A NOVEL PSEUDO-RANDOM SEQUENCE GENERATOR FOR CRYPTOGRAPHIC APPLICATIONS USING LOGISTIC MAP VARIANTS WITH DYNAMIC THRESHOLDING

Tho Nguyen Van[1], Dung Le Dinh[1], Tung Dam Minh[1], Phong Nguyen Thanh[1], Bao Ngoc Do Thanh[2]*

[1]The University of Danang - VN-UK Institute for Research and Executive Education, Vietnam
[2]Duy Tan University, Vietnam

*Corresponding author: baongocdt@gmail.com

(Received: January 14, 2025; Revised: February 28, 2025; Accepted: March 06, 2025)

DOI: 10.31130/ud-jst.2025.23(6A).019E

Abstract - This paper presents an innovative approach for generating pseudo-random sequences for cryptographic applications, utilizing a modified logistic map and a dynamic thresholding mechanism. The generator is defined in terms of a logistic map, where the logistic function value is decomposed into binary values based on a dynamic threshold. A dynamic thresholding technique is introduced to convert the continuous chaotic outputs into binary sequences, ensuring better balance and reduced bias. Analytical and experimental evaluations show that this approach enhances the randomness of the generated sequences while maintaining simplicity and efficiency. Its lightweight nature and high efficiency make it especially suitable for resource-constrained environments such as IoT devices, where secure and efficient cryptographic solutions are critical. The combination of chaotic systems and dynamic thresholding showcases the potential of this approach as a lightweight, efficient, and secure solution for cryptographic sequence generation in modern applications.

Key words - Pseudo-random sequence; logistic map; dynamic thresholding; cryptography; autocorrelation

1. Introduction

Random number generation plays a fundamental role in various domains of science and technology, especially in cryptography, secure communications, and statistical simulations. In cryptographic systems, random sequences are essential for generating secure encryption keys, initialization vectors, and nonces. A high level of randomness is necessary to ensure the unpredictability and security of cryptographic protocols. Random number generators (RNGs) are typically classified into two main categories: True Random Number Generators (TRNGs) and Pseudo-Random Number Generators (PRNGs). TRNGs rely on physical phenomena, such as radioactive decay or thermal noise, to generate unpredictable sequences, while PRNGs use deterministic algorithms to generate sequences that mimic randomness [1-3].

Despite the inherent unpredictability of TRNGs, their reliance on physical phenomena makes them less efficient and less portable than PRNGs in many practical applications. Consequently, PRNGs have become the preferred choice in cryptographic systems due to their computational efficiency and reproducibility. However, the challenge lies in designing PRNGs that achieve high randomness while avoiding any detectable patterns that could compromise their security. PRNGs rely on deterministic algorithms to generate sequences that simulate randomness. Achieving high-quality randomness in PRNGs is a challenging task, as any predictable pattern in the sequence can lead to vulnerabilities in cryptographic systems. To address this issue, various approaches to PRNG design have been developed over the years, such as Linear Feedback Shift Registers [4-7], Cryptographic PRNGs [8-10], Chaos-Based PRNGs [13-18], Hybrid PRNGs [19-21], etc.

1.1. Related Work

Over the years, numerous approaches have been developed to design secure and efficient Pseudo-Random Number Generators (PRNGs). These approaches leverage diverse mathematical models, chaotic systems, and cryptographic techniques. Below, we summarize key research contributions and their distinguishing characteristics.

1.1.1. Linear Feedback Shift Registers (LFSRs)

LFSRs are among the earliest PRNGs introduced for cryptographic applications. LFSRs remain a fundamental approach to PRNGs due to their simplicity and efficiency in hardware implementations. LFSRs operate by iteratively applying a feedback function to a register, resulting in a deterministic sequence. LFSRs are computationally efficient and easy to implement, making them suitable for hardware-based systems [4]. However, their linear nature and finite periodicity make them predictable, limiting their security in modern applications. To address these limitations, Nonlinear Feedback Shift Registers (NLFSRs) have been proposed. For example, Golic presented enhancements to feedback functions, introducing nonlinearity to improve randomness [5]. NLFSRs offer better resistance to cryptanalysis but at the cost of increased computational complexity. Gammel et al. proposed an NLFSR-based design for lightweight cryptographic applications, improving both randomness and resistance to cryptanalysis [6]. Newer LFSR-based methods, such as those by Kumari et al., have targeted resource-constrained devices by combining LFSRs with a physical unclonable function (PUF), achieving a balance between randomness and efficiency [7].

