CHAOS
Theory and Applications
in Applied Sciences and Engineering

e-ISSN: 2687-4539
RESEARCH ARTICLE
Volume:2 Number:2 Pages:58-68

A Chaos Based Pseudo-Random Bit Generator Using Multiple Digits Comparison

Lazaros Moysis *,[1], **Aleksandra Tutueva** †,[2], **Christos Volos** *,[3] **and Denis Butusov** **,[4]

*Laboratory of Nonlinear Systems, Circuits & Complexity (LaNSCom), Physics Department, Aristotle University of Thessaloniki, Thessaloniki, Greece,
†Department of Computer-Aided Design, Saint Petersburg Electrotechnical University "LETI", 5, Professora Popova st., 197376 Saint Petersburg, Russia,
**Youth Research Institute, Saint-Petersburg Electrotechnical University 'LETI', 5, Professora Popova st., 197376 Saint Petersburg, Russia.

ABSTRACT This work presents a simple method of designing pseudo-random bit generator by generating multiple bits per iteration from the decimal part of a chaotic map. This is done by extracting the decimal part of the state in each iteration and comparing each digit separately to a threshold value. This way, more than one bits can be generated in each iteration, in contrast to most well-known generators based on discrete-time chaotic maps, which generate only one bit. The method is tested on multiple maps and it is seen that for most, around 8 digits can be extracted each time, so that the final bitstream passes all NIST tests. The generated PRBG is then studied through a simple image encryption application, by combining shuffling and the XOR operation.

KEYWORDS
Chaos,
Pseudo-random bit generator (PRBG),
1D chaotic maps,
Encryption,
Security analysis.

INTRODUCTION

It has been well documented that chaotic systems are suitable in a plethora of applications related to encryption, like text, image and video encryption, watermarking, user authentication, secure communications and many more, see for example Strogatz (2018); Stojanovski and Kocarev (2001); Roy *et al.* (2017); Wang *et al.* (2017, 2019a); Sheng and Wu (2012); Kang *et al.* (2014); Zhang *et al.* (2004); Jamal *et al.* (2013); Boutayeb *et al.* (2002); Özkaynak (2018) and the works cited therein.

The main reason for which chaotic systems have found use in such a plethora of applications is the combination of complexity, with a deterministic nature. Chaotic systems are deterministic systems described by differential/difference equations, that can generate extremely complex behavior, using very few terms. Thus, a user can create very fast simulations and generate values that possess random properties, without relying on stochastic processes or noise from a natural source. Moreover, the sensitivity of chaotic systems to initial conditions can also help in building secure designs that are practically impossible for an outsider to replicate or interpret, without knowing the key parameters of the system, making such designs resistant to brute force attacks Alvarez and Li (2006).

Among the most prominent applications of chaotic systems are pseudo-random bit generators (PRBGs) Stojanovski and Kocarev (2001); Wang *et al.* (2017); Akgül *et al.* (2019); Patidar *et al.* (2009); Ahmad *et al.* (2018); Zhao *et al.* (2019); Datcu *et al.* (2020). In this application, the values of a chaotic system are used to generate a stream of bits, using a simple rule. The aim is to have the final bitstream possess properties similar to a random series, which is tested using the National Institute of Standards and Technology (NIST) statistical test suite Rukhin *et al.* (2001). There are numerous techniques to generate the bitstream from the values of a chaotic map. A common technique is to compare the value of the map in each iteration to a threshold value, and producing a zero or one depending on the result Irfan *et al.* (2020); Ursulean (2004); Volos *et al.* (2013a); Demir and Ergün (2018); El-Naggary and Moussa (2020); Wang and Xie (2012); Kang *et al.* (2014).

Based on this technique, a question that arises is whether we can generate more that one bits per iteration, by comparing each digit in the decimal part of the chaotic map to a threshold value. Also, in case this is possible for a chaotic map, we need to determine what is the maximum number of bits that can be extracted in each iteration. Hence, in this work, we try to give an answer to the above questions, by considering a collection of one-dimensional chaotic maps, and comparing their decimal parts digit by digit, to the threshold value of 5. It is seen that for most maps, around 8 digits can be extracted in each iteration using this technique, so

Manuscript received: 22 June 2020,
Revised: 15 July 2020,
Accepted: 16 July 2020.

[1] lmousis@physics.auth.gr (**Corresponding author**)
[2] avtutueva@etu.ru
[3] volos@physics.auth.gr
[4] dnbutusov@etu.ru

that the resulting bitstream passes all NIST tests.

Finally, we apply the generated bitstreams to the problem of image encryption Gao and Chen (2008); Shaukat *et al.* (2020); Pak and Huang (2017); Zhou *et al.* (2014); Han (2019); Khanzadi *et al.* (2014); Özkaynak (2018); Zhu *et al.* (2018); Moafimadani *et al.* (2019); Kang *et al.* (2014); Ge *et al.* (2019). The encryption consists of two simple steps. The rows and columns of the source image are first shuffled using the values from a PRBG to generate the shuffling matrices. Then, the image is reshaped in vector form and is combined with another bitstream from a PRBG using the *XOR* operator. A series of tests are then performed on the original, shuffled and encrypted images to test how well do the resulting images mask their information. Two test images are considered, a black and white QR code, and a grayscale Lena image.

The rest of the paper is structured as follows: Section 2 presents the proposed PRBG technique. In Section 3 the chaotic maps tested are presented. In Section 4 the statistical test results are shown. In Section 5 the application to image encryption is considered. Section 6 concludes the work.

PSEUDO-RANDOM BIT GENERATION

In the proposed design we will consider several one-dimensional chaotic maps that give a mapping to the interval $[0,1]$. To design a PRBG from each map, we will extract the last $\mathcal{F}$ decimal digits of the value x_i in each iteration, and compare each digit $d_1,...,d_\mathcal{F}$ to the threshold value of 5. This value is chosen since it is the mean value of the numbers zero to nine. Depending on the result, a 0 or 1 will be generated. The individual bits b_i are then combined into a row vector $\mathbf{B}_i$, which is then appended to a single bitstream $\mathcal{B}$. This way, $\mathcal{F}$ bits are generated per iteration, until the desired bitstream length is reached. Of course, for a bitstream of length ℓ, only $\frac{\ell}{\mathcal{F}}+1$ iterations are required. The procedure is outlined in Fig. 1.

Notice, that since there are five digits smaller than 5 (0,1,2,3,4) and four digits greater than it (6,7,8,9), the threshold comparison assigns the bit value of 1 if $d_i \geq 5$ and 0 if $d_i < 5$. This way, the equality is assigned to digits that are greater than 5, and the comparison is balanced.

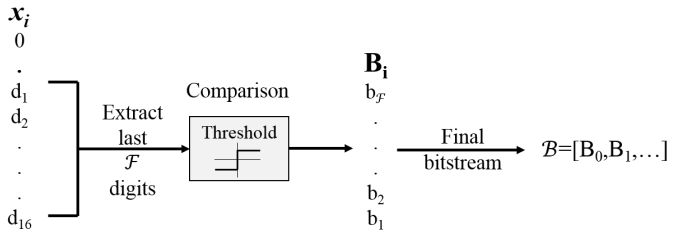


Figure 1 Procedure for bit generation.

Based on this technique, assuming a double-precision 16-digit accuracy, and also that the map used takes values in the range of $[-1,1]$, leaving 16 digits available in the decimal part, a maximum of 16 bits can be generated in each iteration. The question that rises though is if the generated bitstream is random enough. To verify so, starting from $\mathcal{F}=16$ digits, a set of 50×10^6 bistreams is generated and tested through the NIST statistical test suite. This toolbox consists of 15 tests that can verify the randomness of a sequence. For each test, a P-value is returned, and if it is higher than a chosen significance level a, then the test is successful. The significance value here is taken as the default $a=0.01$. The value $a=0.01$ means that for 100 sequences, only one sequence is expected to be rejected. Thus, a P-value≥ 0.01 means that the sequence can be considered random with a confidence of 99% Rukhin *et al.* (2001).

