Received: 25 July 2022 | Revised: 29 November 2022 | Accepted: 8 February 2023

DOI: 10.1002/spy2.304

RESEARCH ARTICLE

WILEY

A resource efficient pseudo random number generator based on sawtooth maps for Internet of Things

Unsub Zia[1] | **Mark McCartney**[1] | **Bryan Scotney**[1] | **Jorge Martinez**[1] | **Ali Sajjad**[2]

[1]School of Computing, Ulster University, Belfast, UK

[2]Applied Research, British Telecom, Ipswich, UK

Correspondence
Unsub Zia, Ulster University, Belfast, UK.
Email: zia-smu@ulster.ac.uk

Funding information
British Telecommunications; Invest Northern Ireland

Abstract

The strength of cryptographic keys rely on the random number generators (RNGs) to produce random seed values. Unfortunately there are not many RNGs options suitable for Internet of Things (IoTs) scenario, due to limited processing resources and bulk quantity of IoT data that needs to be secured. In this article, we studied sawtooth map which is a chaotic map. However, when implemented on a computer, the sawtooth map results on a non-chaotic orbit due to the finite precision of computation. This can be avoided if we use the sawtooth map as the local map in a coupled map lattice (CML) system. We explore such coupled map systems for randomness through entropy and statistical analysis. Based on the results, we propose a lightweight hybrid pseudo random number generator (PRNG) based on sawtooth based CML system and SPONGENT hashing. The proposed PRNG is thoroughly tested against statistical attacks, entropy analysis, key space analysis and compared with existing state of the art solutions. The results provide evidence that the proposed PRNG produces random numbers that could produce sufficiently strong cryptographic keys for resource constrained IoT devices.

KEYWORDS

coupled map lattice, cryptography, IoT, pseudo random number generator, sawtooth map

1 | INTRODUCTION

Digital data is considered an asset and the evolution of the internet into the Internet of Things (IoT) is giving birth to new types of security issues.[1] The deployment of security primitives relies on random seed values that enhance the robustness of a security algorithm towards attacks.[2] These random seed values can be obtained via two methods, namely true random number generators (TRNGs) or pseudo random number generators (PRNGs). TRNGs are the true random sources of entropy that exist naturally and portray unpredictable behavior.[3] True random sources exist in nature in abundance ranging from atmospheric noise to radioactive decay of elements, but it is hard to capture the required length of naturally occurring random events into cryptographic applications for integrated circuits.[4] True random sequences can also be retrieved form electronic circuitry for instance, ring oscillator based TRNGs,[5] electronic noise based TRNGs,[6] Field Programmable Gate Array (FPGA) based TRNGs,[7] and random-access memory based TRNGs.[8] Despite the benefits electronic circuit based TRNGs provide, they are highly susceptible to side channel attacks because of the intrinsic characteristics of

electronic circuits, for instance voltage drop and power consumption. There is significant evidence in the literature about successful attacks on electronic circuit based TRNGs.

The most common type of attacks include electromagnetic injection attacks,[9] in which the adversary utilizes antenna-based equipment, such as ring oscillators and takes control of the output signals by injecting known electromagnetic frequencies, thus changing true randomness of the system into deterministic patterns.[10] In conventional side channel attacks, the adversary has to study the system in depth and try probing physical characteristics of the circuit manually. A major breakthrough is the utilization of machine learning in cryptanalysis of TRNGs, during which the attacker uses deep learning algorithms to analyze the circuit behavior and can infiltrate with relative ease.[11] The above discussed shortcomings of TRNGs make them unsuitable for cryptographic applications where electronic circuit based random number generation is not an option. For instance, sensor devices used in IoT frameworks are highly resource constrained in nature and are usually deployed in remote conditions where an attacker can easily gain access to the physical device and perform side channel analysis on the electronic circuit. A substitute to TRNGs in such scenarios are PRNGs. PRNGs are mathematical functions or algorithms that are capable of generating long sequences that appear to be random.[12] By the term "appear random" we mean that they are deterministic because they are generated mathematically, yet they are indistinguishable to statistical tests when compared with a true random stream of numbers. In terms of cryptographic applications, the pseudo random sequences are hard to predict without the knowledge of the input (seed) values. Whilst designing a PRNG, the following are the expectations that should be met[13]:

- **Pass empirical tests:** PRNG must be able to pass any type of statistical tests, where long sequences of random numbers are tested for a uniformly distributed samples.
- **Mathematical basis:** Should possess a strong mathematical formulation, that has the capability to generate independent and unique random patterns.
- **Resource efficient:** The PRNG should be designed to be resource efficient. For instance, in Monte Carlo simulations, a massive quantity of random numbers are needed. This might require the RNG to run multiple times. Even in cryptographic applications like system on chip (SoC), RNGs are expected to generate fresh seed values for encryption keys multiple times in short intervals.
- **Multiple streams:** Parallel computation might be used for Monte Carlo calculations and multiple replicas of a RNG would be required. However, it is crucial to ensure that the streams from all parallel PRNGs are independent from each other.
- **Practical concerns:** PRNGs form the backbone of any cryptosystem, thus the installation and seeding process should be kept easy and straightforward. Some PRNGs are quite lightweight with few lines of code, whereas others are complicated designs with resource extensive code implementations. The design should be standard and generic to suit all major cryptographic applications.
- **Reproducibility:** The seed for a specific cryptographic key must be reproducible otherwise the cryptosystem cannot be recovered. Further, testing and debugging require the same stream of random numbers to be generated multiple times.

PRNGs have always been an active area of research and there are a variety of solutions available. Examples include counter-based RNGs,[14] 64-bit maximally equi-distributed PRNGs,[15] FPGA based Hopfield neural networks under electromagnetic radiation PRNGs,[16] and static random access memory physical unclonable functions based FPGA implementations.[17] Insecure hardware implementations such as FPGAs make the PRNG vulnerable and limit their application in cryptographic applications. For instance, a novel attack on FPGA based PRNG was performed using a Hardware Trojan Horse method which exploits the dynamic partial reconfiguration capability in modern FPGA boards.[18] In another study, the researchers performed an attack on a multi-tenant FPGA setting that caused timing constraints violations.[19] The term multi-tenant means that the FPGAs are capable of accommodating multiple implementations simultaneously. These types of attacks are hard to notice as they trigger temporary errors. The unique properties of chaos systems make them a suitable candidate for generating pseudo random sequences.

A significant portion of the literature comprises of chaos based PRNGs,[20] some examples include, PRNGs based on quantum chaotic maps,[21] PRNGs based on mixing of three chaotic maps,[22] PRNGs based on discrete-space chaotic maps,[23] and fractal based PRNGs.[24] In the past chaos based cryptosystems have not been considered for standardization due to the weaknesses and loopholes identified during cryptanalysis of chaotic algorithms. Some examples of successful cryptanalytic studies on chaos based PRNGs include cryptanalysis of a chaotic ring oscillator,[25] cryptanalysis of RNG

based on a chaotic circuit,[26] cryptanalysis of a RNG using continuous time chaos-based oscillator[26] and cryptanalysis of a non-equilibrium point chaotic oscillator based RNG.[27]

Coupled map lattices (CMLs) form a discrete-time multi-dimensional system by utilizing chaotic maps as local maps, on a network.[28] These multi-dimensional systems can generate pseudo random sequences which rely on control parameters and initial conditions of the chaotic maps. In a previous study, we proposed a novel PRNG proposed based on generalized symmetric maps and adaptive control parameters.[29] Despite, the fact that the proposed PRNG has a large key space and passed majority of security tests there were a few shortcomings that needed to be addressed: (i) the proposed algorithm passed NIST tests only for few values of control parameters, (ii) the proposed algorithm was not lightweight as the Jacobian calculation was quite resource expensive. These reasons limit the application of generalized symmetric map based PRNG, and there remains a gap for a better PRNG in terms of randomness and resource efficiency. In this article, a novel and resource effective PRNG is proposed that is based on sawtooth maps. The motivation behind the use of sawtooth maps for proposed model is discussed in the next section.

1.1 | Unique properties of sawtooth map

The sawtooth map can be represented using Equation (1) as below:

$$f(x, \lambda, h) = \lambda x mod(h) \tag{1}$$

where, λ represents the gradient of the tooth in saw-tooth map and is greater than zero ($\lambda > 0$), $xmod(h)$ gives the remainder of x after division by h. As shown in Figure 1, λ is the gradient of the sawtooth map teeth that is, the chopper lines of the map. The unique property of sawtooth map which draws our attention for the PRNG purpose is that, apart from at the points of discontinuity the gradient of the map is a constant λ

$$\frac{dx}{dy} = \lambda \tag{2}$$

Thus, if we have the map of the form:

$$x_{n+1} = f(x_n, \lambda, h) \tag{3}$$

The global Lyapunov (λ_g) for the map can be represented as:

$$\lambda_g = \ln \mid \lambda \mid \tag{4}$$

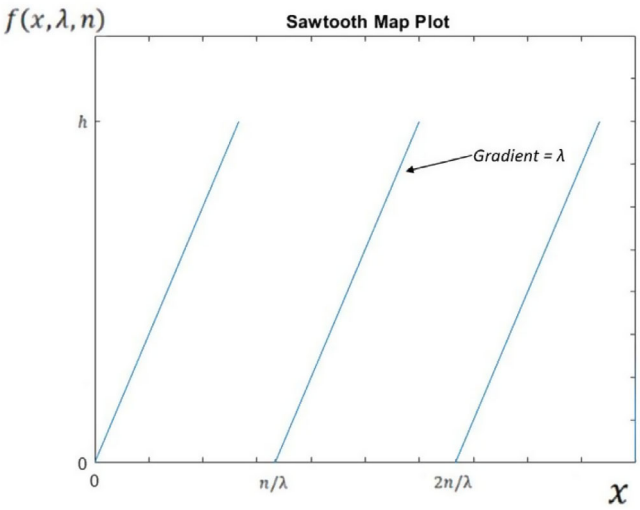

FIGURE 1 Demonstration of sawtooth map plot and its gradient λ

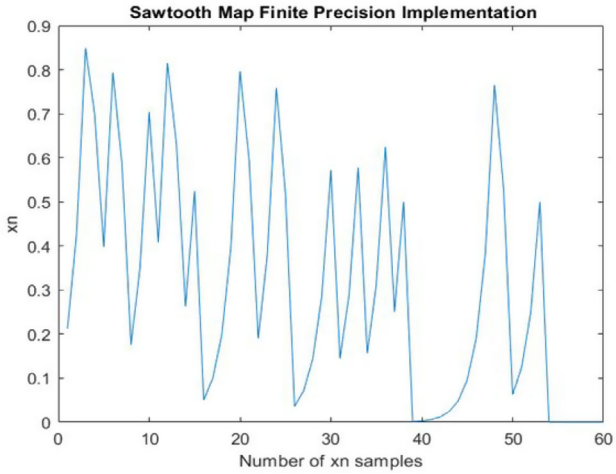

FIGURE 2 Sawtooth map moving out to non-chaotic orbit using finite precision computer implementation

Moreover, if $\lambda > 1$, then $\lambda_g > 0$, and hence the map should be chaotic for the given condition. However, due to finite precision used in computation, the map will eventually end up in a non-chaotic orbit. This behavior can be visualized in Figure 2, where the map eventually approaches a fixed point.

1.1.1 | Sawtooth map as local map in CML system

A CML system comprises of local maps linked together on a lattice. Each local map is linked with other nodes of the lattice using a coupling factor. The simplest type of CML system that can used to understand the CML structure is the nearest neighbors CML model as shown in Equation (5).[30]

$$x_{n+1}(i) = (1-\epsilon)f\left(x_n(i)\right) + \frac{\epsilon}{2}\left[f\left(x_n(i+1)\right) + f\left(x_n(i-1)\right)\right] \tag{5}$$

The nearest neighbors CML system is a diffusive two-way CML model, where the parameter i gives the ith lattice point, which in this case is coupled with two of its neighboring nodes that is, right node $(i+1)$ and left node $(i-1)$. i varies between $[1,L]$ where L is the total number of lattice points. The coupling factor is represented by ϵ, which is a real number in $[0, 1]$. Finally, n is the discrete time step for the CML system. Here, $f(x)$ denotes the local map, which is usually a logistic or tent map but in the proposed model, $f(x)$ is sawtooth map.

The problem of the formally chaotic map approaching a fixed point, as discussed in the previous section can be avoided if we place the sawtooth map on a CML lattice of L nodes. Figure 3 shows the chaotic output from sawtooth map based CML system, y-axis shows the amplitude fluctuations in the values x_n, whereas the x-axis shows the time scale that is, number of iterations. The experiments were generated using following control parameters; $n = 2000$ (number of iterations), $L = 2$ (lattice points), $\lambda = 2$ (gradient), $n = 1$ and $\epsilon = 0.2$ (coupling factor).