1.1.2. Cryptographic PRNGs

Cryptographic PRNGs (C-PRNGs) employ cryptographic primitives to ensure high randomness and unpredictability. These methods often use block ciphers

(e.g., AES-CTR mode) or hash functions (e.g., HMAC-based DRBG) to derive pseudo-random sequences [8]. Cryptographic PRNGs provide strong security guarantees and meet formal standards, such as NIST SP 800-90A [9]. One widely used C-PRNG is the Dual_EC_DRBG, which relies on elliptic curve cryptography to generate random sequences[10]. However, concerns about potential backdoors and performance issues led to their removal.

Cryptographic PRNGs have also evolved to address efficiency concerns. Windarta et al. proposed a lightweight hash-based PRNG optimized for real-time cryptographic protocols, achieving improved performance for embedded systems [11]. Recent advancements in quantum-resistant cryptography have also extended to PRNGs. Kuang et al.introduced a PRNG construction secure against quantum adversaries, leveraging lattice-based cryptographic primitives [12].

Although highly secure, cryptographic PRNGs are often computationally intensive, making them less suitable for resource-constrained environments like IoT devices.

1.1.3. Chaos-Based PRNGs

Chaos-based PRNGs exploit the inherent unpredictability and sensitivity of chaotic systems to initial conditions [13,14]. They have gained popularity due to their lightweight implementation and natural randomness properties. Several chaotic maps have been explored for random number generation:

Alvarez and Li analyzed the cryptographic properties of the Logistic Map and highlighted its sensitivity to parameter tuning [15]. However, its deterministic nature can lead to observable patterns and finite periodicity, requiring post-processing techniques to enhance randomness. Saikia el al. utilized the Tent Map for secure symmetric encryption and demonstrated its ability to produce uniform distributions [16]. Despite this, its deterministic characteristics require additional randomization mechanisms.

Lan et al., have proposed parameter-tuning mechanisms to enhance the unpredictability of Logistic Map-based PRNGs [17]. These include dynamic parameter adjustments and multi-dimensional extensions.

Lightweight chaos-based PRNGs for IoT have gained popularity. For instance, Vijay Kumar and Ashish Girdhar proposed a 2D-Lorenz chaotic map for generating random sequences in real-time encryption protocols, demonstrating superior randomness and efficiency [18].

Despite advancements, chaos-based PRNGs still face challenges such as:

Parameter Sensitivity: Small deviations in parameters can lead to periodic patterns or loss of randomness.

- Deterministic Nature: Without post-processing, chaotic systems remain vulnerable to predictability.

1.1.4. Hybrid PRNGs

Hybrid PRNGs combine multiple chaotic systems or integrate chaotic maps with cryptographic techniques to address the limitations of standalone systems. These methods aim to leverage the strengths of different approaches while mitigating their weaknesses.

Researchers have explored combining the Logistic Map with the Tent Map to create multi-dimensional systems that enhance randomness. Stojanovski and Kocarev demonstrated improved statistical properties by coupling chaotic maps [19].

Altameem et al. proposed a hybrid PRNG combining chaotic maps with AES-based post-processing to enhance randomness and security for encryption applications. The AES layer effectively eliminated observable patterns in chaotic sequences [20]. To overcome the drawbacks in instability of the true RNGs and periodicity of the pseudo-RNGs, based on an analog–digital hybrid chaotic entropy source, an aperiodic hybrid RNG is proposed by Ming el al. [21] The hybrid source is a nondegenerate chaotic system that consists of a delay-coupled digital chaotic map and the analog anticontrol. The proposed system presents good chaotic behaviors in the digital world and has great advantages when realized on hardware platforms.

While hybrid PRNGs show significant improvements, their performance depends on the selection of chaotic maps, combination strategies, and parameter tuning, making them more complex to design and implement.