If the bitstream fails to pass any of the 15 tests provided, that is, if the returned P-value in any test is lower than $a=0.01$, then the first decimal digit of x_i is discarded in the bit generation, and the bitstream is generated again using the least significant $\hat{\mathcal{F}}=\mathcal{F}-1$ digits. This process continues until the bitstream passes all the tests.

THE UTILIZED MAPS

To test how this method works with different maps, we considered several 1D chaotic maps that all to $[0,1]$ or $[-1,1]$ and can achieve different maximum Lyapunov exponent values. The maps considered are:

The Logistic map May (1976); Phatak and Rao (1995); Pareek *et al.* (2005); Patidar *et al.* (2009); Wang *et al.* (2016); Irfan *et al.* (2020); Han (2019) is given by:

$$x_i = kx_{i-1}(1-x_{i-1}) \tag{1}$$

This is a one parameter map, with a full mapping to $[0,1]$ for $k=4$. Its bifurcation diagram is shown in Fig. 2 and the diagram of Lyapunov exponent (LE) in Fig 3. It is seen that the highest value of LE is around 0.7.

The Sine map Belazi and El-Latif (2017); Pareek *et al.* (2005); Hua and Zhou (2016) is given by:

$$x_i = k\sin(\pi x_{i-1}) \tag{2}$$

This is a single parameter map with a full mapping to $[0,1]$ for $k=1$. Its bifurcation diagram and diagram of LE is shown in Fig. 4 and 5. The map can achieve higher values of LE, although at these parameter values, the mapping is outside the interval $[0,1]$.

The Renyi map Alzaidi *et al.* (2018); Addabbo *et al.* (2007) is:

$$x_i = \mod(kx_{i-1},1) \tag{3}$$

For $k>1$ the map is chaotic and maps to $[0,1]$. Its bifurcation diagram and diagram of LE is shown in Fig. 6 and 7. For $k=10$ the LE can reach a value above 2.

The Chebysev map Liu *et al.* (2016); Stoyanov and Kordov (2015); Huang (2012); Stoyanov (2013) is:

$$x_i = \cos(k\arccos x_{i-1}) \tag{4}$$

For $k>2$ it maps to $[0,1]$. Its bifurcation diagram and diagram of LE is shown in Fig. 8 and 9. For $k=10$ the LE can reach a value above 2.

The Cubic map Pareek *et al.* (2005); Sarmah and Das (2014) is:

$$x_i = kx_{i-1}(1-x_{i-1}^2) \tag{5}$$

For $k=2.6$ it gives a full mapping on $[0,1]$. Its bifurcation diagram and diagram of LE is shown in Fig. 10 and 11. The LE diagram is similar to that of the classic logistic map.

The Cubic-Logistic map Elabady *et al.* (2014) is:

$$x_i = kx_{i-1}(1-x_{i-1})(2+x_{i-1}) \tag{6}$$

Its bifurcation diagram and diagram of LE in Figs. 12 and 13 are very similar to that of the logistic map.

The Logistic-May map Ali and Khan (2019) is:

$$x_i = \mod(x_{i-1}e^{(k+9)(1-x_{i-1})} - (k+5)x_{i-1}(1-x_{i-1}),1) \tag{7}$$

This map gives a constant mapping to $[0,1]$ and its LE can achieve very high values, as seen in Figs. 14 and 15

The Coupled-Sine map Irani *et al.* (2019); Liu *et al.* (2020):

$$x_i = a|\sin(\beta^3\pi x_{i-1})| + (1-a)(1-|\sin(k^3\pi x_{i-1})(1-x_{i-1})|) \tag{8}$$

This map has three parameters and can give a mapping to $[0,1]$ for most values. The bifurcation diagram and diagram of LE is shown in Figs. 16 and 17 with respect to k, for $a = 0.9$, $b = 9$. The LE is high, at around 7.

Finally, we propose a modification of the Sine map, the following Sine-Sinh-Sine map:

$$x_i = k\sin(\pi\sinh(\pi\sin(\pi x_{i-1}))) \tag{9}$$

The bifurcation diagram and diagram of LE is shown in Figs. 18 and 19. The map gives a full mapping on $[-k,k]$, and its LE can achieve high values.

Note that in all of the above simulations, the LE is computed using the method in Bovy (2004).

NIST RESULTS

This package consists of 15 tests that test the randomness of a time series and is the most common tool in the analysis of pseudorandom bit generators.

The test results for each generator based on each map are shown in Table 1. The parameter values in each case are chosen close to the value that gives a full mapping to $[0,1]$. Yet, instead of choosing an integer value for the parameter, a small random value $\mathrm{p} \in [0,1]$ is added using the `rand` command in Matlab. This way, the parameter has a full 16-digit representation, which increases its randomness. From the simulation results, we see that most maps can give a bitstream that passes all tests with around eight to twelve digits extracted in each iteration. This is an indication that for most maps, the choice of eight bits per iteration is safe for random bit generation. So this method is not limited to any specific map, but can be used successfully with different maps as the source for generating the bits, assuming that no more than $\mathcal{F} = 8$ digits are extracted.

Of course, for some maps, some parameter values may yield even more bits per iteration. So in some cases, it may be possible to generate a bitstream that passes all tests with more bits extracted per iteration. For example, the Renyi map can pass almost all tests for up to 13 digits extracted, with only a couple of the subcases of the NonOverlappingTemplates test failing. So in all cases, choosing a lower number of extracted digits as shown in Table 1 is the best choice.

Note also, that it is possible to consider parameter values that give a mapping outside the interval $[-1,1]$. Yet, since in this case there will be less digits in the decimal part of the value x_i, the procedure for digit extraction must be adjusted for that fact. In addition, the number of digits extracted for a successful bitstream will probably be lower.

One exception is the Logistic-May map, that despite having a large LE, fails to produce a NIST sequence that passes the NIST tests. This could be attributed to round-off errors due to the exponential term that appears inside the modulo operator.

Finally, regarding simulation time, depending on the map used, the algorithm takes on average $0.42 - 0.5$ seconds to generate a bitstream of length 10^6 with 8 bits extracted per iteration, tested on an Intel®Core™i5-3337U Processor, with 6bg of RAM, using Matlab 2018b. This was computed by taking the average time of 100 simulations, for each different map. The resulting mean execution time for all maps falls between this interval.

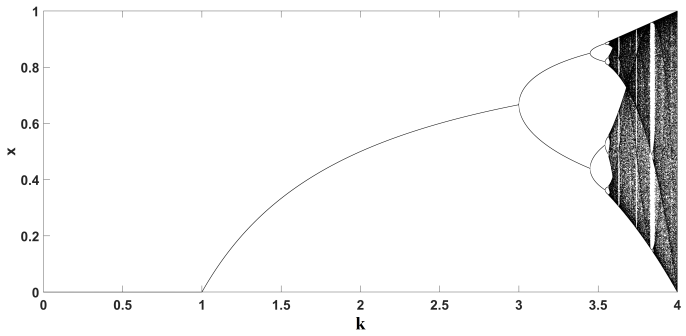


Figure 2 Bifurcation diagram of the Logistic map.

