CHAOS
Theory and Applications
in Applied Sciences and Engineering

e-ISSN: 2687-4539
RESEARCH ARTICLE
Vol.3 / No.1 / 2021 / pp.11-19
https://doi.org/10.51537/chaos.835222

# Designing a Pseudo-Random Bit Generator Using Generalized Cascade Fractal Function

**Shafali Agarwal** *,[1]
*Independent Researcher, 9600 Coit Road, Plano, TX 75025, USA.

**ABSTRACT** A cascade function is designed by combining two seed maps that resultantly has more parameters, high complexity, randomness, and more unpredictable behavior. In the paper, a cascade fractal function, i.e. cascade-PLMS is proposed by considering the phoenix and lambda fractal functions. The constructed cascade-PLMS exhibits the required fractal features such as fractional dimension, self-similar structure, and covering entire phase space by the data sequence in addition to the chaotic properties. Due to the chaotic behavior, the proposed function is utilized to generate a pseudo-random number sequence in both integer and binary format. This is the result of an extreme scalability feature of a fractal function that can be implemented on a large scale. A sequence generator is designed by performing the linear function operation to the real and imaginary part of a cascade-PLMS, cascade-PLJS separately, and the iteration number at which the cascade-PLJS converges to the fixed point. The performance analysis results show that the given method has a large key space, fast key generation speed, high key sensitivity, and strong randomness. Therefore, the scheme can be efficiently used further to design a secure cryptosystem with the ability to withstand various attacks.

**KEYWORDS**
Mandelbrot set
PRNG
Cascade phoenix lambda fractal
Key security analysis
Dynamic behavior

## INTRODUCTION

An internet era extends the security requirement of the digital information transmitted over the unsecured network. Cryptography is one of the most prominent ways to protect the data from illegitimate users (SI 1998). Since the last few years, a chaotic system has attracted researchers to utilize it in the field of cryptography. The dynamical properties of a non-linear chaotic system such as unpredictability, randomness, sensitivity to the minute change in its initial value, ergodicity, complex structure and deterministic dynamics lead it to a secure cryptosystem design. The abovementioned properties encourage to construct of a chaotic system having increased security and high complexity (Devaney 2018).

A fractal is a graphical representation of a chaotic function with complex structure and infinite scaling in each direction. In addition to chaotic behavior, a fractal function possesses more features such as construction in a complex domain, fractional dimension, self-similarity, etc. (Devaney *et al.* 1989; Mandelbrot and Mandelbrot 1982). A hybrid fractal function exhibits the characteristics of seed functions with more controlling parameters. Recently, a composite fractal function has been proposed by the author and discussed the suitability of the function in an image cryptosystem design (Agarwal 2020). Even many hybrid chaotic maps and their applicability in a pseudo-random generator, cryptography, s-box design have been studied by the researchers (Artuğer and Özkaynak 2020; Bai *et al.* 2020; Hua *et al.* 2018; Lynnyk *et al.* 2015; Moysis *et al.* 2020a). Additionally, fractal geometry is widely utilizing in user authentication (Motỳl and Jašek 2011), medical image analysis (Dey *et al.* 2018), and image hashing (Khelaifi and He 2020). Unpredictable behavior and extreme sensitivity towards the change in initial values prefer a fractal function to design a pseudo-random number sequence (PRNG).

**Manuscript received:** 3 December 2020,
**Revised:** 1 February 2021,
**Accepted:** 3 February 2021.

[1] shafali.agarwal@gmail.com

A pseudo word indicates a random sequence calculated using a deterministic system. According to mathematical theory, a deterministic system is predictable. A complex sequence generator including the process to select a seed value can help to enhance the security and reduce the correlation in the generated sequence. A PRNG has a wide range of applicability in various fields such as in the game industry, artificial intelligence, cryptography, statistical simulation, and many more. On the other hand, a true random number sequence is produced by the author by visualizing spontaneous chaotic oscillation of the current through semiconductor superlattices (Bonilla *et al.* 2016).

Recently, Barnsley's chaos game rules were utilized to generate a pseudo-random sequence (Ayubi *et al.* 2020). A complex Newton fractal function was used to generate a secure PRNG due to the strong statistical characteristics and a random phase space (Barani *et al.* 2020). An additional advantage of the map is to have a PRNG in an integer as well as a complex form. A modified logistic map was utilized to generate PRNG in two phases, including initial pseudo-random sequence and normal pseudo-random sequence using the value obtained in the previous phase (Wang and Cheng 2019). Another modified logistic map was successfully applied to generate random bit sequences by performing a comparison between maps, XOR, and bit reversal (Moysis *et al.* 2020b). An original logistic map was coupled with a piecewise map to implement a chaotic pseudo-random number generator (Sahari and Boukemara 2018). To overcome the chaotic degradation that arises due to the computational accuracy, a self-perturbed hyperchaotic system based PRN generator is proposed. The used hyperchaotic map is derived using the classical Lorenz three-dimensional chaotic system (Zhao *et al.* 2019). A similar Lorenz-like Chen chaotic system (Chen and Ueta 1999) was utilized by the author to generate a complex pseudo-random number generator (Hamza 2017). Earlier a PRN generator was proposed using the time series obtained from the generalized Lorenz chaotic map (Lynnyk *et al.* 2015). The author proposed a method in (Moysis *et al.* 2020a) to generate a PRNG by extracting around 8 bits per iteration from the decimal part of the chaotic map. The method was tested on various one-dimensional maps including the logistic map, sine map, Renyi map, Chebyshev map, cubic map, cubic logistic map.