1.2. Motivation and Contribution

The Internet of Things (IoT) has seen remarkable growth over the past decade, becoming one of the core technologies driving the Fourth Industrial Revolution [22,23]. The number of IoT devices has grown exponentially, from a few billion in the early 2010s to tens of billions today [24], leading to the development of comprehensive IoT ecosystems. However, the rapid growth of IoT also brings significant challenges, such as data security, privacy, and resource management [25, 26]. IoT devices often have limited computational power, memory, and energy resources, making it difficult to implement robust cryptographic-grade PRNGs [27, 28]. Furthermore, PRNGs must be lightweight yet secure against attacks, as compromised randomness can lead to vulnerabilities in IoT applications, including data encryption, authentication, and communication protocols [29, 30]. Balancing efficiency, security, and randomness quality is a critical challenge for PRNG design in IoT devices.

Despite significant advancements in PRNG design, achieving high-quality randomness while maintaining computational efficiency remains an open challenge [31, 32]. Chaos-based PRNGs, in particular, offer a promising avenue for lightweight and efficient random number generation [33]. However, their deterministic nature and susceptibility to parameter tuning issues necessitate further improvements [34]. Given the limitations of existing chaos-based PRNGs, there is a need for innovative methods to improve their randomness, reduce correlations, and enhance security without significantly increasing computational complexity [35, 36].

This paper proposes a novel approach to enhance chaos-based PRNGs by integrating a dynamic threshold mechanism with the Logistic Map. Specifically, the proposed PRNG introduces a time-varying threshold to discretize the output of the Logistic Map into binary sequences. The dynamic threshold introduces time-

dependent variations, breaking periodic patterns and reducing correlations in the sequence. Additionally, we add a post-processing technique using XOR operation to further enhance the randomness of the generated strings. Experimental evaluations using standard randomness tests, including the NIST SP 800-22 suite, demonstrate the effectiveness of the proposed PRNG in generating high-quality random sequences suitable for cryptographic applications.

2. Proposed method

The proposed pseudo-random number generator (PRNG) builds upon the chaotic properties of the Logistic Map, combined with a dynamic threshold mechanism and optional post-processing techniques to improve randomness and eliminate observable patterns. This section describes the mathematical foundation, sequence generation process, and the enhancements introduced to address the limitations of traditional chaos-based PRNGs

2.1. Logistic Map as the Core Chaotic System

The Logistic Map is a one-dimensional iterative chaotic system defined as:

$$g_{n+1} = r.g_n.(1 - g_n), g_n \in (0,1) \quad (1)$$

where: g_n is the state of the system at iteration n; r is the control parameter that determines the behavior of the system.

For r∈[3.7,4.0], the Logistic Map exhibits chaotic behavior, characterized by sensitivity to initial conditions, ergodicity, and topological mixing. These properties make it suitable for random number generation. However, the deterministic nature of the Logistic Map can result in observable periodic patterns, especially in its binary representation.

2.2. Dynamic Threshold Mechanism

To mitigate the limitations of static threshold-based PRNGs, we introduce a dynamic threshold mechanism to discretize the chaotic output of the Logistic Map into binary sequences. The threshold varies with time (iteration index n) according to a sinusoidal function:

$$T_n = 0.5 + A.sin\left(\frac{\pi n}{N}\right) \quad (2)$$

where: Tn is the dynamic threshold at iteration n; A is the amplitude of the sinusoidal oscillation, controlling the variability of the threshold; N is the length of the generated sequence.

Static thresholds (T=0.5) in chaotic maps can result in periodic binary sequences or patterns that reduce randomness. The time-dependent threshold disrupts periodicity by introducing variability into the mapping process. Additionally, by modulating the threshold over time, the dynamic mechanism ensures that each iteration introduces additional variability, enhancing randomness and reducing correlations between successive bits.

To ensure accurate encoding, the dynamic threshold must be adjusted appropriately to the changes in the input signal. The dynamic threshold minimizes the chance of producing predictable patterns, thus improving the randomness and security of the encoding system.

The stability of the dynamic threshold is crucial. It should adapt smoothly to changes in the input signal without causing excessive fluctuations or encoding errors. Using an appropriate adjusting function (such as sine or cosine) helps maintain stability throughout the encoding process.

The dynamic threshold can be adjusted flexibly based on the environment and input signal, improving the system's adaptability to changes in operational conditions or environmental factors such as noise or power supply variations. The dynamic threshold can be adjusted based on the Statistical Feedback Mechanism. The PRNG can periodically analyze its output using entropy measurements and randomness tests. If deviations from expected randomness patterns occur, the threshold parameters (such as amplitude or frequency of variation) can be modified accordingly.