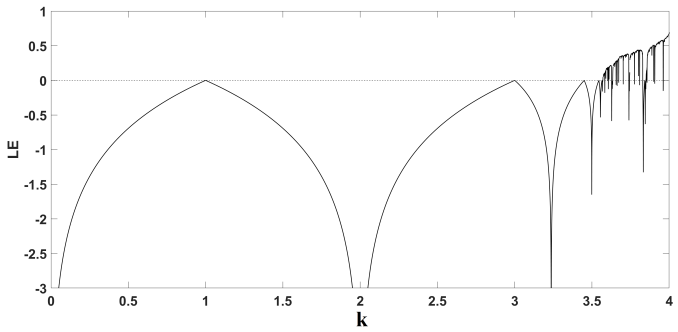


Figure 3 Diagram of Lyapunov exponent of the Logistic map.

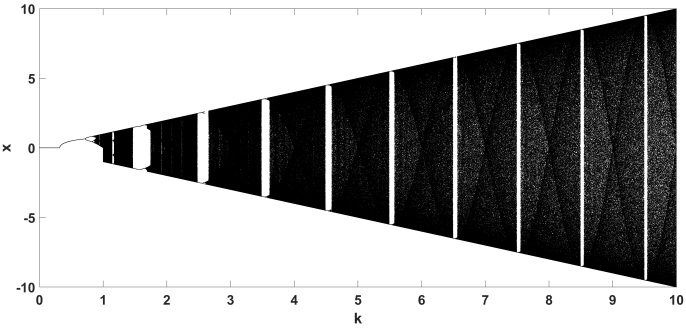


Figure 4 Bifurcation diagram of the Sine map.

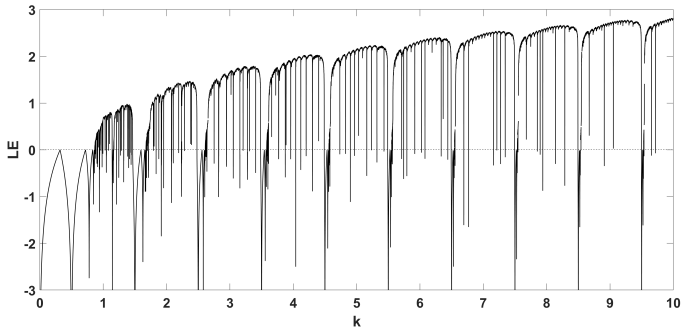


Figure 5 Diagram of Lyapunov exponent of the Sine map.

Table 1 Digit extraction for successful PRBG and test results for a bistream of length $50 \cdot 10^6$.

Test \ Map	Logistic	Sine	Renyi	Chebysev	Cubic	Cubic-logistic	Logistic-May	Coupled-Sine	Sine-Sinh-Sine
Parameter Values	$r = 3.9 + 0.1 \cdot \mathrm{p}$	$k = 0.9 + 0.1 \cdot \mathrm{p}$	$c = 9.9 + 0.1 \cdot \mathrm{p}$	$c = 9.9 + 0.1 \cdot \mathrm{p}$	$k = 2.58 + 0.01 \cdot \mathrm{p}$	$k = 1.57 + 0.01 \cdot \mathrm{p}$	$k = 9 + \mathrm{p}$	$k = 9 + \mathrm{p}$	$k = 0.9 + 0.1 \cdot \mathrm{p}$
Digits Extracted	12	11	8	11	11	9	0	10	11
Frequency	0.616305	0.739918	0.236810	0.574903	0.657933	0.350485	fail	0.851383	0.616305
BlockFrequency	0.030806	0.935716	0.058984	0.122325	0.122325	0.699313	fail	0.851383	0.699313
CumulativeSums	0.816537	0.455937	0.319084	0.534146	0.455937	0.419021	fail	0.616305	0.350485
Runs	0.262249	0.455937	0.739918	0.262249	0.699313	0.319084	fail	0.911413	0.350485
LongestRun	0.045675	0.319084	0.153763	0.419021	0.935716	0.262249	fail	0.494392	0.020548
Rank	0.051942	0.616305	0.319084	0.289667	0.350485	0.262249	fail	0.383827	0.574903
FFT	0.699313	0.236810	0.616305	0.289667	0.137282	0.171867	fail	0.108791	0.289667
NonOverlappingTemplate	0.616305	0.319084	0.739918	0.455937	0.236810	0.494392	fail	0.494392	0.851383
OverlappingTemplate	0.045675	0.534146	0.058984	0.383827	0.236810	0.350485	fail	0.657933	0.816537
Universal	0.319084	0.816537	0.616305	0.816537	0.455937	0.657933	fail	0.213309	0.935716
ApproximateEntropy	0.739918	0.494392	0.699313	0.574903	0.779188	0.085587	fail	0.851383	0.657933
RandomExcursions	0.500934	0.534146	0.468595	0.949602	0.311542	0.324180	fail	0.074177	0.253551
RandomExcursionsVariant	0.074177	0.437274	0.253551	0.976060	0.242986	0.772760	fail	0.232760	0.671779
Serial	0.419021	0.816537	0.383827	0.657933	0.971699	0.289667	fail	0.494392	0.616305
LinearComplexity	0.350485	0.574903	0.002043	0.534146	0.494392	0.040108	fail	0.085587	0.779188

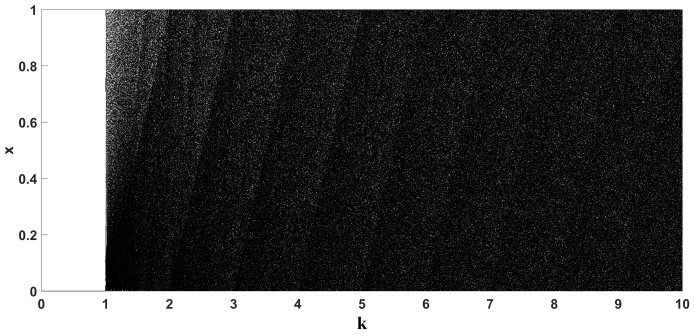


Figure 6 Bifurcation diagram of the Renyi map.

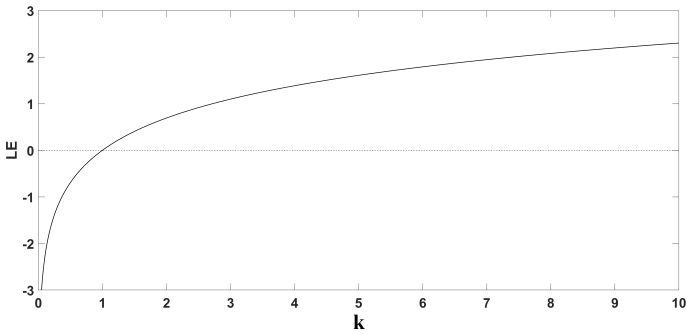


Figure 7 Diagram of Lyapunov exponent of the Renyi map.

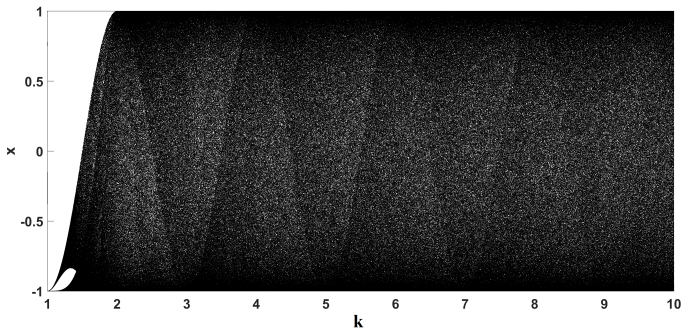


Figure 8 Bifurcation diagram of the Chebysev map.

