Research on a new hybrid random number sequence generator*

Serhii Yanushevskyi[1,†], Heorhii Prokhorov[1,†], Oleksandr Hres[1,†], Yurii Yaremchuk[2,*,†], and Volodymyr Lytvynenko[3,†]

[1] *Yuriy Fedkovych Chernivtsi National University, 2 Kotsiubynskoho Str., 58002 Chernivtsi, Ukraine*
[2] *Vinnytsia National Technical University, 95 Khmelnytsky highway str., 21021 Vinnytsia, Ukraine*
[3] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," 37 Beresteiskyi ave., 03056 Kyiv, Ukraine*

Abstract
The presented study reports the results of developing a hybrid random sequence generation system that combines both hardware and software components to achieve high performance and strong statistical quality. The proposed system integrates three primary components: a conventional webcam serving as a physical entropy source, cellular automata functioning as a nonlinear post-processing mechanism, and the Java SecureRandom library acting as the software-based random number generator. Such an architecture aims to merge the advantages of physical randomness with the reproducibility and speed of software generation. Previous research demonstrated that an ordinary webcam can be utilized to generate random bit sequences with a throughput exceeding 100 Mbit/s. Although these sequences exhibited significant randomness, their statistical parameters did not fully satisfy the core NIST standards. In particular, the value distribution of the generated elements lacked uniformity, despite displaying sufficiently chaotic behavior. While this shortcoming is not critical, further refinement was considered beneficial. To address this issue, linear cellular automata based on chaotic rules were incorporated. This modification improved the statistical quality of the output sequences but resulted in a noticeable reduction in performance.

Keywords
software engineering, cybersecurity, webcam, cellular automata, random number generation, NIST, encryption

1. Introduction

Random number generators (RNG) play a crucial role in modern cryptographic systems, as they serve as the fundamental source of unpredictability required to ensure the confidentiality, integrity, and authenticity of information. By producing data sequences that are statistically random and unpredictable, RNGs enable the generation of cryptographic keys, the establishment of secure network connections, and the maintenance of robust authentication mechanisms [1]. In practice, random numbers are applied in a wide range of domains—from the creation of session keys and digital signatures to load balancing, data integrity verification, initialization vectors, PIN and password generation, and simulation of stochastic processes in scientific and financial computations [2, 3].

Unlike the generation of single random values, the production of random number sequences introduces additional challenges and requirements. Beyond meeting the fundamental NIST SP 800-22 statistical standards, a high-quality random sequence generator must also demonstrate sufficient productivity rate to support real-time cryptographic operations. For practical implementations, a performance rate of at least 100 Mbps is typically considered the minimum threshold for effective integration into modern high-speed communication and security systems. Achieving this balance

* *CPITS-II 2025: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2025, Kyiv, Ukraine*
* Corresponding author.
† These authors contributed equally.

s.yanushevskyi@chnu.edu.ua (S. Yanushevskyi); g.prokhorov@chnu.edu.ua (H. Prokhorov); o.hres@chnu.edu.ua (O. Hres); yurevyar@vntu.net (Y. Yaremchuk); v.lytvynenko@kpi.ua (V. Lytvynenko)
0009-0007-3426-2868 (S. Yanushevskyi); 0000-0001-7810-2785 (H. Prokhorov); 0000-0002-8465-193X (O. Hres); 0000-0002-6303-7703 (Y. Yaremchuk); 0000-0002-1536-5542 (V. Lytvynenko)

between statistical quality and performance is one of the key objectives in the design of advanced RNG systems.

RNGs implemented purely in software are commonly referred to as pseudo-random number generators (PRNGs). These systems rely on deterministic algorithms, meaning that the sequence of generated numbers is completely determined by an initial value, or seed. Although the resulting sequences can appear random, they are not truly unpredictable if the underlying algorithm or seed is known. Nevertheless, software-based RNGs are widely used because they can fully comply with NIST requirements, ensuring excellent uniformity, randomness, and reproducibility under controlled conditions.

One of the major advantages of software-based generators is their predictable performance and scalability. Their output rate and efficiency depend primarily on the computational power of the system, including the CPU clock frequency, number of processing cores, and operating system efficiency in managing multithreaded tasks. Advanced implementations, such as those integrated into programming libraries like Java's SecureRandom or OpenSSL's RAND_bytes, can achieve very high throughput levels, making them suitable for most cryptographic applications..

2. Literature review

The deterministic nature of pseudo-random number generators (PRNGs) remains a fundamental limitation, particularly in applications that demand true unpredictability and high levels of cryptographic strength. Since PRNGs operate according to predefined mathematical algorithms, their output sequences are entirely determined by an initial value, or seed. Consequently, if an attacker is able to infer or reconstruct the internal state or the seeding mechanism of the generator, it becomes possible to predict all subsequent outputs, effectively compromising the entire cryptographic system.

