Security of Infocommunication Systems and Internet of Things

2025 Vol 3, No 1

https://doi.org/10.31861/sisiot2025.1.01007

Received 10 June 2025; revised 25 June 2025; accepted 29 June 2025; published 30 June 2025

Investigation of Statistical Characteristics of Random Number Sequences Generated by a Webcam Using Cellular Automata Functionality and NIST Patterns

Heorhii Prokhorov[1], Mariia Hanzhelo[1], Rostyslav Diachuk[1] and Serhii Yanushevskyi[1,*]

[1]Department of software computer system, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: s.yanushevskyi@chnu.edu.ua)

ABSTRACT At the current stage of development, software-based random number generation is increasingly vulnerable to attacks due to the growing computational power of modern systems. Hardware-based generation relies on reliable stochastic physical phenomena, yet often demonstrates low throughput or an insufficient level of statistical quality. In this study, linear cellular automata are utilized as a post-processing mechanism aimed at improving key statistical properties of the random number sequences. The study presents the results of improving the statistical properties of number sequences obtained from a standard webcam with regard to one of the key cryptographic requirements – uniform distribution of values. Prior analyses revealed that inherent stochastic behavior within webcam sensors produces irregular, non-uniform distributions in the output sequences. This limitation can be mitigated by post-processing using linear cellular automata, specifically employing rules 30, 90, 105, and 150. To evaluate the effectiveness of this method, a comparative analysis was conducted against software-generated sequences produced via the Java SecureRandom class. It is shown that by applying a sufficient number of iterations, it is possible to achieve the required level of uniformity in the distribution of sequence elements. However, the generated sequences did not fully pass the NIST test suite. Statistical uniformity was assessed efficiently using lightweight Java-based libraries, enabling rapid integration even on low-performance hardware such as smartphones. The research findings can be applied in the design of a hardware-based random number generator.

KEYWORDS software engineering, cybersecurity, random number sequences, webcam, cellular automata.

I. INTRODUCTION

The generation of random numbers is fundamental to cryptographic mechanisms, including key creation, secure password generation, and authentication codes. Such generation is subject to a set of requirements defined by standards such as NIST and BSI. However, when it comes specifically to generating random number sequences (RNS), these requirements reach a new qualitative level: they do not override the existing ones but rather introduce additional constraints.

A key challenge in modern software engineering is the generation of RNS with a throughput of at least 100 Mbit/s, with an ideal target of up to 1 Gbit/s. This requirement is fundamental for dynamic steganography – the creation of secure communication channels in which not only the transmitted data are protected, but also the very fact of transmission itself [1]. In such systems, encrypted data are injected into the RNS and transmitted in this form over the communication channel. Therefore, RNS generators must meet stringent criteria that make it extremely difficult not only to extract meaningful content but also to detect the presence of data transmission at all. Ideally, the secure channel constantly transmits random number sequences at a speed of no less than 100 Mbit/s, operating 24/7.

This paper builds upon the findings presented in our earlier work, in which we explored webcam-generated random number sequences and the impact of cellular automata processing [2, 3]. The present study extends that foundation by incorporating a comparative evaluation using NIST statistical tests and assessing the feasibility of practical implementation in resource-constrained environments.

II. THE WEBCAM AS A DRIVER OF RNS GENERATION

The technique for extracting a random number sequence from a webcam frame, previously introduced in, has been refined for the current study and used as a baseline for comparative analysis [3].

Previous studies have demonstrated that a standard webcam operating in SVGA mode (800 × 600) is capable of generating 1 440 000 signed byte values per frame in the range [-128…+127], which corresponds to 11 520 000 bits [2, 3].

Extrapolating this to a frame rate of 25 frames per second yields a throughput of nearly 300 Mbit/s.

An experimental analysis of the correlation between two consecutive frames (0.04 s apart) showed a correlation of up to 30% in complete darkness and less than 10% under normal lighting conditions. This indicates a high entropy (randomness) of the generated RNS elements and a very long recurrence period – practically infinite.

However, the statistical characteristics were found to be not entirely satisfactory. In particular, the value distribution of the elements was not uniform, which contradicts the cryptographic requirement for randomness.

For comparison, a reference RNS was generated using the SecureRandom library in the Java programming language, which demonstrated a clearly uniform distribution of values [3]. The deviation from the mean was

just 1.2%. Such sequences fully meet all statistical requirements, with the exception of unpredictability.

