Security
of Infocommunication Systems
and Internet of Things

2024 Vol 2, No 2

https://doi.org/10.31861/sisiot2024.2.02010

Received 16 December 2024; revised 20 December 2024; accepted 28 December 2024; published 30 December 2024

Research Into the Efficiency of Processing a Numerical Random Sequence by Chaotic-type Cellular Automata

Heorhii Prokhorov* **and Denys Trembach**

Software Engineering Department, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: g.prokhorov@chnu.edu.ua)

ABSTRACT The article presents the results of a study of the processing of a sequence of random numbers obtained from a single webcam frame for compliance with one of the requirements of information cryptosecurity: uniformity of distribution. For the experiment an ordinary domestic web camera was used. The functionality of extraction a random number sequence from a frame of a web camera was discussed in the previous investigation. Java language provided software support. Before the main investigation a random number sequence generated by Java class was explored. The practically permissible level of uniformity of distribution for this sequence was obtained. For Java class SecureRandom this level is equal to 1.7%. For the purposes of experiment a high-speed sequence processing functionality based on cellular automata has been developed. As an origin of experiment an occasional snapshot of the web camera was used. As a result, it was found that a processing by chaotic rules (rules 30, 90, 105, 150) of linear cellular automata improves the quality of distribution of elements of a sequence almost to ideal level. It was found that the rule 30 provides the highest efficiency and productivity, but does not differ much from other rules. All the results were supported by statistical calculations and drawing custom graphical histograms especially designed for the experiment with a help of Java statistical and graphical classes. It was established that by selecting the number of iterations, the required level of uniformity of the distribution of sequence elements can be obtained. In theory a high-resolution web camera can provide a productivity of 2 Gbit/s. The results of the investigation can become the basis for the development of an affordable high-speed reliable hardware-software generator of a sequence of random numbers.

KEYWORDS software engineering, crypto-resistance, cellular automata, random number sequences, webcam.

I. INTRODUCTION

Random number generation plays an important role in ensuring cryptographic security, for example, the generation of passwords, keys, PIN-codes. Such generation is subject to a set of requirements approved by NIST or BSI protocols. However, when generating a sequence of random numbers (RNS), these demands are reaching a new level.

At the forefront of modern requirements is the urgent task of software engineering – the generation of RNS with a performance of at least 100 Mbit/s, and it is desirable to increase the speed to 1 Gbit/s. This requirement is the basis of steganography - the creation of secure data transmission channels, where not only the transmitted data is protected, but also the fact of transmission itself i.e., see [1]. In this case, the encrypted data is "dissolved" in a sequence of random numbers and is transmitted to the consumer in this form. If such a transmission is intercepted, it will be extremely difficult to calculate whether the packet contains an informative component or whether it is simply random content to overload the interception channels.

As for the generated RNS themselves, they must meet all the requirements for cryptosecurity: the level of chaos (randomness), a long cycle period, and a uniform distribution of elements. The big problem at the moment is precisely balancing all these requirements in a single device – a high-performance RNS generator (GRNS).

II. ANALYSIS OF CURRENT LITERARY DATA AND FORMULATION OF THE PROBLEM

Three approaches are used to generate random numbers.

The first approach – software – is based on specialized mathematical algorithms of software engineering. Unfortunately, software generators are to some extent predictable. As shown in study [2], mathematical proofs of the unsatisfactory cryptoresistance of pseudo-random sequences are available. The algorithm for generating a pseudo random sequence is publicly available, for example, for the Java Oracle documentation is on open access [3, 4], which makes it theoretically possible to attack the encryption algorithm.

Thus, we can say that the software method of generating random numbers, due to its predictability, is not completely crypto-resistant, although it fully satisfies all other requirements for cryptographic information security (CIS).

The second approach – hardware – is built using physical devices that use any physical stochastic noise sources. For example, in several recent studies [5, 6] a beta-emission counter is used to generate random numbers. This approach is completely crypto-resistant but requires additional expensive and exotic equipment.

The above limitations lead to the conclusion that it is necessary to study the possibility of using a simple, affordable webcam as the basis for a reliable, high-

performance RNS generator. A similar idea was already considered in study [7], but at that time (2014) the theoretical possible performance was limited to 200 Mbit/s, and the maximum webcam mode did not exceed VGA (640×480). But modern requirements recommend a performance of at least 100 Mbit/s, and preferably 1 Gbit/s.

In recent studies [8, 9] the statistical characteristics of the RNS obtained from a webcam were considered, and it turned out that with all the positive aspects, such a characteristic as a uniform distribution of elements by value remains unsatisfactory, which is unacceptable from the point of view of the requirements of the CIS.