In this paper, the cascading of two fractal functions is proposed with the applicability of the function in the design of a pseudo-random number generator. The emergence of the chaotic characteristics of two maps provides a more complex environment to produce a PRNG. The change in any single parameter realizes to a completely new data sequence, which is the foremost requirement of a secure PRNG. The main contribution in the paper can be summarized as follows:

1. A cascade structure of the fractal function is implemented using Phoenix and lambda fractal functions.
2. The dynamical behavior of the proposed cascade-PLMS is thoroughly investigated by analyzing its dimension, self-similar structure, trajectory, and cobweb diagram.
3. A method to generate a pseudo-random number sequence is proposed by using a combination of a cascade-PLMS, cascade-PLJS fractal function, and a fixed-point value resultant the execution of a particular cascade-PLJS.
4. The randomness and security of the generated PRNG are verified with various tests such as key space, key sensitivity, correlation value, autocorrelation analysis, information entropy, etc.

The rest of the paper is organized as follows. The structure of the proposed cascade-PLMS, and cascade PLJS and their dynamical properties are studied in section 2. In section 3, the generated fractal functions are applied to produce a pseudo-random bit sequence. In section 4, the randomness and security performance of the generated PRNG are analyzed. Finally, the paper is concluded with a discussion of future work direction in section 5.

## A CASCADE FRACTAL FUNCTION AND IT'S DYNAMICAL BEHAVIOR ANALYSIS

A cascade fractal function (Cascade-FF) is designed by considering two seed functions (for example $F_1(x)$ and $F_2(x)$) connected in the series. For each iteration, the output of $F_1(x)$ is fed into the $F_2(x)$ as input, and the output of $F_2(x)$ is fed as an input to the $F_1(x)$. A repetitive output value feeding to each other until the number of iteration limit gets over (Zhou *et al.* 2014). Mathematically, for functions $F_1(x)$ and $F_2(x)$, a cascade-FF is defined as follows:

$$x_{n+1} = F_1(F_2(x_n)) \tag{1}$$

where $F_1(x)$ and $F_2(x)$ two seed functions which can be the same or different. A function is known as a cascade with itself if the same functions are using in the cascade-FF design. In that case, the function definition will be:

$$x_{n+1} = F_1(F_1(x_n)) \tag{2}$$

A cascade-FF has the ability to exhibit different structures while changing the order of contributed seed functions. Such as:

$$x_{n+1} = F_1(F_2(x_n)) \tag{3}$$

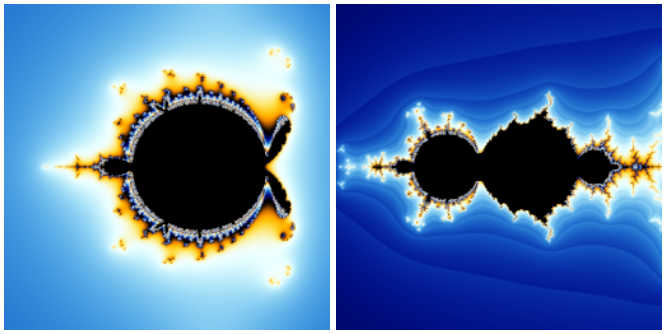
and

$$x_{n+1} = F_2(F_1(x_n)) \tag{4}$$

The paper focuses on the single aspect of designing a cascade-FF using phoenix and lambda fractal function. Let's recall the mathematical definition of the phoenix fractal and lambda fractal functions respectively (Peitgen *et al.* 2006):

$$\begin{aligned} z_{(n+1)} &= z_n^a + z_n^b c + p z_{(n-1)} \\ z_{(n+1)} &= c z_n (1 - z_n)^{(w-1)} \end{aligned} \tag{5}$$

where $c \in C$, and $-1 < p < 1$ with $z_0 \neq 0$. The fractal images generated by executing both functions are shown in Figure 1.