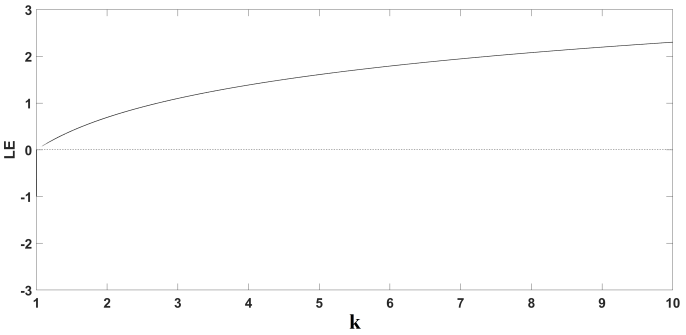


Figure 9 Diagram of Lyapunov exponent of the Chebysev map.

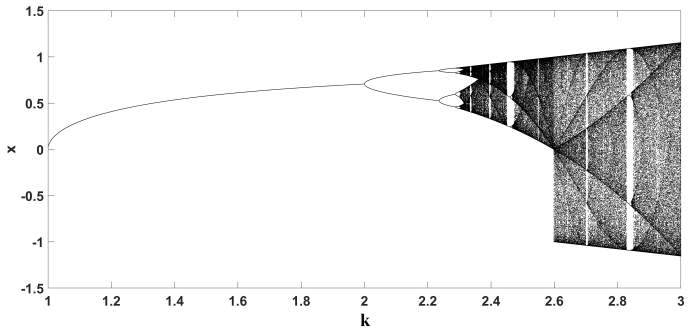


Figure 10 Bifurcation diagram of the Cubic map.

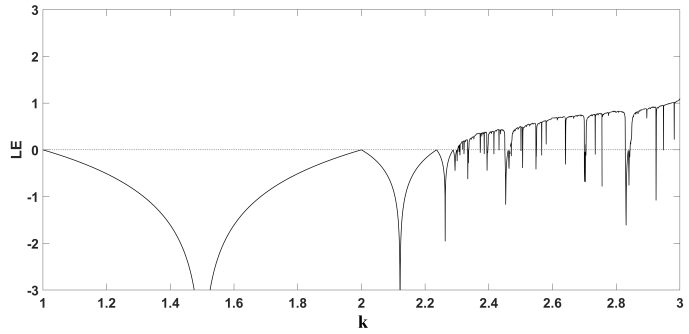


Figure 11 Diagram of Lyapunov exponent of the Cubic map.

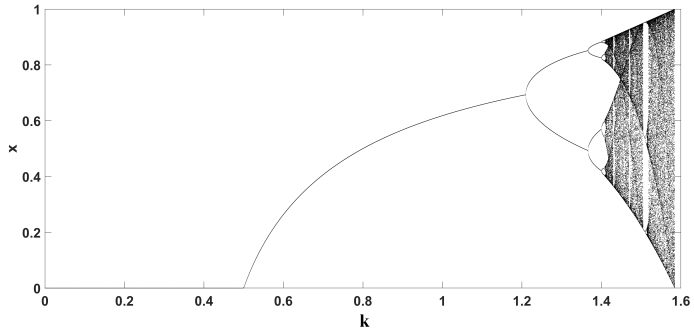


Figure 12 Bifurcation diagram of the Cubic-Logistic map.

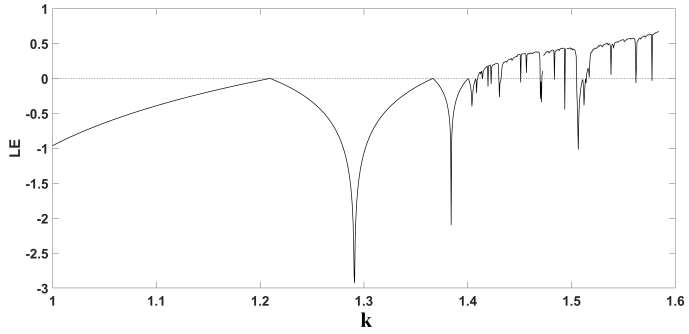


Figure 13 Diagram of Lyapunov exponent of the Cubic-Logistic map.

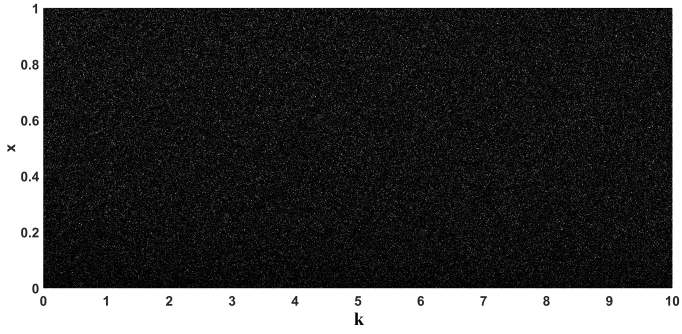


Figure 14 Bifurcation diagram of the Logistic-May map.

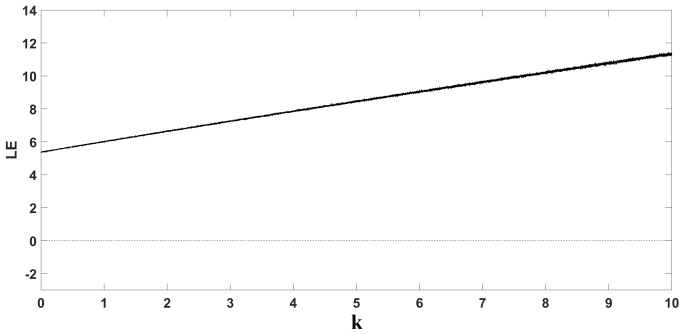


Figure 15 Diagram of Lyapunov exponent of the Logistic-May map.

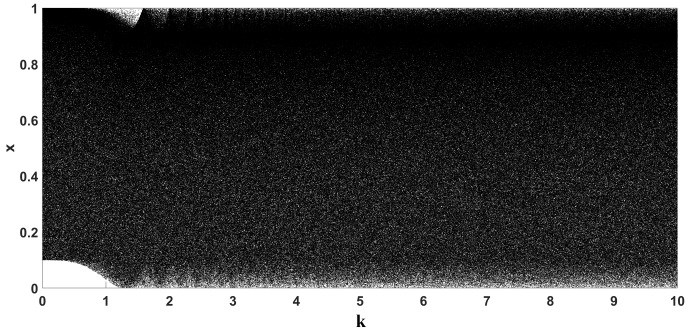


Figure 16 Bifurcation diagram of the Coupled Sine map.

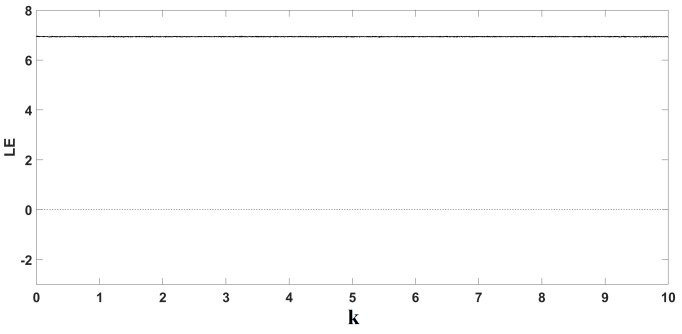


Figure 17 Diagram of Lyapunov exponent of the Coupled Sine map.

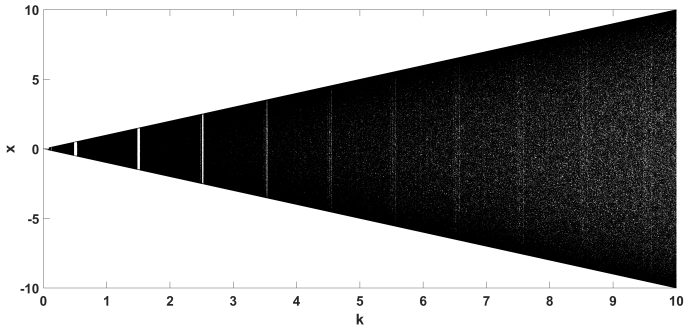


Figure 18 Bifurcation diagram of the Sine-Sinh-Sine map.