This inherent vulnerability emphasizes the necessity of developing hybrid random number generation systems, which integrate software algorithms with physical entropy sources such as electronic or thermal noise, optical sensors, micro-phone input, or user interaction data. These physical process-es introduce a level of unpredictability that cannot be reproduced by deterministic algorithms alone, thereby enhancing the entropy and security of the generated sequences.

From a theoretical perspective, a PRNG based on mathematical computation can achieve an unbounded or effectively infinite sequence length, meaning it can continuously produce random numbers without entropy depletion. However, this advantage comes at the cost of predictability – since the generation process follows a known algorithmic model, its randomness is ultimately simulated rather than intrinsic. Thus, despite the potential for high performance and reproducibility, the primary drawback of purely algorithmic approaches lies in their susceptibility to reverse engineering and the absence of true physical randomness [4, 5].

True random number generators (TRNGs) generate random numbers from the inherent chaos of real physical stochastic processes, such as photocurrent, thermal noise in resistors and diodes, radioactive decay, etc. [6]. The main advantage is complete randomness and unpredictability.

The main limitations of true random number generators (TRNGs) stem from their relatively low performance rates and, in some cases, suboptimal statistical characteristics when compared to their software-based analogues. Because TRNGs rely on the measurement of physical phenomena—such as thermal noise, radioactive decay, or optical fluctuations—their generation speed is inherently constrained by the physical process itself and the sampling frequency of the measuring system. Moreover, these physical entropy sources often require precise calibration, environmental stability, and complex circuitry to ensure consistent output quality. As a result, TRNG devices tend to be costly, technically intricate, and difficult to maintain in long-term or large-scale implementations [7].

From the standpoint of software engineering and applied cryptography, this situation creates a strong motivation to investigate hybrid and adaptive methods of random sequence generation. Such approaches aim to combine the high entropy and unpredictability of hardware-based sources with the high performance, flexibility, and scalability of software algorithms. In recent years, research in this direction has focused on integrating physical entropy sources—for example, webcams, microphones, or hardware sensors—with software-based post-processing modules that refine and normalize the statistical properties of the output sequences.

Therefore, exploring new hardware—software architectures for random number generation is both theoretically significant and practically relevant. These hybrid systems can potentially overcome the trade-off between true randomness and computational efficiency, providing secure, high-speed, and statistically reliable random sequences suitable for modern cryptographic, data protection, and simulation applications [8, 9].

3. Main part

In previous studies [10, 11], the webcam was examined as a potential source of entropy for random number generation. The ability of a standard consumer webcam to produce image data at a rate of approximately 25 frames per second in SVGA resolution (800×600 pixels) provides a substantial data throughput, making it an attractive candidate for use in high-performance random sequence generators. Each captured frame contains numerous sources of micro-level randomness, including sensor noise, variations in lighting, and subtle environmental fluctuations, which together contribute to a significant level of entropy.

Subsequent experimental investigations have confirmed that webcam-based entropy generation exhibits strong randomness and high unpredictability of output sequences. Under optimal conditions, theoretical calculations suggest that a high-resolution webcam can achieve data rates of up to 5 Gbps, making it one of the most promising low-cost physical entropy sources for practical cryptographic and simulation applications.

Nevertheless, this approach is not without limitations. The statistical characteristics of the generated sequences, when evaluated according to NIST SP 800-22 standards, reveal certain deviations from ideal uniformity. Specifically, the distribution of generated elements tends to be non-uniform, although the overall sequence remains sufficiently chaotic and unpredictable, as illustrated in Figure 1. This imbalance does not completely disqualify the webcam as an entropy source but indicates the need for additional post-processing or normalization techniques to improve compliance with statistical quality requirements [12].

