



Article

Cryptographic Grade Chaotic Random Number Generator Based on Tent-Map

Ahmad Al-Daraiseh ¹, Yousef Sanjalawe ², Salam Al-E'mari ³, Salam Fraihat ^{4,*}, Mohammad Bany Taha ⁵
and Muhammed Al-Muhammed ¹

- ¹ Computer Science Department, School of Information Technology, American University of Madaba, Amman 11821, Jordan; a.daraiseh@aum.edu.jo (A.A.-D.); m.almuhammed@aum.edu.jo (M.A.-M.)
 - ² Cybersecurity Department, School of Information Technology, American University of Madaba, Amman 11821, Jordan; y.sanjalawe@aum.edu.jo
 - ³ Information Security Department, Faculty of Information Technology, University of Petra, Amman 11196, Jordan; salam.ammari@uop.edu.jo
 - ⁴ Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman P.O. Box 346, United Arab Emirates
 - ⁵ Department of Data Science and Artificial Intelligence, School of Information Technology, American University of Madaba, Amman 11821, Jordan; m.taha@aum.edu.jo
- * Correspondence: s.fraihat@ajman.ac.ae

Abstract: In recent years, there has been an increasing interest in employing chaotic-based random number generators for cryptographic purposes. However, many of these generators produce sequences that lack the necessary strength for cryptographic systems, such as Tent-Map. However, these generators still suffer from common issues when generating random numbers, including issues related to speed, randomness, lack of statistical properties, and lack of uniformity. Therefore, this paper introduces an efficient pseudo-random number generator, called State-Based Tent-Map (SBTM), based on a modified Tent-Map, which addresses this and other limitations by providing highly robust sequences suitable for cryptographic applications. The proposed generator is specifically designed to generate sequences with exceptional statistical properties and a high degree of security. It utilizes a modified 1D chaotic Tent-Map with enhanced attributes to produce the chaotic sequences. Rigorous randomness testing using the Dieharder test suite confirmed the promising results of the generated keystream bits. The comprehensive evaluation demonstrated that approximately 97.4% of the tests passed successfully, providing further evidence of the SBTM's capability to produce sequences with sufficient randomness and statistical properties.

Keywords: random number generator; Tent-Map; chaotic; Dieharder



Citation: Al-Daraiseh, A.; Sanjalawe, Y.; Al-E'mari, S.; Fraihat, S.; Bany Taha, M.; Al-Muhammed, M. Cryptographic Grade Chaotic Random Number Generator Based on Tent-Map. *J. Sens. Actuator Netw.* **2023**, *12*, 73. <https://doi.org/10.3390/jsan12050073>

Academic Editor: Lei Shu

Received: 2 September 2023

Revised: 22 September 2023

Accepted: 26 September 2023

Published: 10 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid progress of the Internet in recent years, there has been a rising need for enhanced information security in various fields. As a result, security concerns have gained increasing attention [1]. The reliability and unpredictability of encryption algorithms and key generation are pivotal for ensuring the effectiveness of encryption systems [2–5]. Random numbers are essential elements in the majority of cryptographic algorithms, and the usage of a random number generator (RNG) holds significant value in the domain of information security. One notable application of RNGs is their crucial role in generating parameters for public key cryptographic systems, like ECC and RSA, as well as facilitating image encryption [6].

In the past few years, there has been a proliferation of image-based cryptographic schemes that leverage the principles of confusion and diffusion. These encryption schemes consist of two key phases. The first phase, referred to as confusion, involves scrambling the positions of pixels to disrupt any correlation between them. The second phase, diffusion,

employs reversible operations to modify the pixel values [7]. The confusion and diffusion processes can be repeated for multiple rounds denoted by m and n , with both m and n being greater than zero. Within these processes, pseudo-random number sequences (PRNSs) generated by chaotic maps play a crucial role [8,9]. Random numbers (RN) play a crucial role in modern cryptography as they facilitate the encryption of information through ciphers and ensure security by utilizing private keys and certificates. Their importance extends across various engineering domains, including asymmetric encryption, digital algorithms in cryptography, system testing, and statistical Monte Carlo methods [10]. RNGs are utilized to generate these RNs or confidential keys and can be categorized as deterministic (also known as pseudo-random number generators or PRNGs) or nondeterministic (also known as true random number generators or TRNGs). PRNGs produce random numbers through algorithms that rely on a seed as the input, while TRNGs generate random numbers using physical sources [11].

Many techniques were developed for generating PRNSs. One approach involves employing a Chebyshev chaotic map to generate PRNS [12]. In this particular method, the eight-bit planes of an eight-bit gray level image are stored separately, resulting in a matrix size eight times that of the original image. The PRNS then performs a permutation of the pixel bits. However, this scheme necessitates sorting the lengthy chaotic sequence, which introduces additional computational overhead. In the subsequent part of the scheme, the bit planes undergo another permutation using Arnold's cat map. Unfortunately, the number of rounds required to achieve the desired outcomes is not specified. While this dual permutation process generates a diffusion effect on the encrypted image, it is important to note that permutation-only ciphers are susceptible to cryptanalysis. Consequently, the overall scheme exhibits a high level of complexity but lacks security. Therefore, an alternative scheme that incorporates both permutation and diffusion is necessary. As mentioned earlier, chaotic maps are the essential building blocks for creating encryption algorithms, particularly for image encryption, by serving as the PRNG. These mathematical functions generate highly unpredictable patterns, beginning from an initial seed value. Figure 1 illustrates the recent advancements in utilizing chaotic maps in nonlinear dynamic systems, including pseudo-random number generation and encryption. Pseudo-random number generators are found in both hardware and software implementations, with Field Programmable Gate Array (FPGA) and microcontroller-based implementations utilizing platforms such as Arduino and Raspberry Pi. Encryption applications primarily focus on image ciphering, with only a few instances involving audio [13].

Quantum Random Number Generation (QRNG) is at the forefront of cutting-edge research within the field of quantum information science. This captivating discipline leverages the inherent unpredictability of quantum mechanics to create genuinely random numbers [14]. In stark contrast to classical random number generators, which often rely on deterministic algorithms, QRNG capitalizes on the fundamental tenets of quantum physics, including the uncertainty principle and the superposition of quantum states, to yield inherently unpredictable outcomes. The utility of QRNG extends across diverse domains, encompassing cryptography, secure communications, and simulations [15]. As the demand for robust, unassailable encryption techniques escalates, QRNG holds the promise of assuming a pivotal role in safeguarding the digital realm. It achieves this by generating random numbers that are naturally impervious to predictability and exploitation. Researchers in this field persistently challenge the limits of our comprehension of quantum phenomena, striving to harness these phenomena for pragmatic, real-world applications that have the potential to revolutionize secure data transmission and computational integrity in the future [14,15].

This current study is driven by the significant attention received by chaotic cryptography in recent years, evident from the numerous cipher systems proposed and discussed in the literature that make use of chaotic maps. The appeal of incorporating chaotic maps in cipher systems arises from several key attributes. Firstly, chaotic maps display a high sensitivity to initial conditions and control parameters. Secondly, the evolution of their

orbits is unpredictable. Additionally, these maps can be relatively easily implemented in both hardware and software, resulting in high encryption rates. These attributes are closely associated with crucial cryptographic properties, such as confusion and diffusion, as well as balance and the avalanche effect [16]. Despite its advantages, the original Tent-Map faces several limitations when it comes to generating random numbers, encompassing drawbacks in terms of speed, randomness, a lack of statistical properties, and the absence of uniformity [17–20]; therefore, this paper introduces and implements an innovative methodology for designing and constructing a PRNG that generates cryptographic-grade random numbers while prioritizing efficiency, compactness, and simplicity. The remaining sections of this paper are structured as follows. Section 2 presents a comprehensive review of the literature and discusses the state-of-the-art random number generator algorithms. Section 3 provides an introduction to the preliminaries of the Tent-Map. The proposed random number generator is discussed in Section 4. In Section 5, the experimental settings and evaluation results are presented. Section 6 presents the contributions to sensor and actuator networks. Finally, Section 7 concludes with the key findings, limitations, and suggestions for future work.

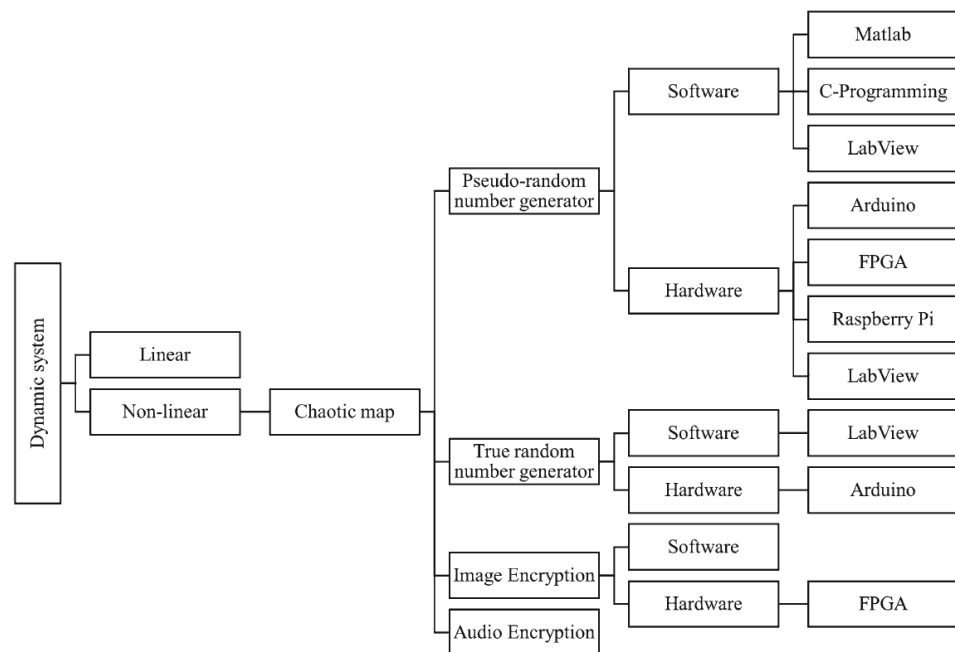


Figure 1. Applications of chaotic maps [13].

2. Related Works

Chaotic maps find applications in various domains of information security, including stream ciphers [21,22], block ciphers [23,24], hashing [25,26], steganography, and digital watermarking [27–30]. Consequently, a wide range of chaotic maps is available for generating pseudo-random numbers, with the logistic map being a popular choice among researchers due to its extensive exploration. Researchers have utilized various modified versions of the logistic map for pseudo-random number generation, such as the pseudo-randomly enhanced logistic map (PELM), which incorporates optimizations to improve randomness [31], and the floating-point-based modified logistic chaotic system, which employs floating-point arithmetic for increased precision [32]. The hyper-chaotic modified robust logistic map (HC-MRLM) introduces hyper-chaotic behavior to enhance complexity [33]. The modified logistic map, with an increased key-space range, expands the parameter range to generate more diverse sequences [34]. The nonlinear digitalized modified logistic map involves digitization techniques for digital implementation [35]. Furthermore, the optimized logistic map employing perturbation operation utilizes perturbations to enhance chaotic behavior and sensitivity to initial conditions [36].