(a) (b)

**Figure 1** a) Phoenix fractal b) Lambda fractal.

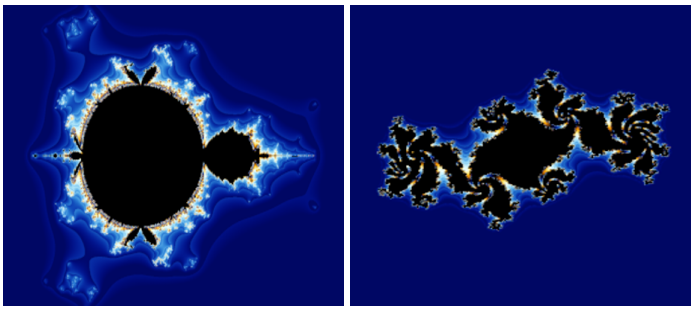
### Cascade-PLMS and Cascade-PLJS

A cascade-PLMS function is proposed to have a more complicated chaotic structure that is controlled by many parameters as compared to an individual. Too many parameters give the flexibility to have a more random and unpredictable output sequence by varying its value. By considering the phoenix fractal as $F_1(x)$ and lambda fractal as $F_2(x)$), a cascade-PLMS is defined as follows:

$$\begin{aligned} tempz &= z_n^a + z_n^b c + pz_{(n-1)} \\ z_{(n+1)} &= c * tempz(1 - tempz)^{(w-1)} \end{aligned} \tag{6}$$

All variables have their usual meaning except tempz. It represents an intermediate value of the phoenix function which has fed to the lambda function as input. The cascade-PLMS function is a set of c values for which the orbit of starting value i.e. $z_n$ remains bounded under the function iteration. The proposed cascade-PLMS function is utilized to generate a pseudo-random number sequence with the parameter values $z_0 = 0.09$, $p = -0.03$, $a = 2$, $b = 1$, and $w = 3$.

A cascade phoenix lambda Julia set (cascade-PLJS) is nothing but a fractal image of the same function for a fixed $c$ value starting with a nonzero z value. The paper has shown a cascade-PLJS image for $c = (0.7444196429, 0.6863839286)$. Both fractal images are plotted for the above-given parameter values using the UltraFractalTM and shown in Figure 2. A repetitive execution of the function with a fixed c value makes it converge to a fixed-point attractor, depending on whether the c value lies inside the cascade-PLMS image or outside of it. The convergence rate of the function varies for different c values. The iteration number at which the cascade-PLJS converges will be utilized in the pseudo-random number generation method.

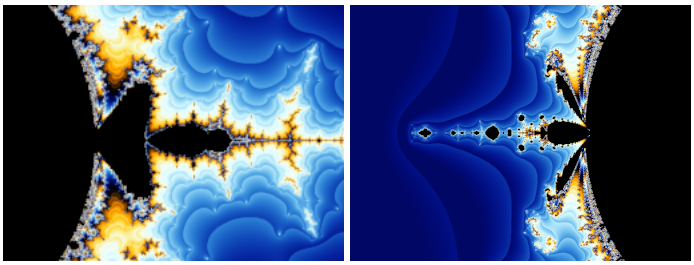


(a) (b)

**Figure 2** a) Cascade-PLMS b) Cascade-PLJS.

### Dynamical Properties Analysis of Cascade-PLMS

***Self-Similar Structure*** A fractal image is well-known to have a self-similar structure at a wide range of different scales. The beauty of a Mandelbrot set is to have infinite information on a small area of interest. As you zoom into the set, you will get newer fascinating images. A new cascade-PLMS is supposed to create an artistically appealing fractal image that also exhibits new patterns upon further exploration. Figure 3 shows randomly selected fractal images obtained by zooming the cascade-PLMS function.



(a) (b)

**Figure 3** (a)-(b) Zoomed version of cascade-PLMS

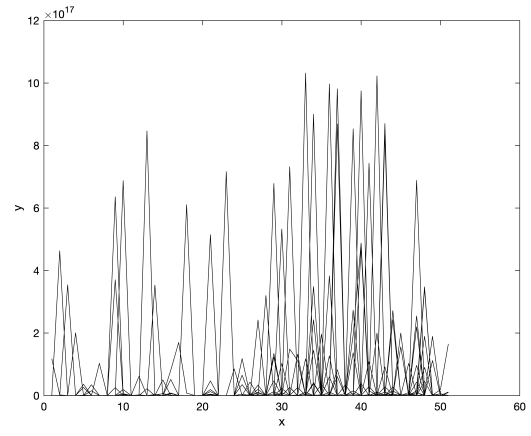
***Fractal dimension*** According to Felix Hausdorff (Czyz 1994), rough and broken fractal images should have an "in-between" dimension. This is a common way to measure the complexity of a fractal image boundary. A non-regular two-dimensional fractal image is supposed to have a dimension value between one and two. Recently, the author developed a user interface to calculate the fractal dimension using the box-counting method (Çimen *et al.* 2020). If a fractal image is superimposed by a grid of $N$ squares to occupy the $E$ number of edges, the fractal dimension can be calculated as:

