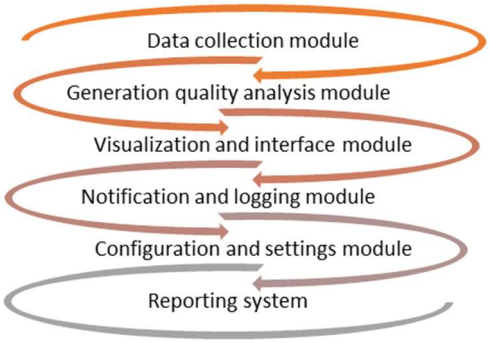

Figure 1: The main components of the monitoring system

Data collection module. This module gathers data from various random number generation sources, including both PRNGs and hardware generators. Data can be collected from local or remote generating systems. The module facilitates real-time data collection, in addition to storing historical data for subsequent analysis. Data sources can be generators in cryptographic systems, IoT devices, mobile applications, or other systems that rely on PRNG.

Generation quality analysis module. The analysis module assesses the quality of randomness in the collected numbers. It uses statistical methods to detect correlations, and predictable patterns and checks whether the generation meets the criteria of randomness. Methods that can be utilized in this module: Chi-square test and Pearson's test to test for uniform distribution; autocorrelation analysis to check dependencies between numerical sequences; multivariate tests for analyzing correlations between several parameters; Entropy test for evaluating the degree of unpredictability in numbers.

Visualization module and user interface design. Offers an interface for visualizing monitoring results. The graphical interface should show indicators such as entropy level, distribution uniformity, frequency deviations, and other quality metrics. Types of visualizations that can be implemented: Histograms and distribution graphs that show the distribution of numbers and reveal possible deviations from uniformity; heat maps of correlations that visualize dependencies between different random number generations; real-time monitoring shows current generation performance and quality metrics, allowing for immediate detection of deviations or anomalies.

Notification and logging module. This module is responsible for logging events and notifying about deviations in the generation. It provides logging of all generation processes and provides the ability to view historical data for in-depth analysis. If serious deviations are detected, the system sends a notification to the administrator or interested parties via email, mobile application, or other means of communication.

Configuration and settings module. This module enables the configuration of various parameters for the monitoring system, including data collection frequency, alert threshold values, selection of statistical tests for analysis, and user interface settings. The system should support flexible configuration for different types of generators and usage scenarios, allowing it to be adapted to specific needs.

Reporting system. Automatic generation of detailed reports on the quality of random number generation. These reports can be saved as PDF or other formats, allowing detailed analysis of the generation history and making it available to interested parties. Reports usually include the following factors: randomness metrics, detected deviations, and recommendations for improving the quality of generation.

The use of a monitoring system is particularly useful for the following industries:

- **Cryptographic systems** where the reliability of random number generation is critical for security.
- **IoT devices**, where constrained resources may impact the quality of generation.
- **Mobile applications** that utilize random number generation for security purposes or gaming.

Here are the key advantages of the system.

Increased reliability. Continuous monitoring ensures the stable operation of generators, helping to avoid failures and anomalies.

Instant reaction to deviations. Thanks to built-in notifications, the system allows you to quickly react to any failures in the generation process.

Real-time analysis. The system supports real-time data collection and analysis, which allows you to quickly obtain information about the quality of random numbers.

This system provides an opportunity to flexibly configure the generation of random numbers to ensure their high quality, convenient visualization, and timely detection of problems in real conditions of use.

6. Modeling the operation of the random number generation monitoring system

6.1. Overview of the system's general algorithm

Let's examine the key stages of the random number generation monitoring system (Fig. 2).

System initialization. The system is initiated and configured to monitor random number generation. The sources of random number generation, whether software or hardware generators, are identified.

Data collection. The system collects numerical sequences from generators in real-time. Data collection is conducted based on pre-defined intervals or events.