The binary sequence is generated as follows:

$$b_n = \begin{cases} 1, if\ g_n > T_n \\ 0, if\ g_n \le T_n \end{cases} \quad (3)$$

Here, g_n is the output of the Logistic Map, and T_n is the dynamic threshold at iteration n.

Figure 1 illustrates the simulation results of generating a bit sequence using a dynamic threshold for the Logistic Map. In Figure 1.a, the chaotic values gn of the Logistic Map oscillate unpredictably between 0 and 1, reflecting the map's chaotic nature, while the dynamic threshold Tn varies smoothly in a sinusoidal pattern. This dynamic threshold introduces additional complexity compared to a static threshold, enhancing variability in the decision process. Figure 1.b shows the resulting bit sequence, derived by comparing g_n with T_n. The alternating pattern of 0's and 1's in the bit sequence reflects the chaotic behavior of gn and the influence of the dynamic threshold. This approach aims to improve randomness by reducing repetitive patterns often associated with static thresholds, though further statistical tests are needed to confirm the quality of randomness in the generated sequence.

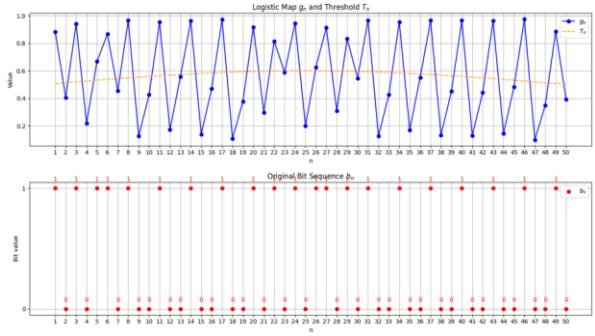

Figure 1: Generate PRNG sequence with dynamic threshold.

2.3. Post-Processing for Improved Randomness

Although the dynamic threshold improves randomness, additional post-processing can further enhance the quality of the generated sequence by eliminating residual correlations or dependencies. In this work, we explore XOR-based post-processing: The binary sequence is processed by applying an XOR operation between adjacent bits:

$$b_{2n'} = b_{2n} \oplus b_{2n-1} \quad (4)$$

This operation reduces dependencies between successive bits and ensures uniform bit transitions.

Benefits of XOR Post-Processing:

- Breaks Patterns: Removes residual linear or periodic correlations.

- Enhances Uniformity: Improves the distribution of transitions between 0 and 1.

- Increases Security: Adds another layer of randomness to resist reverse engineering.

3. Experimental evaluation

This section presents the experimental evaluation of the proposed PRNG. The experiments were designed to assess the quality of randomness, statistical properties, and performance of the generated sequences. Standard randomness tests, such as the NIST SP 800-22 statistical test suite, were employed to rigorously evaluate the sequences. The results were compared with other PRNGs to highlight the advantages of the proposed method.

3.1. Experimental Setup

3.1.1. Parameters and Configuration

The following parameters were used in the experiments:

- Logistic Map Parameters: r=3.9. The control parameter was set in the range of [3.7, 4.0], as this ensures chaotic behavior in the Logistic Map

- Initial state g0=0.654321, chosen randomly in the interval (0,1). A randomly chosen value within (0,1) was used to prevent periodic patterns.

Dynamic Threshold Parameters:

- Threshold function: The threshold function $T_n = 0.5 + A.sin\left(\frac{\pi n}{N}\right)$ was selected with A = 0.1 to introduce controlled variability in sequence generation.

- Sequence Length: Each test used a sequence of 1,000,000 bits, which is a sufficient length to validate statistical properties. N=1,000,000 bits were generated for each test.

- Test Repetitions: Each test was conducted 100 times, and the average results were recorded to reduce bias

These parameter choices ensure that the results presented in the experimental evaluation are statistically valid and applicable to real-world cryptographic applications.

3.1.2. Implementation Details

The algorithm was implemented in Python using efficient numerical libraries.

All randomness tests were conducted using standard libraries and tools, including the NIST SP 800-22 statistical test suite.

The tests were performed 100 times and the average was calculated.