The logistic map typically exhibits chaotic behavior within a narrow range of control parameter values, specifically between 3.57 and 4. However, efforts have been made to expand this parameter range and enhance the chaotic behavior of the logistic map. In a recent study focused on pseudo-random bit generation, logistic maps were utilized to generate multimodal maps, leading to an increased range of chaotic behavior within the parameter interval [37]. Additionally, researchers have explored the application of different chaotic maps, such as the Lorenz map, in their investigations.

Numerous electronic and hardware-based implementations of chaos-based PRNGs have been introduced in the literature. Matheus et al. presented a gate-level hardware implementation of a PRNG based on an exponential chaotic map [38]. Their hardware design utilized an FPGA device for the proposed PRNG system. There are several FPGA-based implementations of chaotic PRNGs, including those using a four-wing memristive hyperchaotic system and Bernoulli map [39], a reconfigurable chaotic PRNG based on FPGA [40], a chaos-based bitwise dynamical PRNG on FPGA [41], and an FPGA implementation of a chaos-based PRNG for secure communication [42]. FPGA devices are often preferred by algorithm designers for prototyping as they allow code execution at the gate level using hardware descriptive languages. While FPGA implementations are common for prototyping, real-world applications typically employ microcontrollers. Therefore, several proposals have focused on chaos-based PRNGs using microcontrollers. For example, a recent proposal introduced an analog circuit and microcontroller-based PRNG application based on a new easily realizable 4D chaotic system [43]. Another study realized a chaotic random bit generator using a microcontroller [44], and a hardware implementation of initials-boosted coexisting chaos in a two-dimensional sine map [45]. These proposals leverage microcontrollers to implement chaos-based PRNGs in practical applications. Chaos-based PRNGs find applications in various security domains, including stream ciphers [46], block ciphers [47], secure communications [48], image encryption [49], and video encryption [50], among others.

However, despite their popularity and successful applications, chaos-based PRNGs have certain shortcomings. Yeniçeri et al. conducted a study highlighting a vulnerability in a random number generator based on a time delay differential equation [51]. By predicting values in advance and coupling them with future states of a time-delay-based chaotic system, they were able to synchronize the system and generate signals similar to the original ones, posing a potential attack. Another study focused on the cryptanalysis of chaos-based PRNGs, identifying system weaknesses and suggesting improvements [52,53].

In the context of hardware-based attacks, Youling et al. conducted cryptanalysis on chaos-based cryptosystems [54]. They implemented a chaos-based cryptosystem on a microcontroller device and performed side-channel attack analysis and correlation power analysis. These attacks involve studying the execution time and power consumption of the cryptographic algorithm. Vulnerability and security analyses have also been performed in various other studies on chaos-based PRNGs within cryptographic applications. For example, researchers conducted a security analysis of an efficient chaos-based PRNG for video encryption applications [55]. Additionally, a comprehensive vulnerability analysis was performed for a chaos-based PRNG in another study [56,57]. These investigations contribute to understanding the vulnerabilities and security considerations associated with chaos-based PRNGs in cryptographic contexts.

Also, a novel method for generating pseudo-random sequences using coupled map lattices is presented in [58]. It addresses the limitation of chaotic maps, which have a restricted chaotic behavior range, limiting their use in cryptography. The proposed method introduces generalized symmetric maps with adaptive control parameters, allowing users to choose any symmetric chaotic map while ensuring independent and random output sequences. A lattice-based structure connects local maps to their neighboring nodes, increasing sequence complexity. The method's effectiveness is evaluated through various techniques, demonstrating a large key space, the generation of pseudo-random sequences, and suitability for IoT devices. The model employs a spatiotemporally coupled map

lattice system with adaptive values derived from local map accumulation points, ensuring consistent chaotic behavior. Detailed explanations of the system’s workings are provided in subsequent sections. Based on the reviewed literature, it is evident that the majority of recent research proposals focus on using well-known maps, such as the logistic map or Tent-Map, as local maps in cryptographic systems. These maps, including the logistic map and Tent-Map, belong to the family of symmetric chaotic maps, which also encompass higher-order maps. However, it is worth noting that the Tent-Map, in particular, has been observed to exhibit a strong dependency on the control variable μ for its chaotic behavior. To remedy this and other limitations, an efficient chaotic random number generator is proposed in this paper. The main goal of this paper is to design a generator that produces cryptographic-grade random numbers while prioritizing efficiency, compactness, and simplicity.

3. Preliminaries

The exploration of one-dimensional dynamical systems that exhibit chaotic behavior has been an area of active research. One of the elementary examples of such systems is the Tent-Map, which is a noninvertible, piecewise linear discrete map in one dimension. The Tent-Map displays chaotic dynamics and can be mathematically represented as follows [59]:

$$X_{i+1} = f(x_i, \mu) = \begin{cases} f_L(x_i, \mu) = \mu x_i, & \text{if } x_i < 0.5 \\ f_R(x_i, \mu) = \mu(1 - x_i), & \text{otherwise} \end{cases} \quad (1)$$

In the defined system, the function f operates on the interval $[0, 1]$ and maps it to itself (i.e., x_i belongs to $[0, 1]$ for all i greater than or equal to zero). The initial condition denoted as x_0 and the control parameter, denoted as μ , play crucial roles in Equation (1). The initial condition represents the starting value, while the control parameter determines the behavior of the system. Specifically, μ is a positive real constant within the range $[0, 2]$. The collection of real values $x_0, x_1, \dots, x_n, \dots$ is referred to as the orbit of the system. For every x_0 , there exists a corresponding orbit. The nature of the dynamical behavior exhibited by the system varies depending on the value of the control parameter μ , ranging from predictable to chaotic.

Figure 2 depicts the iterative procedure employed to generate the orbit of the Tent-Map for the given initial condition $x_0 = 0.1$ and control parameter $\mu = 1.9998$. By examining this figure, it becomes apparent that the function $f(x_i, \mu)$ reaches its maximum value when x_i equals 0.5 [59].

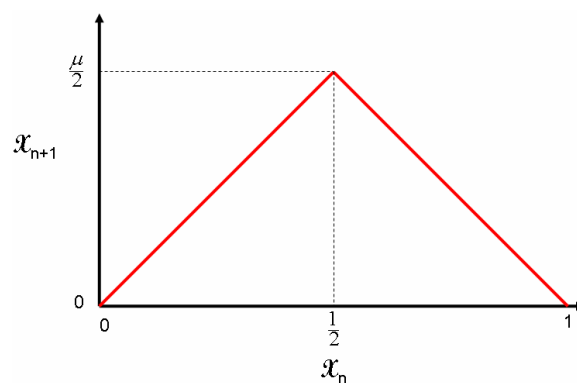


Figure 2. Graph of Tent-Map function.

As previously mentioned, one way to qualitatively analyze the Tent-Map is through its bifurcation diagram, which illustrates its asymptotic behavior. The bifurcation diagram represents the orbits of the map as a function of the control parameter μ , displaying the potential ranges of values for x_i at different values of i , where i represents an index or a counting variable. Figure 3 shows the bifurcation diagram of the system for the range of control parameters for $\mu \in [1, 2]$ [60,61].

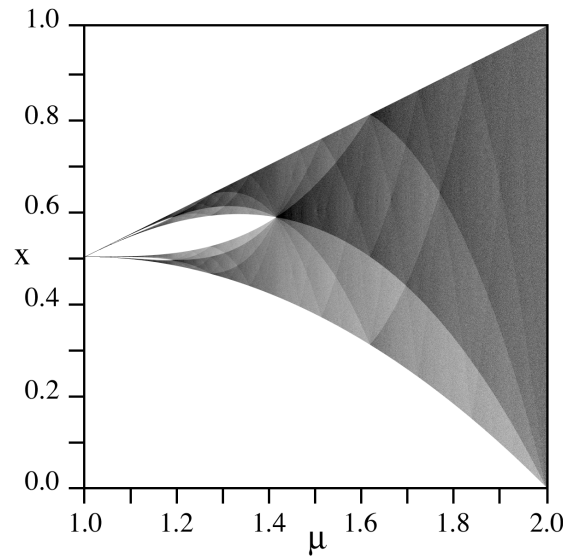


Figure 3. Bifurcation diagram of Tent-Map for $\mu \in [1, 2]$.

In the following section, a comprehensive analysis of the bifurcation diagram of the system is provided.

1. When the control parameter μ belongs to the interval $[0, 1)$, the system exhibits a single fixed point at $x = 0$, regardless of the initial conditions x_0 . Consequently, the trajectories originating from any point within the interval $[0, 1]$ will converge to $x = 0$.
2. When the control parameter μ equals one, the system possesses fixed points for all values of x except for $x = 0.5$. In other words, the trajectories are attracted to various fixed points, excluding the value $x = 0.5$.
3. When the control parameter μ exceeds one, the system features two unstable fixed points: one at $x = 0$ and the other at $x = \mu / (\mu + 1)$. The trajectories in this case are repelled by both fixed points, resulting in complex and chaotic behavior.
4. When the control parameter μ falls between one and the square root of two, the system exhibits a characteristic where certain intervals, ranging from $\mu \sqrt{\mu^2 / 2}$ to μ^2 , map onto themselves. This collection of intervals represents the Julia set of the system. As the value of μ exceeds the square root of two, these intervals merge together, forming a single connected Julia set.
5. Within the range of control parameters from 1 to 2, the behavior of the system becomes chaotic. All orbits become unstable, and the interval between $\mu \sqrt{\mu^2 / 2}$ and μ^2 contains both periodic and nonperiodic points.
6. When μ is equal to two, System (1) demonstrates fully chaotic behavior. It maps the interval $[0, 1]$ onto itself. The iterations produced by System (1) exhibit statistically uncorrelated noise with a uniform independent and identically distributed [62–64] distribution. In other words, the natural invariant density for the system is equal to one.

Unlike the logistic map and some other chaotic maps, it is noteworthy to mention that the behavior of the iterates in the system transitions directly from stable fixed-point behavior when μ is less than one to chaotic behavior when μ is greater than one, without undergoing a period doubling phenomenon.

Lyapunov exponents play a crucial role in analyzing nonlinear dynamics and distinguishing between chaotic and nonchaotic motion. Therefore, Lyapunov exponents are employed to assess whether a particular value of μ induces chaotic evolution in the system [65]. The Lyapunov exponent, denoted as λ , is calculated as follows [59]:

$$\lambda(x_0) = \frac{1}{k} (\ln|\dot{f}(\mu, x_0)| + \ln|\dot{f}(\mu, x_1)| + \dots + \ln|\dot{f}(\mu, x_{k-1})|). \quad (2)$$

Clearly,

$$f(x_i, \mu) \begin{cases} f_L(x_i, \mu) = \mu, & \text{if } x_i < 0.5 \\ f_R(x_i, \mu) = -\mu, & \text{if } x_i > 0.5 \end{cases} \tag{3}$$

Moreover, it should be noted that $f_0(\mu, 0.5)$ is undefined. Therefore, for any orbit of the system that does not include 0.5, the Lyapunov exponent, denoted as $\lambda(x_0)$, is equal to the natural logarithm of the control parameter μ . Specifically, $\lambda(x_0) > 0$ when $\mu > 1$, and $\lambda(x_0) < 0$ when $\mu < 1$. This observation aligns perfectly with the bifurcation diagram presented in Figure 2, confirming the instability of fixed points and periodic points when $\mu > 1$. Notably, the fact that $\lambda(x_0)$ is always positive for $\mu > 1$ implies that the bifurcation diagram of the system (as evident from Figure 2) lacks periodic windows typically observed in the case of a logistic map. Figure 4 displays the Lyapunov exponents of the system as a function of its control parameter μ .