1.1.2 | Constant Lyapunov behavior of sawtooth map

The general equation for the CML system can be constructed by assuming ith node on the lattice at the nth timestep is x_n as shown below:

$$x_n^i = \sum_{j=1}^{L} a_{ij} f\left(x_n^{(j)}, \lambda, h\right) \tag{6}$$

where, $\sum_{j=1}^{L}\left(a_{ij}\right) = 1$ and i and j are in the range $[1, L]$ since A is a square matrix. For convenience a_{ij} can be represented as $A = a_{ij}$. The matrix $\mathbf{A}$ has ith row and jth column represented by a_{ij}, the Jacobian of the system can be

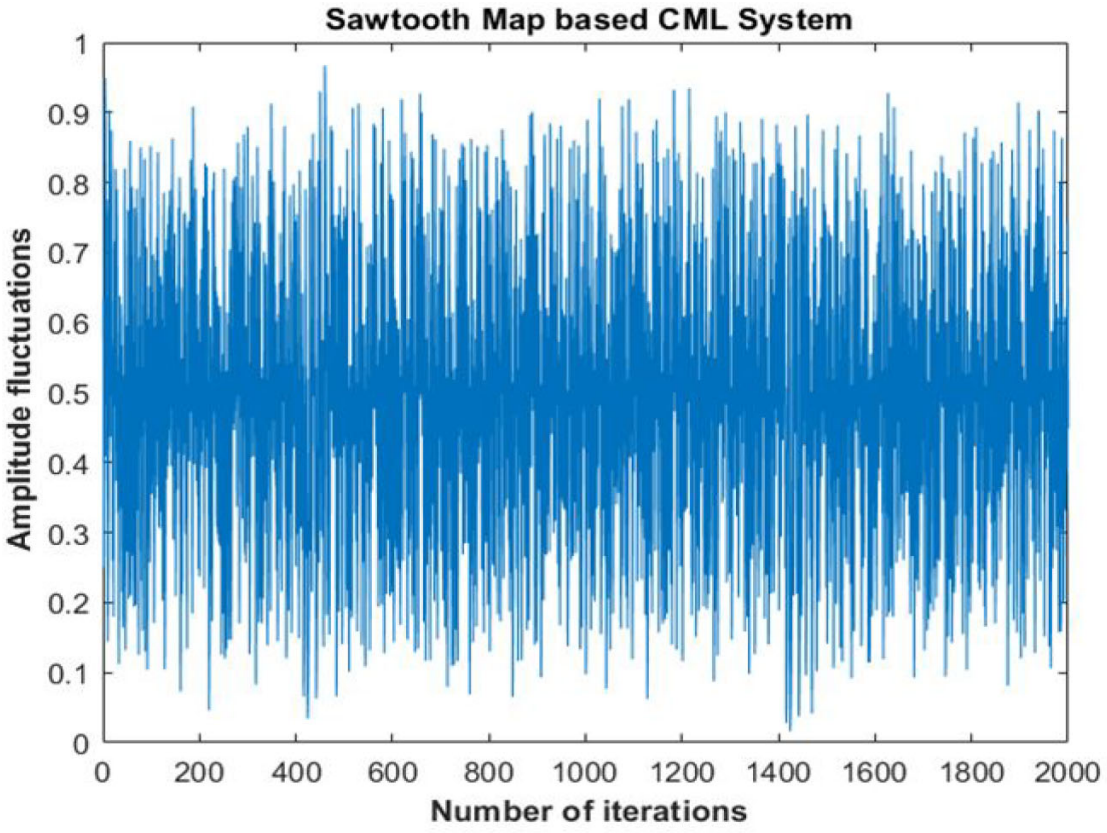

FIGURE 3 Chaotic output of CML system with sawtooth map as local map

calculated as follows:

$$\boldsymbol{J} = \lambda \boldsymbol{A} \tag{7}$$

Recall that λ and A are constant which result in a constant Jacobian matrix. This property of the system has multiple benefits for cryptographic applications:

1. **Fast calculation of $\boldsymbol{f}(\boldsymbol{x}, \lambda, \boldsymbol{h})$:** The system can be scaled such that $\boldsymbol{h} = \boldsymbol{1}$, (meaning $\boldsymbol{x}_n \in [\boldsymbol{0}, \boldsymbol{1}]$), therefore $\boldsymbol{f}(\boldsymbol{x}, \lambda, \boldsymbol{h})$ amounts to taking the remainder of $\lambda \boldsymbol{x}$ after division by 1.
2. **Control Lyapunov exponent:** The largest Lyapunov exponent ($ln \mid \lambda \mid$) can be chosen to be any large value of choice.
3. **Eigenvalues calculated only once:** The Jacobian ($\lambda \boldsymbol{A}$), is independent of $\boldsymbol{x}_n^i$ and so the eigenvalues need to be calculated only once allowing the KS entropy to be evaluated easily and quickly.
4. **Multi-dimensional hyper-chaotic system:** The eigenvalues of the Jacobian are $\lambda_i = \lambda \bullet \lambda_i^A$, where λ_i^A are the eigenvalues of matrix A. This means that if the smallest absolute value of an eigenvalue of the eigenvalues of A is λ_L^A, where the order of eigenvalue is assumed to be $\left(\lambda_1^A| \geq |\lambda_2^A| \geq \ldots |\lambda_L^A\right)$. Then, provided the λ is picked such that $\left(\lambda|\lambda_1^A| > 1\right)$, the system will be hyperchaotic in L dimensions. Further, one can induce hyper-chaos in as many dimensions up to L by simply increasing value of λ.

1.1.3 | Comparison with Logistic and GSM based CML

It is a crucial requirement for a PRNG to generate random output that is independent of previously generated sequences. Thus, to test the correlation in the output sequence sawtooth based CML system, we plotted the time series data generated by logistic based CML systems, GSM based CML systems and a random matrix for white noise as a reference model. In Figure 4, the columns denote the values of x_n for each lattice point respectively, whereas the rows represent

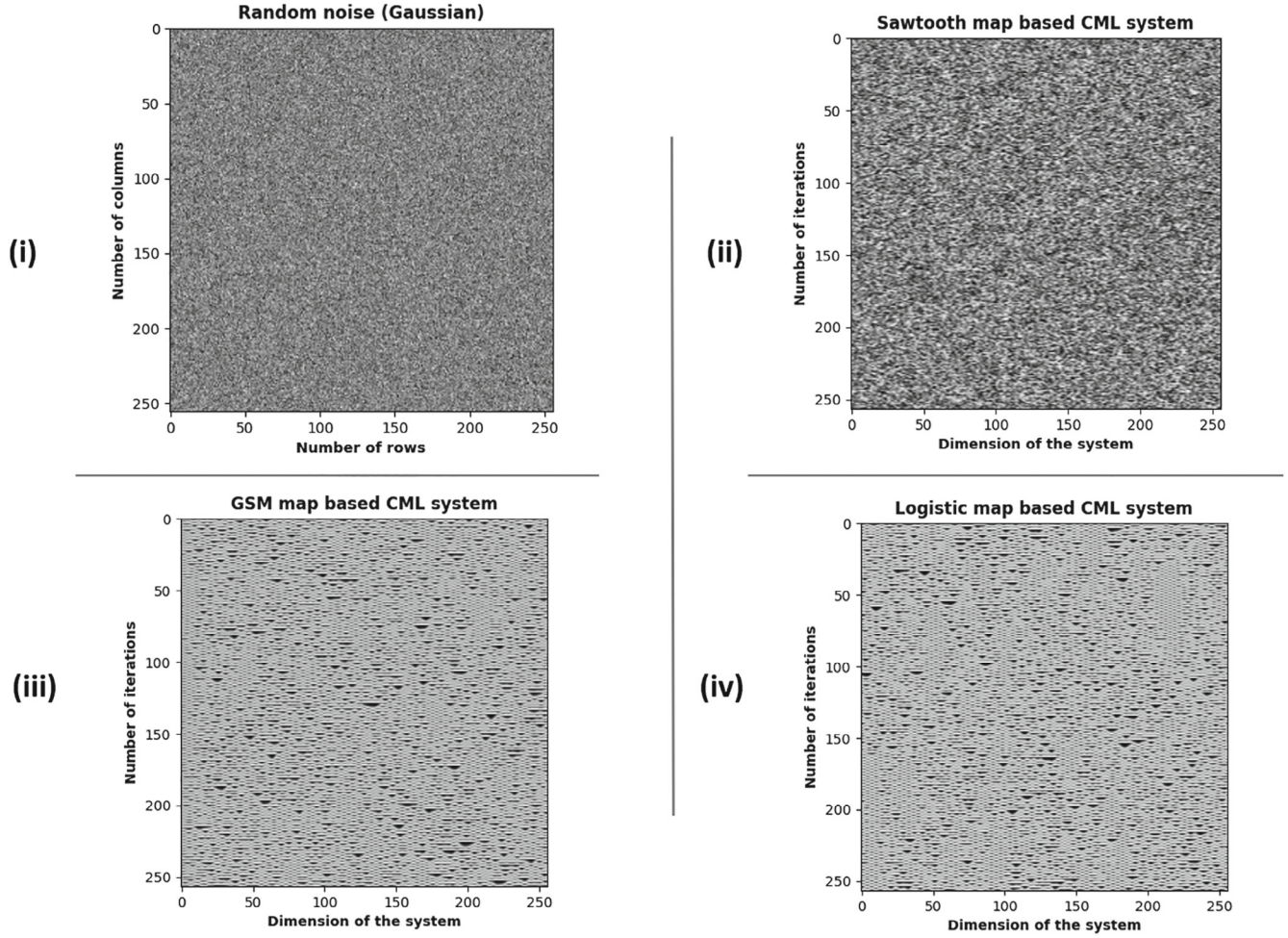

FIGURE 4 Shows 2-D plots of time series data generated using CML system with different local maps (logistic, GSM, and sawtooth) to compare with random noise

the values of x_n for every iteration. The idea behind the plotting of x_n values for every iteration and not discarding the previous iteration data was to visualize the output of the system with respect to time and look for any correlation or dependence between the iterations. The random noise matrix has been plotted as a reference model for comparison using the numpy.random.normal() function available in python programming language. This function draws random samples from a normal (Gaussian) distribution. Figure 4 (i) shows a random matrix plot of size 256×256, mean value $\mu = 0$ and standard deviation $\sigma = 0.1$ that is compared with the output of CML system with different local maps. The results for CML systems with different local maps were generated using following parameters as input to the system. Figure 4 (ii) *sawtoothmap*: 256 lattice points, 256 iterations, 1000 iterations astransient cut and values for control parameters are $\lambda = 10$, $h = 1$, and $\epsilon = 0.5$, (iii) *GSMmap*: 256 lattice points, 256 iterations, 1000 iterations as transient cut and values for control parameters are $\alpha = 1.5$, $\beta = adaptive$, $\gamma = 0.25$, $\epsilon = 0.5$ and (iv) *logisticmap*: 256 lattice points, 256 iterations, 1000 iterations as transient cut and value for control parameter are $r = 4$ and $\epsilon = 0.5$.

This can be observed from Figure 4 that the 2-D plots of sawtooth map appear show similar visual output to white noise which is considered random as all the frequencies are uniformly distributed. However, the logistic and GSM based CML systems exhibit partially random behavior, as there are few repetitive patterns that can be seen in the 2-D plots Figure 4.

This visual interpretation concludes that sawtooth based CML system values are independent from previous iteration values and are random in nature (ie, uniformly distributed). This property makes sawtooth map based CML systems a suitable candidate to be used as RNG. To further examine the credibility of sawtooth based CML systems for random number generation application, a thorough analysis was performed using statistical and entropy tests in the following sections.

1.1.4 | Main contributions of this research

This study mainly contributes in proposal of lightweight PRNG for IoTs, however it also adds value in the field of chaos theory and cryptography. Few highlights from this study are listed as below:

- **Identification of unique properties of sawtooth maps for a PRNG proposal:** We realized and identified unique properties of sawtooth maps, that could be best suited to for a PRNG proposal. (i) The global Lyapunov exponents of the sawtooth maps can be controlled manually, which is a useful property to ensure the chaotic output of the map.(ii) Unlike most other chaotic maps, the global Lyapunov exponents remain constant during the iterative solution of the map, leading to fewer computations and utilization of fewer resources. This unique property added another motivation towards using sawtooth maps for a PRNG proposal.
- **Finite precision problem of sawtooth maps:** The unique properties of the sawtooth maps have been well studied theoretically. However, when implemented using a computer, the finite precision problem leads the sawtooth maps to leave chaotic orbit thus not maintaining properties of sawtooth maps claimed theoretically. In this study, we propose the sawtooth map to be used as local map inside a CML system to resolve the issue. Moreover, a CML system comprises of lattice based structure, which connects several sawtooth map nodes together forming a complex multi-dimensional system which is harder to predict and could be suitable for security applications.
- **Serialization and randomization of the PRNG output:** The output from a sawtooth map based CML system is a stream of chaotic output sequences and is unpredictable if the input parameters for the CML system are not unknown. However, for cryptographic applications, random number are used as seed values and need to be of specified length. Thus, we proposed the use of a lightweight hashing function that would allow serialization of output data stream from CML system into fixed size bitstreams for cryptographic applications. Moreover, since hash functions are one way collision resistant functions, hashing the data stream from a sawtooth based CML system would add an additional phase of randomization, making it impossible for the adversary to predict the pattern.
- **Security testing of the proposed PRNG:** For cryptographic applications, it is crucial to test for the quality of random number sequences. Since a sawtooth based CML system is a multi-dimensional system, we tested the behavior of the system using Kolmogorov Sinai (KS) entropy. Moreover, the quality of output sequences was tested using standardized statistical and entropy test suites made available by National Institute of Standards and Technology (NIST).
- **Performance testing for resource constrained applications** The unique properties of a sawtooth based CML system makes it unique in terms of reduced resource utilization. However a thorough experimentation using has been performed to test the feasibility of proposed solution for resource constrained applications such as IoT.

2 | RELATED WORK

PRNGs play a vital role in securing systems by providing pseudo random seed values for cryptographic primitives. A variety of proposals have been presented for chaos based PRNGs in the literature. A significant number of inventions are based around logistic map-based pseudo random number generation.[31-36] The limitation in using logistic maps for random number generation is the short range of control parameter that is, [3.57,4]. However, researchers have presented several solutions to improve the parameter space for chaotic output in logistic map based PRNGs. Garcia et al conducted a study on a pseudo random bit generator, in which logistic maps have been used to generate multimodal maps to increase the parameter space for the chaotic region.[37] The logistic map is extensively studied and is the usual choice of scientists when proposing new PRNGs but other maps like the Tinkerbell map,[38] Chirikov map,[39] Lorenz map,[40] Chen map,[41] lemniscate map[42] and sawtooth map[43-45] have also been used for PRNG solutions.

As discussed in the previous section, chaos based PRNGs have not gained popularity in industrial applications after being broken several times cryptanalytically.[46] Kaneko proposed that, chaotic maps if used as lattice nodes in a CML system can generate high entropy independent sequences.[47] Several studies have been performed suggesting improvements to Kaneko's architecture of CML systems for PRNG applications. In a recent study, an improved CML based PRNG was proposed, based on the idea of mixing state variables.[48] Xiupin et al introduced a Chebyshev map based time-varying delay to CML systems for novel PRNG.[49] Several research studies are based on the construction of PRNGs using CML systems in conjunction with supporting mathematical schemes. Xing et al proposed an idea for a PRNG based on CML systems in conjunction with chaotic iteration.[50] Where chaotic iteration is a convergence theorem that could be useful for

generating chaos, proposed by Devaney et al.[51] In another study, a combination of CML system and elementary cellular automata has been used in construction of a PRNG.[52]

CMLs are classified as spatio-temporal systems, in which the dimensions of the system could be controlled by specifying the lattice count. There are yet other characteristics of CML systems that need to be explored. In generalized symmetric map paper,[29] a novel CML based PRNG has been proposed, in which the idea was to utilize any chaotic map as a local map from the symmetric chaotic map family. As chaotic maps exhibit chaotic output only for a certain range of control parameters, an adaptive scheme was introduced that automatically selected control parameters based on the selection of map and initial conditions for the map. The proposed method had a large key space and passed major attack models with high entropy values. Despite the benefits discussed, the proposed CML based PRNG with GSM had a few shortcomings. The major drawback with the proposed PRNG was not able to pass NIST randomness tests with full accuracy. Even though for some values, it did pass all 15 tests, for some of control parameters this was not the case. As noted earlier, another limitation with this type of CML based PRNG is that KS Entropy calculation requires the product of individual Jacobian matrices, which in the case of high dimensional systems could lead to large matrix multiplication problems which could be resource greedy and time consuming. Even though the concept of an adaptive parameter resolved the problem of maintaining the selected map in chaotic orbits but the Lyapunov exponents of the maps could not be controlled. As discussed in previous section, sawtooth maps possess some unique characteristics, which could be used to counteract the shortcomings with the CML based PRNG with GSM.