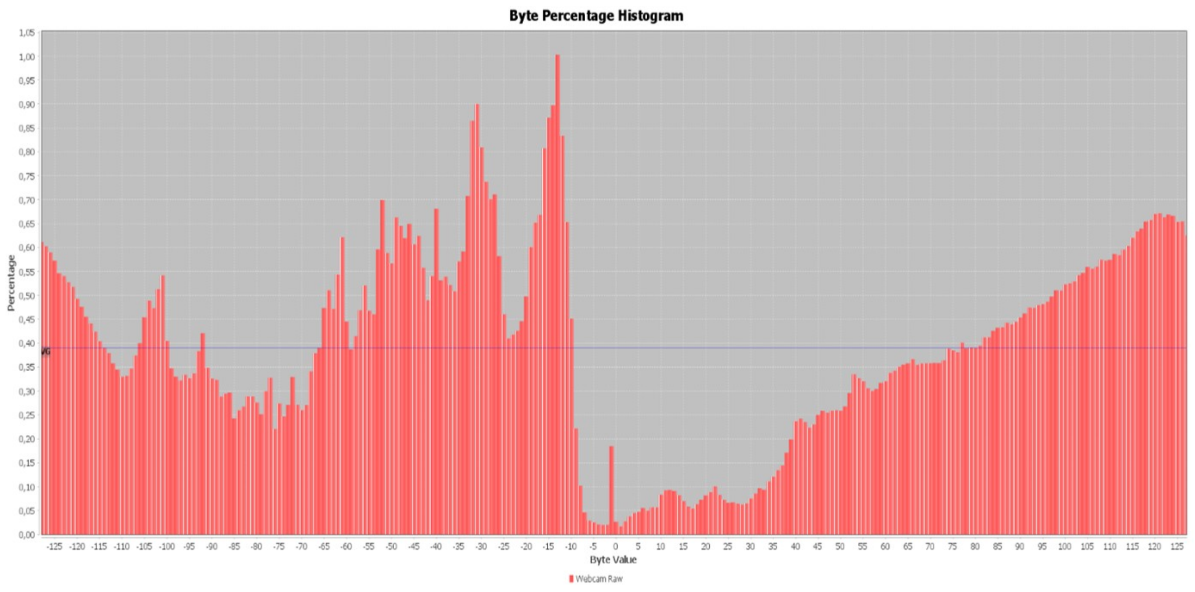


Figure 1: Histogram of typical distribution of elements of RNS obtained from one frame of web camera

The histogram shown in Figure 1 illustrates the value distribution of the random sequence generated using a webcam-based RNG. The diagram reveals that the occurrence frequency of certain byte values varies noticeably across the output stream. For instance, byte 15 appears in only 0.1% of cases, whereas byte 55 occurs approximately 1.0% of the time. Such deviations indicate that the overall distribution of values is not perfectly uniform, which suggests the presence of bias within the generated sequence. In some limited contexts, for example, non-cryptographic stochastic simulations, such irregularities may be considered acceptable. However, for applications that require strict compliance with NIST SP 800-22 statistical standards, this behavior is regarded as unsatisfactory and requires correction.

The overall shape of the distribution is chaotic and irregular, which in itself can be interpreted as a positive characteristic, since higher chaotic behavior often correlates with improved randomness and unpredictability. Nonetheless, this same property can also lead to unintended side effects. Specifically, the distinctive statistical pattern of the generated sequence may serve as a unique "fingerprint" of the particular webcam used, reflecting the inherent noise characteristics of its optical sensor and electronics. In certain scenarios, especially those involving privacy-sensitive or security-critical systems, such device-specific signatures may be undesirable, as they could potentially be exploited to identify or trace the entropy source.

Nevertheless, generating raw random number sequence (RNS) output in SVGA mode already demonstrates a data rate of up to 200 Mbps, which is a notable performance indicator for a low-cost, consumer-grade entropy source. By increasing the camera's resolution and frame capture rate, the achievable rate can be significantly improved. Theoretical estimations indicate that, under optimal conditions and with modern high-resolution webcams operating at full HD or higher frame rates, the data generation speed could potentially reach up to 5 Gbps. This performance level positions webcams as a promising foundation for the development of high-performance hybrid random sequence generators, combining physical entropy acquisition with software-based post-processing to achieve both speed and statistical robustness [13, 14].

To mitigate the disadvantages of raw random sequence generation, the computational power of cellular automata (CA) was employed. Among the well-known chaotic rules (30, 90, 105, 150), rule 30 was selected as the most chaotic one [15].

As demonstrated in previous research [16, 17], applying first 10–20 iterations of cellular automata–based post-processing is sufficient to adjust the distribution histogram of webcam-generated random sequences to a statistically satisfactory and nearly uniform shape, as shown in Figure 2.

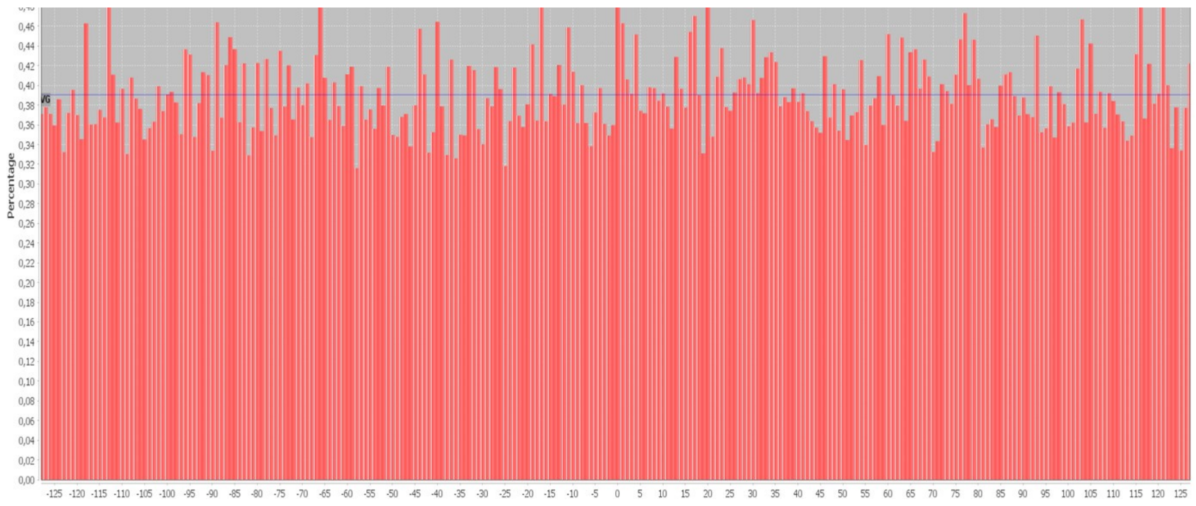


Figure 2: Histogram of value distribution of elements of RNS obtained from one frame of web camera and processed by CA-30 (10 iterations)