For the case of a frame captured under normal lighting conditions, the value distribution histogram is shown in Fig. 1.

As seen in Fig. 1, the histogram of a typical frame's value distribution is neither uniform nor Gaussian. This indicates unpredictability, but not true randomness. Certain byte values – for instance, within the range [0...25] – are completely absent. As a result, standard deviation analysis is not applicable in this case.

To address this issue, post-processing of the RNS using chaotic-type cellular automata (CA) was proposed, in particular, the "Rule 30" method [4, 5].

In terms of Boolean algebra, this method is expressed as:

$$result = left\ XOR\ (center\ OR\ right). \quad (1)$$

After processing the RNS with this functionality, the resulting histogram is shown in Fig. 2.

As shown in Fig. 2, the histogram exhibits a distribution that can be described as "chaotically uniform". The deviation from the mean is approximately 12%. With further processing – specifically 50 iterations – the histogram becomes almost indistinguishable from that of SecureRandom (the ideal case).

The time required for a single iteration was measured. For SVGA mode (11 520 000 bits), one iteration takes approximately 60 microseconds. Therefore, applying 10 iterations slows down the generation process by roughly a factor of 16.

The cellular automata functionality was implemented in Java using low-level bitwise operations provided by the core language. This approach operates at the kernel level and requires minimal memory and processing time. Therefore, it is entirely feasible to utilize all available CPU cores, allowing the processing time to be reduced proportionally through parallelization [6].

Due to the simplicity of their operation, cellular automata are often referred to as "crypto-primitives", which makes it possible to implement this functionality in virtually any programming language including Assembly, C, or Python even on microcontrollers such as Arduino.

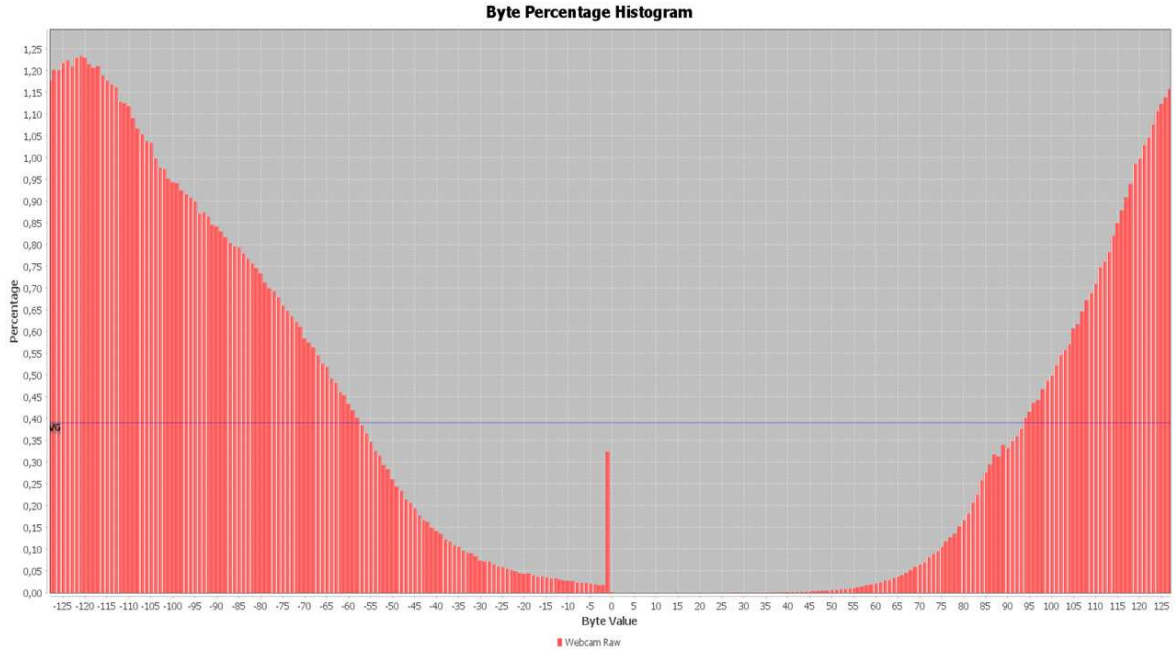

FIG. 1. Histogram of RNS element distribution obtained from a webcam.