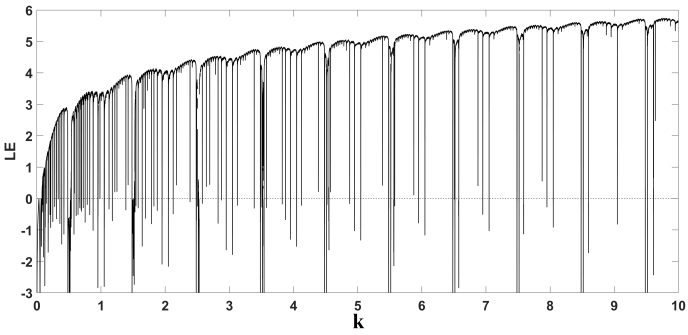


Figure 19 Diagram of Lyapunov exponent of the Sine-Sinh-Sine map.

APPLICATION TO IMAGE ENCRYPTION

In this section, we apply the designed PRBG to the problem of image encryption. The method that will be followed consists of two simple steps. In the first step, a shuffle algorithm will be applied to rearrange the rows and columns of the source image. This will be done to reduce the correlation between adjacent pixels in the original image. The shuffling algorithm uses an approach similar to the one in Gao and Chen (2008). Then, the image will be encrypted using the XOR operator, which is very common in image encryption Gao and Chen (2008); Volos *et al.* (2013b); Özkaynak (2018); Zhu *et al.* (2018); Moafimadani *et al.* (2019); Kang *et al.* (2014); Ge *et al.* (2019). In both steps, the proposed PRBGs will be utilised.

Given a $M \times N$ source image in black and white, represented by a matrix $\mathcal{I}$ with entries of 0 or 255, representing the colour of each pixel, the process is outlined in the following steps.

Shuffling

First, a shuffling on the original image is performed. To do so, we consider the Sine-Sinh-Sine map (9), with parameter $k = M$. Then, we iterate the map and compute $p_j = \lfloor \mod(x_i, M) \rfloor$, until M discrete values are generated in the interval $[0, M-1]$. So if a value is generated twice, it is discarded the second time. The set $\{p_1 + 1, ..., p_M + 1\}$ represents the rearrangement of the rows of the original image. So the 1st row is moved to the row $p_1 + 1$, the 2nd row is moved to $p_2 + 1$ and so on. This procedure can be represented by left multiplication $L\mathcal{I}$, where L is an invertible matrix where each row has zero entries and 1 in the position p_j.

Similarly, setting $k = N$, and using the value of the last iteration of the Sine-Sinh-Sine map as the initial condition x_0, the map (9) is iterated again, computing $q_j = \lfloor \mod(x_i, N) \rfloor$ in each iteration, until N discrete values are generated in the interval $[0, N-1]$. The resulting set $\{q_1 + 1, ..., q_N + 1\}$ represents the rearrangement of the columns of the original image. This procedure can also be represented by right multiplication $\mathcal{I}R$, where R is an invertible matrix where each column has zero entries and 1 in the position q_j.

Overall, the resulting shuffled image is $\mathcal{G} = L\mathcal{I}R$. Note that the shuffling procedure can be repeated more times if desired. The combined shuffling of m times can be written in matrix form as $\mathcal{G} = L_m \cdots L_1 \mathcal{I} R_1 \cdots R_m$.

Encryption

After the image is shuffled, its columns are stacked to one another, so that the image $\mathcal{G}$ is reshaped to a vector of length $M \times N$. To easily convert the vector to binary form, we replace the values of 255 with 1. Then, the resulting vector $\mathcal{V}$ is combined with a bitstream X of equal length generated with the technique proposed in the previous section, using the XOR operation, yielding the encrypted vector $\mathcal{E} = \mathcal{V} \oplus X$. This vector can now be safely transmitted in a secure communications scheme.

Decryption

The decryption process follows the reverse procedure of the previous steps. First, we perform $\hat{\mathcal{E}} = \mathcal{E} \oplus X$ on the encrytped image to obtain the original vector. We then replace 1 with 255. Then reshape the vector to a $M \times N$ matrix $\hat{\mathcal{G}}$. The original image can then be obtained as $\hat{\mathcal{I}} = L^{-1} \hat{\mathcal{G}} R^{-1}$.

Note that for grayscale or RGB images, the same procedure can be repeated with only minor adjustments. For a grayscale image, in the encryption part, the resulting $M \times N$ vector with values in $[0, 255]$ can be transformed to a binary vector first, with a resulting vector of length $8 \times M \times N$. For an RBG image, the same procedure can be performed for each of the three submatrices of each colour.

Key Space

As for the key space, the procedure utilizes the Sine-Sinh-Sine map with fixed parameter, and also any one of the proposed 1D maps to generate the bitstream. Most of the proposed maps have only only one parameter, so the overall key space consists of three parameters, the initial value of the Sine-Sinh-Sine map, and the parameter value and initial value of the map used in the PRBG. So, assuming a 16-digit accuracy, an upper bound for the key space is $10^{48} = (10^3)^{16} \approx (2^{10})^{16} = 2^{160}$, which is higher that the 2^{100} required to resist brute force attacks Alvarez and Li (2006).

Note also that in this work, we chose to use well known maps with very simple dynamics, like the Logistics map and the Sine map, which have only one parameter. Using different one-dimensional maps with more parameters will certainly increase the key space.

Simulation Analysis

For the simulation of the above procedure, we consider a 200×200 black and white QR code image. The code provides a link to the Wikipedia entry for QR codes. For the shuffling and XOR operations, the Sine-Sinh-Sine map (9) is used. After successful encryption, we perform a series of tests to the original, shuffled and encrypted images, to test the performance of the encryption.

Figure 20 shows the original, shuffled and encrypted images. In Fig. 21, the histograms of the original and encrypted images are shown. It is seen that the encrypted image has a more uniform distribution between the 1s and 0s, which is an indication of randomness. The shuffled image has a histogram identical to the original image, so this is not shown.

To study the correlation between adjacent pixels, we compute the correlation coefficients of the original, shuffled, and encrypted images, for horizontal, vertical, and diagonal pixels. In unencrypted images, the correlation between adjacent pixels is high, so the encrypted images should have a correlation coefficient close to zero, which is an indication of randomness. The results are shown in Table 2. Indeed, the shuffled and encrypted images have correlation close to zero, with the encrypted image having the lowest values among the three, as expected. The correlation coefficients are computed using the following formulas

$$E(x) = \frac{1}{L}\sum_{i=1}^{L} x_i \tag{10}$$

$$D(x) = \frac{1}{L}\sum_{i=1}^{L}(x - E(x))^2 \tag{11}$$

$$cov(x,y) = \frac{1}{L}\sum_{i=1}^{L}(x_i - E(x))(y_i - E(y)) \tag{12}$$

$$\gamma(x,y) = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{13}$$

where x, y are the gray values of two adjacent pixels and L the number of adjacent pixels considered.

Finally, the information entropy is computed. For an image $\mathcal{S}$, it is given by

$$H(\mathcal{S}) = -\sum_{i=0}^{2^n-1} p(s_i)\log_2 p(s_i) \tag{14}$$

where $p(s_i)$ is the possibility of occurence of gray level i, $s_i = i$, for $i = 0, ..., 2^n$. In the grayscale case, since there are 256 values for the pixels, the information entropy of a random image should be close to $\log_2 256 = 8$, Zhu *et al.* (2018); Moafimadani *et al.* (2019). Here, since the images tested are binary, a random image should have information entropy close to 1. The information entropy of the original and shuffled images is 0.99833, while for the encrypted image is 0.99998, so it is closer to 1.