Figure 3 indicates that the resulting distribution histogram closely approximates the normal (Gaussian) distribution for continuous values or the Poisson distribution for discrete ones.

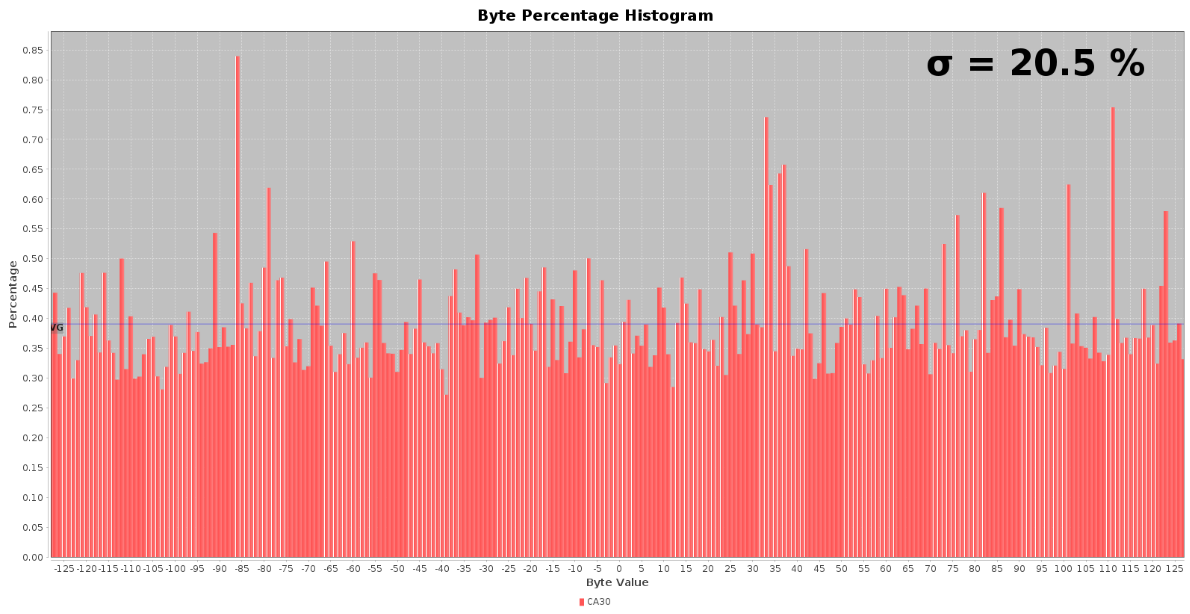
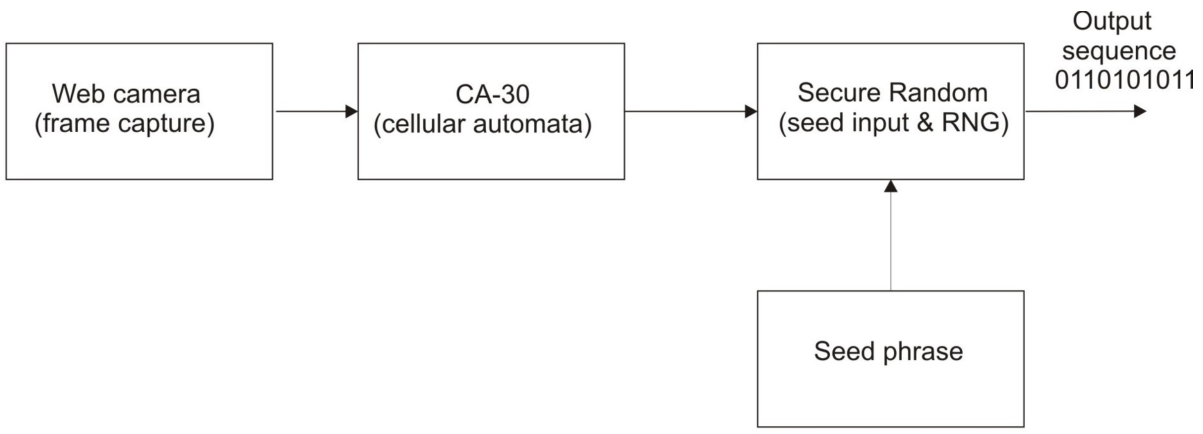


Figure 3: Histogram of value distribution of elements of RNS obtained from one frame of web camera and processed by CA-30 (20 iterations)

However, this improvement in statistical quality comes at the cost of performance degradation. Each iteration introduces additional computational overhead, significantly increasing the total processing time. Experimental results show that after approximately 10 iterations, the overall generation speed of the system decreases by about an order of magnitude compared to the initial configuration. In the specific experimental setup, SVGA resolution at 25 frames per second, the effective throughput was reduced from approximately 200 Mbps to around 20 Mbps.