To our best knowledge, sawtooth map based CML systems have not been studied thoroughly as a source of random number generation for IoT. In a recent study, the authors did propose a pseudo random bit generator and claimed to use "sawtooth map as local map for CML" and also claimed it to be "efficient" solution.[53] However the logistic map was used for Lyapunov analysis instead of the sawtooth map. Also, the NIST randomness tests performed have not been evaluated to a satisfactory level, as they did not provide details about the system behavior for varying initial conditions and control parameters. Although the authors use the term "efficient," the proposed algorithm has not been implemented on any hardware (such as a microcontroller or FPGA) to validate its real efficiency. In this study, we propose a novel PRNG using sawtooth based CML system and SPONGENT hashing. The proposed sawtooth based CML system has been thoroughly studied using Lyapunov and KS entropy analysis. In later sections, the proposed PRNG is tested for NIST randomness tests on range of possible control parameters and efficiency has been fairly evaluated using real resource constrained IoT devices.

3 | BEHAVIORAL ANALYSIS OF THE SAWTOOTH BASED CML SYSTEM

CML systems are high dimensional spatio-temporal systems that behave uniquely for different types of local maps and initial conditions. Therefore, before proposing solutions utilizing the sawtooth based CML systems, it is crucial to understand what range of control parameters could be used to get the desired behavior from the system. Feigenbaum diagram analysis has already been performed in Reference 54, however to perform detailed behavioral analysis of sawtooth based system, Lyapunov analysis and Kolmogorov Sinai entropy analysis has been explained in the sections below.

3.1 | KS entropy analysis of the system

When designing RNG systems, the most crucial step is to measure the entropy of the system. Entropy could be understood as the degree of randomness or disorder in a system. It is not straightforward to calculate the entropy of dynamical systems that changes state with respect to time. Kolmogorov-Sinai (KS) entropy is considered to be a reliable measure of entropy for chaotic systems, where positive KS entropy denotes presence of chaos in the system.[55] The calculation of KS entropy relies on the Lyapunov spectra of the CML system. The Lyapunov exponents from the Lyapunov spectra can be defined in the form of an ordered set ($\lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \lambda_L$), where λ is the Lyapunov exponent and L is the dimension of the CML system. From the ordered set, only positive Lyapunov exponents are added up and divided by the dimension of the system L, the mathematical relation for calculation of KS entropy is shown below:

$$Ent_{\mu} = \frac{1}{L}\sum_{\lambda_{i>0}} \lambda_i \tag{8}$$

where, λ_i are the Lyapunov exponents calculated with respect to eigenvalues i. $\boldsymbol{L}$ is the dimension of the system and $\boldsymbol{Ent}_{\mu}$ represents the KS entropy. Lyapunov analysis is one of the most effective methods to characterize the level of stability or instability in a dynamical system. Specially in chaos-based dynamical systems, the largest Lyapunov can be referred as the decisive variable and classify the degree of chaos produced.[56] As discussed in Section 1.1.2, the Jacobian of the system can be calculated as $\boldsymbol{J} = \lambda \boldsymbol{A}$, where λ and matrix $\boldsymbol{A}$ are constant and the eigenvalues are independent of $\boldsymbol{x_n^i}$ thus, the Jacobian is constant and does not have to be recalculated for every iteration. In the next section KS entropy values for proposed system have been realized for different types of matrix $\boldsymbol{A}$.

3.1.1 | KS entropy analysis using coupled nearest neighbors matrix A

The term coupled nearest neighbors for matrix $\boldsymbol{A}$ refer to the adjacent elements of the matrix $\boldsymbol{A}$ connected to immediate neighbors (left and right) using chosen coupling strength. The matrix $\boldsymbol{A}$ below shows that, only middle element $(1-\epsilon)$ and its neighboring elements $\left(\frac{\epsilon}{2}\right)$ have non-zero values whereas, rest of the elements in a given row are zeros.

$$A = \begin{bmatrix} (1-\epsilon) & \left(\frac{\epsilon}{2}\right) & 0 & & 0 & \left(\frac{\epsilon}{2}\right) \\ \left(\frac{\epsilon}{2}\right) & (1-\epsilon) & \left(\frac{\epsilon}{2}\right) \dots & & 0 & 0 \\ 0 & \left(\frac{\epsilon}{2}\right) & (1-\epsilon) & \left(\frac{\epsilon}{2}\right) & & 0 \\ & \vdots & & \ddots & & \vdots \\ \left(\frac{\epsilon}{2}\right) & 0 & 0 \dots & & \left(\frac{\epsilon}{2}\right) & (1-\epsilon) \end{bmatrix}$$

The KS entropy has been analyzed using nearest neighbor coupling approach as shown in matrix $\boldsymbol{A}$ defined above. Figure 5 shows the a three dimensional KS entropy plot for sawtooth map based CML system generated using following control parameters: $\lambda = [\mathbf{0}, \mathbf{10}]$ $\boldsymbol{L} = \mathbf{100}$, $\boldsymbol{h} = \mathbf{1}$ and $\epsilon = [\mathbf{0}, \mathbf{1}]$.

This can be observed from Figure 5, the higher values of λ, correspond to increasing KS entropy values and roughly linear trend. This behavior can be explained with the link between amplitude of positive Lyapunov exponents and stability of the system. In contrast the positive Lyapunov exponents with higher amplitude showing strong chaotic behavior of the system, negative Lyapunov exponents denote the system is not in chaotic orbit. For coupling factor (ϵ), the KS entropy seems to be reaches maximum amplitude for minimum ($\epsilon = 0$) and maximum ($\epsilon = 1$) values of ϵ and seems to have lower KS entropy for ϵ values around 50 % coupling strength ($\epsilon = 0.5$). This behavior can be interpreted by understanding the impact of increasing coupling strength between the lattice nodes. The least value for coupling factor (ie, $\epsilon = \mathbf{0}$) denotes the sawtooth map based lattice nodes of the CML are not linked together at all. However increasing ϵ towards maximum (ie, $\epsilon = \mathbf{1}$) strengthens the bond between lattice nodes, thus resulting in a strongly connected chaotic system.

3.1.2 | KS entropy analysis using globally coupled matrix A

In contrast to nearest neighbor coupling, where each node is linked to only immediate neighboring lattice nodes, in a globally coupled CML system, every lattice node is linked to another lattice node with the specified coupling strength. This can be visualized using the globally coupled matrix A as shown below:

$$A = \begin{bmatrix} (1-\epsilon) & \left(\frac{\epsilon}{L-1}\right) & \left(\frac{\epsilon}{L-1}\right) & & \left(\frac{\epsilon}{L-1}\right) & \left(\frac{\epsilon}{L-1}\right) \\ \left(\frac{\epsilon}{L-1}\right) & (1-\epsilon) & \left(\frac{\epsilon}{L-1}\right) \dots & & \left(\frac{\epsilon}{L-1}\right) & \left(\frac{\epsilon}{L-1}\right) \\ \left(\frac{\epsilon}{L-1}\right) & \left(\frac{\epsilon}{L-1}\right) & (1-\epsilon) & \left(\frac{\epsilon}{L-1}\right) & & \left(\frac{\epsilon}{L-1}\right) \\ & \vdots & & \ddots & & \vdots \\ \left(\frac{\epsilon}{L-1}\right) & \left(\frac{\epsilon}{L-1}\right) & \left(\frac{\epsilon}{L-1}\right) \dots & & (1-\epsilon) & \left(\frac{\epsilon}{L-1}\right) \end{bmatrix}$$

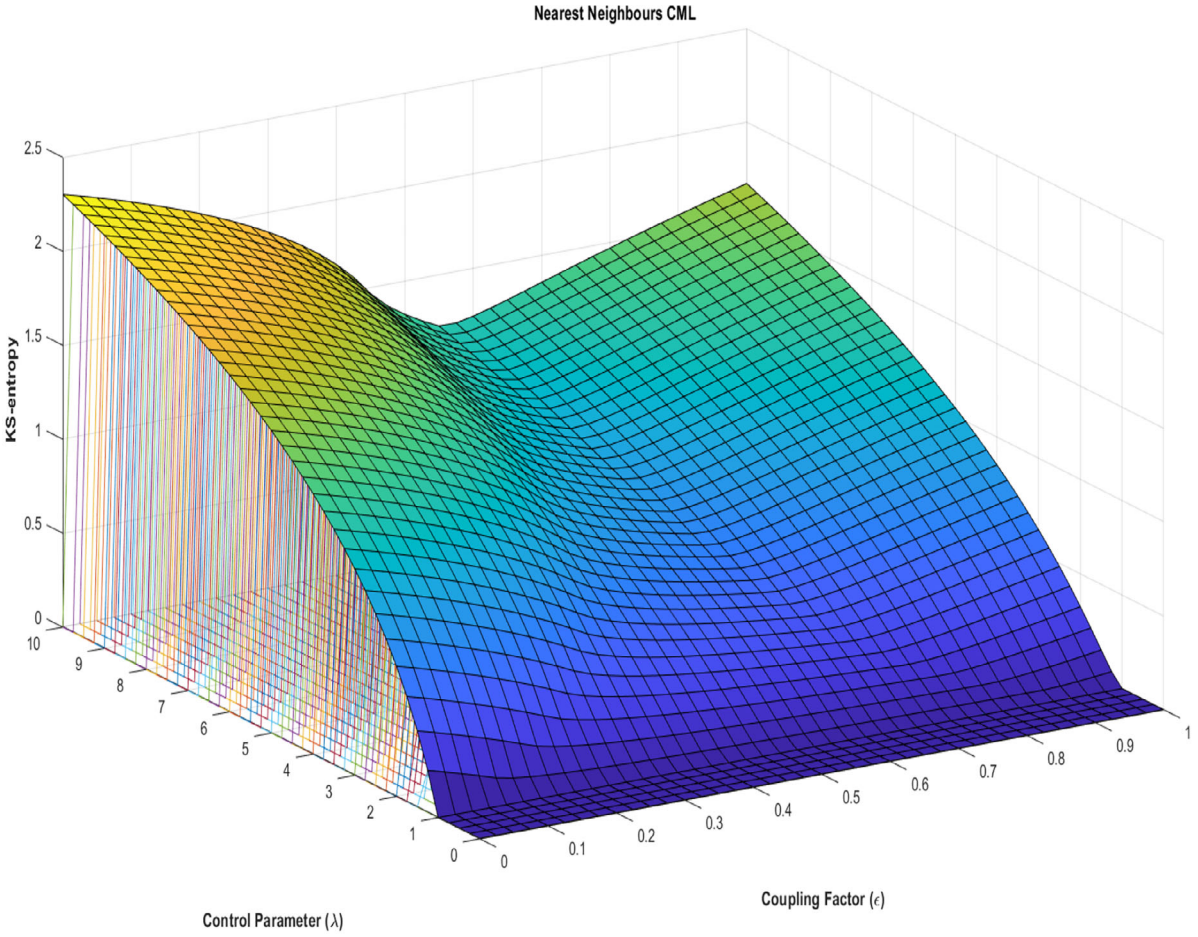

FIGURE 5 KS entropy analysis of sawtooth based CML system for matrix A coupled using nearest neighbors approach with control parameters, $L = 100$, $h = 1$, $\lambda = [0, 10]$, and $\epsilon = [0, 1]$

In Figure 6, KS entropy has been analyzed for the globally coupled neighbors approach using matrix $\boldsymbol{A}$. The KS entropy has been calculated using following parameters: $\lambda = [\mathbf{0}, \mathbf{10}]$), $\boldsymbol{L} = \mathbf{100}$, $\boldsymbol{h} = \mathbf{1}$ and $\epsilon = [\mathbf{0}, \mathbf{1}]$. This can be observed that for increasing values of ϵ, the KS entropy of the system shows a decreasing trend. This behavior can be understood by looking at the Lyapunov spectra of globally connected CML system as explained in an earlier study.[54] The equation below shows the global Lyapunovs (λ_g^1 and λ_g^2) and their dependance on local Lyapunov exponents (λ) for a fully connected CML lattice. This can be observed that for increasing values of ϵ towards maximum (ie, $= \mathbf{1}$), would result in the values of λ_g^2 to be non-positive, thus impacting the KS entropy to drop.

$$\lambda_g^1 = \ln \mid \lambda \mid, \qquad \lambda_g^2 = \ln \left| \lambda \left(1 - \frac{\epsilon N}{N-1} \right) \right| \tag{9}$$

3.1.3 | KS entropy analysis using random matrix A

To further analyze the behavior of the proposed model, the system has been tested using random matrix A. As shown in the matrix below, every element of the matrix A has been generated at random ($rand_{ij}$) and is independent of any other elements and the coupling factor (ϵ). The random values are generated such that the sum of every row of matrix A must be equal to 1, that satisfies $\sum_{j=1}^{n} rand_{ij} = 1$. The KS entropy has been analyzed for proposed method using the sample random matrix A shown below. The parameters used to generate results shown in Figure 7 are as follows: $\lambda = [0, 10]$, $L = 100$, $h = 1$ and $\epsilon = [0, 1]$.