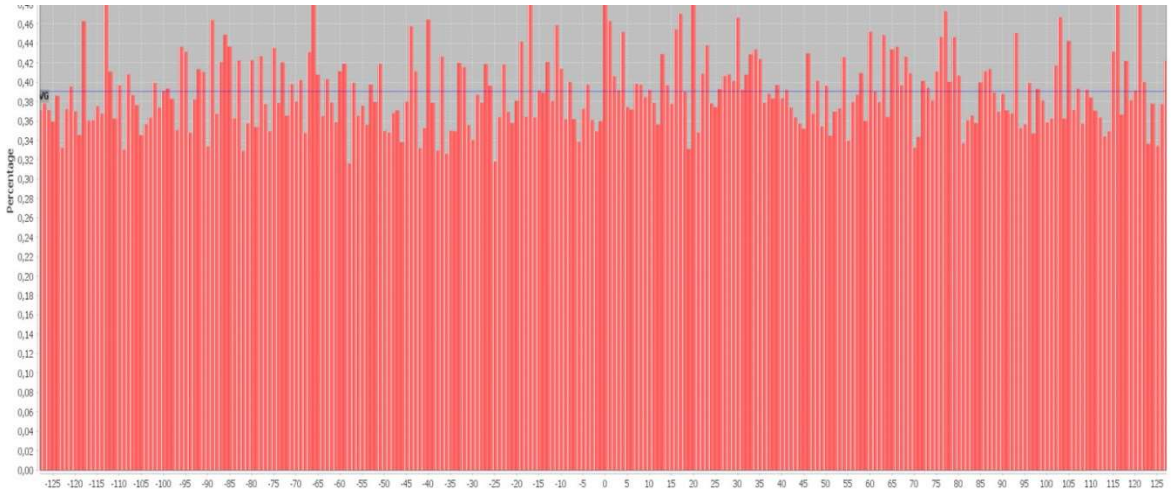

FIG. 2. Histogram of RNS element distribution after CA-30 processing (10 iterations).

In this experiment, no parallel computing techniques were applied in order to preserve a clean conceptual model, which can later be optimized for specific tasks and computational resources.

III. NIST TESTS

The suite includes 15 statistical tests [7]. The RNS under evaluation contained 13 million random byte elements (equivalent to 100 Mbit), generated by the webcam and processed through 10 iterations of CA-30.

TABLE 1. Preliminary NIST Test Results.

Test Name	Test Passed (%)
Frequency (Monobit)	PASSED
Block Frequency	PASSED
Cumulative Sums	PASSED
Longest Run of Ones	PASSED

The remaining 11 tests were not passed. For more accurate statistical evaluation, testing should be conducted on at least 100 such RNS samples.

IV. OPTIMIZATION OF THE GENERATION PROCESS

Despite the fact that the generated RNS do not pass all NIST tests, they possess a critically important property – high throughput.

In certain scenarios involving the transmission of encrypted data – such as dynamic steganography – the relevance or validity of the transmitted information may have a very short lifespan. At the same time, the process of breaking cryptographic protection may require a significantly longer period. Consequently, in such cases, the quality of the RNS may be far from ideal, yet still acceptable in terms of cryptographic security, provided that the generation process offers sufficiently high throughput.

As noted earlier in Section II, the throughput of a webcam operating in SVGA mode (800 × 600) was measured at nearly 300 Mbit/s. This value was obtained experimentally under laboratory conditions using a standard consumer-grade webcam, and serves to confirm the proof-of-concept feasibility.

If a more powerful webcam is used for the experiment – for example, the Anker PowerConf Web Camera C200, which supports Quad HD mode (2560 × 1440) at 50 Hz – the theoretical throughput increases to the following value:

$$P = 2560 \times 1440 \times 3 \times 8 \times 50 = 4.4\ \text{Gbit/s}. \quad (2)$$

In this mode (again, theoretically), the generator is capable of operating continuously in 24/7 mode.

There exist webcams with even higher resolutions (commonly used in security and surveillance systems). In such cases, the generation throughput may be limited solely by the bandwidth of the computer's USB interface – typically in the range of 20 – 40 Gbit/s.