For performance comparison, the proposed PRNG was tested alongside:

1. A Logistic Map with a static threshold (T=0.5) [12].

2. Tent Map-based PRNG [13].

3. AES-based cryptographic PRNG (CTR mode) [5].

3.1.3. System Configuration

The experiments were conducted on a system with the following specifications:

- Processor: Intel Xeon 2.2 GHz

- RAM: 12 GB

- Operating System: Ubuntu 22.04.3 LTS

3.2. Randomness Tests

3.2.1. NIST SP 800-22 Statistical Test Suite

The NIST SP 800-22 suite is a widely recognized standard for evaluating randomness. It includes a series of statistical tests designed to detect patterns or weaknesses in binary sequences. The following tests were applied:

- Frequency Test: Evaluates the balance of 0s and 1s in the sequence.

- Runs Test: Measures the number of runs (consecutive bits of the same value) and compares it to the expected number.

- Block Frequency Test: Assesses the distribution of bits within fixed-size blocks.

- Approximate Entropy Test: Evaluates the complexity and unpredictability of subsequences.

- Cumulative Sums Test (Cusums): Analyzes the cumulative sum of deviations from the expected mean.

- Serial Test: Checks for patterns in overlapping m-bit subsequences.

3.2.2. Autocorrelation Test

The autocorrelation function evaluates the correlation between a sequence and a lagged version of itself. It is defined as:

$$R(k) = \frac{\sum_{i=1}^{N-k}(b_i-\bar{b})(b_{i+k}-\bar{b})}{\sum_{i=1}^{N}(b_i-\bar{b})^2} \quad (5)$$

Where k is the time lag under consideration, $\bar{b}$ is the mean of the sequence, N is the total length of the sequence, and R(k) is the autocorrelation at lag k.

A well-performing PRNG should exhibit low autocorrelation across all lags, indicating that there are no strong dependencies between bits over time. If significant correlations exist, the sequence may exhibit patterns that compromise its unpredictability, making it vulnerable to cryptanalysis.

In this research, the proposed PRNG is evaluated using the autocorrelation test to ensure that:

- There are no periodic patterns in the generated sequence that could lead to predictability.

- Randomness is preserved over different time lags, ensuring that each bit is independent of previous bits.

3.3. Results and Analysis

3.3.1. NIST SP 800-22 Test Results

The results of the NIST randomness tests are summarized in Table 1. The static threshold PRNG exhibited poor performance in the Runs and Serial Tests, highlighting the limitations of using a fixed threshold. The Tent Map PRNG performed better than the static threshold system but still fell short compared to the proposed

method. The proposed PRNG has passed all NIST tests with a pass rate of over 90%, demonstrating its robustness and high randomness. Compared to AES-based PRNG, it gives nearly identical results.

Table 1. *The results of the NIST randomness tests*

Test	P-value			
	Proposed	**Static Threshold**	**Tent Map**	**AES CTR**
Frequency Test	0.252 (pass)	0.005 (fail)	0.015 (pass)	0.697 (pass)
Runs Test	0.123 (pass)	0.000 (fail)	0.000 (fail)	0.322 (pass)
Block Frequency Test	0.547 (pass)	0.132 (pass)	0.012 (pass)	0.703 (pass)
Approximate Entropy Test	0.135 (pass)	0.208 (pass)	0.955 (pass)	0.135 (pass)
Cumulative Sums Test	0.015 (pass)	0.000 (fail)	0.000 (fail)	0.349 (pass)
Serial Test	0.767 (pass)	0.793 (pass)	0.981 (pass)	0.768 (pass)

The Proposed PRNG demonstrates moderate to strong performance in critical tests such as the Frequency Test (p-value 0.252) and Block Frequency Test (p-value 0.547), significantly outperforming Static Threshold PRNG (p-values 0.005 and 0.132) and Tent Map PRNG (p-values 0.015 and 0.012), which exhibit poor randomness. Notably, in the Serial Test (p-value 0.767), the Proposed PRNG achieves comparable results to AES PRNG (0.768) and Static Threshold PRNG (0.793), demonstrating robust bit independence in patterns. Despite a slight underperformance in the Approximate Entropy Test (p-value 0.135) and Cumulative Sums Test (p-value 0.015), where Tent Map PRNG excels (p-values 0.955 and 0.981), the Proposed PRNG still exhibits better overall randomness compared to Static Threshold PRNG and Tent Map PRNG, which struggle in other critical tests such as Runs and Frequency. While AES PRNG remains the most consistent performer across all tests, the Proposed PRNG offers a strong balance of randomness and performance, making it a competitive alternative in scenarios that do not require cryptographic-level randomness but demand better statistical properties than simpler PRNGs like Static Threshold and Tent Map.