Data pre-processing. Collected data is sequenced for further analysis. The accuracy of the collected data is verified to ensure there are no omissions or errors.

Analysis of generation quality. Statistical tests are applied to the collected data to check for randomness:

Check for anomalies. The analysis results are compared against reference indicators. If deviations or anomalies are identified (indicating non-compliance with randomness criteria), the system triggers a response.

Decision on anomalies. If no anomalies are detected, the system continues to collect and log data. If anomalies are detected, the system initiates a response procedure.

Actions when anomalies are detected

- **Notification**: the system alerts the administrator or the individual responsible for security systems about any identified issues.
- **Automatic actions**: an adjustment attempt is possible (restarting the generator or changing the entropy source).
- **Problem logging**: details of the anomaly are captured for further analysis.

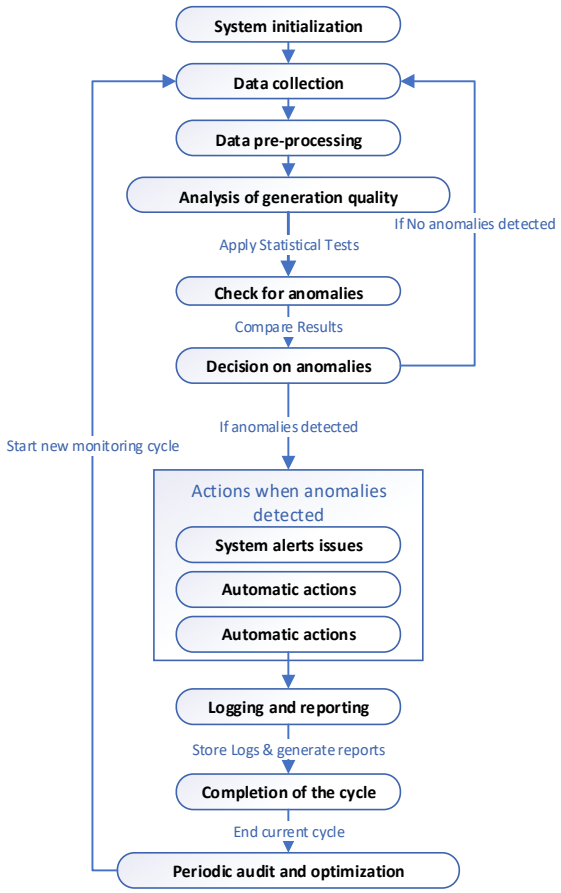

Figure 2: The main stages of the general scheme of the random number generation monitoring system

Logging and reporting. All monitoring actions and results are stored in logs. The system automatically generates reports on the status of random number generation (daily, weekly, etc.).

Completion of the cycle. The system ends the current monitoring cycle and starts a new one.

Periodic audit and optimization. Periodically, the system conducts an in-depth audit of the operation of generators for further improvement of settings or algorithms.

The algorithm is aimed at automatic quality control of random number generation with minimal user intervention. The system can quickly react to deviations, ensuring stability and reliability of generation in critical systems.

6.2. Mathematical model of the system for monitoring the generation of random numbers

A mathematical model for a random number generation monitoring system can be constructed using several key components. This model should include a process of data collection, random analysis, anomaly detection, and response.

Let:

- $X(t)$ —is a sequence of random numbers generated at time t.
- $f(X(t))$ —is a function describing the properties of the sequence $X(t)$, which is responsible for checking its randomness.
- T_{test}—is a set of statistical tests for checking randomness (for example, Chi-square test, entropy test).
- P_{anom}—is the probability of an anomaly occurring in the generation process.
- $D(t)$—is the deviation from the randomness reference values at time t.

6.2.1. Modeling the generation of random numbers

The generation of random numbers in the system is described as a set of sequences of numbers:

$$X(t) = \{x_1, x_2, \dots, x_n\}, \quad (1)$$

where $x_i \in [a, b]$ is a single random number within the interval $[a, b]$, generated at time t.