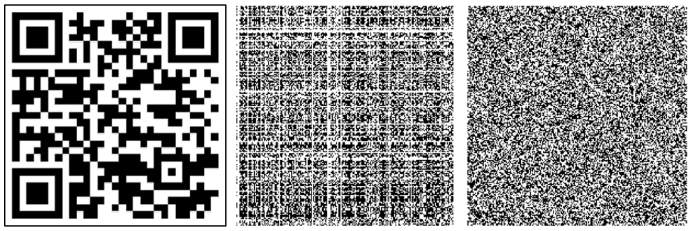
As an additional simulation, we also consider a 256×256 grayscale image of Lena. As mentioned in the Decryption step, the procedure here is exactly the same, with the additional step of transforming the grayscale values in $[0, 255]$ to binary first, and then using XOR. Of course, the step of replacing 255 to 1 is discarded. After encryption, the vector is reshaped appropriately, to obtain the encrypted and decrypted images. The three images are shown in Fig. 22.

The histogram for the original and encrypted images is shown in Fig. 23. As with the QR code, it is seen that the encrypted image has a uniform histogram.

For the correlation analysis, Table 4 shows the correlation coefficients between adjacent pixels.The results are comparable to other works Nosrati *et al.* (2017); Belazi *et al.* (2016); Wang *et al.* (2015, 2019b). Figs. 24, 25, 26 also show the correlation plots between horizontal, vertical, and diagonal adjacent pixels, for a set of 10000 randomly chosen pixels. Indeed, the encrypted image has the most scattered values among adjacent pixels.

Finaly, the information entropy of the original and shuffled images is 7.4467, while for the encrypted image is 7.9968, so it is closer to 8, which means the encrypted image is more random. Table 3 shows the entropy of the encrypted image for different methods.

Overall, it is seen that the proposed PRBGs can easily be applied to the problem of encryption, both in the shuffling and encryption steps, yielding a masked image with good randomness characteristics.



(a) **(b)** **(c)**

Figure 20 (a) Original source image of the QR code, (b) Shuffled image and (c) Encrypted image.

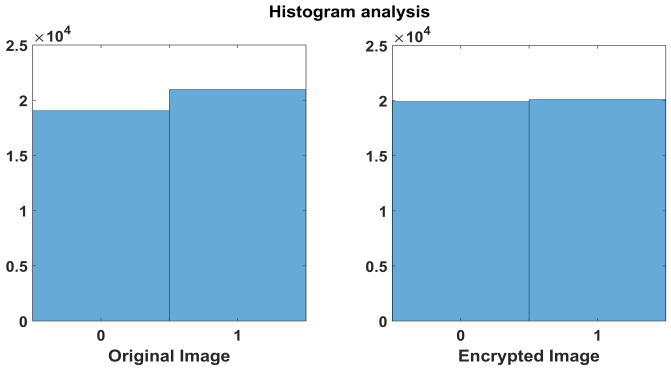


Figure 21 Histogram of the original and encrypted QR image.

Table 2 Correlation coefficients for the QR image encryption.

	Original Image	Shuffled	Encrypted
Horizontal	0.86144	0.12792	0.0036
Vertical	0.85440	0.12305	0.00319
Diagonal	0.73243	0.01335	0.00637

Table 3 Information Entropy for the Lena image.

Original Image	Encrypted
7.4467	7.9968
Nosrati *et al.* (2017)	7.9971
Belazi *et al.* (2016)	7.9963
Wang *et al.* (2015)	7.9970
Wang *et al.* (2019b)	7.9971

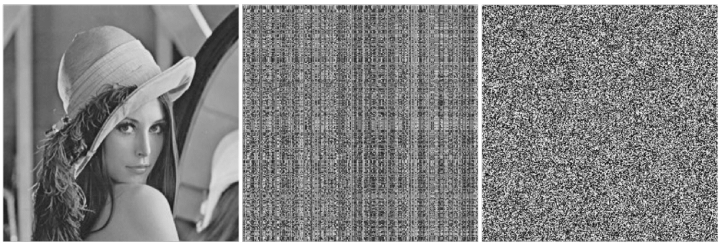


Figure 22 (a) Original Lena source image, (b) Shuffled image and (c) Encrypted image.

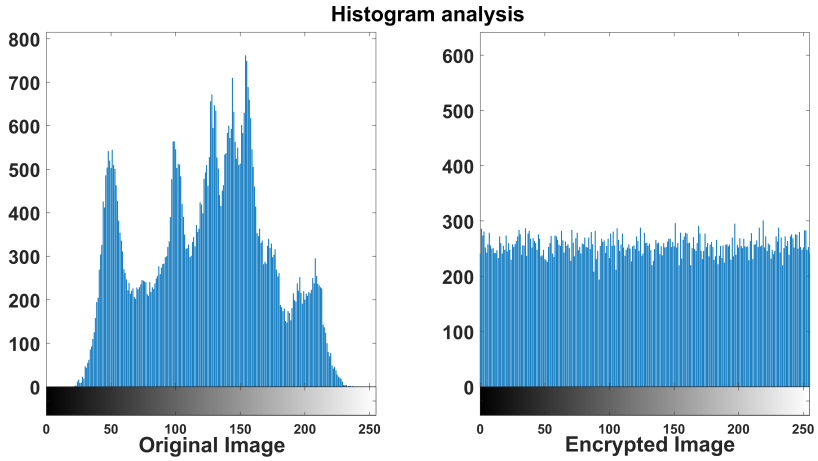


Figure 23 Histogram of the original and encrypted Lena image.

Table 4 Correlation coefficients for the Lena image encryption.

	Original Image	Shuffled	Encrypted
Horizontal	0.9408	0.0835	0.0029
Vertical	0.9695	0.2420	-0.0104
Diagonal	0.9153	0.0053	-0.0049

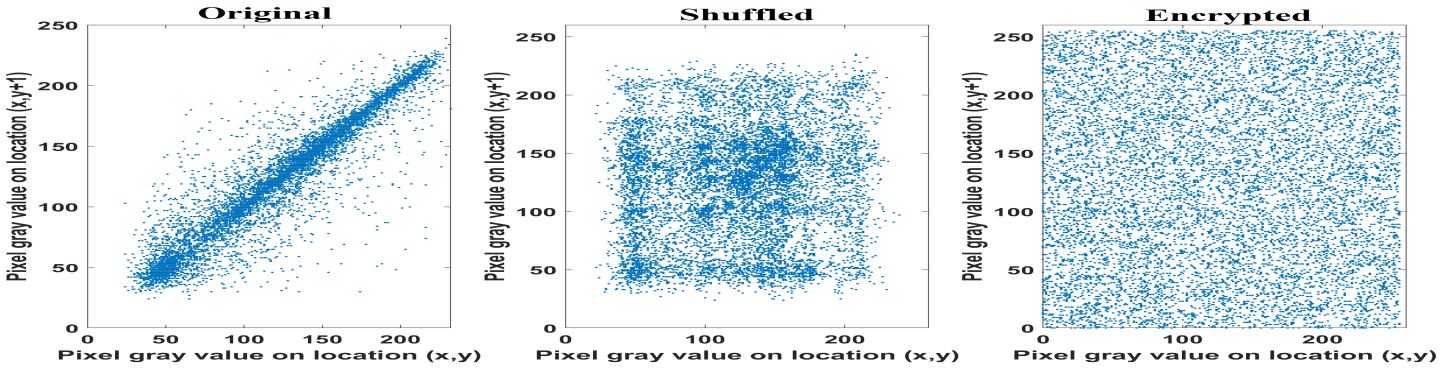


Figure 24 Correlation between horizontal adjacent pixels for the original, shuffled, and encrypted Lena image.