$$\dim = \frac{\log N}{\log E} \tag{7}$$

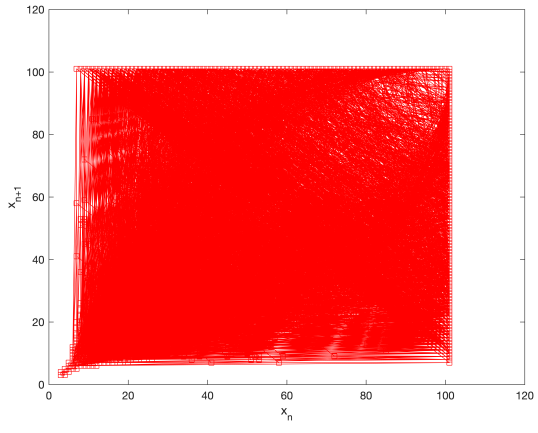
The fractal dimension for several cascade-PLMS was calculated to verify the fractional structure of the proposed system. The obtained results were able to satisfy the requirement of a fractal function. The fractal dimension of the

proposed cascade-PLMS function for the above-discussed parameters is 1.1535.

***Trajectory and Cobweb diagram*** A cobweb and trajectory diagrams are used to display the successive iterations of a function. The only difference is that a cobweb diagram presents the function behavior of a one-dimensional map whereas a trajectory diagram is used to show the path of the generated number sequence of the multi-dimensional map. The chaotic behavior of a function can be justified by distributing the generated sequence over time in the entire phase space. A cascade-PLMS fractal image is generated based on the number of iterations required to bound the initial value within the image. At the same time, a sequence of a complex number is also generated on the execution of the function for each initial value. Therefore, the below Figure 4 shows a cobweb diagram to show the occupancy of the space by the iteration values and also a trajectory diagram to present the relationship between real and imaginary values. It can be stated that the produced data covers the entire phase space in both diagrams.




(a)




(b)

**Figure 4** a) Trajectory diagram b) Cobweb diagram.

## APPLICATION TO PSEUDO-RANDOM BIT GENERATION

The pseudo-random number generator is implemented by considering the above proposed cascade-PLMS and its corresponding cascade-PLJS functions. All randomness tests verify the suitability of the proposed cascade functions to generate an unpredictable number sequence. A pictorial representation of the proposed method can be seen in Figure 5.

The process starts by executing both the functions using the initial values set within the respective value range. Here, the cascade-PLMS function generates a sequence by considering initial values ($z_0$,$a, b, c, p, w$) as $(0.09, 2, 1, 0, -0.03, 3)$ while the $c$ value is considered $(0.7444196429, 0.6863839286)$ in cascade-PLJS assuming other values same as in cascade-PLMS. The detailed method of the proposed technique is described as follows:

**Step 1:** *Calculate zdataMS and zdataJS as a set of a complex number after executing the cascade-PLMS and cascade-PLJS using the above-mentioned initial values set respectively.*

**Step 2:** *Calculate the fixed point of the cascade-PLJS function for a given c value and record the maximum iteration number (Itr) at which the fixed point is obtained.*

**Step 3:** *Separate real and imaginary parts of the zdataMS into zdataMSreal and zdataMSimg and convert it into a one-dimensional array.*

**Step 4:** *Repeat step 3 using zdataJS and obtained zdataJSreal and zdataJSimg in a one-dimensional vector.*

**Step 5:** *Perform the linear function operation on the real number sequence of both the functions and Itr as follows:*

$$updatedRealSeq = zdataMSreal * Itr + zdataJSreal \quad (8)$$

**Step 6:** *Perform the same linear function operation on the imaginary number sequence of both the functions and Itr as follows:*

$$updatedImgSeq = zdataMSimg * Itr + zdataJSimg \quad (9)$$

**Step 7:** *Convert float numbers to an integer by executing the given function separately for real sequence and imaginary sequence as follows:*

$$\begin{aligned} IntRealSeq &= round((updatedrealSeq * 2^{14}) mod 256) \\ IntImgSeq &= round((updatedImgSeq * 2^{14}) mod 256) \end{aligned} \quad (10)$$

**Step 8:** *At last, a pseudo-random sequence is computed by concatenating both the sequences obtained prior using the following function:*

$$\begin{aligned} PRNG(2j) &= IntRealSeq(i) \\ PRNG(2j+1) &= IntImgSeq(i) \end{aligned} \quad (11)$$