3.3.2. Autocorrelation Test

Figure 2 shows the comparison plot of the average autocorrelation of the proposed PRNG with LogisticStatic, TentMap and AES_CTR over delays from 1 to 5. The proposed PRNG demonstrates consistently low autocorrelation values close to zero, outperforming LogisticStatic and TentMap while closely approaching the performance of AES_CTR. Compared to LogisticStatic, which shows significant fluctuations (e.g., -0.2 at lag 1 and 0.2 at lag 2) before stabilizing at higher lags, the proposed PRNG maintains stable and minimal autocorrelation across all lags, indicating improved randomness and reduced bit dependency. In contrast, TentMap exhibits consistently high autocorrelation (~0.4 at lag 1) that gradually decreases with higher lags, reflecting substantial dependencies between bits and inferior randomness. While AES_CTR achieves the best performance with near-zero autocorrelation at all lags due to its cryptographic design, the proposed PRNG closely matches this standard, offering a strong alternative in terms of randomness quality and outperforming LogisticStatic and TentMap by a wide margin.

The experimental results show that the proposed PRNG maintains consistently low autocorrelation values, outperforming traditional chaotic PRNGs while approaching the performance of cryptographic-grade generators like AES-CTR. This confirms the effectiveness of the dynamic threshold in disrupting correlations and improving randomness quality.

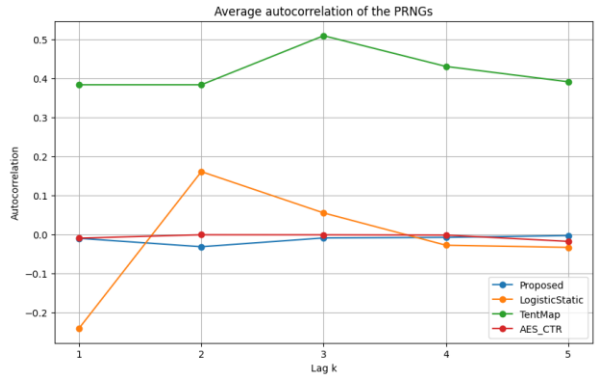

Figure 2. *Average autocorrelation of PRNGs*

3.3.3. Performance Metrics

Table 2. *The performance comparison of PRNGs*

Metric	Proposed PRNG	Static Threshold PRNG	Tent Map PRNG	AES PRNG
Generation Speed (µs/bit)	0.823	0.666	0.964	1.709
CPU Usage (%)	87%	87%	89%	96%
Memory Usage (MB)	8.08	8.06	8.07	15.8

The performance comparison is summarized in Table 2. The Proposed PRNG demonstrates a strong balance between speed, CPU usage, and memory efficiency, making it highly suitable for lightweight applications. With a generation speed of 0.823 µs/bit, it is only slightly slower than the Static Threshold PRNG (0.666 µs/bit) while outperforming the Tent Map PRNG (0.964 µs/bit) and the AES PRNG (1.709 µs/bit). It achieves low CPU usage (87%), comparable to Static Threshold PRNG and lower than Tent Map PRNG (89%) and AES PRNG (96%). Additionally, its memory usage (8.08 MB) is as efficient as Static and Tent Map PRNGs, significantly outperforming AES PRNG (15.8 MB). These results highlight the Proposed PRNG as an optimal solution, offering a robust trade-off between performance and resource consumption, particularly for real-time and resource-constrained applications like IoT.

4. Conclusion

This paper introduces a novel pseudo-random sequence generator based on a modified Logistic Map with a dynamic threshold that adjusts over time. Experimental results, including extensive testing using the NIST SP800-22 statistical test suite, show that the proposed method

outperforms traditional Logistic Map with static threshold and Tent Map-based PRNG in terms of reducing autocorrelation and improving statistical independence. The dynamic threshold introduced in our approach plays a key role in mitigating inherent flaws in the Logistic Map, such as autocorrelation and non-uniformity in long sequences. Additionally, the proposed method demonstrates superior performance efficiency, achieving a competitive generation speed with low CPU and memory usage, making it highly suitable for resource-constrained environments. These characteristics, combined with its statistical robustness, position our generator as an ideal solution for secure and lightweight random number generation in modern cryptographic systems and real-time applications such as the Internet of Things (IoT).