4. Hybrid generation algorithm

This paper proposes a hybrid approach to pseudo-random number sequence generation based on the integration of a webcam, cellular automata, and the *SecureRandom* library. The structural diagram of the proposed hybrid generator RNG is given



Web camera (frame capture)
CA-30 (cellular automata)
Secure Random (seed input & RNG)
Output sequence 0110101011
Seed phrase

Figure 4: The schema of an aforementioned hybrid algorithm

A random number generation module based on a webcam. A post-processing mechanism for the generated sequences using chaotic-type cellular automata. A software pseudo-random number generator based on the SecureRandom function (with the use of initial seed conditions).

The extraction method for random number sequences (RNS) from webcam frames is detailed in [11, 14]. The received sequence of bits from the webcam is post-processed according to certain rules of cellular automata, after which the obtained sequences act as initial conditions for generating sequences of random numbers using the SecureRandom library and the function...

For this purpose, the research employs the built-in Java SecureRandom library—a well-established component designed to provide cryptographically strong random number generation. SecureRandom serves as a core element of the Java Cryptography Architecture (JCA) and can integrate multiple entropy sources and algorithms depending on the selected security provider. This study presents a comprehensive analysis of the library's security features, internal mechanisms, and entropy acquisition methods, as described in the literature [18, 19].

The most reliable method of random number sequence (RNS) generation involves using a pre-generated list of random numbers as a seed. In this investigation, the seed list was derived from the aforementioned hybrid generation method:

```
SecureRandom secure = new SecureRandom();
byte[] randomsFromWebCam = new byte[440];
secure.setSeed(randomsFromWebCam);
```

The Java code fragment presented above illustrates the initial implementation of the SecureRandom object, which serves as the core component of the proposed random sequence generation system. The SecureRandom class is part of the standard Java Development Kit (JDK) and therefore requires no additional Maven dependencies or external libraries, ensuring full portability and seamless integration across various software environments [20–22].

To evaluate the quality and statistical robustness of the generated sequences, the NIST SP 800-22 test suite was employed. The results of these tests serve as a benchmark for comparing software-based randomness with the hybrid webcam–cellular automata–SecureRandom approach proposed in this study.

In this implementation, the value 440 is used as a so-called magic number, representing the minimum seed length required to satisfy the NIST SP 800-22 standard for adequate entropy initialization. This seed ensures that the generator starts with a sufficiently large pool of random bits, providing a robust foundation for producing statistically sound and cryptographically secure random sequences. The chosen seed length aligns with established cryptographic best practices, achieving a balance between security, performance, and compatibility within existing Java-based security frameworks.

The paper presents a study of sequences generated by the secure random function. The study covers 10 RNS with a volume of 100 MB (20). The results of the study are presented in Table 1.

In this work, the sequences generated by a hybrid generator were tested using NIST tests. For this purpose, an array of 100 sequences, each 100 MB in size, was generated. The research results are presented in Table 2.

For the experimental analysis, 100 random number sequences (RNS) were generated using the aforementioned hybrid algorithm shown on Figure 4. The size of each RNS is equal to 100 MB.

According to the experimental results, the proposed hybrid generation method, unlike a generator, based on a webcam and a secure random function, successfully produces statistically secure random sequences. As demonstrated by the study, out of the 15 standard tests included in the NIST SP 800-22 suite, all tests were passed at a high level, confirming the overall reliability and robustness of the generated sequences.

In this study, a 440-byte seed was employed, which satisfies the minimum entropy requirement but may still limit performance in certain high-complexity tests. Therefore, additional research is necessary to determine the optimal trade-off between seed size, generation throughput, and the overall statistical quality of the produced random number sequences.

Table 1
NIST testing results (WebCam & SecureRadom)

#	Test name	True Percentage
1	Frequency (Monobit)	100
2	Block Frequency	100
3	The Runs Test	100
4	Longest Run of Ones	100
5	Binary Matrix Rank	100
6	Discrete Fourier	100
7	Non-overlapping Template Matching	20
8	Overlapping Template Matching	80
9	Universal Statistical	100
10	Linear Complexity	100
11	Serial	100
12	Approximate Entropy	100
13	Cumulative Sums	100
14	Random Excursions	50
15	Random Excursions Variant	60

The obtained NIST testing results are presented in Table 2.