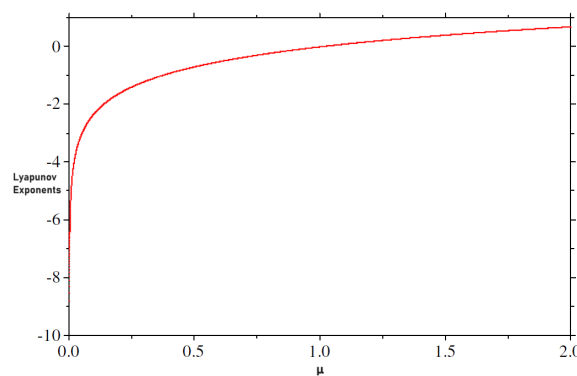


Figure 4. The Lyapunov exponents of Tent-Map across the range of control parameter values μ from 0.0001 to -2 .

To ensure chaotic behavior and fulfill cryptographic requirements, the control parameter μ should be selected to be greater than one. In stream cipher applications, the iterate values of the system need to exhibit random-like properties and cover the entire range from 0 to 1. Therefore, the choice of the control parameter μ is crucial in generating iterate values that are distributed uniformly from 0 to 1. As the control parameter approaches two, the iterate values demonstrate a chaotic distribution that spans a dense point set within a finite interval converging toward (0, 1). This behavior has been extensively studied in the literature [62–64].

Figure 5 demonstrates the high sensitivity of the system to both its initial condition and control parameters, which is a crucial requirement for cryptographic applications. Two sequences of iterates, $\{X_i\}_{i=1}^{100}$ and $\{\tilde{X}_i\}_{i=1}^{100}$, are generated by the system with a fixed control parameter of 1.9999 but different initial conditions, $x_0 = 0.1$ and $x_0 = 0.100001$. The figure clearly illustrates that, even after a small number of iterations, the two sequences diverge significantly.

In stream cipher applications, chaotic systems like this system often utilize the initial condition and control parameter as the secret key, denoted as K, for the cipher system. Prior to communication, this key must be shared between the communicating parties and kept confidential. The security of a cipher system relies on the secrecy of its key, ensuring protection against cryptanalytic attacks from potential eavesdroppers. While many chaotic cipher systems based on a single chaotic map, including those utilizing the Tent-Map [66,67], have been shown to be insecure for cryptographic applications [68,69], the favorable properties exhibited by the Tent-Map in this system motivate its implementation in the design of practical and secure ciphers.

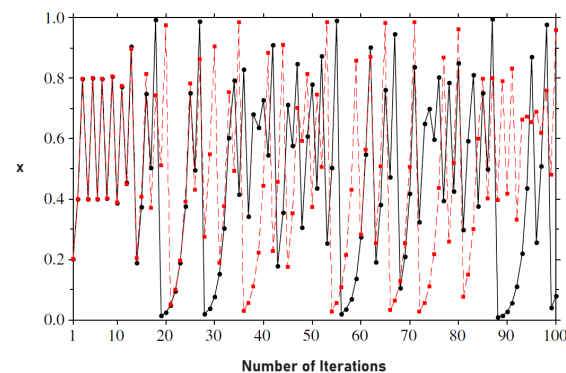


Figure 5. Time series plot for two sequences of iterates, satisfying $|x_0 - x'_0| = 10^{-6}$. The solid line stands for $(x_0, \mu) = (0.1, 1.9999)$, while the dashed line stands for $(0.100001, 1.9999)$.

4. Proposed Scheme—State-Based Tent-Map (SBTM)

The goal of this article is to provide an innovative methodology for designing and constructing a chaotic pseudo-random number generator (CPRNG) that generates cryptographic grade random numbers yet is very efficient, small, and simple. Tent-Map function is a one-dimensional chaotic map that exhibits chaotic behavior and is widely used in dynamical systems and cryptography. It is defined by the equation: $f(x) = r \times \min(x, 1 - x)$, where r is a parameter that determines the nature of the map. The function takes an input value x between 0 and 1 and generates a new output value based on the minimum of x and $1 - x$, scaled by the parameter r . The Tent-Map function displays sensitive dependence on initial conditions, meaning that even small changes in the input can lead to significantly different outputs, contributing to its chaotic properties. It has applications in generating pseudorandom numbers, data encryption, and secure communication protocols. However, the Tent-Map faces significant limitations, including the following: (i) a restricted set of control parameters used as security keys, rendering them susceptible to cyberattacks that can be easily overcome through brute force methods, (ii) a limited chaotic range of their control parameters and/or periodic behavior for certain values within their parameter range, and (iii) the nonuniform distribution of chaotic sequences generated by 1D chaotic systems [17].

The Tent-Map, chosen for its simplicity and efficiency, serves as the fundamental component of the proposed algorithm (SBTM). It not only addresses the shortcomings of the original Tent-Map but also generates high-quality random numbers suitable for cryptographic purposes. In the SBTM, a circular array of numbers is introduced as the state, offering great flexibility in terms of initial values and the number of elements. Several arrangements were tested, all yielding outstanding outcomes, with certain arrangements outperforming others, as demonstrated in Section 5. There are multiple options for selecting the initial values of the state array, and two proposals are presented, with emphasis on the first approach:

- (a) The array's initial values consist of square roots of prime numbers that are less than a given value, denoted as n .
- (b) The initial values are derived from the digits of Pi multiplied by a renowned constant like e (2.718281828459045).

Figure 6 displays the state representation for both versions.

In the left diagram (a), a circular array with five elements is depicted, initialized with the square roots of 2, 3, 5, 7, and 11 (random numbers). On the right diagram (b), a five-element state is shown, initialized with five digits of Pi (1, 4, 1, 5, and 9) multiplied by the constant e .

The algorithm is characterized by its simplicity. It commences by initializing the state, as illustrated in Figure 6, and proceeds by multiplying each element by a factor, denoted as f . The value of f , such as 1000, 10,000, or 100,000, serves to mitigate the influence of extremely small seed or control variables. It is important to note that in Figure 6b, a one is added to

the digits of Pi to prevent a state element from being zero. Additionally, multiplying each element by the constant e produces random fractions. The number of elements in the array is intentionally chosen to be a prime number. Moreover, an index, denoted as i , is randomly initialized based on the seed. Lastly, a variable named sign is initialized to one.

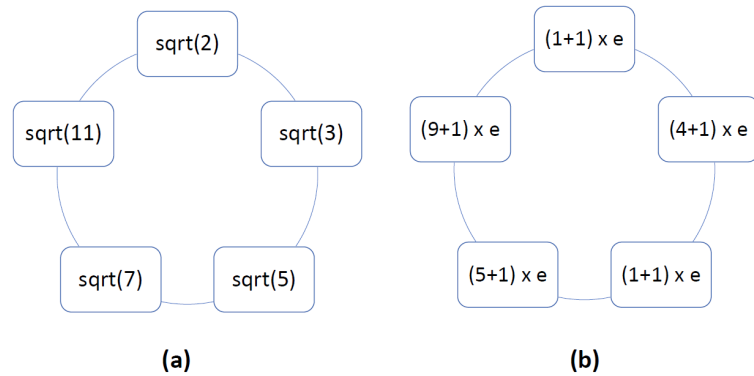


Figure 6. State of SBTM CPRNG.

Once the state array and other variables have been initialized, the process of generating random numbers can commence by supplying a seed, denoted as s , and a control variable, denoted as μ , as the inputs. To circumvent the potential issue of handling extremely small values of μ , or worse, $\mu = 0$, and considering that the algorithm accepts μ as any positive real number, μ is adjusted based on Equation (4) in the following manner:

$$\mu = (\mu + 1) \times e. \tag{4}$$

The algorithm employs Equation (5) to generate the next value r_{n+1} , according to the following equation:

$$r_{n+1} \begin{cases} r_n \times \mu \times state[i], & \text{if } n < 0.5 \\ (1 - r_n) \times \mu \times state[i], & \text{otherwise} \end{cases} \tag{5}$$

In this equation, state refers to the circular array, i is a valid index in the array, μ is the control variable, and r_n is the previous random value starting with the seed s . r_{n+1} could be a number greater than zero since the state elements are large numbers. To bring it down to a value in the range (0, 1), the fraction part is used as shown in Equation (6).

$$r_{n+1} = r_{n+1} - int(r_{n+1}) \tag{6}$$

To update the state of the SBTM, the algorithm performs either an addition or a subtraction of r_{n+1} to the state element indexed by the index i , as depicted in Equations (7) and (8).

$$sign = sign \times -1 \tag{7}$$

$$state[i] = state[i] + (sign \times r_{n+1}) \tag{8}$$

If the state element i deviates from its initial value by a threshold t (set at 500), an automatic reset is triggered. During this step, t is added or subtracted from element i to restore it to a value near its initial value, as shown in Equation (9).

$$state[i] \begin{cases} state[i] + t, & \text{if } state[i] < initial_state[i] - t \\ state[i] - t, & \text{if } state[i] > initial_state[i] + t \end{cases} \tag{9}$$

Finally, the value of the index i is updated, as shown in Equation (10), and r_{n+1} is returned.

$$i = (i + 1) \% \text{size}(\text{State}) \quad (10)$$

The pseudocode of the algorithm is presented in Algorithm 1.

Algorithm 1 SBTM Pseudocode

Require: $Seed, \mu$

Ensure: *Random number*

```

1: SBTM ( $seed, \mu$ )
2: Initialize the state array
3: Initialize sign and  $i$ 
4: Set  $r_n = seed$ 
5: function GENERATE( $r_{n+1}(r_n)$ )
6:   def Generate  $r_{n+1}(r_n)$ 
7:     Use Equation (1) to generate  $r_n + 1$ 
8:     Set  $r_{n+1} = fraction(r_{n+1})$ 
9:     Update the sign
10:    Update the state
11:    Automatic reset if needed
12:    Update the index variable
13:     $r_n = r_{n+1}$ 
14:    return  $r_{n+1}$ 
15: end function

```

Discussion of Design Decisions

Variables, constants, and steps used in the SBTM were chosen based on the following considerations:

- (a) Number of elements in the circular array: The goal was to minimize the size of the generator's state, as increasing its size is undesirable. Various experiments were conducted with different numbers of elements to examine their impact on the generated sequence. All the values used were prime numbers, starting from two. Recommendations for the optimal size are provided in Section 5.
- (b) Initialization of the state: This part was deemed crucial but finding the optimal set of initial values proved to be challenging. The rationale behind the chosen values was the need for irrationality. Specifically, the square roots of prime numbers were considered as if they met this criterion. While higher-order roots could potentially be as effective, further experimentation with them was left for future work. In version b, the individual digits of Pi were employed. The attractiveness of Pi lies in its irrational nature and the random arrangement of its digits. Since the digits of Pi do not possess fractional parts and could include zeros, the approach taken was to add one to each digit to avoid zeros and then multiply them by e (2.718) to introduce a fractional component. It is believed that any other irrational value could also be suitable, but the experimentation was focused solely on e . Although only 10 values (1–10) exist, the arrangement of these values in the state array had an influence on the generated sequences.
- (c) The factor f : It is evident that if the control variable μ approaches zero, the sequence will diminish. To mitigate the impact of this phenomenon, the initial values of the state are multiplied by a factor, such as 1000, 10,000, or 100,000. Selecting larger values will affect the random sequence since the majority of the digits stored in the double data type become part of the whole number, which is truncated in our algorithm, as demonstrated in Equation (2). Recommendations for the optimal factor values are provided in Section 5.
- (d) Equation (2): The numbers generated by Equation (1) are likely to exceed one. To ensure that the generated value remains within the desired limits of 0 and 1, the whole

part of the number is discarded and only the fraction is considered. This mechanism effectively prevents backtracking, aligning with the principles outlined in “NIST Special Publication 800-90A Revision 1” [70].