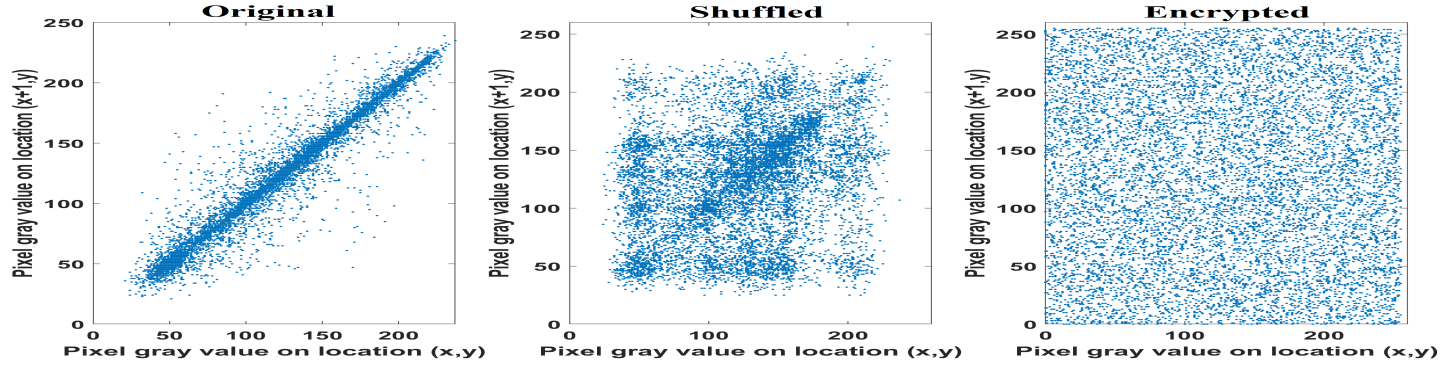


Figure 25 Correlation between vertical adjacent pixels for the original, shuffled, and encrypted Lena image.

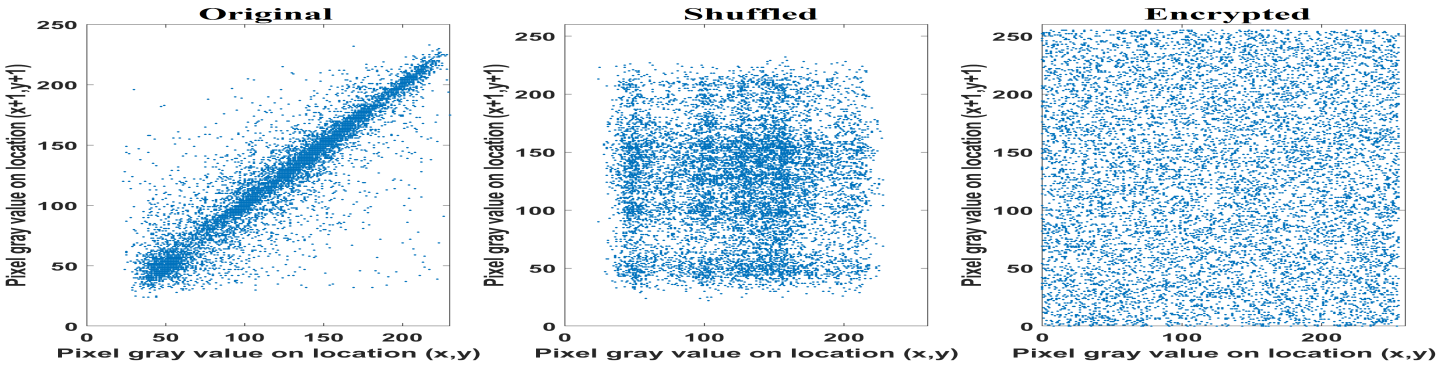


Figure 26 Correlation between diagonal adjacent pixels for the original, shuffled, and encrypted Lena image.

CONCLUSIONS

In this work, the extraction of multiple bits per iteration of a chaotic map has been studied, by comparing each digit in the decimal part to a threshold value. For most maps tested, around 8 digits can be extracted in each iteration. The application to image encryption was then considered.

Future works will consider the application of this technique to continuous chaotic systems. Here, the choice of different numerical integration method and also the step size will surely effect the number of digits that can be extracted. Also, the combination of multiple maps to generate more bits per iteration is an interesting topic for future studies. Lastly, one of our future goals would be to implement the proposed PRBG in an FPGA setup Volos (2013); Stoyanov and Ivanova (2019); Ketthong and San-Um (2014); Akgul *et al.* (2017), and to also study how fast the bitstream can be generated.

Acknowledgments

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Reinforcement of Postdoctoral Researchers - 2nd Cycle" (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

The authors would like to thank the anonymous reviewers for their insightful comments that helped improve the final work.

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

LITERATURE CITED

Addabbo, T., M. Alioto, A. Fort, A. Pasini, S. Rocchi, *et al.*, 2007 A class of maximum-period nonlinear congruential generators derived from the rényi chaotic map. IEEE Transactions on Circuits and Systems I: Regular Papers **54**: 816–828.

Ahmad, M., M. Doja, and M. S. Beg, 2018 A new chaotic map based secure and efficient pseudo-random bit sequence generation. In *International Symposium on Security in Computing and Communication*, pp. 543–553, Springer.

Akgül, A., C. Arslan, and B. Arıcıoğlu, 2019 Design of an interface for random number generators based on integer and fractional order chaotic systems. Chaos Theory and Applications **1**: 1–18.

Akgul, A., C. Li, and I. Pehlivan, 2017 Amplitude control analysis of a four-wing chaotic attractor, its electronic circuit designs and microcontroller-based random number generator. Journal of Circuits, Systems and Computers **26**: 1750190.

Ali, K. M. and M. Khan, 2019 Application based construction and optimization of substitution boxes over 2d mixed chaotic maps. International Journal of Theoretical Physics **58**: 3091–3117.

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

Alzaidi, A. A., M. Ahmad, M. N. Doja, E. Al Solami, and M. S. Beg, 2018 A new 1d chaotic map and β-hill climbing for generating substitution-boxes. IEEE Access **6**: 55405–55418.

Belazi, A., A. A. Abd El-Latif, and S. Belghith, 2016 A novel image encryption scheme based on substitution-permutation network and chaos. Signal Processing **128**: 155–170.

Belazi, A. and A. A. A. El-Latif, 2017 A simple yet efficient s-box method based on chaotic sine map. Optik **130**: 1438–1444.

Boutayeb, M., M. Darouach, and H. Rafaralahy, 2002 Generalized state-space observers for chaotic synchronization and secure communication. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications **49**: 345–349.

Bovy, J., 2004 Lyapunov exponents and strange attractors in discrete and continuous dynamical systems. Theoretica Phys. Project, Catholic Univ. Leuven, Flanders, Belgium, Tech. Rep **9**: 1–19.

Datcu, O., C. Macovei, and R. Hobincu, 2020 Chaos based cryptographic pseudo-random number generator template with dynamic state change. Applied Sciences **10**: 451.

Demir, K. and S. Ergün, 2018 An analysis of deterministic chaos as an entropy source for random number generators. Entropy **20**: 957.

El-Naggary, A. I. and K. H. Moussa, 2020 Pseudorandom bit generator based on chaotic parameter hopping chaos. International Journal of Engineering and Technology **12**: 1136–1143.

Elabady, N., H. Abdalkader, M. Moussa, and S. F. Sabbeh, 2014 Image encryption based on new one-dimensional chaotic map. In *2014 International Conference on Engineering and Technology (ICET)*, pp. 1–6, IEEE.

Gao, T. and Z. Chen, 2008 A new image encryption algorithm based on hyper-chaos. Physics Letters A **372**: 394–400.