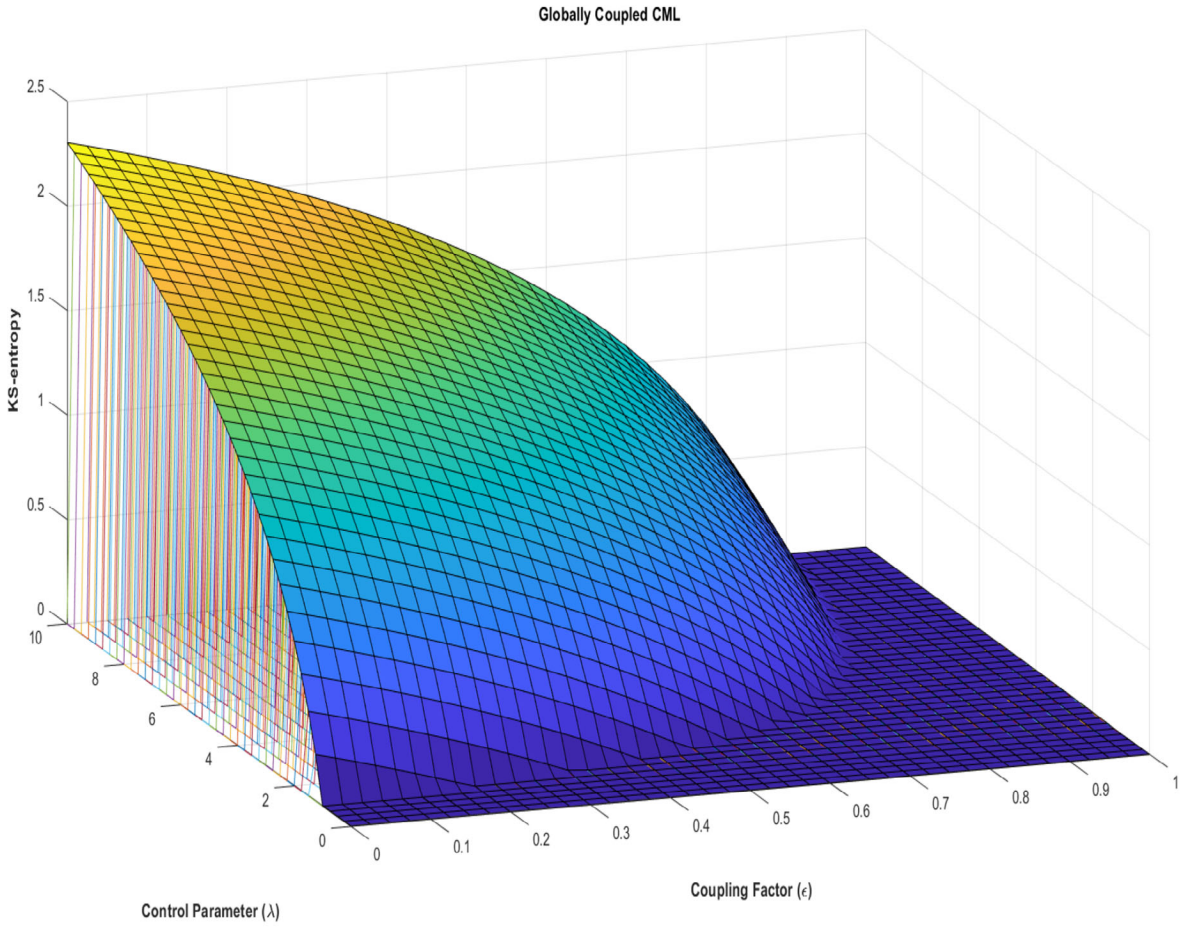

FIGURE 6 KS entropy analysis of sawtooth based CML system for globally coupled matrix A for control parameters, $L = 100$, $h = 1$, $\lambda = [0, 10]$, and $\epsilon = [0, 1]$

$$A = \begin{bmatrix} 0.3418 & 0.0064 & 0.2436 & 0.0559 & 0.0051 & 0.1452 & 0.0966 & 0.0238 & 0.0351 & 0.0466 \\ 0.0080 & 0.3221 & 0.1204 & 0.1100 & 0.0423 & 0.0624 & 0.1165 & 0.0418 & 0.0686 & 0.1078 \\ 0.1876 & 0.0743 & 0.2581 & 0.2079 & 0.0005 & 0.1457 & 0.0207 & 0.0128 & 0.0438 & 0.0486 \\ 0.0513 & 0.0810 & 0.2478 & 0.2656 & 0.0291 & 0.2015 & 0.0129 & 0.0375 & 0.0480 & 0.0254 \\ 0.0225 & 0.1511 & 0.0027 & 0.1409 & 0.2458 & 0.2080 & 0.0627 & 0.0165 & 0.0781 & 0.0716 \\ 0.1286 & 0.0443 & 0.1676 & 0.1944 & 0.0414 & 0.2002 & 0.1183 & 0.0117 & 0.0265 & 0.0669 \\ 0.1407 & 0.1361 & 0.0391 & 0.0205 & 0.0205 & 0.1947 & 0.2268 & 0.0308 & 0.0726 & 0.1180 \\ 0.1057 & 0.1487 & 0.0741 & 0.1814 & 0.0165 & 0.0589 & 0.0938 & 0.1835 & 0.0202 & 0.1173 \\ 0.0913 & 0.1431 & 0.1481 & 0.1360 & 0.0457 & 0.0780 & 0.1297 & 0.0118 & 0.1724 & 0.0440 \\ 0.0882 & 0.1635 & 0.1194 & 0.0524 & 0.0305 & 0.1429 & 0.1532 & 0.0500 & 0.0320 & 0.1681 \end{bmatrix}$$

A random matrix A of size 10×10 has been generated using random values and used for the KS entropy analysis as shown in Figure 7. This can be observed that with random matrix A, the KS entropy of the system increases with increase in λ values, however the coupling factor does not have any impact on the entropy of the system. This behavior can be explained by recalling the fact that random matrix A is independent of coupling factor. Thus, the evidence can be observed in Figure 7 that increasing ϵ values do not influence KS entropy. The KS entropy only depends on λ values,

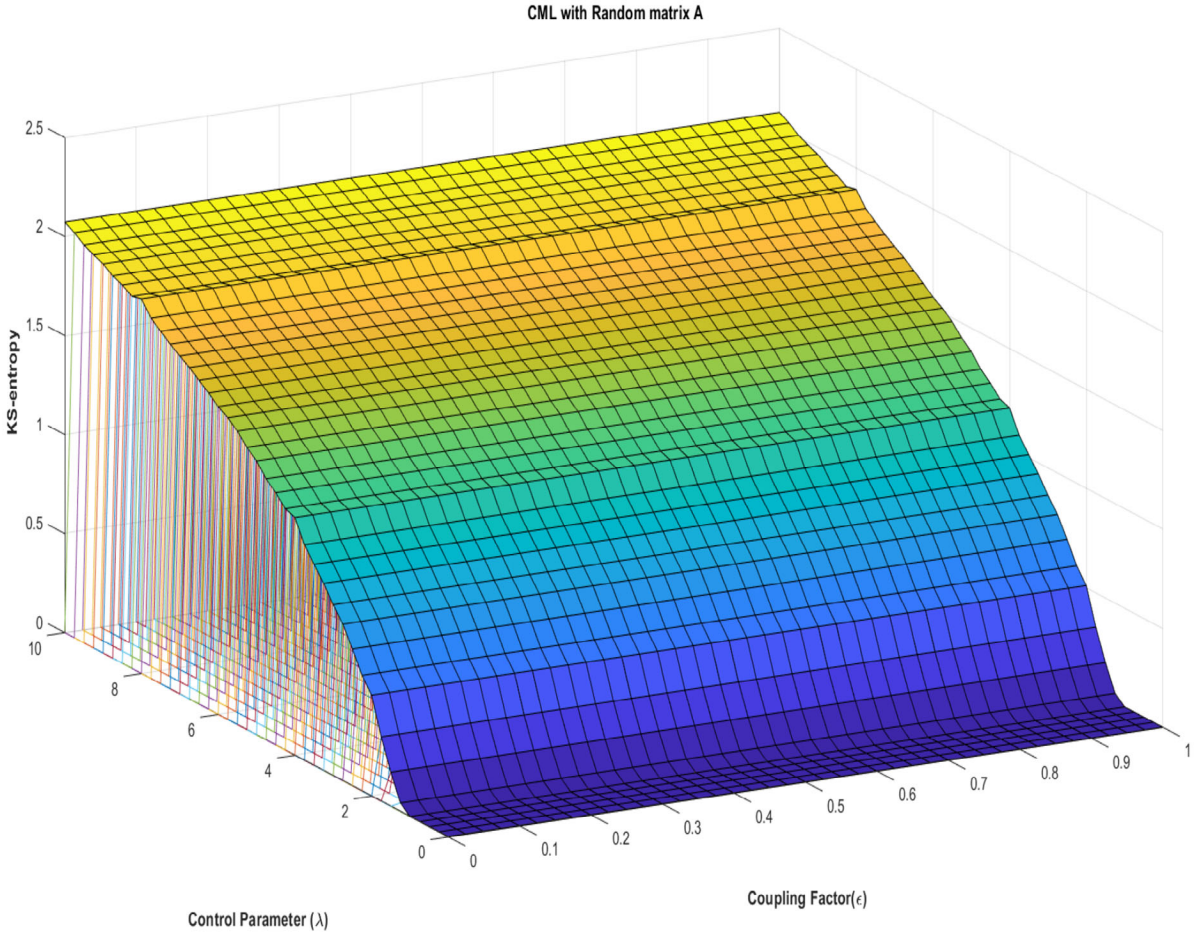

FIGURE 7 KS entropy analysis of sawtooth based CML system for random matrix A for control parameters, $L = 100$, $h = 1$, $\lambda = [0, 10]$, and $\epsilon = [0, 1]$

where increasing λ increases KS entropy. As discussed earlier, higher λ values denote larger Lyapunov exponents, thus higher entropy of the system.

3.1.4 | Time calculation for KS entropy matrix

The main idea behind the selection of different types of matrix A is to test the proposed system behavior for when different types of matrix A are used. As discussed in the earlier sections, the proposed method does not require re-calculation of Jacobian matrices. Since one of the most important characteristics a PRNG must possess is fast and lightweight generation of pseudo random sequences, the feature of having constant Jacobians saves significant computational power. For further analysis, we also examined the time consumed if the size of matrix A was increased, and how that can be compared with other types of matrix A including nearest neighbors, globally coupled and random matrix types. In the experiments performed in the previous sections, the size chosen for matrix A a was 10×10 matrix. However, for fair testing of the proposed system we chose the size of matrix A to be 50×50. Figure 8 summarizes the results of the experiments performed for KS entropy calculation.

Thus, it can be observed that for a 50×50 matrix, the time taken by the proposed method is approximately same, which means that no matter what type of matrix A is chosen as input, the performance of the system would not be affected. Moreover, reducing the matrix size leads to faster calculations as could be seen in case of random matrix of dimension 20×20, which consumes almost 0.459 s to calculate the KS entropy of the system for the whole range of coupling factors. Therefore, smaller sizes are preferred for the system to work efficiently.

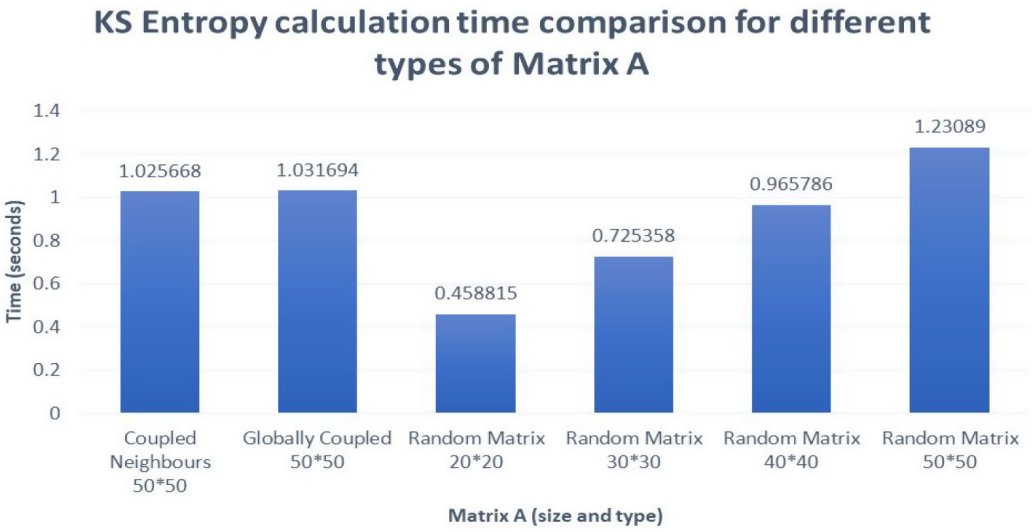

FIGURE 8 Time comparison between different types and sizes of matrix A used for KS entropy calculations

4 | PSEUDO RANDOM NUMBER GENERATION USING SAWTOOTH BASED CML

4.1 | Benefits of using hashing in PRNG

In cryptography, a hash function works as a mapping function that transforms input of any arbitrary length and outputs a fixed size string depending on the hash function key.[57] This output is referred to as the message digest or hash value. A hash function can be represented as:

$$h = hash(m) \tag{10}$$

where, m is the input string, *hash* is the hashing function and h is the unique hashed image of the input m. The unique nature of hashing functions makes them a good fit for pseudo random number generation, some of the main benefits are discussed as below:

- **Ciphered random seed values:** One of the main benefits using hash functions is that once random seed values are hashed, they are masked with the hashing key and can be shared over an IoT network. Since the hash function is a one-way function, it is hard to predict the seed values from the hashed image.
- **Fixed length output:** Considering that the CML output is a time series data, it is challenging to use it in its original form. It is always a good idea to propose a generic solution for ease of application and installation. Hash functions can take an arbitrary length input sequence and generate a fixed length unique output. This could allow serialization of data into any size of output required depending on the application.
- **Additional entropy (re-randomization):** Hash functions can be used as a source for additional entropy to the random sequences generated from a CML system. The hash functions can be thought of as adding a degree of re-randomization to the pseudo random sequences, as they can change the characteristics of inputs. For instance, if a signal following a normal distribution is given as input to a hash function, the output distribution would have a uniform distribution.
- **Reduced memory footprint:** When considering design of PRNGs for resource constrained devices, one important aspect is to consider the memory footprint generated by the seed values. The unique feature of the hash function is that no matter what size the input is, the output will always be fixed length. This length can be chosen with respect to the memory storage capabilities of the device. Therefore, one can choose the output of the hashing function to be 32-bit, 64-bit or any other size of output length depending on the memory requirements of the target device.