III. THE PURPOSE AND OBJECTIVES OF THE RESEARCH

The purpose of the work is: to study the possibility of improving the characteristics of the RNS, namely the characteristics of the distribution of elements.

The tasks of the study are as follows:

- to investigate the form of the distribution of the original RNS obtained from the web camera frame;
- to investigate the results of RNS processing by a cellular automaton (CA) of a chaotic rule (30, 90, 105, 150);
- to determine the intensity of the use of CA to improve the statistical characteristics of RNS.

The random number generator implemented in this work was developed as part of a cryptographic system for protecting the information transfer channel using steganography.

IV. MATERIALS, CONDITIONS AND RESEARCH METHOD

A. Research equipment.

- Desktop:
  - CPU: AMD Ryzen 5 5600 4.4ghz,
  - RAM: 16gb 3200mhz,
  - SSD: Kingston NV1 250gb,
  - GPU: Nvidia GeForce GTX 1660ti;
- Anker Powerconf Web Camera C200. QQVGA (176×144); QVGA (320×240); VGA (640×480); SVGA (800×600); HD (1280×720); Full HD (1920×1080); Quad HD (2560×1440);
- webcam com.github.sarxos.webcam version 0.3.12;
- software: OS Ubuntu 22 LTS, 64 bit; Java Amazon Corretto 17.0.5; IntelliJ IDEA 2023.3.4 (Ultimate Edition); package com.github.sarxos.webcam version 0.3.12 – frame capture; javax.imageio package – video image processing; package java.security.SecureRandom – software generation of a random sequence.

B. Research methods.

The research was based on the results of previous works [8, 9], which described in detail the method of extracting RNS from a web camera frame. The functionality for calculating and visualizing the statistical characteristics of RNS was improved.

Selecting the methods for improving the statistical characteristics of RNS, cryptoprimitives were considered - linear cellular automata (CA), see [10]. Fig. 1 graphically illustrates the basics of the chaotic type CA functionality – rules 30, 90, 105, 150.

Fig. 1 shows the illustrated results of the CA operation and the results of the operation at 20 iterations. The initial conditions are given only one black cell (logical 0) in the center of the input horizontal sequence (upper horizontal row). All four rules demonstrated chaos (unpredictability) at the 20th iteration, but it is visually noticeable that the ratio of ones and zeros (50:50) is more inherent in rule 30 (CA-30). This is one of the requirements of the CIS for sequences of random numbers.

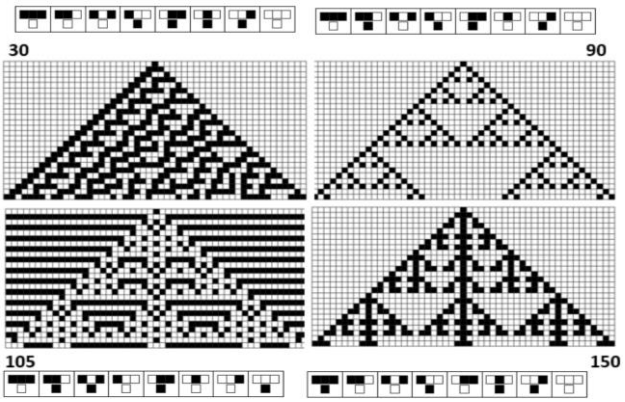

FIG. 1. Graphical illustration of the operation of chaotic cellular automata (CA) with rules 30, 90, 105, 150 over 20 iterations.

C. Conditions of experiment.

To provide clear experiment an ordinary occasional web camera snapshot was created, Fig. 2.

FIG. 2. An occasional snapshot from my window.

There were no special selections or preparations to shoot a snapshot. The only requirement was only to make a colorful frame. Time is 11.00 Kyiv time, 14 Dec, East direction.

This frame later formed a basis for investigation the method of improvement statistical characteristics using cellular automata.

V. THE RESULTS OF THE INVESTIGATION OF THE GENERATED SEQUENCES

The SecureRandom class of Java language is specially designed for generating crypto-resistant RNS [3]. It guarantees the consistency of all aspects of cryptosecurity: - productivity, distribution, chaos - except for one thing: its periodicity and predictability can be easily discerned.

Elements of the ideal uniform sequence take values in the range [-128 .. +127] – a total of 256 values, which corresponds to the byte data type of Java programming. Also, in ideally generated RNS, the presence of each value must be strictly equal to 1/256 or 0.385%. In practice, such a precision is unreachable. If a sequence is generated, for example, for 100 thousand elements, the presence value must be adjusted in the sequence exactly 390 times

(100 000 : 256 = 390.625). To provide such a precision in practice is a complicated task. However, on practice certain deviations from the ideal are permitted. Let's investigate these practically permissible deviations in software generated RNS.