Acknowledgement: This work is supported by The University of Danang- VN-UK Institute for Research and Executive Education (grant number T2023-VNUK-01).

REFERENCES

[1] B. Qi, Y.-M. Chi, H.-K. Lo, and L. Qian, "High-speed quantum random number generation by measuring phase noise of a single-mode laser", *Optics Communications,* vol. 325, pp. 165–169, 2010.

[2] M. Bucci, L. Germani, R. Luzzi, P. Tommasino, A. Trifiletti, and M. Varanonuovo, "A high-speed IC random-number source for smartcard microcontrollers", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 50, no. 11, pp. 1373–1380, Nov. 2003.

[3] C. S. Petrie and J. A. Connelly, "A noise-based IC random number generator for applications in cryptography", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 615–621, May 2000.

[4] C. E. Shannon, "Communication theory of secrecy systems", *Bell System Technical Journal,* vol. 28, no. 4, pp. 656–715, 1949.

[5] B. M. Gammel, R. Gottfert, and O. Kniffler, "An NLFSR-based stream cipher", *in Proc. IEEE Int. Symp. Circuits and Systems* (ISCAS), Kos, Greece, 2006, pp. 4 pp.-2920, doi: 10.1109/ISCAS.2006.1693235.

[6] P. Kumari and B. Mondal, "Lightweight encryption with data and device integrity using NLFSR and PUF for the Internet of Medical Things", *Internet of Things,* vol. 25, 101041, 2023, doi: 10.1016/j.iot.2023.101041.

[7] NIST, "Recommendation for random number generation using deterministic random bit generators", *NIST Special Publication 800-90A*, 2015.

[8] E. Barker, J. Kelsey, and N. Ferguson, "SP 800-90A: Recommendation for random number generation using deterministic random bit generators", *NIST*, 2012.

[9] D. Bernstein, T. Lange, and R. Niederhagen, "Dual EC: A Standardized Back Door", *in Cryptography and Security: From Theory to Practice,* Springer, 2016, pp. 347–366, doi: 10.1007/978-3-662-49301-4_17.

[10] S. Windarta, S. MT, K. Ramli, B. Pranggono, and T. Gunawan, "Lightweight cryptographic hash functions: Design trends, comparative study, and future directions*", IEEE Access*, vol. 10, pp. 82272–82294, 2022, doi: 10.1109/ACCESS.2022.3195572.

[11] R. Kuang, D. Lou, A. He, C. Mchenzie, and M. Redding, "Pseudo quantum random number generator", *in Proc. IEEE Quantum Computing and Engineering Conf. (QCE)*, 2021, doi: 10.1109/QCE52317.2021.00053.

[12] E. N. Lorenz, "Deterministic Nonperiodic Flow", *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–148, Mar. 1963.

[13] T. N.Van *et al.,* "A method for fast synchronization of chaotic systems and its application to chaos-based secure communication", *Journal of Science and Technology - The University of Danang*, vol. 20, no. 12.2, pp. 1–5, Dec. 2022, doi: 10.31130/ud-jst.2022.541ICT.

[14] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems", *Int. Journal of Bifurcation and Chaos,* vol. 16, no. 08, pp. 2129–2151, 2006.

[15] M. Saikia and B. Baruah, "Chaotic map-based image encryption in spatial domain: A brief survey", *in Proc. Int. Conf. Cryptography and Security Applications*, 2017, pp. 578–591.

[16] R. Lan, J. He, S. Wang, Y. Liu, and X. Luo, "A parameter-selection-based chaotic system", *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 65, no. 11, pp. 1796–1800, Nov. 2018, doi: 10.1109/TCSII.2018.2865255.

[17] V. Kumar and A. Girdhar, "A 2D logistic map and Lorenz-Rossler chaotic system-based RGB image encryption approach", *Multimedia Tools Appl*., vol. 80, no. 3, pp. 3749–3773, Jan. 2021, doi: 10.1007/s11042-020-09854-x.

[18] T. Stojanovski and L. Kocarev, "Chaos-based random number generators-Part I: Analysis", *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 48, no. 3, pp. 281–288, Mar. 2001.