The data presented refer to a single webcam and a single USB port. Under horizontal scaling (by increasing the number of webcams and USB ports) the overall throughput increases exponentially.

As previously noted, modern cryptographic applications (particularly steganography) typically require throughput in the range of 0.1 Mbit/s to 1.0 Mbit/s. Therefore, it becomes theoretically feasible to integrate a quality assessment module into the random number generation process. Alternatively, in the case of Java, parallel computation can be employed.

This does not imply the full suite of 15 NIST tests, but rather the use of basic statistical checks – for example, those provided by the DescriptiveStatistics library in the Java programming language [8]. In this case, it is sufficient to instantly calculate the standard deviation of the generated sequence to obtain an approximate estimate of RNS quality. If the quality is found to be unsatisfactory, the sequence is post-processed using cellular automata methods as previously described [9].

In an ideal theoretical scenario, where steganographic encoding requires 100 Mbit/s [10] while the generator is capable of delivering 40 Gbit/s, there exists a 400-fold time reserve for both dynamic quality control and post-processing using cellular automata.

In a practical scenario, a standard USB port provides a data rate of only 480 Mbit/s, which allows for several dozen processing iterations to be performed in order to improve the statistical characteristics of the sequence.

If the quality control process relies on the DescriptiveStatistics library, the cellular automata-based post-processing can be implemented on even the simplest microcontroller, such as an Arduino. A standard modern smartphone running Android already includes the DescriptiveStatistics library within the OS kernel, which further simplifies the implementation of dynamic quality evaluation.

V. CONCLUSION

Compared to our previous work [3], the current results demonstrate a more rigorous validation framework through partial NIST testing and practical throughput measurements.

The use of cellular automata significantly slows down the process of generating random number sequences. Even with a relatively small number of iterations (10), the data processing reduces the sequence generation speed, which can be critical for high-speed systems.

In SVGA mode with 10 iterations of CA-30 processing, the throughput reaches 20 Mbit/s.

Despite the performance reduction, cellular automata improve the uniformity of random sequences.

The NIST tests are passed only partially. This remains a subject for further research and refinement.

High generation throughput can compensate for suboptimal RNS quality.

The bitwise operations used in CA methods enable the implementation of this functionality on even the simplest microcontrollers.

AUTHOR CONTRIBUTIONS

H.P. – conceptualization, investigation, supervision; M.H., R.D. – software, resources, writing-original draft preparation, visualization, validation; S.Y. – methodology, conceptualization, writing-review and editing, supervision, investigation.

COMPETING INTERESTS

The authors declare that they have no competing interests.

REFERENCES

[1] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, and L. E. Bassham III, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Spec. Publ. 800-22 Rev. 1a, Apr. 2010.

[2] D. Dobrovolsky, D. Hanzhelo, H. Prokhorov, and D. Trembach, "Research the level of chaotic and reliability in webcam-generated random number sequences," *SISIOT*, vol. 2, no. 1, p. 01004, Aug. 2024, doi: 10.31861/sisiot2024.1.01004.

[3] D. Hanzhelo and H. Prokhorov, "Investigation of statistical characteristics of numerical random sequence obtained from a web camera frame," *Herald of Khmelnytskyi Natl. Univ. Techn. Sci.*, vol. 337, no. 3(2), pp. 46–51, 2024, doi: 10.31891/2307-5732-2024-337-3-6.

[4] S. Ostapov, B. Diakonenko, M. Fylypiuk, K. Hazdiuk, L. Shumyliak, and O. Tarnovetska, "Symmetrical cryptosystems based on cellular automata," *Int. J. Comput.*, vol. 22, no. 1, pp. 15–20, Mar. 2023, doi: 10.47839/ijc.22.1.2874.

[5] T. Toffoli and N. Margolus, *Cellular Automata Machines*. Cambridge, MA: MIT Press, 1987.

[6] H. Prokhorov and D. Trembach, "Research into the efficiency of processing a numerical random sequence by chaotic-type cellular automata," *SISIOT*, vol. 2, no. 2, p. 02010, Dec. 2024, doi: 10.31861/sisiot2024.2.02010.

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

[8] F. J. Ibáñez-López, M. Rubio-Aparicio, M. Pedreño-Plana, and M. Sánchez-Martín, "Descriptive statistics and basic graphs tutorial to help you succeed in statistical analysis," *Espiral. Cuadernos del Profesorado*, vol. 17, no. 36, pp. 88–99, 2024, doi: 10.25115/ecp.v17i36.9570.