4.2 | Choice of hashing function

There is a huge variety when it comes to available hash functions on the market, NIST has been actively adding upgrades to hashing algorithms and standardizing procedures ranging from $SHA-1$[58] and $SHA-2$ family algorithms[59] to $SHA-3$ finalists.[60] The conventional cryptographic primitives are designed for devices ranging from smart phones to desktop servers with 32/64-bit processors and more than 1 gigabyte of memory, however in the IoT world, sensor devices are constrained with memory resources of a few kilobytes and equipped with 8/16-bit processors. Therefore, the traditional hash functions used in cryptography like the SHA family or MD5 are not suitable for IoT devices. Considering the need for lightweight hashing algorithms, NIST has already completed a second round to select the best lightweight hashing candidate proposals and is heading towards a further round.[61]

There are several other world class conferences, journals and standardizing institutions actively calling and testing for lightweight hashing functions. The International Standards Organization (ISO) published three lightweight hash algorithms as standards in 2016[62] including SPONGENT,[63] Lesamnta-LW[64] and PHOTON.[65] SPONGENT and Leamnta-LW have the same sponge-based construction design, whereas Lesamnta-LW is based on Merkle-Damgård[66] hashing which relies on dedicated block cipher schemes, for which the compression function is half of the block size. Moreover, PHOTON and SPONGENT are optimized for implementation on hardware, whereas Lesamnta-LW has been optimized for implementation in software.[67]

Considering that SPONGENT and PHOTON have similar properties, a comparison of the characteristics of SPONGENT and Lesamnta-LW to best fit our proposed PRNG algorithm was undertaken. In a study by Shoichi et al, a comparison analysis was performed for Lesamnta-LW using software-based implementations that resulted in 120-bit collision resistance and 54 bytes of RAM utilization, however SPONGENT results revealed 80-bit collision resistance using 1329 gate equivalent (GE).[68] In another recent study, the authors analyzed the security performance (in bits) of famous lightweight hash functions as shown in Figure 9.[70]

Security per bit is crucial to consider when designing cryptographic algorithms, as it provides an estimate of the number of trials the adversary would need to perform in order to decode the original values of a sequence. According to NIST, the recommendation for a cryptographic algorithm to provide security services is provide atleast 112 bit security.[71] In Figure 9[70] it can be observed that SPONGENT provides maximum security per bit equivalent when compared to eight similar lightweight hash algorithms. Ankit et al, implemented lightweight hash functions on FPGA board (Xilinx Spartan-6) and conducted performance analysis among SPONGENT, PHOTON, KECCAK-200 and KECCAK-400.[72] The results deduced that SPONGENT had highest throughput/area ratio and PHOTON showed lowest throughput in the field.

In a recent research state of the art lightweight hash algorithms were compared including SPONGENT and Lesmanta-LW in terms of digest size (output size of a hash function), block size (size of a hashing block), and internal state (chaining value size) as shown in Table 1.[73] It can be observed that SPONGENT provides a variety of digest sizes and block sizes to choose from, as in the case of IoT devices this feature might be very helpful when considering varying resource requirements of different sensor devices. Another important factor to consider when designing an algorithm is the fastest execution time that the algorithm can achieve.

In Reference 74, the authors performed a thorough comparison analysis of lightweight hash functions used in Ethereum based blockchain. SPONGENT-256, PHOTON-256 and KECCAK-256 were compared for three types of execution times that is, real time, user time and system time. The results proved that SPONGENT-256 was the fastest of all

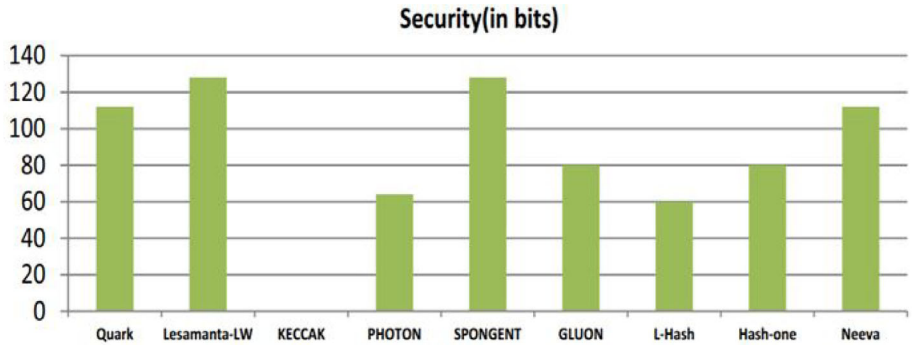

FIGURE 9 Security equivalent comparison (in bits) of famous lightweight hash functions[69]

TABLE 1 Comparison between SPONGENT and Lesamnta-LW based on digest size, block size, and internal state

Algorithms	Digest size (bits)	Block size (bits)	Internal state (bits)	Reference
SPONGENT	80/128/160/224/256	8/16	88/136/176/240/272	67
Lesamnta-LW	256	128	256	68

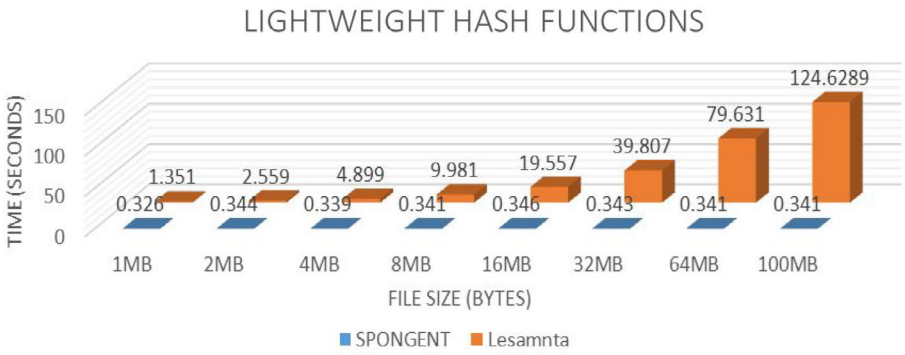

FIGURE 10 Time taken by SPONGENT-256 and Lesamnta-LW-256 to generate digest for different size of data files

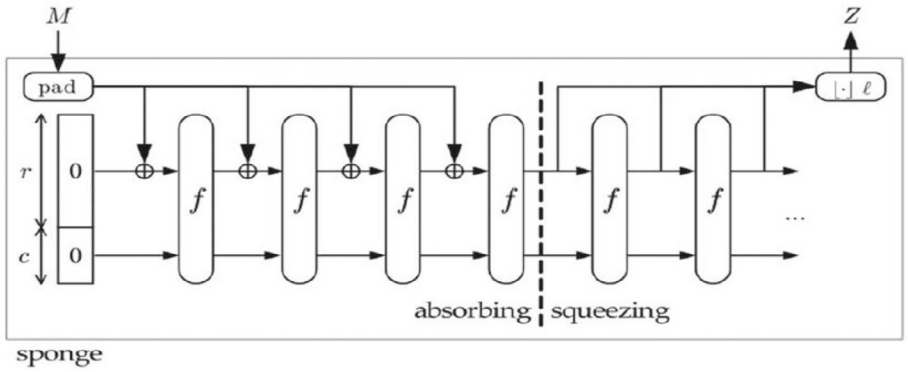

FIGURE 11 The architecture of sponge function[75]

three algorithms with fastest real, user and system time. The data generated in IoT networks is ubiquitous in nature and considering the data is generated 24 h a day by sensors, it is crucial to test the algorithm for its capacity when dealing with continuous stream of data. A comparative analysis has been performed between SPONGENT-256 and Lesamnta-256 implemented on a Raspberrypi 3B+ hardware module for varying data file sizes. The experiments were performed such that the increasing file sizes were given as input for SPONGENT-256 and Lesamnta-256, and it can be observed from Figure 10 that the time taken by Lesamnta-LW increases with file size whereas, for SPONGENT-256, the time taken to generate the hash digest was almost 0.3 s.

4.3 | SPONGENT hashing function

The design and construction of SPONGENT hashing is based on a sponge algorithm that has a finite internal state and uses a fixed-size permutation as shown in Figure 11.[75] It requires an input (variable length) and transforms it to an output (defined length). In general, the process iterates through the states with the addition of the input data and is based on a finite-state machine. The idea of the sponge function was coined by the creator of Keccak algorithm, Bertoni et al.[76]

Its working relies on a P-sponge (unkeyed permutation) or a T-sponge (random function). The function f used in the sponge construction has a variable input length and a predetermined output length. $b = r + c \geq n$ is referred to as the

width of the internal state, where r stands for rate and c is the capacity. Figure 11 shows the summary of three main phases in design of sponge function.

1. **Initialization phase:** The message is padded with a single bit 1, then as many 0 bits as are required, up to a multiple of r bits (eg, if $r = 8$, the 1-bit message 0 becomes 01000000). It is then divided into blocks of r bits.
2. **Absorption phase:** In this step, applications of the function f are spaced out between the r-bit input blocks being *XOR*ed into the state's first r bits. Following the processing of all the input blocks it moves to squeezing phase.
3. **Squeezing phase:** The function f is mixed with the first r bits of the state, which are outputted in blocks at this point. The output's bit count is specified as a component of the procedure. Overall, the input blocks never alter the final c bits of a state, and the final c bits are never produced during the squeeze phase.

Before the absorbing phase, the b-bit 0 is used as the starting value in SPONGENT. The hash size n equals either capacity c or $2c$ in all SPONGENT versions with the exception of SPONGENT-88/80/8. The state's r rightmost bit locations are where the message chunks are *XOR*ed. The hash output includes bits that are in the same r-bit places.

With the parameter selections of the majority of the SPONGENT variants, the assumption is that a permutation-based sponge construction has $n \geq c$ or $c/2 > r$. For a 128-bit hash, the values of parameters are chosen as 128-bit (SPONGENT-128-128-008 - SPONGENT-128/128/8: $n = 128$ bits, $b = 136$ bits, $c = 128$-bits, $r = 8$ bits, $R = 70$).[62]

4.4 | Algorithm design

In this section, the construction and design of the proposed algorithm is explained. Algorithm 1 shows the flow of steps that need to be followed to replicate the results generated in this study for pseudo random number generation using a sawtooth based CML system.

Algorithm 1. Algorithm design for generation of 256-bit pseudo random sequences using sawtooth map based CML system

Function: $f(x)$, $data\_strip(data, length)$
Function Definition:
def f(x):return λx mod(n).
def *data_strip* (data, length): return ($data[0 + i : length + i]$. $foriinrange(0, len(data), length)$)
Output: 256-bit random number sequence.
Initialization:
Set Control parameters: ($\lambda \geq 1$) and ($0 \leq n \leq 1$).
Initialize parameters: x_0, λ and h.
Set the number of time steps = n.
Select the number of lattice points = N
$f(x, \lambda, h) = \lambda x \quad \mod(h)$

Step 1: Discard t number of iterations to avoid transient response.

for n in range t **do**

$$x_{n+1}(i) = (1 - \epsilon)\left(f\left(x_n(i)\right)\right) + \left(\frac{\epsilon}{2}\right)\left(f\left(x_n\right)(i-1)\right) + f\left(x_n(i+1)\right)$$

end for

Step 2: Iterate the CML system for n number of iterations.

for n in range n **do**

$$x_{n+1}(i) = (1 - \epsilon)\left(f\left(x_n(i)\right)\right) + \left(\frac{\epsilon}{2}\right)\left(f\left(x_n\right)(i-1)\right) + f\left(x_n(i+1)\right)$$

end for

Step 3: Round off floating values to nearest integer (0 or 1).

$y = round(x)$

Step 4: Strip down the data stream to random chunks
$z = list(data\_strip(y))$
Step 5: Generate 256-bit hash of sequence using SPONGENT hash.
$K = hash(z)$
Quit

The first phase is the initialization phase, in which initial conditions and control parameters are defined, which are a pre-requisite for the generation of pseudo random number sequences. In Step 1, the main CML equation runs for t iterations. The data generated by these iterations is discarded in order to avoid transient behavior and to allow the sawtooth map to enter chaotic orbits. In Step 2, n iterations are looped over the CML main equation to generate pseudo random sequences. The output data generated using the sawtooth based CML system comprises of rational numbers in range [0, 1], which is rounded to the nearest 0 or 1 in Step 3 for further processing. For IoT applications, fresh seed values of fixed length are required frequently to generate new key pairs, therefore in Step 4, the data stream generated from the CML system is stripped down into chunks of 256 characters.

A function *data_strip* has been defined which shreds down the continuous stream of data values from the CML system data into chunks of arbitrary length at random. Since, the hash function takes arbitrary length input and output fixed sized hashed digest, the function *data_strip* chooses random streams of data of varying length as an input for the hashing function used in Step 5. In the final step, the SPONGENT is used to generate the digest for the pseudo random sequences. The final output K is a 256-bit pseudo random sequence, which can be used as a seed value for any cryptosystem directly. Algorithm 1 can be used to generate a consistent stream of fixed length pseudo random sequences, and the length of input and output sequences can be changed with respect to the requirements of the target cryptosystem. In order to check the effectiveness of the PRNG, the proposed algorithm is tested for randomness in the next section.

4.5 | NIST randomness tests

The National Institute of Standards and Technology (NIST) is a standardizing body for the testing cryptographic primitives, the addition of improvements and the deprecation of the ineffective security solutions.[77] NIST also provides testing tools and benchmarks for researchers and scientists to validate proposed algorithms based on set standards before making them open for public use. For testing of RNGs, NIST has made available a statistical test suite (NIST 800-22 version 2.1.2) for testing the degree of randomness in a sample number sequence.[78] The benchmarking tool comprises of 15 tests to check randomness in a sample data stream, the types of tests and the parameters used are shown in Table 2. The pseudo random sequences generated using Algorithm 1 were tested for randomness using 15 tests from Table 2.

The randomness tests were performed on sawtooth map based nearest neighbor CML system for increasing values of coupling factor ϵ in 0.1, .3, 0.7, 0.9, and λ in 2, 4, 6, 8, 10 to analyze the system of behavior for complete range of control parameters. Figure 12 shows the heat diagram to visualize the number of NIST tests passed for the sawtooth map based CML system. It can be observed that on lower coupling factors the number of tests passed were 14, while for higher coupling factors and increasing λ values the proposed algorithm was able to pass all 15 tests. Increased coupling strength a stronger connection between the CML lattice points and higher values of λ increase the magnitude of the Lyapunov exponents.