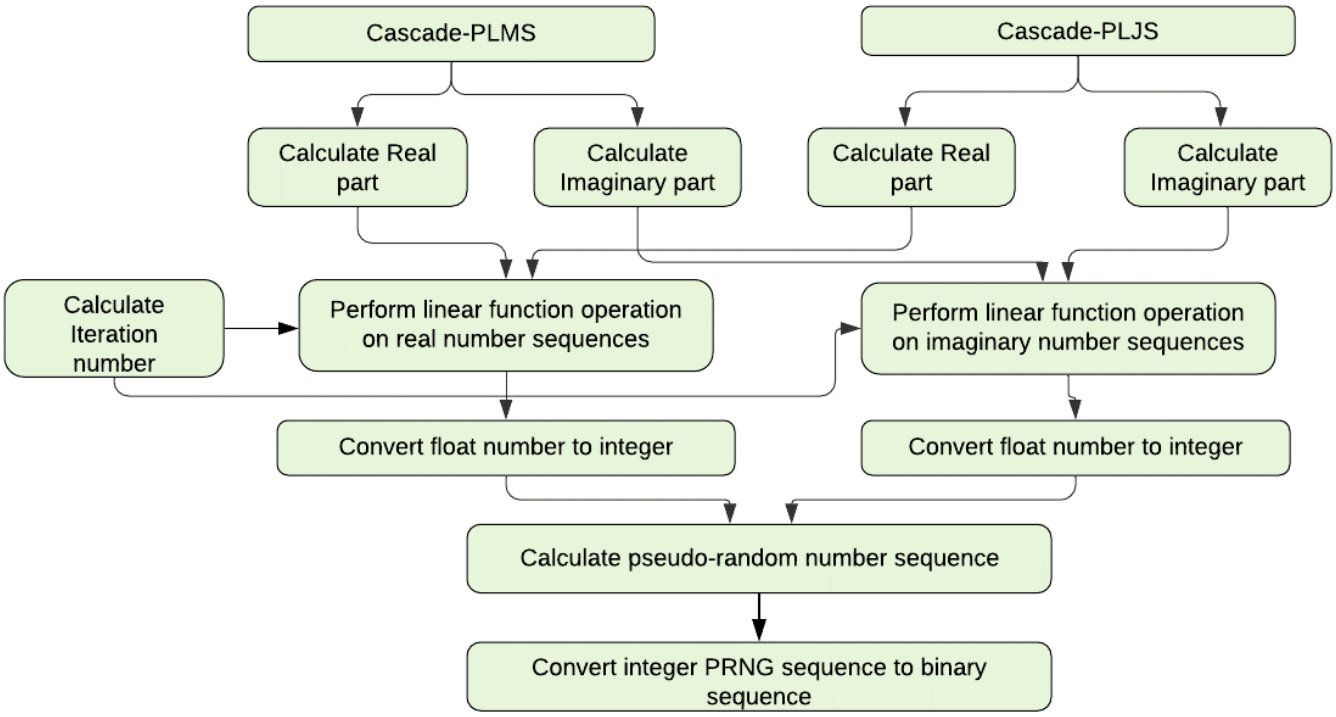



**Figure 5** Block diagram of proposed PRNG method

where $i = 1, 2, \ldots, 500000$ and initialize j=0. As a result, an integer sequence of length $10^6$ is obtained. After converting it into binary form, an 8-bit binary sequence of length $8 * 10^6$ is produced. Hence, eight different binary random number sequences can be generated by combining the digits column-wise. To have a more random outcome, intermediate 500000 values of each real and imaginary data are considered while concatenating the sequence to get a pseudo-random number sequence.

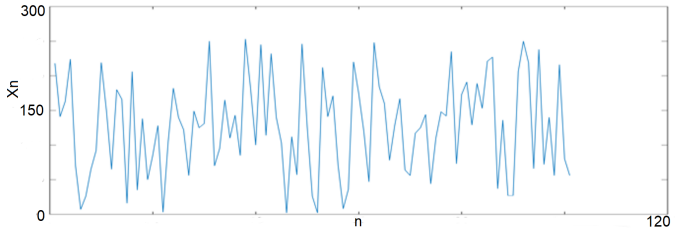



**Figure 6** Visual path of generated number sequence

## RANDOMNESS AND SECURITY ANALYSIS

### Visual Representation of PRN Sequence

A trajectory diagram is used to display the path followed by the sequence generated upon the execution of the function for a particular set of initial values. A non-linear pixel path distributed over the entire phase space represents the chaotic behavior of the map. By selecting an appropriate set of initial values set can lead to producing a random number sequence that does not show the periodic or closed curve behavior. Figure 6 displays a trajectory diagram of randomly selected 500 pixels.

### Key Space

Key space is an important index to indicate a secure cryptosystem. The generator uses a cascade fractal function having a set of initial values and control parameters to generate a pseudo-random sequence. As per the function requirement, a set of values includes $(z_0, a, b, c, p, w)$ and a previous z value. According to the IEEE floating-point standard, a computational precision of a double datatype number is about $10^{15}$. Therefore, the possible key space is calculated as $\left(10^{15}\right)^7 = 10^{105} \approx 2^{320}$. Thus, the available key space is large enough than the prescribed range of $2^{100}$ that is required to resist the brute-force attack (Alvarez and Li 2006).