Table 2
NIST testing results Hybrid generator (WebCam & CA & SecureRandom)

#	Test name	True Percentage
1	Frequency (Monobit)	100
2	Block Frequency	100
3	The Runs Test	100
4	Longest Run of Ones	100
5	Binary Matrix Rank	100
6	Discrete Fourier	100
7	Non-overlapping Template Matching	96
8	Overlapping Template Matching	96
9	Universal Statistical	100
10	Linear Complexity	100
11	Serial	100
12	Approximate Entropy	100
13	Cumulative Sums	100
14	Random Excursions	96
15	Random Excursions Variant	96

In the conducted experiments, the generation performance of the SecureRandom-based implementation reached approximately 2 Gbps, confirming that the proposed hybrid approach effectively combines high statistical reliability with exceptional data throughput.

The research results indicate that the proposed pseudo-random sequence generation method can be applied to the development of cryptographically secure encryption techniques and used in various cryptographic applications.

5. Information encryption method based on a new Hybrid PRNG

This paper also provides an example of using the developed hybrid RNG generator in information encryption methods. Figure 5 shows a diagram of the proposed method of information encryption using a hybrid RNG generator. The proposed method is implemented in Java.

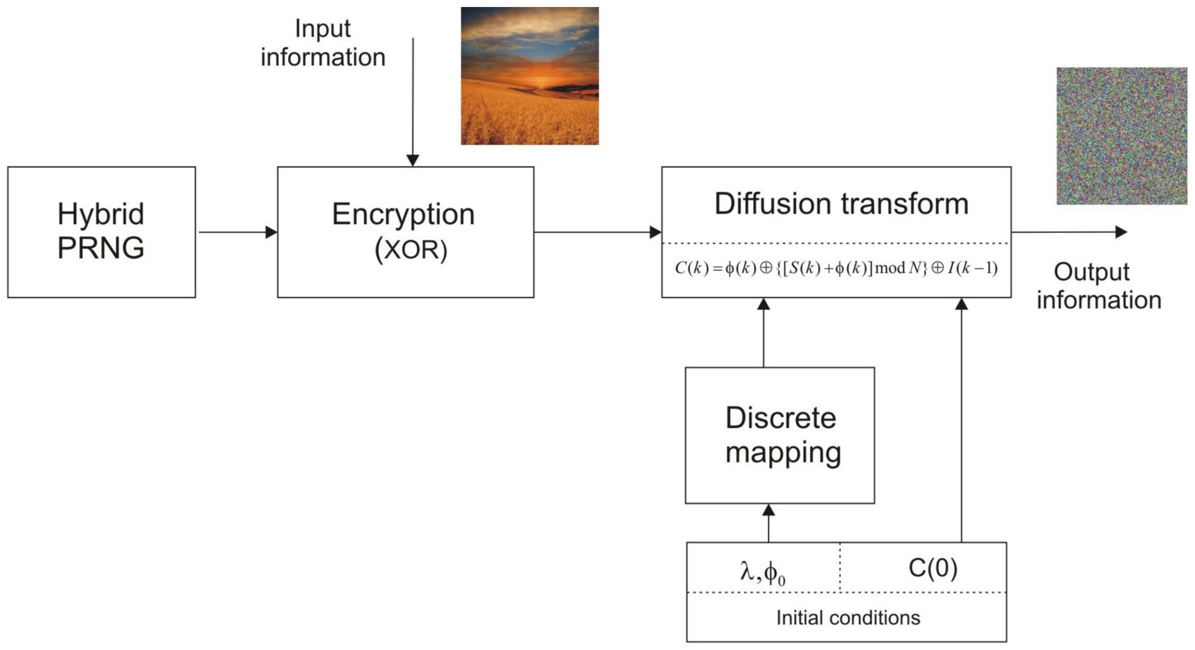


Figure 5: Scheme for implementing the information encryption method based on a hybrid RNG generator

The proposed method is implemented in Java. The encryption algorithm flowchart is shown in Figure 5.

Research on the method using the example of encrypting different types of information. In the case of text message encryption, the original message is converted into 8-bit sequences according to the ASCII code.

The proposed method also allows encrypting graphic information (images). When encrypting images, the R, G, and B color gradations are read from each pixel of the image and represented as a sequence of 8-bit binary numbers.

The generated key sequence of real numbers is converted into an 8-bit binary representation using the following formula [24]:

$$z_n = 2^{-1} b_{n1} + 2^{-2} b_{n2} + \ldots + 2^{-L} b_{nL} \tag{1}$$

where L is binary representation length (L = 8).

In the first stage of the algorithm, bitwise addition of the information and key sequences is performed using the XOR operation.