[19] A. Altameem, P. P. Pillai, T. Suresh, R. C. Poonia, and A. K. J. Saudagar, "A hybrid AES with a chaotic map-based biometric authentication framework for IoT and Industry 4.0", *Systems*, vol. 11, no. 1, pp. 1–28, Jan. 2023, doi: 10.3390/systems11010028

[20] H. Ming, H. Hu, F. Lv, and R. Yu, "A high-performance hybrid random number generator based on a nondegenerate coupled chaos and its practical implementation", *Nonlinear Dynamics,* vol. 111, pp. 1083–1096, 2022, doi: 10.1007/s11071-022-07838-0.

[21] T. Hossain, M. Risalat, M. Khan, A. Rahman, and N. Khan, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises", *Int. Journal for Multidisciplinary Research*, vol. 6, no. 1, 2024, doi: 10.36948/ijfmr.2024.v06i01.22699.

[22] L. Xu, W. He, and S. Li, "Internet of Things in industries: A survey", *IEEE Trans. Industrial Informatics*, vol. 10, pp. 2233–2243, 2014, doi: 10.1109/TII.2014.2300753.

[23] Statista Research Department, "Number of connected IoT devices worldwide 2015–2030", *Statista*, 2023. [Online]. Available: https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/ [Accessed: Nov, 19, 2024].

[24] J. Mohanty, S. Mishra, S. Patra, B. Pati, and C. Panigrahi, "IoT security, challenges, and solutions: A review", *in Handbook of Big Data Technologies*, Springer, 2021, pp. 593–615, doi: 10.1007/978-981-15-6353-9_46.

[25] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things", *Computer Communications*, vol. 57, pp. 1-13, Mar. 2018, doi: 10.1016/j.comcom.2014.12.008.

[26] P. Olubudo, "Cybersecurity Challenges in the Internet of Things (IoT): Analyzing Vulnerabilities and Proposing Solutions for Securing IoT Devices and Networks", *Obafemi Awolowo University*, May 2024. [Online]. Available: ResearchGate. [Accessed: Nov. 19, 2024]

[27] D. Sumit, B. Singh, and P. Jindal, "Lightweight cryptography: A solution to secure IoT", *Wireless Personal Communications,* vol. 112, pp. 1-34, 2020, doi: 10.1007/s11277-020-07134-3.

[28] R. Shaltiel, "An introduction to randomness extractors", *in Foundations of Cryptography: Volume II,* Springer, 2011, pp. 21–41, doi: 10.1007/978-3-642-22012-8_2.

[29] M. Azrour *et al*., "Internet of Things security: Challenges and key issues", *Security and Communication Networks*, vol. 2021, pp. 1-14, 2021, doi: 10.1155/2021/5533843.

[30] M. Chetto, S. El Assad, and M. Farajallah, "A lightweight chaos-based cryptosystem for dynamic security management in real-time overloaded applications", *Int. Journal of Internet Technology and Secured Transactions*, vol. 5, no. 4, pp. 262–275, 2014, doi: 10.1504/IJITST.2014.065187.

[31] N. Gunathilake, A. Al-Dubai, and W. Buchanan, "Recent advances and trends in lightweight cryptography for IoT security", *in Proc. Int. Conf. on Communication and Networking Systems*, 2020, pp. 1–9, doi: 10.23919/CNSM50824.2020.9269083.

[32] S.-M. Fu, Z.-Y. Chen, and Y.-A. Zhou, "Chaos-based random number generators", *Proc. IEEE Conf. Circuits and Systems*, vol. 41, pp. 749–754, 2004.

[33] A. Banerjee, S. Bose, and S. Das, "Parameter sensitivity and improvements in chaos-based PRNGs", *Chaos Theory and Applications,* vol. 7, no. 2, pp. 34-50, Apr. 2018.

[34] W. E. H. Youssef *et al.,* "A secure chaos-based lightweight cryptosystem for the Internet of Things", *IEEE Access,* vol. 11, pp. 123279-123294, 2023, doi: 10.1109/ACCESS.2023.3326476.

[35] T. Stojanovski and L. Kocarev, "Chaos-based random number generators-Part II: Applications", *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl*., vol. 48, no. 4, pp. 389–395, Apr. 2001.