### Key Generation Speed

The proposed PRNG method is implemented on MATLABTM with a MacBook Pro having system configuration 2.6 GHz 6-Core Intel Core i7, and 16 GB memory. The approximate time to produce a random key sequence of size $1000 * 1000$ is 0.2084 sec.

### Key Sensitivity using NBCR Analysis

A key sensitivity test analysis is done to evaluate the impact of the slight change in the input value to its corresponding output value. The sensitiveness of the proposed pseudo-random number generator is tested by executing two tests: 1) visual criterion, 2) the number of bit change rate (NBCR).

To evaluate the visual impact of two sequences, a control parameter value of the function is increased by $10^{-14}$ and others remain constant. Figure 7 shows the reaction of both the sequences generated through initial values and the small perturbed data set. It can be concluded from the figure that the generated number sequence is completely different even by making a small change in the control parameter value. The other test calculates the number of changed bits between two sequences. It is calculated as follows:

$$NBCR = \frac{Ham\_dis(x,y)}{bit\_len} \tag{12}$$

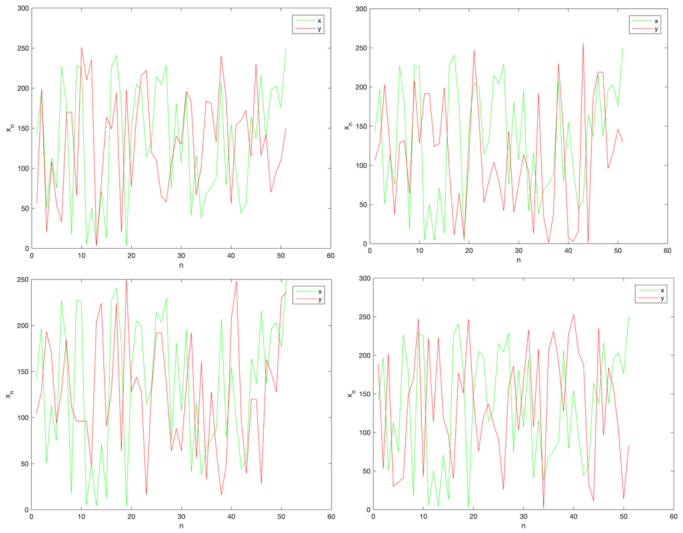


**Figure 7** Graphical representation of key sensitivity analysis of generated sequence produced using original values and altered values

The number of the bit change rate of two different number sequences is expected to be close to 50%. NBCR result in Table 1 indicates that the initially generated pseudo-random sequence is different from the sequence generated after increasing a control parameter value slightly. Therefore, the generated sequences prove the key sensitiveness of the proposed pseudo-random number generator.

### Entropy Analysis

An information entropy concept was introduced by Shannon to describes the randomness and uncertainty in the information system (Shannon 1949). It can be computed using the given function:

$$H(s) = -\sum_{i=0}^{(2^n-1)} p(x_i) log_2[p(x_i)] \tag{13}$$

**Table 1 NBCR value of generated sequence produced using original values and altered values**

| Changed Parameter | NBCR Value |
|---|---|
| Change in initial value z (Seq1) | 49.90 |
| Change in distortion (Seq2) | 49.95 |
| Change in power 'a' (Seq3) | 50.01 |
| Change in power 'w' (Seq4) | 50.04 |

where $p(x_i)$ denotes the probability of occurrence of a symbol $x_i$ in the pseudo-random sequence. If $n$ number of bits are required to represent a symbol, the entropy of the information system is supposed to be close to $n$. A binary sequence requires only one bit to show the symbol, i.e. either zero or one. Therefore, an entropy value of a binary sequence equals to one is considered as ideal value to exhibit the randomness of the sequence.

### Correlation Analysis

A correlation coefficient is calculated to analyze the relationship between the two pseudo-random number sequences. A value close to zero depicts no relationship between the two sequences whereas strongly related sequences have *a* correlation coefficient value near to one. Due to the cascading of the two functions, many parameters are involved in the pseudo-random number generator. Therefore, the correlation analysis is done by varying *a* key at *a* time and keeping constant the other parameters. For two sequences $x$ and $y$, the correlation coefficient is calculated using the given equation:

$$CC(x,y) = \frac{N\sum_{i=1}^{N}(x_i y_i) - \sum_{i=1}^{N}(x_i)\sum_{i=1}^{N}(y_i)}{\sqrt{N\sum_{i=1}^{N}(x_i)^2 - \left(\sum_{i=1}^{n}(x_i)^2\right)\left(N\sum_{i=1}^{N}(y_i)^2 - \left(\sum_{i=1}^{n}(y_i)^2\right)\right)}} \tag{14}$$

Table 2 displayed the effect of changing parameters in terms of correlation coefficient value. Each time a new sequence, i.e. $y$ is generated by adding $\varepsilon = 10^{-14}$ to the previous parameter value and also keeping others as same as before. The process is executed for every parameter in the same way and calculated the corresponding correlation value. The obtained values indicate the high sensitivity of the sequence towards the minute change in the parameter value.