Let's explore a sequence generated by SecureRandom class of the Java programming language and find out what percentage deviation it has, Fig. 3.

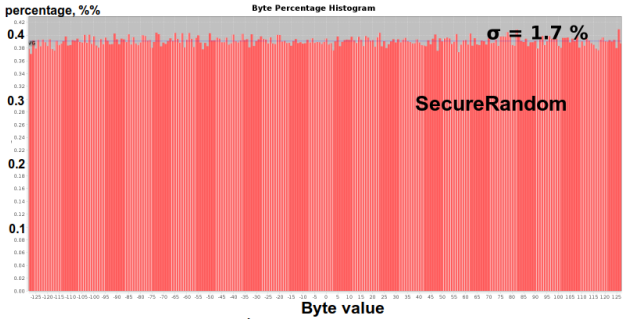

FIG. 3. A histogram of the distribution of elements by RNS value, generated by the SecureRandom class.

Fig. 3 shows the RNS spectrum generated by SecureRandom. The abscise axis (horizontal) shows a range of possible values. For Java, this range is [-128 .. +127]. The vertical axis shows the "presence participation" – the number of bytes of certain value as a percentage.

The standard deviation from mean value (thin blue line at approximately 0.39) is approx. equals to 0.0065 (about 1.6%). We consider this value (1.7%) to be practically permissible one.

Now we can investigate the histogram of values distribution (spectrum) of the original frame shown above on Fig. 2. The method of extraction a sequence of numbers fron a frame of web camera was discussed in [9]. Spectrum of distribution of the number sequence obtained from the frame is demonstrated on Fig. 4.

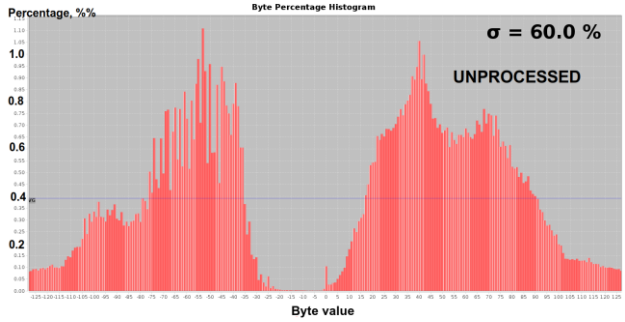

FIG. 4. Histogram of the distribution by value (spectrum) of raw RNS from the snapshot of Fig. 2. Unprocessed.

The histogram on Fig. 4 shows that a minimum in distribution has value -2 and. The local maximum presence was demonstrated by a values of -57 and +38. Such local extrema can play the role of a "fingerprint" for a generator, this fact is rather unwelcome, but not critical one. The distribution is not uniform, but chaotic one, the standard deviation from mean value is approx. equals to 60.0%. This does not satisfy the requirements of CIS, for the reason the original RNS requires extra processing.

Fig. 5 shows the spectrum of the original RNS after 1 cycle (iteration) processing of CA-30.

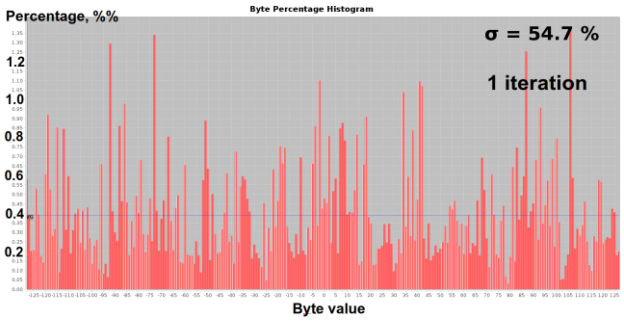

FIG. 5. Histogram of the distribution of values after processing CA-30. 1 iteration.

Fig. 5 shows a significant improvement of the histogram towards uniformity. The standard deviation has decreased from 60% to 54%. However, this is still not satisfactory according to the requirements of the CIS, because the practically permissible level is 1.6%. The following Fig. 6 shows the spectrum after 10 iterations of CA-30 processing.