6.2.2. Modeling the quality of randomness

The randomness test function $f(X(t))$ applies statistical tests to the sequence $X(t)$. For example, for the Chi-square test:

$$f_{\chi^2}(X(t)) = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}, \quad (2)$$

where O_i are the observed frequencies of random numbers, E_i are the expected frequencies of random numbers.

The test results are compared against critical values. If the result surpasses the $\chi^2_{critical}$ threshold, this indicates a deviation from a uniform distribution, and an anomaly is recorded.

Other tests (for example, the entropy test $H(X)$) can estimate the level of entropy:

$$H(X) = \sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \quad (3)$$

where $p(x_i)$ is the probability of the number x_i appearing. A high entropy means a more random sequence.

6.2.3. Modeling the probability of occurrence of anomalies

The probability of an anomaly occurring, denoted as P_{anom}, is influenced by the extent to which the test results deviate from the reference values. If the deviation function $D(t)$exceeds the permissible value D_{max}, an anomaly is considered to have occurred:

$$P_{anom} = P(D(t) > D_{max}). \quad (4)$$

Here $D(t) = |f(X(t)) - f_{target}(X)|$, where $f_{target}(X)$—the reference value of the randomness function.

6.2.4. Modeling the response of the system

If the probability of an anomaly exceeds the permissible $P_{anom} > P_{threshol}$, the system goes into response:

- **Notification:** The system generates a notification for the operator.
- **Automatic intervention:** It is possible to restart the generator or connect a backup source of random number generation.

Formally, the reaction process can be described as follows:

$$R(t) = \begin{cases} 0, P_{anom} \le P_{\text{threshold}} \\ 1, P_{anom} > P_{\text{threshold}} \end{cases}, \quad (5)$$

where $R(t)$ is the system response at time t (0—normal operation, 1—intervention or notification).

6.2.5. Modeling the logging and reporting process

To provide historical analytics, the system keeps a log of all data stored in the form:

$$L(t) = \{X(t), f(X(t)), P_{anom}, R(t)\}. \quad (6)$$

This log allows you to track all events related to the generation of random numbers and generate reports to analyze the monitoring results.

6.2.6. General mathematical model

Mathematically, the model of the random number generation monitoring system can be represented as a set of functions:

1. Generation of a sequence of random numbers

$$X(t) = \{x_1, x_2, \ldots, x_n\}. \quad (7)$$

2. Evaluation of the quality of randomness using tests:

$$f_{\chi^2}(X(t)) = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}, \quad H(X) = \sum_{i=1}^{n} p(x_i) \log_2 p(x_i). \quad (8)$$

3. Probability of anomaly:

$$P_{anom} = P(D(t) > D_{max}). \quad (9)$$

4. System reaction:

$$R(t) = \begin{cases} 0, P_{anom} \le P_{\text{threshold}} \\ 1, P_{anom} > P_{\text{threshold}} \end{cases}. \quad (10)$$

5. Logging and storage of results:

$$L(t) = \{X(t), f(X(t)), P_{anom}, R(t)\}. \quad (11)$$

This mathematical model allows for building a system that automatically collects, analyzes, and controls the quality of random number generation in real-time, providing visualization and responding to anomalies.

7. Overview of the software

7.1. Library of statistical tests

The library of statistical tests is a component of the monitoring server but can be used as a separate product if necessary. The simplest method to utilize it is by adding a ".jar" file to the project during compilation. However, it is advisable to use tools like "Maven" or "Gradle" for automating tasks within Java projects. This avoids manually downloading and compiling the project with the library and is a safer approach.