[9] L. Crocetti, P. Nannipieri, S. Di Matteo, L. Fanucci, and S. Saponara, "Review of methodologies and metrics for assessing the quality of random number generators," *Electronics*, vol. 12, no. 3, p. 723, 2023, doi: 10.3390/electronics12030723.

[10] A. Jammi, Y. Raju, S. Munishankaraiah, and K. Srinivas, "Steganography: an overview," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 10, pp. 5985–5992, 2010.

Heorhii Prokhorov

He had received a Ph.D. in physics and mathematics in 2006. Now is an Associate Professor at Department of software computer system, Yuriy Fedkovych Chernivtsi National University. His research interests include cryptography, coding theory, hardware random number sequences generation.

ORCID ID: 0000-0001-7810-2785

Mariia Hanzhelo

Digital Internationalization adviser and supported ministry of Digital transformation of Ukraine. Heading IT company InDevLab. Now is an PhD student at Department of software computer system, Yuriy Fedkovych Chernivtsi National University. Research interests: Engineering, Cryptography and technology development.

ORCID ID: 0000-0002-6312-740X

Rostyslav Diachuk

Had received BS and MS degrees in software engineering.
Now is a PhD student at Department of software computer system, Yuriy Fedkovych Chernivtsi National University. Research interests: Cryptography, Software development and information protection systems in networks.

ORCID ID: 0000-0002-6259-3302

Serhii Yanushevskyi

Had received BS in Software Engineering and MS degrees in Computer Systems Analyst. Now is an Assistant Professor at Department of software computer system, Yuriy Fedkovych Chernivtsi National University. Research interests: Cryptography, Cybersecurity, Software, Blockchain Technology, Cellular Automata.

ORCID ID: 0009-0007-3426-2868

Дослідження статистичних характеристик послідовностей випадкових чисел згенерованих веб-камерою за функціоналом клітинних автоматів та патернами NIST

Георгій Прохоров[1], Марія Ганжело[1], Ростислав Дячук[1], Сергій Янушевський[1,*]

[1]Кафедра програмного забезпечення комп'ютерних систем, Чернівецький національний університет імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: s.yanushevskyi@chnu.edu.ua)

АНОТАЦІЯ На сучасному етапі програмна генерація випадкових чисел піддається ризику злому через зростання обчислювальної потужності сучасних систем. Генерація апаратного типу базується на надійних стохастичних фізичних явищах, але забезпечує низьку продуктивність або недостатній рівень статистичних показників. Сучасні вимоги до

швидкодії генератора випадкових чисел починаються з величини 100 Мбіт/с. У цьому дослідженні лінійні клітинні автомати використовуються як механізм пост-обробки, спрямований на покращення ключових статистичних властивостей послідовностей випадкових чисел. У дослідженні наведено результати покращення статистичних характеристик послідовності чисел, отриманих зі звичайної веб-камери, щодо дотримання однієї з вимог криптостійкості: рівномірного розподілу елементів за значенням. Попередні аналізи показали, що притаманна стохастична поведінка в світлочутливих елементів веб-камер призводить до неоднорідних розподілу елементів у вихідних послідовностях. Цю перешкоду можна подолати, використовуючи обробку лінійними клітинними автоматами, зокрема правилами 30, 90, 105. 150. Для оцінки ефективності цього методу було проведено порівняльний аналіз із програмно-згенерованими послідовностями, отриманими за допомогою класу Java SecureRandom. Показано, що шляхом вибору кількох ітерацій можна отримати необхідний рівень рівномірності розподілу елементів послідовності за значенням. Проте тести NIST повністю успішно пройти не вдалось. Статистичну однорідність було ефективно оцінено за допомогою маловимогливих бібліотек DescriptiveStatistics мови програмування Java, що дозволило реалізувати швидку інтеграцію функціоналу навіть на низькопродуктивному обладнанні, такому як смартфон з операційною системою Android. Результати дослідження можуть бути використані при проектуванні високошвидкісного апаратного генератора послідовності випадкових чисел.

КЛЮЧОВІ СЛОВА програмна інженерія, кібербезпека, послідовності випадкових чисел, веб-камера, клітинні автомати.