- (e) Updating the state: If the array had been infinite, there would have been no need to update its values. However, due to the limited size of the state array and the desire for more chaotic behavior, updating the state became necessary. Various methods could have been employed for this purpose, but we aimed for a mechanism that was both simple and efficient. Therefore, we chose to add or subtract the newly generated value r_{n+1} to the state element that produced it, based on the alternating sign variable that switched between 1 and -1 . This approach ensured that the elements were updated while maintaining their values close to their initial states, as the addition and subtraction of random values tended to cancel each other out.
- (f) State automatic reset: To maintain alignment with [70] and mitigate the risk of a state element approaching zero or excessively large values, an automatic reset mechanism is implemented. This process ensures that the state element is reset to a value in close proximity to its initial value. By doing so, the algorithm enhances its resistance against adversarial attempts to predict future outputs through the generation of a large number of random values.
- (g) Updating index i variable: This variable determined the next state element to be used in generating the subsequent random value. There were several ways to update this variable, but we opted for a straightforward and efficient approach. By incrementing it by one and utilizing the Mod operator, we ensured that it wrapped back to zero as it approached the size of the state array. This sequential updating method proved to be both efficient and simple.

5. Experiments and Results

Prior to delving into the experiment results, it is crucial to assess the chaotic behavior of the SBTM in comparison to the Tent-Map. This evaluation involves calculating Lyapunov exponents and constructing a Bifurcation diagram. Since the SBTM is a non-differentiable function, the Lyapunov exponent calculation algorithm described in [71] was utilized. To ensure a fair comparison, the same method from [71] was employed to compute Lyapunov exponents for both the SBTM and the original Tent-Map.

Figure 7 showcases the Lyapunov exponents for the original Tent-Map, while Figures 8–10 exhibit the results for the SBTM with distinct ranges of the control variable. It becomes apparent that Figure 7 bears a resemblance to Figure 4, with positive values emerging as μ approaches 1.2. In contrast, the SBTM demonstrates significantly higher positive values for different ranges of μ starting from 0 to 2, 10, and 100, as depicted in Figures 8, 9 and 10, respectively.

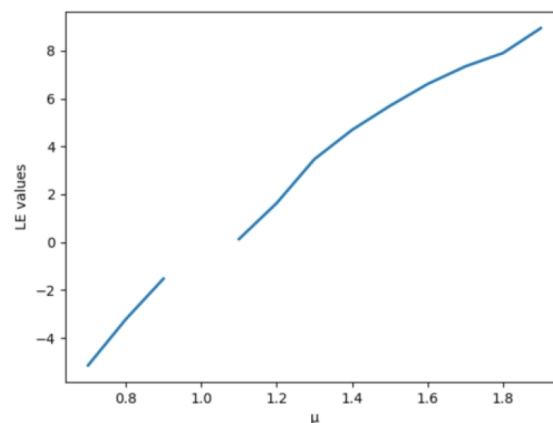


Figure 7. The Lyapunov exponents of Tent-Map, $\mu \in [0, 2]$ using [71].

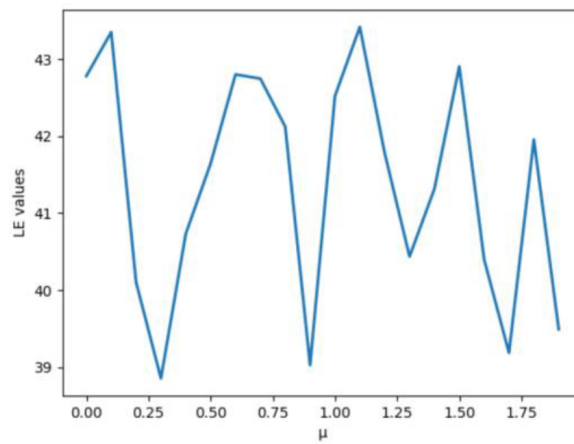


Figure 8. The Lyapunov exponents of SBTM, $\mu \in [0, 2]$ using [71].

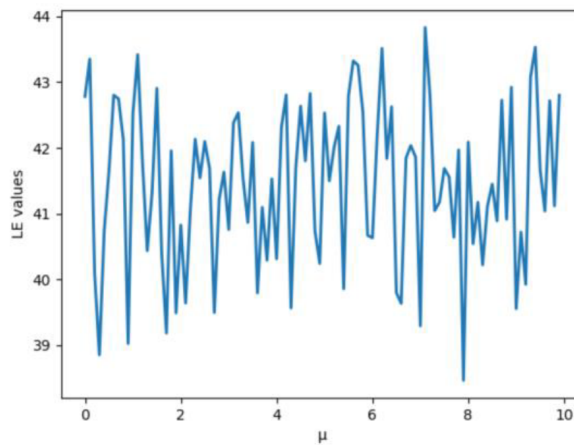


Figure 9. The Lyapunov exponents of SBTM, $\mu \in [0, 10]$ using [71].

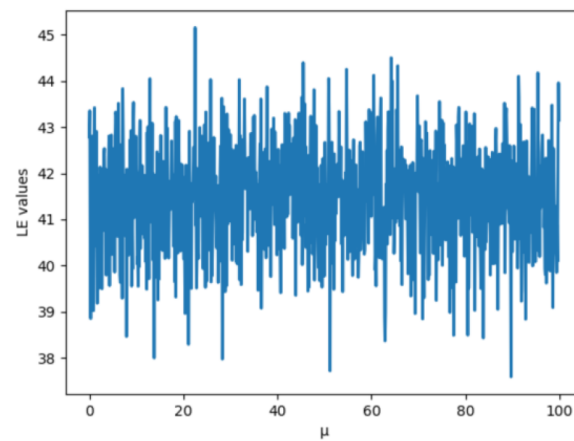


Figure 10. The Lyapunov exponents of SBTM, $\mu \in [0, 100]$ using [71].

Likewise, the Bifurcation diagrams in Figures 11–13 illustrate the extensive chaotic behavior of the SBTM across the entire duration of the three periods $[0, 2]$, $[0, 10]$, and $[0, 100]$. The stark contrast between these diagrams and those of the original Tent-Map algorithm makes it evident that the SBTM exhibits superior chaotic dynamics.

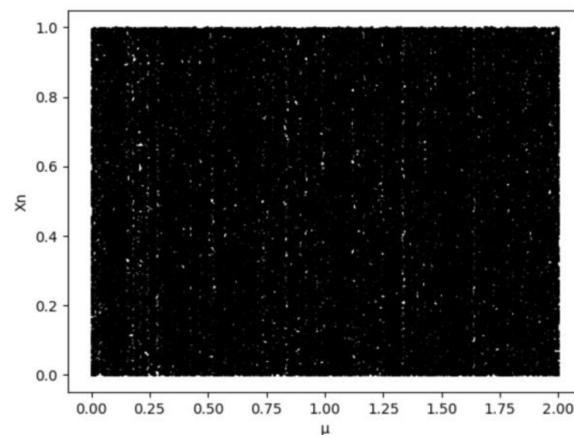


Figure 11. Bifurcation diagram of SBTM for $\mu \in [0, 2]$.

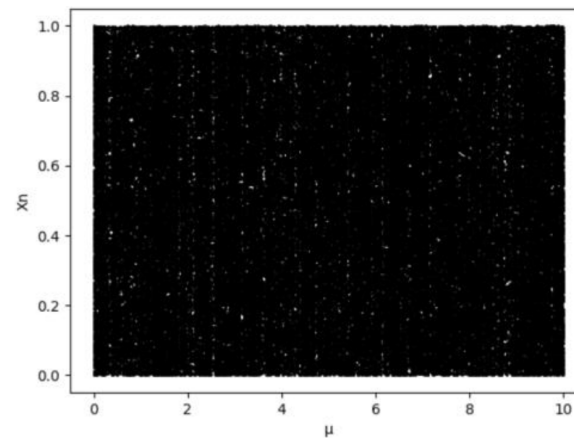


Figure 12. Bifurcation diagram of SBTM for $\mu \in [0, 10]$.

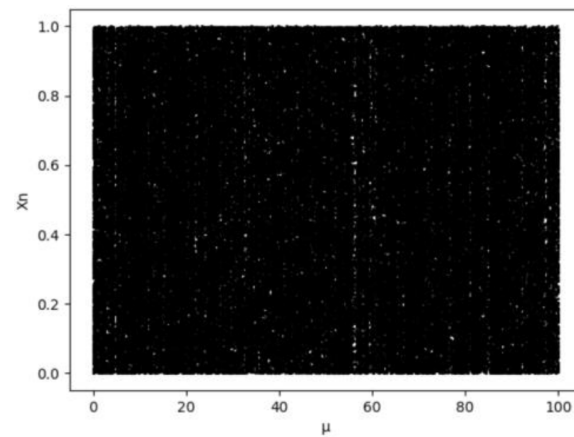


Figure 13. Bifurcation diagram of SBTM for $\mu \in [0, 100]$.

In order to assess the performance of the SBTM, a total of 800 random number sequences were generated. Each sequence comprised two billion unsigned integers, resulting in a size of 8 GB. To generate these sequences, various combinations of control variable values, seed values, and state element values were employed. Specifically, the control variable adopted the values, the seed values, and the number of state elements varied according to the values outlined in Table 1. It is noteworthy to mention that the control variable values were in the period $[0, 2]$ to match those used in the original Tent-Map.

Table 1. Values of control variable, seed, and micro.

Values of control variable	Micro	0.00, 0.20, 0.40, 0.60, 0.80, 1.00, 1.20, 1.40, 1.60, 1.80
Values of seed	Seed	0.01, 0.10, 0.19, 0.28, 0.37, 0.46, 0.55, 0.64, 0.73, 0.82
Values of state	State elements	2, 3, 5, 11, 19, 47, 97, 997

To assess the effectiveness of the proposed SBTM, the Dieharder test suite was used, as it encompasses an extensive collection of statistical tests formulated to evaluate the randomness and quality of random number generators. Dieharder is widely regarded as the best test suite for random number generators due to its comprehensive coverage of statistical tests, adherence to established standards, sensitivity to deviations from randomness, widespread acceptance, and open-source nature [65,72,73]. It offers a range of tests that assess various aspects of randomness, providing a rigorous evaluation of the quality and randomness of random number generators. Its popularity and flexibility make it a trusted tool for researchers and developers in assessing the performance of random number generators. Table 2 below provides an overview of the Dieharder suite tests used to assess the sequences generated by the proposed SBTM.

Table 2. Description of Dieharder tests.