In Maven, you need to define a new repository "jitpack.io" and add the library as a dependency (Fig. 3).

```
<repositories>
    <repository>
        <id>jitpack.io</id>
        <url>https://jitpack.io</url>
    </repository>
</repositories>

<dependencies>
    <dependency>
        <groupId>com.github.Leenocktopus</groupId>
        <artifactId>random-bit-sequence-test</artifactId>
        <version>1.0</version>
    </dependency>
</dependencies>
```

Figure 3: Import the library using Maven

The process is almost identical for Gradle, but the repository should be slightly different. The library does not contain any configuration parameters or settings that must be made before use, so you can perform statistical tests (Fig. 3) simply by calling methods on the library classes.

7.2. Monitoring server

The monitoring server can be used locally for testing, but it is likely to be more useful to deploy it in a cluster, cloud environment, or on local servers in a network where client applications are already deployed (or planned to be deployed in the future).

A simple and working solution would be to use a docker container to deploy the server.

Just like the integration library, the server has several environment variables used for mail and database connections. They must be specified for correct operation.

7.3. Web application

The web application does not contain a "Home Page" per se, so the user will be immediately redirected to the "Random Numbers" page (Fig. 4). This page can be conventionally divided into 2 parts—a random number filter and a table with random numbers.

Figure 4: Graph of the number of random numbers processed by the server

In the upper right corner of the screen, there is a form that enables you to adjust the time parameters of the graphs and display values for the past hour, day, week, or month, as well as select grouping by labels or programs.

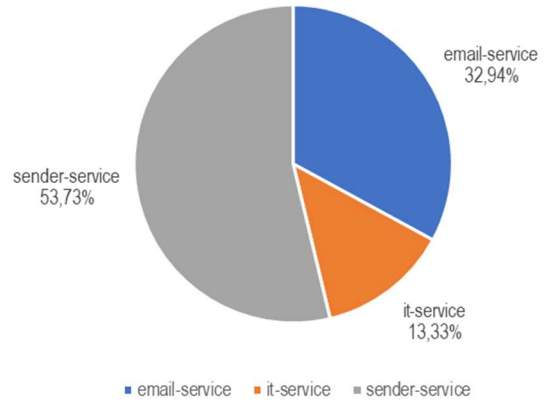

Figure 5: The graph illustrating the distribution of random numbers by client programs

The panels under the heading "Graphs" contain 5 graphs. In Fig. 5 you can see two of them—the number of random numbers processed by the server and the distribution of random numbers by client programs. Additionally, the program features a graph that shows the distribution of random numbers by values for each label. This can help identify whether a generator has a flaw that causes it to produce an excess or deficiency of random numbers within a specific range.

8. Recommendations for improving the reliability of generators

Based on the monitoring and testing results, we will develop recommendations for enhancing random number generation algorithms, focusing on methods to improve their stability and performance in critical systems.

To boost the reliability of random number generators, the following recommendations can be made based on these findings.

Improvement of algorithmic stability of generators.

- **Use of cryptographically stable generators (CSPRNG).** Utilizing generators based on cryptographic algorithms, such as AES or SHA, ensures reliable randomness, even in critical systems like secure communication or data protection.
- **Update generation algorithms.** Consistently update and optimize generators to address emerging attacks or vulnerabilities. This includes improvements to pseudo-random generators such as Xorshift, Mersenne Twister, or newer variants based on block ciphers.

Protection against the influence of external factors.

- **Addition of noise sources (entropy pool).** It is important to supplement generators with external sources of randomness (for example, noise from sensors, and physical processes), which will increase the resistance of the generator to predictable attacks or distortions due to the reduction of internal entropy.
- **Input quality monitoring.** Automated control of input entropy level and periodic updating of noise sources can prevent generation randomness from decreasing.

Minimization of correlations and predictability

- **Regular verification of correlation between generations:** Applying statistical tests to verify the correlation between sequences of numbers will help to identify and eliminate patterns that reduce the reliability of the generator.
- **Increasing the number of random bits:** To increase robustness, it is recommended to generate a larger number of random bits from different independent sources, which reduces the chances of correlation or predictability of the results.