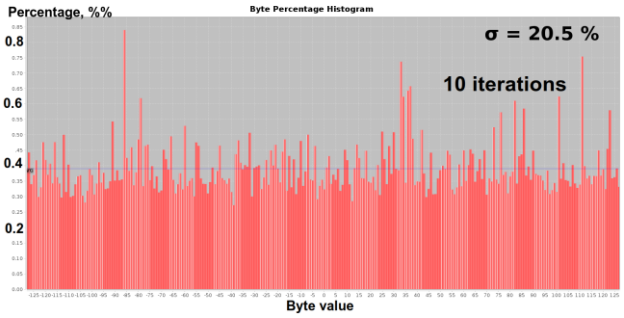

FIG. 6. Histogram of the distribution of values after processing CA-30. 10 iterations.

On Fig. 6 it is noticeable that the chaos in the distribution decreases. The "fingerprint" has completely disappeared. The standard deviation has been decreased to 20.5%. It is possible that such a deviation will be sufficient to accept the uniformity of the distribution as satisfactory.

The following Fig. 7 shows the spectrum after 20 cycles of CA-30 processing.

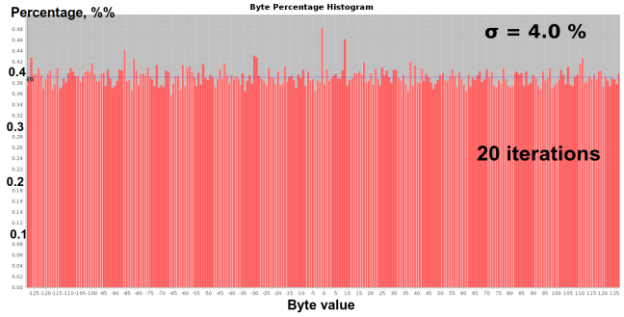

FIG. 7. Histogram of the distribution of values after processing CA-30. 20 iterations.

On Fig. 7, it is clearly visible that after 20 processing cycles (iterations), the spectrum is close to practically permissible, and in numerical value the deviation reached the level of 4.0%, let us recall that for a practically permissible spectrum this deviation is 1.6%. After 50 iterations, the deviation decreased to 0.8%, but the processing time reached 1.5 seconds, which decreases the generation method (speed) by almost two orders of magnitude.

VI. DISCUSSION OF RESEARCH RESULTS

The DescriptiveStatistics class of the Java programming language provides the calculation of a number of statistical characteristics of a sequence, including: the minimum element (min), the maximum element (max), the average value of the distribution (mean), the standard deviation (std dev), the median (median), for example, see [11]. All of them are characteristic of statistics. However, to simplify the material, we used only one parameter – the standard deviation. The smaller its value, the more the distribution resembles a uniform one. But the practically satisfactory level of deviation has not been determined. The ideal practical level is determined, it is equal to 1.6%, but the satisfactory level remains undefined.

CA-30 (rule 30) among other chaotic rules demonstrates the highest speed and the best quality of processing. It has been practically established that 20-30 iterations according to the CA-30 rule lead to an almost ideal state of the distribution of elements by value, and a satisfactory level may be reached in 10 iterations.

The number of iterations significantly affects the performance of the RNS generator. One iteration reduces the performance by half, and 10 iterations - almost 10 times. So, with 10 iterations in SVGA mode, it is possible to actually obtain a performance of 14.4 Mbit/s. In Quad HD mode (2560×1440), and the camera supports this mode, it is theoretically possible to reach the level of 265 Mbit/s.

The discussion does not include consideration of Java computing capabilities. Theoretically, it is possible to carry out RNS processing method using CA in parallel streams, then with an 8-core processor it is possible to reach the level of 2 Gbit/s.

It should be noted separately that the proposed Java method of processing the generated sequence gives an instant statistical characteristic of the distribution of values, unlike [7], where bulky software is used for this. This allows you to use a regular smartphone as the hardware and software basis of the generator.

VII. CONCLUSIONS

1. The distribution of elements by value is random, but individual for each individual device (web camera), which allows it to identify a hardware unit.

2. Processing of RNS by chaotic cellular automata improves the uniformity of the distribution. The best method is the Rule 30 one. The sequence processing time is approximately equal to the generation time.

3. The processing intensity (number of iterations) depends on the specified quality level (uniformity of the distribution). For the practically permissible level at least 20 iterations are required.

General conclusion: processing of sequences of random numbers generated using a web camera can serve as the basis for the development of reliable hybrid hardware-software RNS generator with a performance in the future of up to 2 Gbit/sec.

AUTHOR CONTRIBUTIONS

H.P. – methodology, investigation; D.T. – software, experiment.

COMPETING INTERESTS

The authors are declaring no competing interests.

REFERENCES

[1] A. Jammi, Y. Raju, S. Munishankaraiah, and K. Srinivas, "Steganography: an overview," *International Journal of Engineering Science and Technology*, vol. 2, no. 10, pp. 5985-5992, Dec. 2010.