In Reference 53 the authors test proposed system using NIST statistical suite, but the system behavior was not analyzed using the full range of control parameters, with only a single value of control parameter used to generate the NIST results. Since the system relies on multiple control parameters which act as keys of the cryptosystem, it is crucial to validate systems' random behavior for all possible range of values that could be used as key space of the cryptosystem. In Figure 13, similar experiments as in Figure 12 were performed increasing the number of lattice points to 100. Increasing lattice points increases the number of dimensions of the CML system which results in more complex sequences. It can be observed from Figure 13 that only for two parameter values, the number of passed tests were 14, elsewhere, for all remaining range of parameters passed all 15 randomness tests. It can therefore be deduced that for the proposed algorithm, it is preferable to use a higher number of lattice points, that is, $L > 100$, as this increases the probability that the system passes all 15 tests for randomness.

TABLE 2 Fifteen items of NIST 800-22 test suite (version 2.1.2) and the parameters used in this article

No.	Test type	Parameters
1	Frequency (Monobit) test	
2	Frequency test within a block	Block size $m = 128$
3	Runs test	
4	Test for the longest run of ones in a block	$M = 100\ 000$
5	Binary matrix rank test	
6	Discrete Fourier transform (spectral) test	
7	Non-overlapping template matching test	$m = 9$
8	Overlapping template matching test	$m = 9$
9	Maurer's "Universal Statistical" test	
10	Linear complexity test	$M = 500$
11	Serial test	$m = 16$
12	Approximate entropy test	$m = 10$
13	Cumulative sums (Cusum) test	
14	Random excursions test	
15	Random excursions variant test	sample size = 10 bit streams

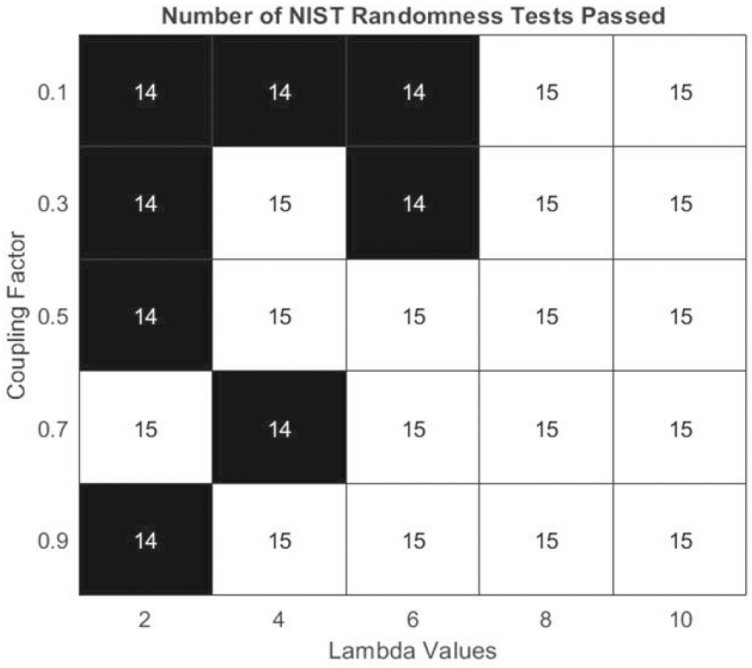

FIGURE 12 NIST statistical test results for sawtooth map based CML system with 10 lattice points

4.6 | NIST entropy tests

NIST specifies in special publication 800-90B the entropy testing of random/PRNGs.[79] There are several ways to compute entropy of an experiment like Hartley, Shannon and others.[80] However NIST's 800-90B recommendation is for the design of entropy sources for cryptographic applications, thus it uses *Min – Entropy* as evaluating measure of entropy. *Min – Entropy* considers the worst case scenario, where the secret can be successfully predicted in the first trial of guess.

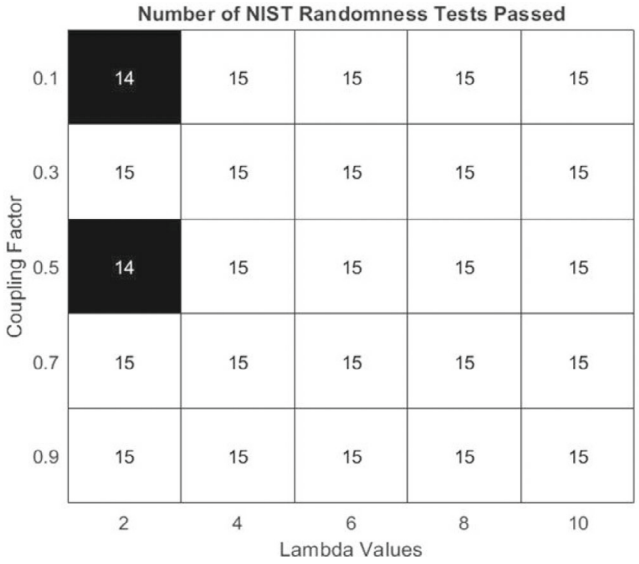

FIGURE 13 NIST statistical test results for sawtooth map based CML system with 100 lattice points

TABLE 3 Entropy results of NIST 800-90B test suite for IID assumption

	Proposed PRNG
File size (bytes)	1 000 000
Raw mean	127.42354
H_{original}	7.627072
$H_{\text{bitstring}}$	0.995163
$\min\left(H_{\text{original}}, 8 \times H_{\text{bitstring}}\right)$	7.627072
Chi square tests	Pass
Length of longest repeated substring test	Pass
IID permutation tests	Pass

The Min-Entropy of an independent discrete random variable X that takes values from the set $A = x1, x2, \ldots, xk$ with probability $Pr(X = x_i) = p_i$ for $i = 1, \ldots, k$ is defined as:

$$H = \min_{1 \le i \le k} \left(-\log_2 p_i\right) = -\log_2 \max_{1 \le i \le k} (p_i) \tag{11}$$

In cases where the entropy H of X is *Min*-Entropy, the probability of any value in X would not be larger than 2^{-H}. For a random variable with k unique and uniform probability distribution ($p1 = p2 = \ldots = pk = 1/k$) can have a maximum possible *Min*-Entropy represented as $\log_2 k$.

Table 3 summarizes the entropy analysis based on the NIST 800-90B test suite results. H_{original} denotes the entropy per byte, whereas $H_{\text{bitstring}}$ represents entropy of a single bit in a bit-string. The minimum value between H_{original} and $H_{\text{bitstring}}$ is considered as the lower bound of entropy referred as *Min*-Entropy of the sample. Chi square tests, length of longest repeated substring tests, and IID permutation tests verify the sample sequence against hypothesis based on independent and identically distributed sequences. The results show the the proposed PRNG passes all the tests. Thirty different sample sequences were generated using the proposed PRNG, with each sample sequence comprising of 1 000 000 bytes (1 MB) of random data. The NIST entropy tests were repeated 30 times for different sample sequences and average values were taken to plot the bar chart for minimum and maximum bounds of the entropy for the proposed PRNG.

The results plot in Figure 14 that the entropy values of the random sequences generated using proposed PRNG are almost unity, thus possessing maximum entropy per bit.

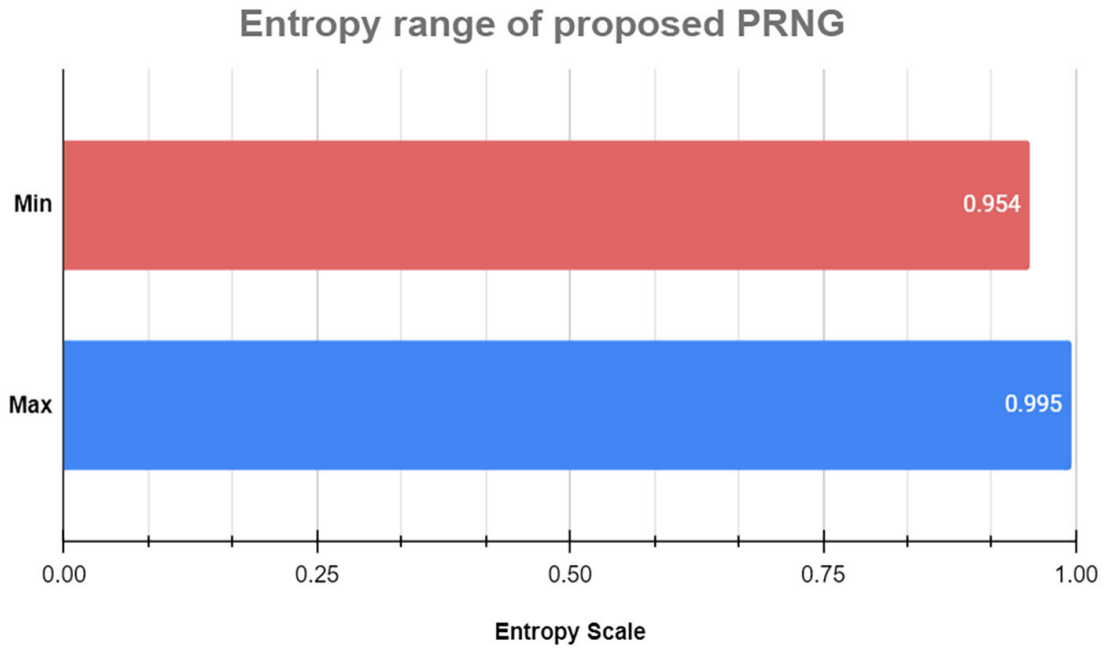

FIGURE 14 Minimum and maximum range of entropy values for proposed PRNG

5 | KEY SPACE ANALYSIS

Key space is defined as the set of any possible keys that can be chosen to secure a system. In general terms, it is possible to say that the higher the key space, the "stronger" the cryptographic key. Here, the term strong refers to the fact that the keys with higher key space are harder to guess. Moreover, one of the basic tests to analyze the strength of a cryptosystem is to perform a key space analysis. RNGs play a basic role in cryptographic applications, helping to generate large key spaces for encryption and decryption keys. In a brute force attack,[81] an attacker has to test all possible choices from the key space to break into the system. In cryptography, the term "key space strength" is used to "measure" its resistance against brute force attacks, any key space smaller than 128-bit is considered weak.[82] For the proposed PRNG, the choice of initial condition and selection of control parameters defines its key space. The experiments were performed using the Python 3.7 programming language with floating point as data type that uses 53 bit representation (16 digits precision that is, 10^{-16}. Thus, the key space for the sawtooth based CML system can be computed using following parameters: $x \in [0, 1]$, $\lambda \in [0, 10^3]$, $h \in [0, 1]$, $n \in [0, 10^{10}]$, $N \in [0, 10^6]$, this can be written in terms of number of possibilities as $(10^{16} \times 10^{16} \times 10^{16} \times 10^{16} \times 10^{16}) = 10^{80} \approx 2^{266}$. Table 4 shows comparison of the proposed PRNG scheme and similar existing PRNG solutions based on key space size and the number of control parameters used.

The compared PRNG solutions in Table 3 include a novel PRNG based on a pseudo randomly enhanced logistic map (PELM),[31] PRNG based on a logistic chaotic system (LCS),[32] PRNG design using a hyper-chaotic modified robust logistic map (HC-MRLM),[33] and pseudo-random bit generator based on multi-modal maps (PELM).[37] This can be observed that the proposed algorithm has a greater number of parameters and has a larger key space then most of existing state of the art solutions discussed in literature.

TABLE 4 Comparison of key space size with similar existing solutions

Algorithms	Key space	Control parameters
PELM[31]	2^{128}	2
LCS[32]	2^{128}	3
HC-MRLM[33]	2^{111}	3
Multi-modal maps[37]	2^{153}	3
Proposed	2^{266}	5

6 | PERFORMANCE TESTING ON IOT DEVICES

The proposed algorithm has been designed to generate a rapid stream of secure and random seed values for IoT devices using few resources. To validate the performance of the proposed algorithm on resource constrained devices we selected a range of prototype IoT sensors that mimic the same characteristics as real IoT sensors. Three Raspberry pi boards were selected with different computational capabilities. Table 6 provides a comparison of Raspberry pi Zero W, 3B+ and pi4 based on technical specifications including processor, memory, connectivity options and power consumption in idle state.[67] The choice of Raspberry pi modules is made in such a way that the proposed algorithm can be tested on extremely resource constrained raspberry pi Zero W model (single core processor and 512 MB RAM) and in advanced Raspberry pi 4 model (Quad core processor and up to 8 GB RAM). Table 5 summarizes the technical specifications of Raspberry pi used for experimentation. The devices were chosen with varying specifications to test the proposed algorithm for resource utilization on a range of devices.

Since the SPONGENT hashing is a standard hashing algorithm and the execution time and resource utilization by the hashing function has already been studied in different studies[67,69,83-85] our contribution in this work is the proposal of a sawtooth map based CML system which has been used in conjunction with SPONGENT hashing for enhanced security. Tables 6–8 shows a summary of results for time taken by the sawtooth based CML system, to generate pseudo random numbers for an increasing number of iterations and lattice points.

It can be observed that the proposed algorithm is lightweight, as the experiments performed on Raspberry pi module show the execution time for pseudo random number generation is quite fast. For 10 lattice points and 10 iterations, the execution time recorded was about 7 ms. The number of lattice points was increased to hundred lattice points, and still the execution time was almost 8 ms. The same experiments when performed on Raspberry pi 3B+ and Raspberry pi 4, the execution time went down to 0.6 ms for 10 lattice points and 10 iterations. The experiments were performed for worst case scenarios with lattice points increased to hundred and number of iterations increased to one thousand, yet the execution

TABLE 5 Technical comparison of different modules of Raspberry pi based on processing, memory, connectivity, and power capabilities

	Raspberry pi zeroW	Raspberry pi 3B+	Raspberry pi 4
Processor	1 GHz single-core CPU	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz	Broadcom BCM2711, Quad-cortex Cortex-A72 (ARMv8) 64-bit SoC @ 1.5 GHz
Memory	512 MB RAM	1 GB RAM	1, 2, 4, or 8 GB of LPDDR4-2400 SDRAM (depending on model)
Bluetooth connectivity	Bluetooth 4.1	Bluetooth 4.2	Bluetooth 5.0
Wireless connectivity	802.11 b/g/n wireless	802.11 b/g/n/ac Wi-Fi	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless
Ethernet	No Ethernet	Ethernet	Gigabit Ethernet
Power consumption (idle state)	100 mA (0.6 W)	350 mA (1.9 W)	540 mA (2.7 W)

TABLE 6 Time taken by sawtooth based CML system (using Raspberry pi Zero W) in generation of pseudo random numbers for increasing number of lattice point and iterations

Raspberry pi Zero W		
Lattice points	**No. of iterations**	**Time taken**
10	10	0.00747
10	100	0.07330
10	1000	0.72680
100	10	0.00836
100	100	0.08205
100	1000	0.82103

TABLE 7 Time taken by sawtooth based CML system (using Raspberry pi 3B+) in generation of pseudo random numbers for increasing number of lattice point and iterations

Raspberry pi 3B+		
Lattice points	**No. of iterations**	**Time taken**
10	10	0.0010
10	100	0.00999
10	1000	0.09879
100	10	0.00119
100	100	0.01150
100	1000	0.11563

TABLE 8 Time taken by sawtooth based CML system (using Raspberry pi 4) in generation of pseudo random numbers for increasing number of lattice point and iterations

Raspberry pi 4		
Lattice points	**No. of iterations**	**Time taken**
10	10	0.00061
10	100	0.00566
10	1000	0.05741
100	10	0.00069
100	10	0.00669
100	1000	0.06905

time stayed less than 1 s with nominal CPU utilization. The results provide strong evidence that the proposed algorithm is lightweight in nature and can be used on low powered IoT devices with limited processing, memory, and power resources.