To increase the impact of changes in the input data on the encrypted data, a modified diffusion transformation was used as an additional encryption stage, which ensures the dependence of the values of the previous bytes of the input information on the subsequent bytes of the encrypted information. The diffusion transformation is determined by the following formula:

$$C(k)=\varphi(k)\oplus\{[S(k)+\varphi(k)]\bmod N\}\oplus I(k-1) \tag{2}$$

where $I(k-1)$ is the previous byte of input data, $\varphi(k)$ is a chaotic function, for which either a logistic mapping or a modified cosine mapping can be used:

$$\varphi(k+1)=3{,}99\,\varphi(k)[1-\varphi(k)] \tag{3}$$

The reverse diffusion process can be described by the equation:

$$S(k)=\{\varphi(k)\oplus C(k)\oplus I(k-1)+N-\varphi(k)\}\bmod N \tag{4}$$

The so-called additional parameters for diffusion conversion are two additional values: $C(0)$ and $\varphi(0)$, the so-called additional parameters for diffusion conversion are two additional values.

The decryption process is performed in reverse order with the same initial condition and control parameter values for discrete mappings used in encryption, since this encryption algorithm is symmetric. The method was tested using color image encryption as an example.

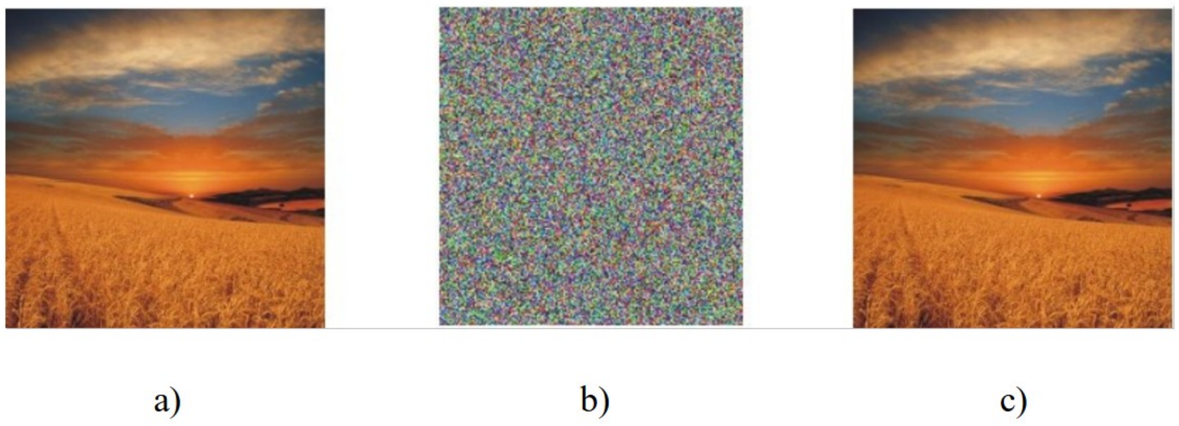


Figure 6: Initial a) encrypted b) and decrypted c) image

Based on the results of image encryption, it can be concluded that a modified diffusion transformation with an additional parameter for encryption should be used as the final stage of encryption. with different values of which, when encrypting and decrypting information, there is no identity between the transmitted and received information, indicating the high cryptographic strength of the developed method.

6. Conclusions

This paper proposes a hybrid RNG generator based on a combination of sequences generated by a web camera, chaotic cellular automata, and the SecureRandom function.

The use of chaotic cellular automata as a mechanism for post-processing sequences generated by a web camera has significantly improved the statistical properties of the generated sequences, and the use of random sequences generated by a web camera as the initial entropy value for the software generator, in particular the Java SecureRandom library, made it possible to create a hybrid generator of pseudo-random sequences that effectively combines the unpredictability of hardware with the efficiency of software, ensuring stable and continuous generation of random sequences.

Research on sequences generated by a hybrid generator using the NIST SP 800-22 test suite showed that all tests were passed at a high level, confirming the overall reliability and stability of the generated sequences. Therefore, this hybrid pseudorandom sequence generator can be used to build cryptographically secure encryption methods and for use in cryptographic applications.

The paper also proposes a method for encrypting information using this hybrid generator as a key sequence generator.

Based on the results obtained using this method on the example of color images encryption, it can be concluded that when encrypting and decrypting information, there is no identity between the transmitted and received information, which indicates the high cryptographic strength of the developed method.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] P. Petriv, I. Opirskyy, N. Mazur, Modern Technologies of Decentralized Databases, Authentication, and Authorization Methods, in: Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3826, 2024, 60–71.

[2] S. Popereshnyak, Y. Novikov, Y. Zhdanova, Cryptographic System Security Approaches by Monitoring the Random Numbers Generation, in: Cybersecurity Providing in Information and Telecommunication Systems II (CPITS-II), 3826, 2024, 301–309.

[3] D. Proskurin, et al., Hybrid RNN-CNN-based Model for PRNG Identification, in: Classic, Quantum, and Post-Quantum Cryptography (CQPC), 3829, 2024, 47–53.

[4] F. Martinez, Attacks on Pseudo Random Number Generators Hiding a Linear Structure, in: S. D. Galbraith (Ed.), Topics in Cryptology – CT-RSA, Lect. Notes Comput. Sci., vol. 13161 (2022) 145–168.