[2] F. Martinez, "Attacks on Pseudo Random Number Generators Hiding a Linear Structure," in *Topics in Cryptology*, S. D. Galbraith, Ed., vol. 13161, *Lecture Notes in Computer Science*, Cham: Springer, 2022, pp. 145–168.

[3] "Class SecureRandom," *Java Platform Standard Edition 8 Documentation*. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html.

[4] C.-H. Hsieh, X. Yao, Q. Zhang, M. Lv, R. Wang, and B. Ni, "BCsRNG: A Secure Random Number Generator Based on Blockchain," *IEEE Access*, vol. 10, pp. 98117-98126, 2022, doi: 10.1109/ACCESS.2022.3206450.

[5] S. Park, B. G. Choi, T. Kang, K. Park, Y. Kwon, and J. Kim, "Efficient hardware implementation and analysis of true random-number generator based on beta source," *ETRI Journal*, vol. 42, no. 4, pp. 518-526, Aug. 2020, doi: 10.4218/etrij.2020-0083.

[6] K. Park, S. Park, B.-G. Choi, et al., "A lightweight true random number generator using beta radiation for IoT applications," *ETRI Journal*, vol. 42, pp. 951–964, 2020, doi: 10.4218/etrij.2020-0119.

[7] R. Li, "A True Random Number Generator algorithm from digital camera image noise for varying lighting conditions," in *SoutheastCon 2015*, Fort Lauderdale, FL, USA, 2015, pp. 1-8, doi: 10.1109/SECON.2015.7132901.

[8] 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.

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

[10] T. Toffoli and N. Margolis, *Cellular Automata Machines*, Cambridge, MA, USA: MIT Press, 1987.

[11] Y. Dong, "Descriptive Statistics and Its Applications," *Highlights in Science, Engineering and Technology*, vol. 47, pp. 16-23, 2023, doi: 10.54097/hset.v47i.8159.

Heorhii Prokhorov

He had received a Ph.D. in physics and mathematics in 2006. Now is a Assistant Professor of Software Engineering Department, Yuriy Fedkovych Chernivtsi National University. His research interests include cryptography, coding theory, hardware random number sequences generation.

ORCID ID: 0000-0001-7810-2785

Denys Trembach

Had received BS and MS degrees in Information Security from Evropejs'kij Universitet Financiv, Ukraine. Now is studying on a Ph.D. in Computer Science, Yuriy Fedkovych Chernivtsi National University. He is currently a security practitioner, mentor, and part-time lecturer. His research interests include cybersecurity, applied AI, chaotic systems dynamics.

ORCID ID: 0000-0001-8095-4186

Дослідження ефективності обробки числової випадкової послідовності клітинними автоматами хаотичного типу

Георгій Прохоров*, Денис Трембач

Кафедра програмного забезпечення комп'ютерних систем, Чернівецький національний університету ім. Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: g.prokhorov@chnu.edu.ua)

АНОТАЦІЯ На сучасному етапі програма, що генерує випадкові числа, ризикує бути зламаною через зростання обчислювальної потужності сучасних систем. Апаратна генерація базується на стохастичних фізичних явищах, але забезпечує низьку продуктивність та незадовільні статистичні параметри. У роботі пропонується використовувати обробку клітинними автоматами для покращення деяких статистичних параметрів послідовностей випадкових чисел, що згенеровані веб-камерою. 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. Раніше було встановлено, що стохастичні процеси, що відбуваються в матриці веб-камери, викликають хаотичний розподіл значень у згенерованій послідовності випадкових чисел. Цю проблему можна подолати за допомогою обчислювальної потужності лінійних клітинних автоматів, особливо правил 30, 90, 105. 150. Ці криптопримітиви відомі як хаотичні, які споживають низьку обчислювальну потужність. Успішність застосування оцінювали у порівнянні з генерацією випадкових чисел програмним методом, зокрема класом SecureRandom мови програмування Java. Показано, що шляхом вибору кількох ітерацій можна отримати необхідний рівень рівномірності розподілу елементів послідовності за значенням. Оцінка величини рівномірності розподілу здійснюється швидко за допомогою статистичної бібліотеки мови програмування Java і може бути реалізована на звичайному смартфоні, операційній системі Android, без використання громіздких статистичних пакетів. Теоретично показано, що веб камера високої роздільної здатності може забезпечити продуктивність генерації випадкових чисел до 2 Гбіт/сек. Результати дослідження можуть бути використані при проектуванні апаратного генератора послідовності випадкових чисел.

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