Test(s)	Description
STS_serial (Serial Test)	The STS_serial test aims to identify any sequential patterns present in the generated sequence. By analyzing overlapping subsequences of a fixed length, the test assesses whether the sequence deviates from randomness. It calculates the frequency of specific patterns within the sequence and compares it to the expected occurrence in a random sequence. Notable deviations can indicate the presence of nonrandom behavior.
STS_runs (Runs Test)	The STS_runs test investigates the presence of consecutive repetitions of the same value, known as "runs," within the sequence. It evaluates whether the number of runs consisting of ones and zeros falls within the anticipated range for a random sequence. If there are substantial deviations from the expected range, it may indicate the existence of nonrandomness or potential bias in the generated sequence.
STS_monobit (Monobit Test)	The STS_monobit test is a fundamental assessment that examines the distribution of ones and zeros within the sequence. Its purpose is to determine if the proportion of ones and zeros is approximately equal, as this is a crucial characteristic of a random sequence. Substantial deviations from a balanced distribution may be indicative of nonrandomness.
RGB_permutations (RGB Permutations Test)	The RGB_permutations test examines the arrangements of triplets formed by consecutive values in the generated sequence. It scrutinizes the presence of unforeseen patterns or regularities within these permutations and compares them to the anticipated distribution in a random sequence. Deviations from the expected randomness can be indicative of nonrandom behavior.
RGB_minimum_distance (RGB Minimum Distance Test)	The RG_minimum_distance test quantifies the minimum Euclidean distance between successive triplets of values in the sequence. It evaluates how the values are dispersed or clustered within the RGB color space. Deviations from the anticipated distances may indicate the presence of patterns or nonrandomness in the generated sequence.
rgb_lagged_sum (RGB Lagged Sum Test)	The rgb_lagged_sum test investigates the cumulative sum of pairs of values with a lag in the RGB sequence. It evaluates the presence of patterns or anomalies in these cumulative sums and compares them to the anticipated distribution in a random sequence. Deviations from randomness can suggest nonrandom behavior within the sequence.
rgb_kstest_test (RGB Kolmogorov–Smirnov Test)	The rgb_kstest_test employs the Kolmogorov–Smirnov test on the RGB sequence to evaluate its adherence to a uniform distribution within the RGB color space. It examines significant deviations from uniformity, which may indicate the presence of nonrandomness or bias in the generated sequence.

Table 2. Cont.

Test(s)	Description
rgb_bitdist (RGB Bit Distribution Test)	The rgb_bitdist test centers around the distribution of bits within the RGB sequence. It examines the occurrence of various bit patterns and compares their frequencies to the expected distribution in a random sequence. Deviations from this expected distribution may suggest nonrandom behavior or bias in the generated sequence.
marsaglia_tsang_gcd (Marsaglia and Tsang GCD Test)	The marsaglia_tsang_gcd test scrutinizes the greatest common divisor (GCD) of value pairs in the sequence. It evaluates the presence of patterns or regularities in the GCD values and compares them to the anticipated distribution in a random sequence. Deviations from randomness can suggest the existence of nonrandom behavior in the generated sequence.
diehard_sums, diehard_squeeze, diehard_runs, diehard_rank_32×32, diehard_parking_lot, diehard_oqso, diehard_operm5, diehard_dna, diehard_crps, diehard_count_1st_byt, diehard_bitstream, diehard_birthday, diehard_3dsphere, diehard_2dsphere, dab_monobit2, dab_filltree2, and dab_filltree	Each test focuses on specific statistical properties and patterns present within the generated sequence. These tests assess characteristics, such as bit distributions, consecutive runs, permutation patterns, spatial distribution, and more. Their purpose is to identify any deviations from randomness and detect potential nonrandomness or biases in the sequence.

In evaluating the performance of the proposed SBTM, Dieharder subjects the generator’s sequences to a battery of 91,200 individual tests (in total) to assess different aspects of randomness and statistical properties. In an analysis of the results obtained, as depicted in Table 3 and summarized in Figure 14, it was found that out of the total tests conducted, 303 tests exhibited failures (only 0.33% of the total), indicating significant deviations from randomness. On the other hand, a substantial majority of 88,830 tests passed (97.4% of the total), demonstrating the generator’s ability to generate sufficiently random sequences. Additionally, 2067 tests were categorized as weak (2.27% of the total). This comprehensive examination of the Dieharder test results provides valuable insights into the strengths and high quality of the tested random number generator.

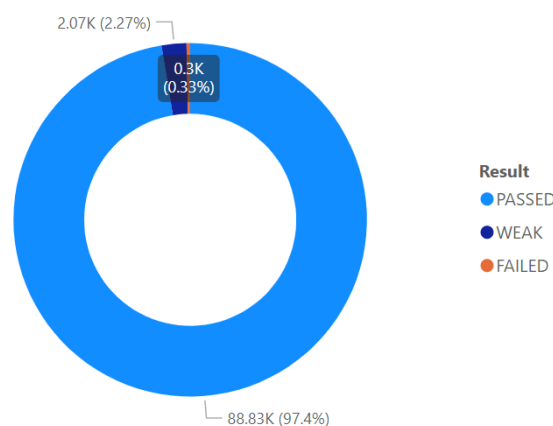


Figure 14. Summary of results percentages of Dieharder tests.

The experimental results in accordance with the values of the control variable, the values of the seed variable, and the values of the state are typically presented in an organized and comprehensible manner in Tables 4, 5 and 6, respectively, to facilitate comprehension and analysis. In this paper, the findings have been structured into three distinct tables: the table of results according to the values of the control variable, Table 4, the table of results

according to the values of the seed variable, Table 5, and the table of results according to the values of the state Table 6. These tables offer a comprehensive summary of the results by presenting the relationship between the failure and success rates and the various control parameters. From Table 4, it is evident that no failed tests were reported when μ values were 0, 0.2, 0.8, and 1.2. Similarly, Table 5 indicates that no failed tests occurred for the seed values of 0.19 and 0.55. Furthermore, Table 6 shows that when the number of elements in the state array was three and five, there were no failed tests, making them optimal choices.

Table 3. Results of Dieharder tests.

Test_Name	Failed	Passed	Weak	Total
dab_bytedistrib	2	792	6	800
dab_dct	3	789	8	800
dab_filltree	8	1573	19	1600
dab_filltree2	8	1572	20	1600
dab_monobit2	5	786	9	800
diehard_2dsphere	0	783	17	800
diehard_3dsphere	0	775	25	800
diehard_birthdays	0	785	15	800
diehard_birthdays	0	785	15	800
diehard_bitstream	1	775	24	800
diehard_count_1s_byt	4	780	16	800
diehard_count_1s_str	4	781	15	800
diehard_craps	6	1558	36	1600
diehard_dna	4	786	10	800
diehard_operm5	0	789	11	800
diehard_opso	2	778	20	800
diehard_oqso	2	779	19	800
diehard_parking_lot	4	785	11	800
diehard_rank_32×32	1	791	8	800
diehard_rank_6×8	1	782	17	800
diehard_runs	6	1554	40	800
diehard_squeeze	3	780	17	800
diehard_sums	3	711	86	800
marsaglia_tsang_gcd	0	1561	39	1600
rgb_bitdist	46	9376	178	9600
rgb_kstest_test	0	787	13	800
rgb_lagged_sum	186	25,505	711	26,400
rgb_minimum_distance	0	3152	48	3200
rgb_permutations	1	3121	78	3200
sts_monobit	0	784	16	800
sts_runs	0	776	24	800
sts_serial	3	23,486	511	24,000
total	303	88,830	2067	91,200

Table 4. The results according to the values of control variable.

Micro	Failed	Passed	Weak	Total
0.00	0	8881	239	9120
0.20	0	8903	217	9120
0.40	41	8845	234	9120
0.60	49	8883	188	9120
0.80	0	8912	208	9120
1.00	40	8874	206	9120
1.20	0	8949	171	9120
1.40	66	8841	213	9120
1.60	39	8889	192	9120
1.80	68	8853	199	9120
Total	303	88,830	2067	91,200

Table 5. The results according to the values of seed variable.

Seed	Failed	Passed	Weak	Total
0.01	38	8895	187	9120
0.10	68	8830	222	9120
0.19	0	8906	214	9120
0.28	48	8871	201	9120
0.37	1	8926	193	9120
0.46	14	8856	250	9120
0.55	0	8923	197	9120
0.64	42	8893	185	9120
0.73	39	8874	207	9120
0.82	53	8856	211	9120
Total	303	88,830	2067	91,200

Table 6. The results according to the values of state.

No. of State Elements	Failed	Passed	Weak	Total
2	1	11,128	271	11,400
3	0	11,159	241	11,400
5	0	11,147	253	11,400
11	1	11,142	257	11,400
19	1	11,135	264	11,400
47	42	11,098	260	11,400
97	37	11,108	255	11,400
997	221	10,913	266	11,400
Total	303	88,830	2067	91,200

The seed and the p -value in Dieharder are interconnected in the evaluation of random number generation. The seed represents the initial value(s) used to initialize the random number generator. It acts as the starting point for generating a sequence of random numbers. On the other hand, the p -value is a statistical measure that assesses the compatibility between the observed data, generated by the random number generator, and the expected distribution assuming randomness. The relationship between the seed and the p -value, as illustrated in Figure 15, arises from the fact that the choice of seed can influence the quality and statistical properties of the generated sequence. Different seed values can lead to distinct sequences of random numbers. Dieharder applies various statistical tests to these sequences to evaluate their randomness and quality. When analyzing the results using Dieharder, the obtained p -value indicates the degree to which the generated sequence aligns with the expected distribution. A higher p -value, closer to one, signifies a better fit with the expected distribution, indicating a higher level of randomness. Conversely, a lower p -value, closer to zero, suggests significant deviations from randomness.

It is important to recognize that the relationship between the seed and the p -value is indirect. The choice of seed can impact the characteristics of the generated sequence, which subsequently affects the outcomes of the statistical tests and the resulting p -values. By carefully selecting suitable seeds, it is possible to generate sequences that demonstrate stronger adherence to randomness and yield higher p -values in the Dieharder tests. Overall, the average p -values across all seeds were found to be very similar.

In addition, this paper investigates the correlation between the average TSamples and the test status, which plays a critical role in assessing the quality and randomness of random number generators. By analyzing the average TSamples values across various test outcomes, this study uncovers noteworthy patterns as illustrated in Figure 16. Notably, tests with an average TSamples of 1.99 million predominantly receive a passed status, indicating a high level of randomness. Conversely, tests characterized by an average TSamples of 2.65 million result in a slightly more failed status. Additionally, tests with an average TSamples of 1.28 million are categorized as weak. This gives clear evidence that

the default values of the Dieharder test suite may not be suitable for all tests, meaning that some tests require more TSamples to pass and others require less to pass.

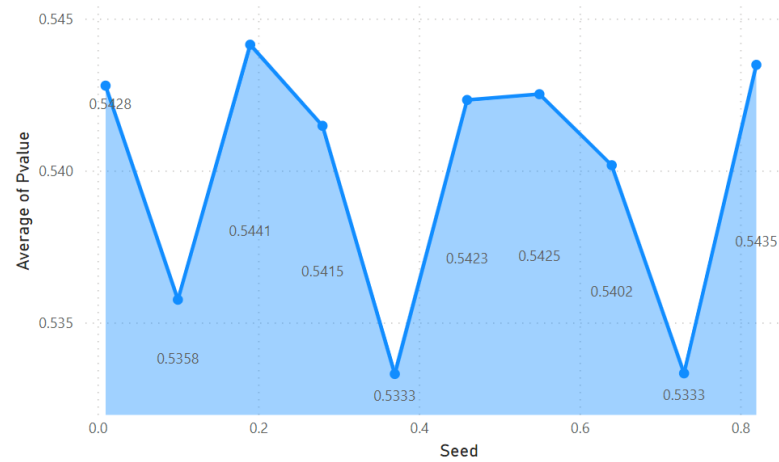


Figure 15. Relationship between values of seed and average p value.