Durability testing in real conditions

- **High-load and stress-testing:** It is important to regularly test generators under real-world operating conditions, particularly in high-load and resource-constrained (power, memory) situations, to verify their robustness.
- **Integration with monitoring systems:** The creation of systems that automatically monitor the operation of generators in real time allows timely detection of possible failures or loss of randomness.

Backup and restoration of the generation system

- **Use of multiple sources of generation:** Creating redundancy systems where generators work in parallel reduces the risks associated with failure of one generator or loss of entropy.
- **Automated switching to other generators in case of failures:** In case of generation problems,

the system should automatically switch to another random number generator or source.

Optimization of computational efficiency

- **Optimization of resource usage:** It is important to configure generators to consume minimum power and memory, which is critical in resource-constrained environments such as IoT. This can be achieved by simplifying or adapting existing algorithms.
- **Development of lightweight algorithms:** Using lightweight algorithms specially optimized for resource-constrained devices will help improve performance and reliability in such systems.

Periodic update and audit of generators

- **Scheduled updates and retesting:** Continuous testing and auditing of generators, including the use of new statistical tests, will help maintain a high level of reliability and identify vulnerabilities to new types of attacks.

These guidelines will enhance the reliability of pseudorandom number generators, particularly in critical systems like cryptographic algorithms, IoT security, and other fields where the quality of randomness is essential for the security and stable operation of systems.

9. Conclusions

Random number generators are an important tool for solving a variety of simulation, numerical methods, cryptography, and programming problems. Generation facilities can adopt one of several available approaches, each with its strengths and weaknesses. Nevertheless, the most critical feature of generators is their capacity to produce truly random numbers, as the security of cryptographic applications and the efficiency and speed of numerical applications hinges on the randomness of these numbers.

Utilizing generation tools necessitates prior research through statistical tests and cryptographic attacks to ensure confidence in the quality of the generated numbers and the security of the tool. During operation, generators sometimes show worse performance than was obtained during initial tests. This may be due to problems in the entropy source, incorrect application, or software implementation. Depending on the generator's specific task, implementing a monitoring system is advisable to identify and address potential defects promptly.

The created monitoring system provides the functions of monitoring the operation of generation means and alerting in case of exceptional situations. Programs or hardware devices connect to a centralized server and send random numbers generated by them for statistical testing and storage for future research. The monitoring system consists of the following components:

A library including 15 NIST tests and 8 multivariate statistics tests. The NIST statistical tests are a comprehensive approach to the verification of random numbers and means of their generation, while the methods of multivariate statistics complement them by providing the possibility to verify short sequences of bits.

An integration library designed to quickly connect a monitoring server and generators or applications containing random number generators. Application integration is done only with the use of metadata and configuration.

The monitoring server primarily functions to aggregate random numbers transmitted by client programs, along with their pre-processing and storage in the database. Additional features include various settings for tracking and notification processes, as well as detailed reports and real-time random number testing.

A web application that is completely based on the functions and application interface of the monitoring server and is designed to provide a convenient interface for users.

The monitoring system is recommended to be used in the case of operation or research of several generators of random numbers and sequences created by them at random. Practical application of the product is possible in:

Cryptography, development, and maintenance of software products and hardware—tracking the operation of autonomous random number generators and programs that use built-in generators;

Scientific research—simultaneous statistical testing of several random number generators, development and testing of new random number generators.

References

[1] O. Shevchenko, et al., Methods of the Objects Identification and Recognition Research in the Networks with the IoT Concept Support, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 2923 (2021) 277–282.

[2] V. Dudykevych, et al., Platform for the Security of Cyber-Physical Systems and the IoT in the Intellectualization of Society, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 449–457.

[3] C. Camara, et al., A True Random Number Generator Based on Gait Data for the Internet of You, IEEE Access, 8 (2020) 71642–71651. doi: 10.1109/ACCESS.2020.2986822.