7 | COMPARISON WITH STATE-OF-THE-ART SOLUTIONS

Despite the promising results from the previous analyses of the proposed system, it is a standard practice to compare the introduced system with existing state-of-the-art solutions. Traditional PRNGs undergo statistical and security testing, but for CML based PRNGs it is crucial to analyze the chaotic behavior of the proposed system as well.

Table 9 shows the summary of comparisons performed between the proposed method and similar CML based PRNG solutions. It can be observed that for a majority of proposed systems the chaos-based analysis has been not performed to a satisfactory level. Considering the dynamic nature of CML systems, it is important to study their chaotic behavior using Lyapunov analysis, visualizing pseudo phase trajectories, KS entropy and bifurcation analysis. If the chaotic behavior of the CML based PRNG is ignored, it is not a good idea to propose it for security applications, since thorough knowledge about the cryptosystem behavior is required for its proper working.

Table 9 also reveals that all on the table, CML based PRNG proposals have been tested for randomness using NIST statistical suite, which is standard for measuring the degree of randomness in a PRNG. However, majority of proposals do not use the entropy tests recommended by NIST. The proposed sawtooth based CML system has been tested using chaotic analysis, statistical analysis, entropy analysis, and security analysis. Moreover, the results produced were of satisfactory level which could be used as a basis to suggest our system is safe to be used for security applications.

TABLE 9 Comparison of proposed CML based PRNG with existing similar solutions on the basis of chaos, security, and statistical analysis technique used to validate the proposed systems

	Reference 48	Reference 49	Reference 51	Reference 52	Reference 29	Proposed method
Local chaotic map	**Logistic map**	**Logistic map**	**Chaotic iteration**	**Logistic map**	**Generalized symm. map**	**Sawtooth map**
Chaos analysis						
Pseudo phase trajectories	×	×	×	✓	✓	✓
Lyapunov exponent	×	×	×	✓	✓	✓
Kolmogorov Sinai entropy	×	✓	×	✓	✓	✓
Bifurcation diagram	×	×	×	✓	✓	×
Security analysis						
Key space	×	×	×	×	✓	✓
Speed	×	×	×	×	✓	✓
Statistical analysis						
NIST randomness tests	✓	✓	✓	✓	✓	✓
NIST entropy tests	×	×	×	×	×	✓
Other randomness tests	×	×	×	×	×	×
Comparison with other PRNG techniques	×	×	×	×	✓	✓

8 | CONCLUSION

For encryption/decryption of data strong cryptographic keys are required. However, the strength of cryptographic keys solely depend on the PRNGs used to generate seed values.

In this article, a lightweight PRNG has been proposed for IoT devices. Sawtooth map is a famous chaotic map, but in practical implementations it never reaches chaotic orbit due to finite data precision. We tested the sawtooth map to be used as local map inside a lattice based structure called CML, where all lattices are connected to one another. The results showed that the proposed system was capable of producing random sequences. We utilize lightweight hashing function SPONGENT to serialize the output data from the CML system. The hashing function adds an extra layer of security, adds entropy conditioning and outputs fixed length random number sequences of desired length.

The proposed method has been tested for randomness and entropy against NIST statistical and entropy benchmarks. The proposed PRNG passed all 15 tests on a wide range of parameter values of the CML system. It also passed the permutation tests and output high entropy output sequences. The proposed PRNG has also been tested for resource utilization and execution time when implemented using resource constrained IoT prototype sensors. The results prove that for a larger number of lattice points and iterations as input parameters, the proposed PRNG was able to generate random sequences in less than a second. Thus, it can be deduced from the experimentation that the proposed approach is lightweight in terms of resource utilization and produces pseudo random sequences that are sufficient to be used in security applications.

9 | FUTURE WORK AND RESEARCH OPPORTUNITIES

The IoT framework relies on IoT sensors that generate data in bulk, thus requiring increased security. To generate fresh and secure cryptographic keys, a reliable supply of lightweight random number seeds is required. The proposed

lightweight PRNG is best suitable for such IoT assisted applications with resource limitations. Application areas include establishing key agreement for drones,[86] securing vehicle to infrastructure communications,[87] provable authentication in Industry 4.0,[88] and preserving privacy in connected health applications.[89]

In coming years, the growth of technologies like IoT, machine learning (ML) and artificial intelligence (AI) will require stronger security measures. A quantum random number generator's (QRNG) ML-based cryptanalysis has recently been presented in a study.[90] To examine the effects of deterministic classical noise on an optical continuous variable QRNG at different phases, they developed a predicted ML analysis. Inherent correlations were successfully recognized by the attack model when the deterministic noise sources were dominant.

The power of ML and AI technologies is also harne beings to improve the security of RNGs against attacks. In 2019, Yiming et al, proposed a ML resistant pseudo-random number generator by using PUF security modules in conjunction with the PRNG.[91] Another interesting work on improving the quality of RNGs to be used for secure communications has been suggested by Wang et al.[92] The article suggests a unique PRNG for use in secure communications based on the innovative Hopfield neural network.

Traditionally, the quality of a PRNG is evaluated using statistical and entropy tests made available by NIST. However, in a few studies the researchers have designed ML based frameworks to test the quality of PRNGs. Tilo Fischer from Fraunhofer Institute for Applied and Integrated Security Weiden, Germany used Artificial Neural Networks (ANN) to test the suitability of PRNGs for secure for cryptographic applications.[93] The results concluded that the proposed ANN based testing tool performed better than existing benchmarking tools like Dieharder. Kubczak et al, took the testing experience of RNGs to an advanced level with the proposal of online tool called "ORANGUTAN" for testing the quality of RNGs.[94] The system makes use of the original random sequences matching technique, and as a result, the user receives evaluation metrics like cost and length ratios that are simple to understand. Thus, it would be interesting to develop ML and AI based RNGs and tools to test the quality of random numbers.

ACKNOWLEDGMENTS

This research is supported by the BTIIC (British Telecom Ireland Innovation Centre) project, funded by British Telecom and Invest Northern Ireland.

CONFLICT OF INTEREST STATEMENT

The authors have no conflicts of interest to declare. All co-authors have seen and agreed with the contents of the manuscript. We certify that the submission is original work and is not under review at any other publication.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ETHICS STATEMENT

This article does not contain any studies with human participants or animals performed by any of the authors.

ORCID

Unsub Zia https://orcid.org/0000-0003-1422-2408

REFERENCES

1. Sengupta J, Ruj S, Bit SD. A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. *J Netw Comput Appl*. 2020;149:102481. doi:10.1016/j.jnca.2019.102481
2. Baldanzi L, Crocetti L, Falaschi F, et al. Cryptographically secure pseudo-random number generator IP-core based on SHA2 algorithm. *Sensors*. 2020;20(7):1869. doi:10.3390/s20071869
3. Stipčević M, Koç ÇK. True random number generators. In: Koç Ç, ed. *Open Problems in Mathematics and Computational Science*. Cham: Springer; 2014:275-315. doi:10.1007/978-3-319-10683-0_12
4. Agnew GB. Random sources for cryptographic systems. In: Chaum D, Price WL, eds. *Advances in Cryptology — EUROCRYPT' 87*. Berlin: Springer; 1987:77-81. doi:10.1007/3-540-39118-5_8
5. Buchovecká S, Lórencz R, Kodýtek F, Buček J. True random number generator based on ring oscillator PUF circuit. *Microprocess Microsyst*. 2017;53:33-41. doi:10.1016/j.micpro.2017.06.021
6. Epstein M, Hars L, Krasinski R, Rosner M, Zheng H. Design and implementation of a true random number generator based on digital circuit artifacts. In: Walter CD, Koç ÇK, Paar C, eds. *Cryptographic Hardware and Embedded Systems - CHES 2003*. Berlin: Springer; 2003:152-165. doi:10.1007/978-3-540-45238-6_13

7. Kohlbrenner P, Gaj K. An embedded true random number generator for FPGAs. Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM; 2004:71-78. 10.1145/968280.968292
8. Huang CY, Shen WC, Tseng YH, King YC, Lin CJ. A contact-resistive random-access-memory-based true random number generator. *IEEE Electron Device Lett*. 2012;33(8):1108-1110. doi:10.1109/LED.2012.2199734
9. Markettos AT, Moore SW. The frequency injection attack on ring-oscillator-based true random number generators. In: Clavier C, Gaj K, eds. *Cryptographic Hardware and Embedded Systems - CHES 2009*. Berlin: Springer; 2009:317-331. doi:10.1007/978-3-642-04138-9_23
10. Bayon P, Bossuet L, Aubert A, et al. Contactless electromagnetic active attack on ring oscillator based true random number generator. In: Schindler W, Huss SA, eds. *Constructive Side-Channel Analysis and Secure Design*. Berlin: Springer; 2012:151-166. doi:10.1007/978-3-642-29912-4_12
11. Yu Y, Moraitis M, Dubrova E. Can deep learning break a true random number generator? *IEEE Trans Circuit Syst II Express Briefs*. 2021;68(5):1710-1714. doi:10.1109/TCSII.2021.3066338
12. Rotenberg A. A new pseudo-random number generator. *J ACM*. 1960;7(1):75-77. doi:10.1145/321008.321019
13. Kennedy T. Monte Carlo methods-a special topics course. Tucson: University of Arizona; 2016. https://www.math.arizona.edu/∼tgk/mc/book.pdf.
14. Widynski B. Squares: a fast counter-based RNG. arXiv preprint arXiv:2004.06278, 2020. doi: 10.48550/arXiv.2004.06278
15. Harase S, Kimoto T. Implementing 64-bit maximally equidistributed F2-linear generators with Mersenne prime period. *ACM Trans Math Softw*. 2018;44(3):1-11. doi:10.1145/3159444
16. Yu F, Zhang Z, Shen H, et al. Design and FPGA implementation of a pseudo-random number generator based on a Hopfield neural network under electromagnetic radiation. *Front Phys*. 2021;9:302. doi:10.1109/ACCESS.2019.2956573
17. Chen S, Li B, Zhou C. FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator. *Microprocess Microsyst*. 2018;59:57-68. doi:10.1016/j.micpro.2018.02.001
18. Johnson AP, Chakraborty RS, Mukhopadhyay D. A novel attack on a FPGA based true random number generator. Proceedings of the WESS'15: Workshop on Embedded Systems Security. ACM; 2015:1-6. 10.1145/2818362.2818368
19. Mahmoud D, Stojilović M. Timing violation induced faults in multi-tenant FPGAs. Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE; 2019:1745-1750. https://ieeexplore.ieee.org/abstract/document/8715263
20. Zia U, McCartney M, Scotney B, et al. Survey on image encryption techniques using chaotic maps in spatial, transform and spatiotemporal domains. *Int J Inf Secur*. 2022;21:917-935. doi:10.1007/s10207-022-00588-5
21. Akhshani A, Akhavan A, Mobaraki A, Lim SC, Hassan Z. Pseudo random number generator based on quantum chaotic map. *Commun Nonlinear Sci Numer Simul*. 2014;19(1):101-111. doi:10.1016/j.cnsns.2013.06.017
22. François M, Grosges T, Barchiesi D, Erra R. Pseudo-random number generator based on mixing of three chaotic maps. *Commun Nonlinear Sci Numer Simul*. 2014;19(4):887-895. doi:10.1016/j.cnsns.2013.08.032
23. Lambić D, Nikolić M. Pseudo-random number generator based on discrete-space chaotic map. *Nonlinear Dyn*. 2017;90(1):223-232. doi:10.1007/s11071-017-3656-1
24. Ayubi P, Setayeshi S, Rahmani AM. Deterministic chaos game: a new fractal based pseudo-random number generator and its cryptographic application. *J Inf Secur Appl*. 2020;52:102472. doi:10.1016/j.jisa.2020.102472
25. Ergün S. Cryptanalysis of a chaotic ring oscillator based random number generator. Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). IEEE; 2018:1498-1501. https://ieeexplore.ieee.org/document/8456080.
26. Ergün S. Cryptanalysis of a chaos based random number generator: lessons learned. Proceedings of the 2017 International Conference on Signals and Systems (ICSigSys). IEEE; 2017:112-116. https://ieeexplore.ieee.org/abstract/document/7967023.
27. Erbay C. Cryptanalysis of the chaotic oscillator based random number generator for bank authenticator device. *Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Dergisi*. 2020;22(2):591-599. doi:10.25092/baunfbed.742618
28. Kaneko K. Overview of coupled map lattices. *Chaos*. 1992;2(3):279-282. doi:10.1063/1.165869
29. 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(2):1-17. doi:10.1007/s42452-021-04919-4
30. Kaneko K. Period-doubling of kink-antikink patterns, quasiperiodicity in antiferro-like structures and spatial intermittency in coupled logistic lattice: towards a prelude of a "field theory of chaos". *Prog Theor Phys*. 1984;72(3):480-486. https://academic.oup.com/ptp/article/72/3/480/1906788
31. Murillo-Escobar M, Cruz-Hernández C, Cardoza-Avendaño L, Méndez-Ramírez R. A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. *Nonlinear Dyn*. 2017;87(1):407-425. doi:10.1007/s11071-016-3051-3
32. Wang L, Cheng H. Pseudo-random number generator based on logistic chaotic system. *Entropy*. 2019;21(10):960. doi:10.3390/e21100960
33. Irfan M, Ali A, Khan MA, et al. Pseudorandom number generator (PRNG) design using hyper-chaotic modified robust logistic map (HC-MRLM). *Electronics*. 2020;9(1):104. doi:10.3390/electronics9010104
34. Hemdan AM, Faragallah OS, Elshakankiry O, Elmhalaway A. A fast hybrid image cryptosystem based on random generator and modified logistic map. *Multimed Tools Appl*. 2019;78(12):16177-16193. doi:10.1007/s11042-018-6948-7
35. Chen SL, Hwang T, Lin WW. Randomness enhancement using digitalized modified logistic map. *IEEE Trans Circuit Syst II Express Briefs*. 2010;57(12):996-1000. doi:10.1109/TCSII.2010.2083170
36. Liu J, Liang Z, Luo Y, et al. A hardware pseudo-random number generator using stochastic computing and logistic map. *Micromachines*. 2021;12(1):31. doi:10.3390/mi12010031