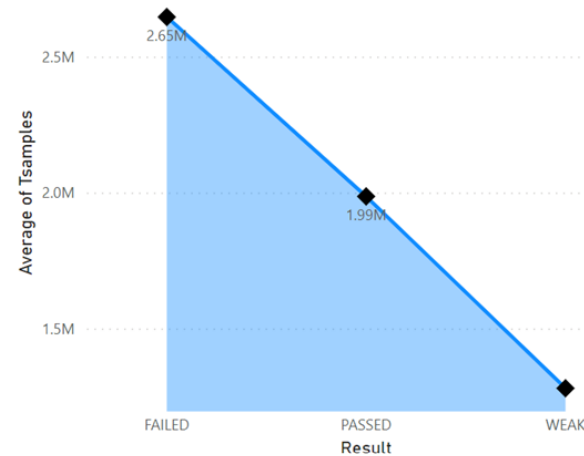


Figure 16. Relationship between average of TSamples and result.

In sum, the design choices made during the development of the SBTM algorithm have played a vital role in establishing its significance and robustness in generating efficient random sequences. The selection of the variables, constants, and steps was a deliberate process aimed at optimizing the performance of the generator. Several important factors guided these decisions. Firstly, the number of elements in the circular array was carefully chosen to minimize the state size, ensuring efficiency. Prime numbers were employed as the array size, and experiments were conducted to assess their impact on the generated sequences. Optimal sizes were determined to achieve desirable results. Secondly, the initialization of the state was recognized as a crucial step, leading to the exploration of irrational values. The square roots of prime numbers were utilized, leveraging their irrational nature and random arrangement. Further investigations could explore alternative irrational values. Thirdly, the introduction of the factor "f" addressed the issue of diminishing sequences when the seed approached zero. Multiplying the initial state values by a factor helped mitigate this phenomenon.

Furthermore, Equation (6) was incorporated to ensure that the generated values remained within the range of 0 and 1, avoiding backtracking and adhering to established principles of random number generators. Also, an automatic reset mechanism was integrated to enhance resistance against prediction attempts. This process ensured that state elements returned to values close to their initial states, aligning with established standards and reducing the risk of extreme values or element depletion. These design decisions col-

lectively contribute to the significance and robustness of the SBTM algorithm in generating efficient sequences with desirable random properties.

Analysis of Failed Tests

It is crucial to emphasize that a failed test does not necessarily imply a poor sequence; it simply indicates that the P-value is either extremely small, close to zero, or extremely high, close to one. It should be noted that the authors did not modify the default parameters of the Dieharder software Version 3.31.1. and reported the results as they were generated. To recap, a comprehensive testing process was conducted on a total of 800 sequences. Each sequence underwent a battery of 114 tests using the Dieharder software, consisting of 31 unique tests and multiple repetitions with varying Ntuple values. The distribution of these unique tests and the corresponding frequency of repetition for each can be found in Table 7. It is noteworthy that the NIST tests are a subset of these overall assessments, specifically encompassing the sts_serial tests.

Table 7. Unique Dieharder tests and their frequency.

Test	Times Repeated with Different Ntuple or Other Parameters
diehard_birthdays	1
diehard_operm5	1
diehard_rank_32x32	1
diehard_rank_6x8	1
diehard_bitstream	1
diehard_opso	1
diehard_oqso	1
diehard_dna	1
diehard_count_1s_str	1
diehard_count_1s_byt	1
diehard_parking_lot	1
diehard_2dsphere	1
diehard_3dsphere	1
diehard_squeeze	1
diehard_sums	1
diehard_runs	2
diehard_craps	2
marsaglia_tsang_gcd	2
sts_monobit	1
sts_runs	1
sts_serial	30
rgb_bitdist	12
rgb_minimum_distance	4
rgb_permutations	4
rgb_lagged_sum	33
rgb_kstest_test	1
dab_bytedistrib	1
dab_dct	1
dab_filltree	2
dab_filltree2	2
dab_monobit2	1

Among the 800 sequences that underwent testing, only 11 sequences exhibited one or more failed tests, highlighting the robust performance of the SBTM given the rigorous nature of Dieharder testing. Furthermore, a total of 739 sequences demonstrated one or more weak test results, while 59 sequences showed no failed or weak test outcomes. The absence of failed or weak tests in these sequences is typically associated with exceptionally strong and lengthy (approximately 64 billion numbers, equivalent to 256 GB) sequences generated by a robust generator, like AES_OFB. A summary of these results is provided in Table 8.

Table 8. Summary of test results by sequence.

Description	Count	Comments
Perfect sequences	59	No failed or weak tests
Sequences with weak tests	739	9 of which have also failed tests
Sequences with failed tests	11	9 of which have also weak tests
Sequences with failed but no weak tests	2	Have failed tests but no weak tests
All sequences	800	-

To gain a deeper understanding of the sequences with failed tests, Table 9 presents additional details. Among the sequences, four experienced only one failed test out of the 114 tests performed, while seven sequences encountered multiple failed tests. Notably, the last sequence, 997_1.8_0.1, recorded the highest number of failed tests. It becomes evident that when the number of state elements is large (≥ 47), a few sequences may exhibit more failed tests. This observation is supported by Table 10, which showcases the relationship between the number of failed tests and the number of state elements. Notably, when the state had 3 or 5 elements, no failed tests occurred, whereas the highest number of failed tests was observed for sequences with 997 state elements. Another noteworthy observation from Table 9 is that `rgb_lagged_sum` is the test that had the highest number of failures. This outcome is not surprising, as each sequence is subjected to `rgb_lagged_sum` testing 33 times (as indicated in Table 7), with each test utilizing a different NTuple value. It is important to note that the sequences that failed did not fail in all 33 tests; rather, only a few tests within that set resulted in failure.

Table 9. Sequence vs. failed tests.

Sequence (State_μ_Seed)	No of Failed Tests	Failed Tests Name/s	Number of Unique Tests Failed
11_0.6_0.01	1	<code>rgb_lagge_sum</code>	1
19_1.6_0.37	1	<code>sts_serial</code>	1
2_1.6_0.46	1	<code>sts_serial</code>	1
47_1.0_0.64	1	<code>sts_serial</code>	1
997_1.4_0.46	13	<code>rgb_lagged_sum</code> , <code>rgb_bitdist</code> , <code>dab_dct</code> , <code>dab_filltree</code> , <code>dab_filltree2</code> , <code>dab_monobit2</code>	6
97_1.6_0.01	37	<code>rgb_lagged_sum</code> , <code>diehard_squeeze</code> , <code>diehard_sums</code> , <code>diehard_runs</code> , <code>diehard_craps</code> , <code>rgb_bitdist</code>	6
997_1.0_0.73	39	<code>rgb_lagged_sum</code> , <code>diehard_squeeze</code> , <code>diehard_sums</code> , <code>diehard_runs</code> , <code>diehard_craps</code> , <code>rgb_bitdist</code>	6
47_0.4_0.64	41	<code>diehard_dna</code> , <code>diehard_count_1s_str</code> , <code>diehard_count_1s_byt</code> , <code>diehard_parking_lot</code> , <code>rgb_bitdist</code> , <code>rgb_lagged_sum</code> , <code>dab_monobit2</code>	7
997_0.6_0.28	48	<code>rgb_lagged_sum</code> , <code>diehard_dna</code> , <code>diehard_count_1s_str</code> , <code>diehard_count_1s_byt</code> , <code>diehard_parking_lot</code> , <code>rgb_bitdist</code> , <code>dab_filltree</code> , <code>dab_filltree2</code> , <code>dab_monobit2</code>	9

Table 9. Cont.

Sequence (State_μ_Seed)	No of Failed Tests	Failed Tests Name/s	Number of Unique Tests Failed
997_1.4_0.82	53	rgb_lagged_sum, diehard_opso, diehard_oqso, diehard_dna, diehard_count_1s_str, diehard_count_1s_byt, diehard_parking_lot, rgb_bitdist, dab_bytedistrib, dab_dct, dab_filltree, dab_filltree2, dab_monobit2	13

Table 10. State elements vs. failed tests.

State Elements	Failed Tests
2	1
3	0
5	0
11	1
19	1
47	42
97	37
997	221

Providing an exhaustive list of all the sequences with weak tests proves challenging due to their large number (739). It is essential to note that adjusting the default parameters of Dieharder would result in reporting all weak results as passed. However, it is important to emphasize that the default parameters were not modified in this case. Furthermore, generating longer sequences has the potential to reduce the number of weak tests, as observed in numerous individual sequences. However, this approach would significantly increase the testing time and necessitate computers with greater resources.

Analyzing the 739 sequences that yielded a total of 2067 weak test results, the average number of weak tests per sequence is 2.8. The minimum number of weak tests recorded is 1, while the maximum is 17. Table 11 outlines the distribution of weak tests across the sequences, demonstrating that the majority of sequences have only 1, 2, 3, or 4 weak tests.

Table 11. Weak tests vs. no. of sequences.

No. Weak Tests	No. Sequences
1	143
2	232
3	166
4	110
5	46
6	25
7	7
8	6
9	2
10	1
17	1

However, there is one sequence with 10 weak tests and another with 17 weak tests. To see if there is a correlation between the number of state elements and the number of weak tests, Table 12 provides insight. The table shows that the weak tests encountered by sequences generated by different sizes of the state are very close. However, in agreement

with the conclusions above, when the state is three and five the least number of weak tests is noticed.

Table 12. Results of new states.

State	No. of Weak Tests
2	271
3	241
5	253
11	257
19	264
47	260
97	255
997	266
total	2067

To ensure the robustness of the SBTM and address any concerns regarding sequences with high failed tests, a thorough investigation was conducted. Firstly, the seven sequences with high failed tests were identified from Table 9. In order to ascertain if these failures were mere coincidences rather than a systematic issue, 28 similar sequences were generated, as presented in Table 13. Each newly generated sequence featured a slight modification in either the control variable or the seed. For instance, if the original value of the control variable was 1.4, the newly generated sequences would use values such as 1.4000000001 and 1.3999999999 (i.e., the original value plus and minus one billionth), and so forth.

The results displayed in Table 13 conclusively demonstrate that the failures observed in the original sequences were indeed coincidental and not indicative of a systematic flaw in the algorithm. Among the 28 newly generated sequences, none exhibited any failed tests, and the number of weak tests remained like that of the original 739 sequences. This further reinforces the confidence in the algorithm’s performance and reliability.

Table 13. Results of new sequences.