[4] T. Zanotti, Guidelines for the Design of Random Telegraph Noise-based True Random Number Generators, IEEE Transactions on Device and Materials Reliability, 24(2) (2024) 184–193. doi: 10.1109/TDMR.2024.3394576.

[5] A. Kumar, A. Mishra, Evaluation of Cryptographically Secure Pseudo Random Number Generators for Post Quantum Era, in: IEEE 7th International Conference for Convergence in Technology (I2CT) (2022) 1–5. doi: 10.1109/I2CT54291.2022.9824543.

[6] K. Banerjee, P. Dasgupta, Acceptance and Random Generation of Event Sequences under Real Time Calculus Constraints, Design, Automation & Test in Europe Conference & Exhibition (DATE) (2014) 1–6. doi: 10.7873/DATE.2014.267.

[7] D. Novazrianto, et al., Design Automation of Single Photon Counting Method for Quantum Random Number Generation, in: 9th International Conference on Information and Communication Technology (ICoICT) (2021) 411–416, doi: 10.1109/ICoICT52021.2021.9527529.

[8] L. Carreira, et al., Low-Latency Reconfigurable Entropy Digital True Random Number Generator with Bias Detection and Correction, IEEE Transactions on Circuits and Systems I: Regular Papers, 67(5) (2020) 1562–1575. doi: 10.1109/TCSI.2019.2960694.

[9] D. Origines, A. Sison, R. Medina, A Novel Pseudo-Random Number Generator Algorithm based on Entropy Source Epoch Timestamp, International Conference on Information and Communications Technology (ICOIACT) (2019) 50–55. doi: 10.1109/ICOIACT46704.2019.8938509.

[10] B. Schneier, NIST's Post-Quantum Cryptography Standards Competition, IEEE Security & Privacy, 20(5) (2022) 107–108. doi: 10.1109/MSEC.2022.3184235.

[11] M. Herrero-Collantes, J. C. Garcia-Escartin, Quantum random number generators, Reviews of Modern Physics, 89 (2016).

[12] B. Perach, S. Kvatinsky, An Asynchronous and Low-Power True Random Number Generator using STT-MTJ, IEEE International Symposium on Circuits and Systems (ISCAS) (2020) 1–11. doi: 10.1109/ISCAS45731.2020.9181042.

[13] B. Narayanapuram, J. Panda, A New Side Channel Resistant Hybrid PUF Based Light Weight True Random Number Generator, in: IEEE 3rd International Conference on Technology, Engineering, Management for Societal Impact using Marketing, Entrepreneurship and Talent (TEMSMET) (2023) 1–6. doi: 10.1109/TEMSMET56707.2023.10150018.

[14] L. Huang, H. Zhou, K. Feng, Quantum Random Number Cloud Platform, NPJ Quantum Information 7(107) (2021). doi: 10.1038/ s41534-021-00442-x.

[15] M. Sharma, et al., Security on Cloud Computing Using Pseudo-random Number Generator Along with Steganography, Artificial Intelligence and Applied Mathematics in Engineering Problems, 43 (2020) 654–665. doi: 10.1007/978-3-030-36178-5_54.

[16] V. Shapoval, et al., Automation of Data Management Processes in Cloud Storage, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 410–418.

[17] Y. Martseniuk, et al., Automated Conformity Verification Concept for Cloud Security, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 25–37.

[18] Z. Hu, et al., Bandwidth Research of Wireless IoT Switches, in: IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (2020) 546–550. doi: 10.1109/tcset49122.2020.2354922.

[19] V. Masol, S. Popereshnyak, Joint Distribution of Some Statistics of Random Bit Sequences, Cybernetics and Systems Analysis, 57(1) (2021) 139–145.

[20] V. Masol, S. Popereshnyak, Checking the Randomness of Bits Disposition in Local Segments of the (0, 1)-Sequence, Cybernetics and Systems Analysis, 56(3) (2020) 513–520.