**Table 2 Correlation coefficient value of generated sequence produced using original values and altered values**

| Changed Parameter | Correlation coefficient |
|---|---|
| Change in initial value z (Seq1) | 0.0034 |
| Change in distortion (Seq2) | 0.0024 |
| Change in power 'a' (Seq3) | -0.0003 |
| Change in power 'w' (Seq4) | 0.0005 |

### Autocorrelation Analysis

Autocorrelation analysis is carried out to measure the similarity between a sequence $OS$ and its corresponding shifted sequence $OSS$. The formula to calculate autocorrelation of a sequence with size $N$ is given as:

$$AC = \frac{M1 - M2}{N} \tag{15}$$

where $M1$ and $M2$ refer to the number of matches and mismatches between the $OS$ and $OSS$ respectively. A value that falls in a range $[-1, 1]$ depicts a highly random number sequence with a small correlation with itself. A graphical view of pixel autocorrelation can be seen in Figure 8.

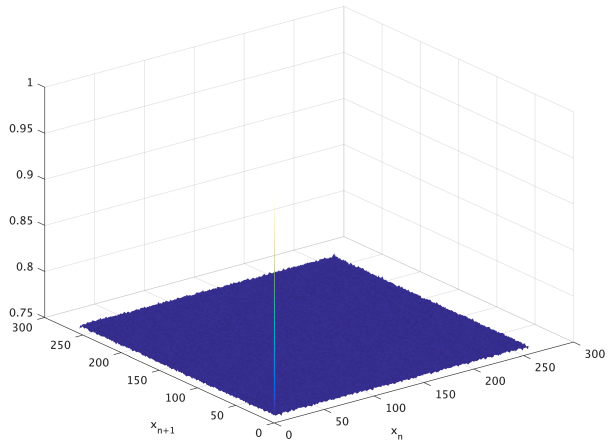



**Figure 8** Autocorrelation analysis of number sequence

### Performance Comparison with Existing Encryption Algorithms

A comparative analysis of the proposed scheme with the other existing PRNG methods is discussed in the section. The comparison is mainly focused on the evaluation parameters such as key space, entropy, number of bit change rate, and adjacent pixels correlation. Table 3 listed the data of various considered PRNG methods along with the proposed scheme to show a relative view of obtained results. The performance parameters considered in the table proved good agreement of the proposed PRNG algorithm from a highly efficient and security view.

## CONCLUSION

A cascade fractal function can be designed by combining any two existing fractals. The paper analyzed the dynamical behavior of cascade-PLMS function by considering phoenix and lambda fractals. The benefit of combining two nonlinear functions is to have a more complex structure that is further utilized to propose a pseudo-random number generator. A linear function operation was applied to the cascade-PLMS, cascade-PLJS, and the number of iterations got as a result of obtaining a fixed point of the cascade-PLJS. It is of interest to generate a PRNG which is an integer and is convertible to the 8-bit binary sequence. By considering the arrangement of the column-wise bit of the data, eight simultaneous binary number sequences can be utilized in further application. Aiming at the security of the generated PRNG, a slight change in any system parameter leads to a completely new pseudo-random number sequence.

The proposed concept of a new cascade fractal function opens the door for the researchers to analyze the feasibility of the model using the other existing fractal functions. The choice of fractal surely affects the complexity outcome based on the corresponding function combination. Further, the obtained PRNG can be applied to the cryptographic application including creating watermarks, casinos, encoding digital contents, and many more. It's also aiming to study how fast a bitstream can be generated so that it can be utilized in the hardware implementation.

**Table 3 Performance comparison of the proposed PRNG method with the existing methods**

| Encryption Algorithm | Key Space | Entropy | NBCR | Correlation Coefficient |
|---|---|---|---|---|
| Proposed | $2^{320}$ | 7.9864 | 49.97 | 0.0016 |
| Ref. (Zhao *et al.* 2019) | $2^{70}$ | 7.9896 | 49.74 | - |
| Ref. (Barani *et al.* 2020) | $2^{588}$ | 7.9937 | 50.13 | 0.0003 |
| Ref. (Ayubi *et al.* 2020) | $2^{232}$ | - | - | 0.0586 |
| Ref. (Wang and Cheng 2019) | variable | 7.9692 | 51.92 | - |
| Ref. (Agarwal 2018) | $2^{145}$ | - | - | 0.0041 |

## CONFLICTS OF INTEREST

The author declares that there is no conflict of interest regarding the publication of this paper.

## LITERATURE CITED


Agarwal, S., 2018 Cryptographic key generation using burning ship fractal. In *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*, pp. 1–6.

Agarwal, S., 2020 A new composite fractal function and its application in image encryption. Journal of Imaging **6**: 70.