Sequences (State_μ_Seed)	Failed (Original)	Seq with New Micro	Failed	Weak	Seq New Seed	Failed	Weak
997_1.4_0.46	13	$1.4 + 1^9$	0	3	$0.46 + 1^9$	0	3
97_1.6_0.01	37	$1.4 - 1^{-9}$	0	2	$0.46 - 1^{-9}$	0	2
		$1.6 + 1^9$	0	2	$0.01 + 1^9$	0	2
997_1.0_0.73	39	$1.6 - 1^{-9}$	0	1	$0.01 - 1^{-9}$	0	2
		$1 + 1^9$	0	3	$0.73 + 1^9$	0	2
47_0.4_0.64	41	$1 - 1^{-9}$	0	2	$0.73 - 1^{-9}$	0	2
		$0.4 + 1^9$	0	2	$0.64 + 1^9$	0	6
997_0.6_0.28	48	$0.4 - 1^{-9}$	0	2	$0.64 - 1^{-9}$	0	6
		$0.6 + 1^9$	0	2	$0.28 + 1^9$	0	3
997_1.4_0.82	53	$0.6 - 1^{-9}$	0	3	$0.28 - 1^{-9}$	0	4
		$1.4 + 1^9$	0	2	$0.82 + 1^9$	0	3
997_1.8_0.1	68	$1.4 - 1^{-9}$	0	4	$0.82 - 1^{-9}$	0	1
		$1.8 + 1^9$	0	4	$0.1 + 1^9$	0	4
		$1.8 - 1^{-9}$	0	3	$0.1 - 1^{-9}$	0	3

6. Contributions to Sensor and Actuator Networks

Tent-Map RNGs, derived from the mathematical Tent-Map function, have emerged as influential tools in the realm of sensor and actuator networks (SANs). These RNGs exhibit chaotic behavior and are known for their pivotal role in enhancing the capabilities of SANs. In this section, we will explore the significant contributions and implications of integrating Tent-Map RNGs into SANs.

1. **Enhancing Network Security:**
Security is a paramount concern within SANs, particularly in applications such as military networks, healthcare systems, and critical infrastructure monitoring. Tent-Map RNGs make substantial contributions to SAN security by furnishing a robust source of randomness. They excel in resisting statistical tests, thus ensuring that the generated random numbers remain unpredictable and resistant to potential attacks. This, in turn, bolsters the confidentiality and integrity of data transmission throughout the network.
2. **Energy-Efficient Operations:**
Energy efficiency stands as a pivotal aspect of SANs, especially when dealing with battery-powered sensor nodes. Tent-Map RNGs offer a distinct advantage in this regard as they demand minimal computational resources for random number generation. This low computational overhead makes them particularly well-suited for resource-constrained SAN devices, ultimately extending battery life and sustaining the network's operational longevity.
3. **Facilitating Distributed Coordination:**
SANs frequently comprise a multitude of sensor and actuator nodes necessitating seamless coordination. Tent-Map RNGs empower these nodes to generate random values, thereby aiding tasks such as time synchronization, the selection of routing protocols, and data aggregation. The introduction of randomness plays a crucial role in averting synchronization challenges and enhancing overall network performance.
4. **Enhancing Fault Tolerance:**
SANs operate in diverse environments where they may encounter factors like environmental fluctuations, hardware failures, or signal interference. Tent-Map RNGs contribute to fault tolerance by injecting randomness into decision-making processes. This randomness equips the network to adapt and recover swiftly from unforeseen events, ensuring continued functionality even in adverse conditions.
5. **Preserving Privacy in Data Aggregation:**
Data aggregation is a fundamental operation in SANs, where sensor nodes gather and transmit data to a central node or base station. Tent-Map RNGs can be employed to introduce controlled noise into the collected data. This preserves data privacy while still permitting meaningful aggregation at the central node. This approach safeguards sensitive information without compromising the network's overall efficiency.

Random number generators based on the Tent-Map algorithm have made substantial contributions to the realm of sensor and actuator networks. Their unique attributes, including unpredictability, energy efficiency, and fault tolerance, position them as invaluable tools for enhancing the security and functionality of SANs. As SANs continue to evolve and play pivotal roles in various applications, the integration of Tent-Map RNGs is anticipated to remain a prominent and influential aspect of their design and operation.

7. Conclusions and Future Work

This paper proposed and investigated the performance of the State-Based Tent-Map as a random number generator. The SBTM was developed to address the shortcomings of the original Tent-Map in generating random numbers, including issues related to speed, randomness, lack of statistical properties, and lack of uniformity. The sequences generated by the SBTM underwent comprehensive testing using the Dieharder test suite, which rigorously evaluated their statistical properties and adherence to randomness. The results revealed that a significant majority of the tests, approximately 97.4%, were successfully passed, indicating the generator's ability to produce sequences with sufficient randomness. While a small portion of the tests exhibited failures and weaknesses, these findings provide valuable insights for further improving the SBTM's performance. Overall, this paper highlights the strengths and potential areas of enhancement for the SBTM, contributing to the understanding and development of reliable random number generation techniques. Application-specific customizations are a potential avenue for the SBTM. Researchers can

explore adapting and tailoring the SBTM algorithm to address the unique requirements of diverse application domains. Future studies can also investigate the modification or extension of the algorithm to align with specific needs in fields like Monte Carlo simulations, gaming, cryptography, or statistical modeling. By fine-tuning the SBTM to suit these applications, its performance, efficiency, and suitability can be optimized.

Author Contributions: Conceptualization, A.A.-D. and Y.S.; methodology, S.A.-E., S.F., M.B.T. and M.A.-M.; formal analysis, A.A.-D. and Y.S.; investigation, S.A.-E., S.F., M.B.T. and M.A.-M.; resources, S.A.-E., S.F., M.B.T. and M.A.-M.; writing—original draft preparation, A.A.-D. and Y.S.; writing—review and editing, M.B.T. and M.A.-M.; supervision, A.A.-D. and Y.S.; funding acquisition, S.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All the experiment results data in this manuscript are available on: https://drive.google.com/drive/folders/1LRdHQnxpLwI5UGFLFQTbMXfHH7KiL7wS?usp=drive_link.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Rathore, M.S.; Poongodi, M.; Saurabh, P.; Lilhore, U.K.; Bourouis, S.; Alhakami, W.; Hamdi, M. A novel trust-based security and privacy model for internet of vehicles using encryption and steganography. *Comput. Electr. Eng.* **2022**, *102*, 108205. [CrossRef]
- Gupta, R.K.; Almuzaini, K.K.; Pateriya, R.K.; Shah, K.; Shukla, P.K.; Akwafo, R. An improved secure key generation using enhanced identity-based encryption for cloud computing in large-scale 5G. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 7291250. [CrossRef]
- Nisha, C.; Monoth, T. Analysis of spatial domain image steganography based on pixel-value differencing method. In *Soft Computing for Problem Solving: SocProS 2018*; Springer: Singapore, 2020; Volume 2, pp. 385–397.
- Karampidis, K.; Kavallieratou, E.; Papadourakis, G. A review of image steganalysis techniques for digital forensics. *J. Inf. Secur. Appl.* **2018**, *40*, 217–235. [CrossRef]
- Hosny, K.M.; Zaki, M.A.; Lashin, N.A.; Hamza, H.M. Fast colored video encryption using block scrambling and multi-key generation. *Vis. Comput.* **2022**, 1–32. [CrossRef]
- Singh, L.; Singh, A.K.; Singh, P.K. Secure data hiding techniques: A survey. *Multimed. Tools Appl.* **2020**, *79*, 15901–15921. [CrossRef]
- Kaur, S.; Singh, S.; Kaur, M.; Lee, H.N. A systematic review of computational image steganography approaches. *Arch. Comput. Methods Eng.* **2022**, *29*, 4775–4797. [CrossRef]
- Sharma, M.; Ranjan, R.K.; Bharti, V. A pseudo-random bit generator based on chaotic maps enhanced with a bit-XOR operation. *J. Inf. Secur. Appl.* **2022**, *69*, 103299. [CrossRef]
- Wang, Y.; Gong, J.; Wang, M.; Jiang, G. A pseudo-random number generator for integer chaotic map. *J. Beijing Univ. Posts Telecommun.* **2022**, *45*, 58.
- Rustad, S.; Andono, P.N.; Shidik, G.F. Digital image steganography survey and investigation (goal, assessment, method, development, and dataset). *Signal Process.* **2022**, *206*, 108908.
- Fridrich, J. *Steganography in Digital Media: Principles, Algorithms, and Applications*; Cambridge University Press: Cambridge, UK, 2009.
- Bhavani, Y.; Kamakshi, P.; Kavya Sri, E.; Sindhu Sai, Y. A survey on image steganography techniques using least significant bit. In *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2021*; Springer Nature: Singapore, 2022; pp. 281–290.
- Rahman, S.; Uddin, J.; Khan, H.U.; Hussain, H.; Khan, A.A.; Zakarya, M. A novel steganography technique for digital images using the least significant bit substitution method. *IEEE Access* **2022**, *10*, 124053–124075. [CrossRef]
- Eaton, M.; Hossameldin, A.; Birrittella, R.J.; Alsing, P.M.; Gerry, C.C.; Dong, H.; Pfister, O. Resolution of 100 photons and quantum generation of unbiased random numbers. *Nat. Photonics* **2023**, *17*, 106–111. [CrossRef]
- Xu, B.; Chen, Z.; Li, Z.; Yang, J.; Su, Q.; Huang, W.; Guo, H. High speed continuous variable source-independent quantum random number generation. *Quantum Sci. Technol.* **2019**, *4*, 025013. [CrossRef]
- Ding, J.; Chen, K.; Wang, Y.; Zhao, N.; Zhang, W.; Yu, N. Discop: Provably Secure Steganography in Practice Based on “Distribution Copies”. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; pp. 2238–2255.
- Daoui, A.; Yamni, M.; Chelloug, S.A.; Wani, M.A.; El-Latif, A.A.A. Efficient image encryption scheme using novel 1D multiparametric dynamical tent map and parallel computing. *Mathematics* **2023**, *11*, 1589. [CrossRef]
- Khalil, N.; Sarhan, A.; Alshewimy, M.A. An efficient color/grayscale image encryption scheme based on hybrid chaotic maps. *Opt. Laser Technol.* **2021**, *143*, 107326. [CrossRef]