[5] M. Cornejo, S. Ruhault, (In)security of Java SecureRandom Implementations, in: Journées Codage et Cryptographie, 2014. https://www-fourier.ujf-grenoble.fr/JC2/exposes/ruhault.pdf

[6] S. Park, B. G. Choi, T. Kang, K. Park, Y. Kwon, J. Kim, Efficient Hardware Implementation and Analysis of true Random-Number Generator based on Beta Source, ETRI J., 42(4) (2020) 518–526. doi:10.4218/etrij.2020-0083

[7] T. Toffoli, N. Margolus, Cellular Automata Machines, MIT Press, Cambridge, MA, 1987.

[8] S. Ostapov, et al., Symmetrical Cryptosystems based on Cellular Automata, Int. J. Comput., 22(1) (2023) 15–20. doi:10.47839/ijc.22.1.2874

[9] H. Prokhorov, D. Trembach, Research into the Efficiency of Processing a Numerical Random Sequence by Chaotic-Type Cellular Automata, SISIOT, 2(2) (2024) 02010. doi:10.31861/sisiot2024.2.02010

[10] D. V. Hanzhelo, G. V. Prokhorov, Investigation of Inter-Frame Correlation of Webcam-generated Chaos, Meas. Comput. Devices Technol. Process., 2 (2024) 154–159. doi:10.31891/2219-9365-2024-78-18

[11] R. Diachuk, Y. Dobrovolsky, D. Hanzhelo, H. Prokhorov, D. Trembach, Research of the Level of Chaoticity and Reliability in Webcam-Generated Random Number Sequences, SISIOT, 2(1) (2024) 01004. doi:10.31861/sisiot2024.1.01004

[12] National Institute of Standards and Technology, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Spec. Publ. 800-22 Rev. 1, Aug. 2008.

[13] Y. Dobrovolsky, et al., Development of a Hash Algorithm based on Cellular Automata and Chaos Theory, East.-Eur. J. Enterp. Technol., 5(9) (2021) 48–55. doi:10.15587/1729-4061.2021.242849

[14] V. Maksymovych, O. Harasymchuk, I. Opirskyy, The Designing and Research of Generators of Poisson Pulse Sequences on base of Fibonacci Modified Additive Generator, in: Advances in

Intelligent Systems and Computing, Springer International Publishing, Cham, 2018, 43–53. doi:10.1007/978-3-319-91008-6_5

[15] D. Hanzhelo, H. Prokhorov, Investigation of Statistical Characteristics of Numerical Random Sequence Obtained from a Web Camera Frame, Herald Khmelnytskyi Natl. Univ. Techn. Sci., 337(3/2) (2024) 46–51. doi:10.31891/2307-5732-2024-337-3-6

[16] Oracle, Java SE 8 Documentation, Class SecureRandom, 2021. https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html

[17] K. İnce, Security analysis of Java SecureRandom library, in: 2nd Int. Conf. Access Recent Adv. Eng. Digitization (ARACONF), 2021. doi:10.31590/ejosat.900956

[18] F. J. Ibáñez-López, M. Rubio-Aparicio, M. Pedreño-Plana, M. Sánchez-Martín, Descriptive Statistics and Basic Graphs Tutorial to Help You Succeed in Statistical Analysis, Espiral. Cuad. Profesorado, 17(36) (2024) 88–99. doi:10.25115/ecp.v17i36.9570

[19] L. Crocetti, P. Nannipieri, S. Di Matteo, L. Fanucci, S. Saponara, Review of Methodologies and Metrics for Assessing the Quality of Random Number Generators, Electronics, 12(3) (2023) 723. doi:10.3390/electronics12030723

[20] A. Jammi, Y. Raju, S. Munishankaraiah, K. Srinivas, Steganography: An Overview, Int. J. Eng. Sci. Technol., 2(10) (2010) 5985–5992.

[21] Oracle, Class SecureRandom, Java Platform Standard Edition 8 Documentation. https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html

[22] C.-H. Hsieh, X. Yao, Q. Zhang, M. Lv, R. Wang, B. Ni, BCsRNG: A Secure Random Number Generator based on Blockchain, IEEE Access, 10 (2022) 98117–98126. doi:10.1109/ACCESS.2022.3206450

[23] G. V. Prokhorov, R. L. Diachuk, M. G. Hanzhelo, S. V. Yanushevskyi, New Approaches to the Design of a Hybrid Random Sequence Generator, in: Phys. Technol. Probl. Transm. Process. Storage Inf. Infocommun. Syst.: 10th Int. Sci. Pract. Conf., Chernivtsi, 2025, 134–135.

[24] O. V. Hres, M. I. Skrypskyi, V. M. Kosovan, H. M. Rozorinov, Research of Pseudorandom Sequence Generators based on Discrete Mappings, Herald Khmelnytskyi Natl. Univ. Techn. Sci., 4 (2017) 243–251.