37. García-Martínez M, Campos-Cantón E. Pseudo-random bit generator based on multi-modal maps. *Nonlinear Dyn*. 2015;82(4):2119-2131. doi:10.1007/s11071-015-2303-y
38. Stoyanov B, Kordov K. Novel secure pseudo-random number generation scheme based on two tinkerbell maps. *Adv Stud Theor Phys*. 2015;9(9):411-421. 10.12988/astp.2015.5342
39. Tutueva A, Pesterev D, Karimov A, Butusov D, Ostrovskii V. Adaptive Chirikov map for pseudo-random number generation in chaos-based stream encryption. Proceedings of the 2019 25th Conference of Open Innovations Association (FRUCT). IEEE; 2019:333-338. https://ieeexplore.ieee.org/abstract/document/8981516.
40. Al-Hazaimeh OM, Al-Jamal MF, Alhindawi N, Omari A. Image encryption algorithm based on Lorenz chaotic map with dynamic secret keys. *Neural Comput Appl*. 2019;31(7):2395-2405. doi:10.1007/s00521-017-3195-1
41. Hu H, Liu L, Ding N. Pseudorandom sequence generator based on the Chen chaotic system. *Comput Phys Commun*. 2013;184(3):765-768. doi:10.1016/j.cpc.2012.11.017
42. Saber M, Eid MM. Low power pseudo-random number generator based on lemniscate chaotic map. *Int J Electr Comput Eng*. 2021;11(1):863-871. doi:10.11591/ijece.v11i1.pp863-871
43. Dastgheib MA, Farhang M. A digital pseudo-random number generator based on sawtooth chaotic map with a guaranteed enhanced period. *Nonlinear Dyn*. 2017;89(4):2957-2966. doi:10.1007/s11071-017-3638-3
44. Alioto M, Bernardi S, Fort A, Rocchi S, Vignoli V. An efficient implementation of PRNGs based on the digital sawtooth map. *Int J Circuit Theory Appl*. 2004;32(6):615-627. doi:10.1002/cta.299
45. Li CY, Chang TY, Huang CC. A nonlinear PRNG using digitized logistic map with self-reseeding method. Proceedings of the 2010 International Symposium on VLSI Design, Automation and Test. IEEE; 2010:108-111. https://ieeexplore.ieee.org/abstract/document/5496703.
46. Özkaynak F, Yavuz S. Security problems for a pseudorandom sequence generator based on the Chen chaotic system. *Comput Phys Commun*. 2013;184(9):2178-2181. doi:10.1016/j.cpc.2013.04.014
47. Kaneko K. Spatiotemporal chaos in one-and two-dimensional coupled map lattices. *Phys D Nonlinear Phenom*. 1989;37(1-3):60-82. doi:10.1016/0167-2789(89)90117-6
48. Wang P, Qiu J. A pseudorandom bit generator based on mixing of state variable of CML. Proceedings of the 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE; 2017:331-335. https://ieeexplore.ieee.org/abstract/document/8285000.
49. Lv X, Mu N, Liao X. A pseudo-random number generator based on delay coupled map lattice. Proceedings of the 2018 Eighth International Conference on Information Science and Technology (ICIST). IEEE; 2018:377-381. https://ieeexplore.ieee.org/abstract/document/8426130.
50. Xy W, Qin X. A new pseudo-random number generator based on CML and chaotic iteration. *Nonlinear Dyn*. 2012;70(2):1589-1592. doi:10.1007/s11071-012-0558-0
51. Bahi JM, Guyeux C. Hash functions using chaotic iterations. *J Algorithm Comput Technol*. 2010;4(2):167-181. doi:10.1260/1748-3018.4.2.167
52. Dong Y, Zhao G. A spatiotemporal chaotic system based on pseudo-random coupled map lattices and elementary cellular automata. *Chaos Solitons Fractal*. 2021;151:111217. doi:10.1016/j.chaos.2021.111217
53. Liang R, Tan X, Zhou H, Wang S. An efficient parallel pseudorandom bit generator based on an asymmetric coupled chaotic map lattice. *Pramana*. 2015;85(4):617-627. doi:10.1007/s12043-014-0905-4
54. McCartney M. Chaos on the hypercube and other places. *Int J Math Educ Sci Technol*. 2022;2022:1-9.
55. Frigg R. In what sense is the Kolmogorov-Sinai entropy a measure for chaotic behaviour? Bridging the gap between dynamical systems theory and communication theory. *Br J Philos Sci*. 2004;53:411-434. https://www.jstor.org/stable/3541669
56. Liu HF, Yang YZ, Dai ZH, Yu ZH. The largest Lyapunov exponent of chaotic dynamical system in scale space and its application. *Chaos*. 2003;13(3):839-844. doi:10.1063/1.1596556
57. Yi X. Hash function based on chaotic tent maps. *IEEE Trans Circuit Syst II Express Briefs*. 2005;52(6):354-357. doi:10.1109/TCSII.2005.848992
58. Gallagher P, Director A. Secure hash standard (SHS). FIPS PUB; 1995; 180:183. http://www.cs.haifa.ac.il/∼orrd/IntroToCrypto/online/fips180-3_final.pdf.
59. Dworkin MJ. SHA-3 standard: permutation-based hash and extendable-output functions; 2018. https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions.
60. Preneel B. The state of hash functions and the NIST SHA-3 competition. In: Yung M, Liu P, Lin D, eds. *Information Security and Cryptology*. Berlin: Springer; 2008:1-11. doi:10.1007/978-3-642-01440-6_1
61. Turan MS, McKay K, Chang D, et al. Status report on the second round of the NIST lightweight cryptography standardization process. Technical report. Gaithersburg: NIST; 2021. 10.6028/NIST.IR.8369
62. ISO/IEC 29192-5:2016; 2021.
63. Bogdanov A, Knežević M, Leander G, Toz D, Varıcı K, Verbauwhede I. SPONGENT: a lightweight hash function. In: Preneel B, Takagi T, eds. *Cryptographic Hardware and Embedded Systems – CHES 2011*. Berlin: Springer; 2011:312-325. doi:10.1007/978-3-642-23951-9_21
64. Hirose S, Ideguchi K, Kuwakado H, Owada T, Preneel B, Yoshida H. A lightweight 256-bit hash function for hardware and low-end devices: Lesamnta-LW. In: Rhee KH, Nyang D, eds. *Information Security and Cryptology - ICISC 2010*. Berlin: Springer; 2010:151-168. doi:10.1007/978-3-642-24209-0_10
65. Guo J, Peyrin T, Poschmann A. The PHOTON family of lightweight hash functions. In: Rogaway P, ed. *Advances in Cryptology – CRYPTO 2011*. Berlin: Springer; 2011:222-239. doi:10.1007/978-3-642-22792-9_13

66. Merkle RC. One way hash functions and DES. In: Brassard G, ed. *Advances in Cryptology — CRYPTO' 89 Proceedings*. New York: Springer; 1989:428-446. doi:10.1007/0-387-34805-0_40
67. Hirose S, Kuwakado H, Yoshida H. Provable-security analysis of authenticated encryption based on Lesamnta-LW in the ideal cipher model. *IEICE Trans Inf Syst*. 2021;104(11):1894-1901. doi:10.1587/transinf.2021NGP0008
68. Hirose S, Ideguchi K, Kuwakado H, Owada T, Preneel B, Yoshida H. An AES based 256-bit hash function for lightweight applications: Lesamnta-LW. *IEICE Trans Fundam Electron Commun Comput Sci*. 2012;95(1):89-99. doi:10.1587/transfun.E95.A.89
69. Sebastian N, Anjali V. Implementation of SPONGENT variants using FPGA. *J Emerg Technol Innov Res*. 2014;1:96-100. http://www.jetir.org/papers/JETIR1402008.pdf
70. Dhanda SS, Singh B, Jindal P. Lightweight cryptography: a solution to secure IoT. *Wirel Pers Commun*. 2020;112(3):1947-1980. doi:10.1007/s11277-020-07134-3
71. Barker E, Burr W, Polk W, Smid M. *Recommendation for Key Management: Part 1: General*. Gaithersburg: National Institute of Standards and Technology, Technology Administration; 2006. doi:10.6028/NIST.SP.800-57pt1r5
72. Shah A, Engineer M. A survey of lightweight cryptographic algorithms for IoT-based applications. In: Tiwari S, Trivedi M, Mishra K, Misra A, Kumar K, eds. *Smart Innovations in Communication and Computational Sciences*. Singapore: Springer; 2019:283-293. doi:10.1007/978-981-13-2414-7_27
73. Biryukov A, Perrin L. State of the art in lightweight symmetric cryptography. Cryptology ePrint Archive; 2017. https://eprint.iacr.org/2017/511.
74. Apriani M, Sari RF. Performance comparison of spongent and photon hashing algorithms in Ethereum-based blockchain system. Proceedings of the 2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE). IEEE; 2021:564-569. https://ieeexplore.ieee.org/abstract/document/9616831.
75. Bogdanov A, Knezevic M, Leander G, Toz D, Varici K, Verbauwhede I. SPONGENT: the design space of lightweight cryptographic hashing. *IEEE Trans Comput*. 2012;62(10):2041-2053. doi:10.1109/TC.2012.196
76. Bertoni G, Daemen J, Peeters M, Assche GV. Keccak. Cryptology ePrint Archive, Paper 2015/389; 2015. https://eprint.iacr.org/2015/389
77. Weber AG. The USC-SIPI image database: Version 5; 2006. http://sipi.usc.edu/database/.
78. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report. McLean, VA: Booz-Allen and Hamilton Inc; 2001. https://apps.dtic.mil/sti/pdfs/ADA393366.pdf.
79. Turan MS, Barker E, Kelsey J, McKay KA, Baish ML, Boyle M. *Recommendation for the Entropy Sources Used for Random Bit Generation*. Gaithersburg: National Institute of Standards and Technology; 2018. doi:10.6028/nist.sp.800-90b
80. Bromiley P, Thacker N, Bouhova-Thacker E. Shannon entropy, Renyi entropy, and Information. Statistics and Inf Series (2004-004); 2004; 9:10-42. https://www.academia.edu/32317926/Shannon_Entropy_Renyi_Entropy_and_Information.
81. Kaspersky. Brute Force attack: Definition and examples. 2021. https://www.kaspersky.com/resource-center/definitions/brute-force-attack.
82. II E, SYM D, ECRYPT II. *European Network of Excellence in Cryptology*; 2010. https://www.cosic.esat.kuleuven.be/ecrypt/ecrypt2/documents/D.SPA.20.pdf.
83. Abyaneh MRS. Security Analysis of Lightweight Schemes for RFID Systems [dissertation]. Bergen: The University of Bergen; 2012. https://core.ac.uk/download/pdf/30897794.pdf.
84. Meuser T, Schmidt L, Wiesmaier A. Comparing lightweight hash functions-PHOTON & Quark. Download Hrz Tu-Darmstadt De; 2015. https://download.hrz.tu-darmstadt.de/media/FB20/Dekanat/Publikationen/CDC/2015-07-06_TR_PhotonQuark.pdf.
85. Yalçın T, Kavun EB. On the implementation aspects of sponge-based authenticated encryption for pervasive devices. In: Mangard S, ed. *Smart Card Research and Advanced Applications*. Berlin: Springer; 2012:141-157. http://cardis.org/proceedings/cardis_2012/CARDIS2012_11.pdf
86. Gope P, Sikdar B. An efficient privacy-preserving authenticated key agreement scheme for edge-assisted internet of drones. *IEEE Trans Veh Technol*. 2020;69(11):13621-13630. doi:10.1109/TVT.2020.3018778
87. Prateek K, Altaf F, Amin R, Maity S. A privacy preserving authentication protocol using quantum computing for V2I authentication in vehicular ad hoc networks. *Secur Commun Netw*. 2022;2022:2022. doi:10.1155/2022/4280617
88. Garg S, Kaur K, Kaddoum G, Choo KKR. Toward secure and provable authentication for Internet of Things: realizing industry 4.0. *IEEE Internet Things J*. 2019;7(5):4598-4606. doi:10.1109/JIOT.2019.2942271
89. Wang Z. Blind batch encryption-based protocol for secure and privacy-preserving medical services in smart connected health. *IEEE Internet Things J*. 2019;6(6):9555-9562. doi:10.1109/JIOT.2019.2929803
90. Truong ND, Haw JY, Assad SM, Lam PK, Kavehei O. Machine learning cryptanalysis of a quantum random number generator. *IEEE Trans Inf Forensics Secur*. 2018;14(2):403-414. doi:10.1109/TIFS.2018.2850770
91. Wen Y, Yu W. Machine learning-resistant pseudo-random number generator. *Electron Lett*. 2019;55(9):515-517. doi:10.1049/el.2019.0485
92. Wang YH, Shen ZD, Zhang HG. Pseudo random number generator based on hopfield neural network. Proceedings of the 2006 International Conference on Machine Learning and Cybernetics. IEEE; 2006:2810-2813. https://ieeexplore.ieee.org/abstract/document/4028539.
93. Fischer T. Testing cryptographically secure pseudo random number generators with artificial neural networks. Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). IEEE; 2018:1214-1223. doi:10.1109/TrustCom/BigDataSE.2018.00168

94. Kubczak P, Wozniak W, Nikonowicz J, Matuszewski L, Jessa M. An online platform for testing and evaluating random number generators. Proceedings of the 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE; 2021:1-6. https://ieeexplore.ieee.org/abstract/document/9559127.

How to cite this article: Zia U, McCartney M, Scotney B, Martinez J, Sajjad A. A resource efficient pseudo random number generator based on sawtooth maps for Internet of Things. *Security and Privacy*. 2023;e304. doi: 10.1002/spy2.304