Alvarez, G. and S. Li, 2006 Some basic cryptographic requirements for chaos-based cryptosystems. International journal of bifurcation and chaos **16**: 2129–2151.

Artuğer, F. and F. Özkaynak, 2020 A novel method for performance improvement of chaos-based substitution boxes. Symmetry **12**: 571.

Ayubi, P., S. Setayeshi, and A. M. Rahmani, 2020 Deterministic chaos game: A new fractal based pseudo-random number generator and its cryptographic application. Journal of Information Security and Applications **52**: 102472.

Bai, S., L. Zhou, M. Yan, X. Ji, and X. Tao, 2020 Image cryptosystem for visually meaningful encryption based on fractal graph generating. IETE Technical Review pp. 1–12.

Barani, M. J., P. Ayubi, M. Y. Valandar, and B. Y. Irani, 2020 A new pseudo random number generator based on generalized newton complex map with dynamic key. Journal of Information Security and Applications **53**: 102509.

Bonilla, L. L., M. Alvaro, and M. Carretero, 2016 Chaos-based true random number generators. Journal of Mathematics in Industry **7**: 1–17.

Chen, G. and T. Ueta, 1999 Yet another chaotic attractor. International Journal of Bifurcation and chaos **9**: 1465–1466.

Çimen, M. E., Z. GARİP, Ö. F. Boyraz, I. Pehlivan, M. Z. YILDIZ, *et al.*, 2020 An interface design for calculation of fractal dimension. Chaos Theory and Applications **2**: 3–9.

Czyz, J., 1994 *Paradoxes of measures and dimensions originating in Felix Hausdorff's ideas*. World Scientific.

Devaney, R., 2018 *An introduction to chaotic dynamical systems*. CRC Press.

Devaney, R. L., J. A. Yorke, L. Keen, K. T. Alligood, M. F. Barnsley, *et al.*, 1989 *Chaos and Fractals: The Mathematics Behind the Computer Graphics: The Mathematics Behind the Computer Graphics*, volume 1. American Mathematical Soc.

Dey, N., A. S. Ashour, H. Kalia, R. Goswami, and H. Das, 2018 *Histopathological image analysis in medical decision making*. IGI Global.

Hamza, R., 2017 A novel pseudo random sequence generator for image-cryptographic applications. Journal of Information Security and Applications **35**: 119–127.

Hua, Z., F. Jin, B. Xu, and H. Huang, 2018 2d logistic-sine-coupling map for image encryption. Signal Processing **149**: 148–161.

Khelaifi, F. and H. He, 2020 Perceptual image hashing based on structural fractal features of image coding and ring partition. Multimedia Tools and Applications pp. 1–20.

Lynnyk, V., N. Sakamoto, and S. Čelikovskỳ, 2015 Pseudo random number generator based on the generalized lorenz chaotic system. IFAC-PapersOnLine **48**: 257–261.

Mandelbrot, B. B. and B. B. Mandelbrot, 1982 *The fractal geometry of nature*, volume 1. WH freeman New York.

Motỳl, I. and R. Jašek, 2011 Advanced user authentication process based on the principles of fractal geometry. In *Proceedings of the 11th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision (ISCGAV'11)*, pp. 109–112.

Moysis, L., A. Tutueva, K. Christos, and D. Butusov, 2020a A chaos based pseudo-random bit generator using multiple digits comparison. Chaos Theory and Applications **2**: 58–68.

Moysis, L., A. Tutueva, C. Volos, D. Butusov, J. M. Munoz-Pacheco, *et al.*, 2020b A two-parameter modified logistic map and its application to random bit generation. Symmetry **12**: 829.

Peitgen, H.-O., H. Jürgens, and D. Saupe, 2006 *Chaos and fractals: new frontiers of science*. Springer Science & Business Media.

Sahari, M. L. and I. Boukemara, 2018 A pseudo-random

numbers generator based on a novel 3d chaotic map with an application to color image encryption. Nonlinear Dynamics **94**: 723–744.
Shannon, C. E., 1949 Communication theory of secrecy systems. The Bell system technical journal **28**: 656–715.
SI, W. S., 1998 Cryptography and network security: Principles and practice.
Wang, L. and H. Cheng, 2019 Pseudo-random number generator based on logistic chaotic system. Entropy **21**: 960.
Zhao, Y., C. Gao, J. Liu, and S. Dong, 2019 A self-perturbed pseudo-random sequence generator based on hyperchaos. Chaos, Solitons & Fractals: X **4**: 100023.
Zhou, Y., Z. Hua, C.-M. Pun, and C. P. Chen, 2014 Cascade chaotic system with applications. IEEE transactions on cybernetics **45**: 2001–2012.

***How to cite this article:*** Agarwal, S. *Designing a Pseudo-Random Bit Generator Using Generalized Cascade Fractal Function*. Chaos Theory and Applications, 3(1), 11-19, 2021.