19. Hazell, P.; Mather, P.; Longstaff, A.; Fletcher, S. Digital System Performance Enhancement of a Tent Map-Based ADC for Monitoring Photovoltaic Systems. *Electronics* **2020**, *9*, 1554. [[CrossRef](#)]
20. Kanwal, S.; Inam, S.; Othman, M.T.B.; Waqar, A.; Ibrahim, M.; Nawaz, F.; Hamam, H. An effective color image encryption based on Henon map, tent chaotic map, and orthogonal matrices. *Sensors* **2020**, *22*, 4359. [[CrossRef](#)]
21. Zheng, J.; Hu, H. A highly secure stream cipher based on analog-digital hybrid chaotic system. *Inf. Sci.* **2022**, *587*, 226–246. [[CrossRef](#)]
22. Maolood, A.T.; Gbashi, E.K.; Mahmood, E.S. Novel lightweight video encryption method based on ChaCha20 stream cipher and hybrid chaotic map. *Int. J. Electr. Comput. Eng.* **2022**, *12*, 4988–5000. [[CrossRef](#)]
23. Alawida, M.; Teh, J.S.; Mehmood, A.; Shoufan, A. A chaos-based block cipher based on an enhanced logistic map and simultaneous confusion-diffusion operations. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 8136–8151. [[CrossRef](#)]
24. Amdouni, R.; Gafsi, M.; Guessmi, R.; Hajjaji, M.A.; Mtibaa, A.; Bourennane, E.B. High-performance hardware architecture of a robust encryption block-cipher algorithm based on different chaotic maps and DNA sequence encoding. *Integration* **2022**, *87*, 346–363. [[CrossRef](#)]
25. El-Meligy, N.E.; Diab, T.O.; Mohra, A.S.; Hassan, A.Y.; El-Sobky, W.I. A novel dynamic mathematical model applied in hash function based on DNA algorithm and chaotic maps. *Mathematics* **2022**, *10*, 1333. [[CrossRef](#)]
26. Zellagui, A.; Hadj-Said, N.; Ali-Pacha, A. A new hash function inspired by sponge construction using chaotic maps. *J. Discret. Math. Sci. Cryptogr.* **2022**, 1–31. [[CrossRef](#)]
27. Liu, J.; Wang, Y.; Han, Q.; Gao, J. A sensitive image encryption algorithm based on a higher-dimensional chaotic map and steganography. *Int. J. Bifurc. Chaos* **2022**, *32*, 2250004. [[CrossRef](#)]
28. Bhandari, M.; Panday, S.; Bhatta, C.P.; Panday, S.P. Image steganography approach based ant colony optimization with triangular chaotic map. In Proceedings of the 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM), Gautam Buddha Nagar, India, 23–25 February 2022; Volume 2, pp. 429–434.
29. Wang, K.; Gao, T.; You, D.; Wu, X.; Kan, H. A secure dual-color image watermarking scheme based 2D DWT, SVD and Chaotic map. *Multimed. Tools Appl.* **2022**, *81*, 6159–6190. [[CrossRef](#)]
30. Hosny, K.M.; Darwish, M.M. Robust color image watermarking using multiple fractional-order moments and chaotic map. *Multimed. Tools Appl.* **2022**, *81*, 24347–24375. [[CrossRef](#)]
31. Murillo-Escobar, M.; Cruz-Hernández, C.; Cardoza-Avenidaño, L.; Méndez-Ramírez, R. A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. *Nonlinear Dyn.* **2017**, *87*, 407–425. [[CrossRef](#)]
32. Wang, L.; Cheng, H. Pseudo-random number generator based on logistic chaotic system. *Entropy* **2019**, *21*, 960. [[CrossRef](#)]
33. Hemdan, A.M.; Faragallah, O.S.; Elshakankiry, O.; Elmhalaway, A. A fast hybrid image cryptosystem based on random generator and modified logistic map. *Multimed. Tools Appl.* **2019**, *78*, 16177–16193. [[CrossRef](#)]
34. Chen, S.L.; Hwang, T.; Lin, W.W. Randomness enhancement using digitalized modified logistic map. *IEEE Trans. Circuits Syst. II Express Briefs* **2010**, *57*, 996–1000.
35. Liu, J.; Liang, Z.; Luo, Y.; Cao, L.; Zhang, S.; Wang, Y.; Yang, S. A hardware pseudo-random number generator using stochastic computing and logistic map. *Micromachines* **2020**, *12*, 31. [[CrossRef](#)]
36. García-Martínez, M.; Campos-Cantón, E. Pseudo-random bit generator based on multi-modal maps. *Nonlinear Dyn.* **2015**, *82*, 2119–2131. [[CrossRef](#)]
37. García-Martínez, M.; Ontañón-García, L.; Campos-Cantón, E.; Čelikovský, S. Hyperchaotic encryption based on multi-scroll piecewise linear systems. *Appl. Math. Comput.* **2015**, *270*, 413–424. [[CrossRef](#)]
38. Stoyanov, B.; Kordov, K. Novel secure pseudo-random number generation scheme based on two tinkerbelle maps. *Adv. Stud. Theor. Phys.* **2015**, *9*, 411–421. [[CrossRef](#)]
39. Tutueva, A.; Pesterev, D.; Karimov, A.; Butusov, D.; Ostrovskii, V. Adaptive Chirikov map for pseudo-random number generation in chaos-based stream encryption. In Proceedings of the 2019 25th Conference of Open Innovations Association (FRUCT), Helsinki, Finland, 5–8 November 2019; pp. 333–338.
40. Cardoso, M.B.; da Silva, S.S.; Nardo, L.G.; Passos, R.M.; Nepomuceno, E.G.; Arias-Garcia, J. A new PRNG hardware architecture based on an exponential chaotic map. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–5.
41. Yu, F.; Li, L.; He, B.; Liu, L.; Qian, S.; Huang, Y.; Cai, S.; Song, Y.; Tang, Q.; Wan, Q.; et al. Design and FPGA implementation of a pseudorandom number generator based on a four-wing memristive hyperchaotic system and Bernoulli map. *IEEE Access* **2019**, *7*, 181884–181898. [[CrossRef](#)]
42. Rezk, A.A.; Madian, A.H.; Radwan, A.G.; Soliman, A.M. Reconfigurable chaotic pseudo random number generator based on FPGA. *AEU-Int. J. Electron. Commun.* **2019**, *98*, 174–180. [[CrossRef](#)]
43. Garcia-Bosque, M.; Pérez-Resca, A.; Sánchez-Azqueta, C.; Aldea, C.; Celma, S. Chaos-based bitwise dynamical pseudorandom number generator on FPGA. *IEEE Trans. Instrum. Meas.* **2018**, *68*, 291–293. [[CrossRef](#)]
44. Hobincu, R.; Datcu, O. FPGA implementation of a chaos based PRNG targeting secret communication. In Proceedings of the 2018 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 8–9 November 2018; pp. 1–4.
45. Kaçar, S. Analog circuit and microcontroller based RNG application of a new easy realizable 4D chaotic system. *Optik* **2016**, *127*, 9551–9561. [[CrossRef](#)]

46. Volos, C.K. Chaotic random bit generator realized with a microcontroller. *J. Comput. Model.* **2013**, *3*, 115–136.
47. Bao, H.; Hua, Z.; Wang, N.; Zhu, L.; Chen, M.; Bao, B. Initials-boosted coexisting chaos in a 2-D sine map and its hardware implementation. *IEEE Trans. Ind. Inform.* **2020**, *17*, 1132–1140. [[CrossRef](#)]
48. Nesa, N.; Ghosh, T.; Banerjee, I. Design of a chaos-based encryption scheme for sensor data using a novel logarithmic chaotic map. *J. Inf. Secur. Appl.* **2019**, *47*, 320–328. [[CrossRef](#)]
49. Liu, Z.; Wang, Y.; Zhao, Y.; Zhang, L.Y. A stream cipher algorithm based on 2D coupled map lattice and partitioned cellular automata. *Nonlinear Dyn.* **2020**, *101*, 1383–1396. [[CrossRef](#)]
50. Wang, X.; Bao, X. A novel block cryptosystem based on the coupled chaotic map lattice. *Nonlinear Dyn.* **2013**, *72*, 707–715. [[CrossRef](#)]
51. Peng, Z.; Yu, W.; Wang, J.; Zhou, Z.; Chen, J.; Zhong, G. Secure communication based on microcontroller unit with a novel five-dimensional hyperchaotic system. *Arab. J. Sci. Eng.* **2021**, *47*, 813–828. [[CrossRef](#)]
52. Som, S.; Dutta, S.; Singha, R.; Kotal, A.; Palit, S. Confusion and diffusion of color images with multiple chaotic maps and chaos-based pseudorandom binary number generator. *Nonlinear Dyn.* **2015**, *80*, 615–627. [[CrossRef](#)]
53. Xu, H.; Tong, X.; Meng, X. An efficient chaos pseudo-random number generator applied to video encryption. *Optik* **2016**, *127*, 9305–9319. [[CrossRef](#)]
54. Yeniçeri, R.; Kilinç, S.; Yalçın, M.E. Attack on a chaos-based random number generator using anticipating synchronization. *Int. J. Bifurc. Chaos* **2015**, *25*, 1550021. [[CrossRef](#)]
55. Lambić, D.; Janković, A.; Ahmad, M. Security analysis of the efficient chaos pseudo-random number generator applied to video encryption. *J. Electron. Test.* **2018**, *34*, 709–715. [[CrossRef](#)]
56. Ergün, S. Cryptanalysis and improvement of a chaos based random number generator. In Proceedings of the 2016 International Symposium on Electronics and Smart Devices (ISESD), Bandung, Indonesia, 29–30 November 2016; pp. 199–202.
57. Luo, Y.; Zhang, D.; Liu, J.; Liu, Y.; Cao, Y.; Ding, X. Cryptanalysis of chaos-based cryptosystem from the hardware perspective. *Int. J. Bifurc. Chaos* **2018**, *28*, 1850114. [[CrossRef](#)]
58. Zia, U.; McCartney, M.; Scotney, B.; Martinez, J.; Sajjad, A. A novel pseudo-random number generator for IoT based on a coupled map lattice system using the generalised symmetric map. *SN Appl. Sci.* **2022**, *4*, 48. [[CrossRef](#)]
59. Kanso, A. Self-shrinking chaotic stream ciphers. *Commun. Nonlinear Sci. Numer. Simul.* **2011**, *16*, 822–836. [[CrossRef](#)]
60. Matthews, R. On the derivation of a “chaotic” encryption algorithm. *Cryptologia* **1989**, *13*, 29–42. [[CrossRef](#)]
61. Pecora, L.M.; Carroll, T.L. Synchronization in chaotic systems. *Phys. Rev. Lett.* **1990**, *64*, 821. [[CrossRef](#)] [[PubMed](#)]
62. Beck, C.; Schögl, F. *Thermodynamics of Chaotic Systems*; Cambridge University Press: Cambridge, UK, 1995.
63. Ott, E. *Chaos in Dynamical Systems*; Cambridge University Press: Cambridge, UK, 2002.
64. Schuster, H.G.; Just, W. *Deterministic Chaos: An Introduction*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
65. Zhang, Z. A Multi-Threaded Cryptographic Pseudorandom Number Generator Test Suite. Ph.D. Thesis, Naval Postgraduate School, Monterey, CA, USA, 2016.
66. Habutsu, T.; Nishio, Y.; Sasase, I.; Mori, S. A secret key cryptosystem by iterating a chaotic map. In *Advances in Cryptology—EUROCRYPT’91: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, 8–11 April 1991*; Proceedings 10; Springer: Berlin/Heidelberg, Germany, 1991; pp. 127–140.
67. Alvarez, E.; Fernández, A.; Garcia, P.; Jiménez, J.; Marcano, A. New approach to chaotic encryption. *Phys. Lett. A* **1999**, *263*, 373–375. [[CrossRef](#)]
68. Beham, E. Cryptanalysis of the Chaotic-map Cryptosystem. In *Advances in Cryptology—EUROCRYPT’91: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, 8–11 April 1991*; Springer: Berlin/Heidelberg, Germany, 1991.
69. Alvarez, G.; Montoya, F.; Romera, M.; Pastor, G. Cryptanalysis of a chaotic encryption system. *Phys. Lett. A* **2000**, *276*, 191–196. [[CrossRef](#)]
70. Barker, E.; Kelsey, J. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)*; US Department of Commerce, Technology Administration, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007.
71. Wolf, A.; Swift, J.B.; Swinney, H.L.; Vastano, J.A. Determining Lyapunov exponents from a time series. *Phys. D Nonlinear Phenom.* **1985**, *16*, 285–317. [[CrossRef](#)]
72. Brown, R.; Eddelbuettel, D.; Bauer, D.D. *Dieharder: A Random Number Test Suite*; Duke University Physics Department Durham: Durham, NC, USA, 2018.
73. Patidar, V.; Sud, K. A novel pseudo random bit generator based on chaotic standard map and its testing. *Electron. J. Theor. Phys.* **2009**, *6*, 327–344.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.