Ge, R., G. Yang, J. Wu, Y. Chen, G. Coatrieux, *et al.*, 2019 A novel chaos-based symmetric image encryption using bit-pair level process. IEEE Access **7**: 99470–99480.

Han, C., 2019 An image encryption algorithm based on modified logistic chaotic map. Optik **181**: 779–785.

Hua, Z. and Y. Zhou, 2016 Image encryption using 2d logistic-adjusted-sine map. Information Sciences **339**: 237–253.

Huang, X., 2012 Image encryption algorithm using chaotic chebyshev generator. Nonlinear Dynamics **67**: 2411–2417.

Irani, B. Y., P. Ayubi, F. A. Jabalkandi, M. Y. Valandar, and M. J. Barani, 2019 Digital image scrambling based on a new one-dimensional coupled sine map. Nonlinear Dynamics **97**: 2693–2721.

Irfan, M., A. Ali, M. A. Khan, M. Ehatisham-ul Haq, S. N. Mehmood Shah, *et al.*, 2020 Pseudorandom number generator (prng) design using hyper-chaotic modified robust logistic map (hc-mrlm). Electronics **9**: 104.

Jamal, S. S., T. Shah, and I. Hussain, 2013 An efficient scheme for digital watermarking using chaotic map. Nonlinear Dynamics **73**: 1469–1474.

Kang, Q., K. Li, and J. Yang, 2014 A digital watermarking approach based on dct domain combining qr code and chaotic theory. In *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 331–337, IEEE.

Ketthong, P. and W. San-Um, 2014 A robust signum-based piecewise-linaer chaotic map and its application to microcontroller-based cost-effective random-bit generator. In *2014 International Electrical Engineering Congress (iEECON)*, pp. 1–4, IEEE.

Khanzadi, H., M. Eshghi, and S. E. Borujeni, 2014 Image encryption using random bit sequence based on chaotic maps. Arabian Journal for Science and engineering **39**: 1039–1047.

Liu, L., S. Miao, M. Cheng, and X. Gao, 2016 A pseudorandom bit generator based on new multi-delayed chebyshev map. Information Processing Letters **116**: 674–681.

Liu, Y., Z. Qin, X. Liao, and J. Wu, 2020 Cryptanalysis and enhancement of an image encryption scheme based on a 1-d coupled sine map. NONLINEAR DYNAMICS .

May, R. M., 1976 Simple mathematical models with very compli-

cated dynamics. Nature **261**: 459–467.
Moafimadani, S. S., Y. Chen, and C. Tang, 2019 A new algorithm for medical color images encryption using chaotic systems. Entropy **21**: 577.
Nosrati, K., C. Volos, and A. Azemi, 2017 Cubature kalman filter-based chaotic synchronization and image encryption. Signal Processing: Image Communication **58**: 35–48.
Özkaynak, F., 2018 Brief review on application of nonlinear dynamics in image encryption. Nonlinear Dynamics **92**: 305–313.
Pak, C. and L. Huang, 2017 A new color image encryption using combination of the 1d chaotic map. Signal Processing **138**: 129–137.
Pareek, N., V. Patidar, and K. Sud, 2005 Cryptography using multiple one-dimensional chaotic maps. Communications in Nonlinear Science and Numerical Simulation **10**: 715–723.
Patidar, V., K. K. Sud, and N. K. Pareek, 2009 A pseudo random bit generator based on chaotic logistic map and its statistical testing. Informatica **33**.
Phatak, S. and S. S. Rao, 1995 Logistic map: A possible random-number generator. Physical review E **51**: 3670.
Roy, S., S. Chatterjee, A. K. Das, S. Chattopadhyay, S. Kumari, *et al.*, 2017 Chaotic map-based anonymous user authentication scheme with user biometrics and fuzzy extractor for crowdsourcing internet of things. IEEE Internet of Things Journal **5**: 2884–2895.
Rukhin, A., J. Soto, J. Nechvatal, M. Smid, and E. Barker, 2001 A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-Allen and Hamilton Inc Mclean Va.
Sarmah, H. K. and M. C. Das, 2014 Various bifurcations in a cubic map. International Journal of Advanced Scientific and Technical Research **3**: 827–846.
Shaukat, S., A. Arshid, A. Eleyan, S. A. Shah, J. Ahmad, *et al.*, 2020 Chaos theory and its application: An essential framework for image encryption. Chaos Theory and Applications **2**: 15–20.
Sheng, S. and X. Wu, 2012 A new digital anti-counterfeiting scheme based on chaotic cryptography. In *2012 International Conference on ICT Convergence (ICTC)*, pp. 687–691, IEEE.
Stojanovski, T. and L. Kocarev, 2001 Chaos-based random number generators-part i: analysis [cryptography]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications **48**: 281–288.
Stoyanov, B., 2013 Pseudo-random bit generator based on chebyshev map. In *AIP Conference Proceedings*, volume 1561, pp. 369–372, American Institute of Physics.
Stoyanov, B. and T. Ivanova, 2019 Chaosa: Chaotic map based random number generator on arduino platform. In *AIP Conference Proceedings*, volume 2172, p. 090001, AIP Publishing LLC.
Stoyanov, B. and K. Kordov, 2015 Image encryption using chebyshev map and rotation equation. Entropy **17**: 2117–2139.
Strogatz, S. H., 2018 *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC Press.
Ursulean, R., 2004 Reconsidering the generalized logistic map as a pseudo random bit generator. Elektronika ir elektrotechnika **56**.
Volos, C. K., 2013 Chaotic random bit generator realized with a microcontroller. Journal of Computations & Modelling **3**: 115–136.
Volos, C. K., I. Kyprianidis, and I. Stouboulos, 2013a Text encryption scheme realized with a chaotic pseudo-random bit generator. Journal of Engineering Science & Technology Review **6**.
Volos, C. K., I. M. Kyprianidis, and I. N. Stouboulos, 2013b Image encryption process based on chaotic synchronization phenomena. Signal Processing **93**: 1328–1340.
Wang, X., Ü. Çavuşoğlu, S. Kacar, A. Akgul, V.-T. Pham, *et al.*, 2019a S-box based image encryption application using a chaotic system without equilibrium. Applied Sciences **9**: 781.
Wang, X., L. Liu, and Y. Zhang, 2015 A novel chaotic block image encryption algorithm based on dynamic random growth technique. Optics and Lasers in Engineering **66**: 10–18.
Wang, X.-Y. and Y.-X. Xie, 2012 A design of pseudo-random bit generator based on single chaotic system. International Journal of Modern Physics C **23**: 1250024.
Wang, X.-Y., J.-J. Zhang, F.-C. Zhang, and G.-H. Cao, 2019b New chaotical image encryption algorithm based on fisher–yatess scrambling and dna coding. Chinese Physics B **28**: 040504.
Wang, Y., Z. Liu, J. Ma, and H. He, 2016 A pseudorandom number generator based on piecewise logistic map. Nonlinear Dynamics **83**: 2373–2391.
Wang, Z., A. Akgul, V.-T. Pham, and S. Jafari, 2017 Chaos-based application of a novel no-equilibrium chaotic system with coexisting attractors. Nonlinear Dynamics **89**: 1877–1887.
Zhang, J., L. Tian, and H.-M. Tai, 2004 A new watermarking method based on chaotic maps. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 2, pp. 939–942, IEEE.
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., L. Bao, and C. P. Chen, 2014 A new 1d chaotic system for image encryption. Signal processing **97**: 172–182.
Zhu, C., G. Wang, and K. Sun, 2018 Improved cryptanalysis and enhancements of an image encryption scheme using combined 1d chaotic maps. Entropy **20**: 843.

How to cite this article: Moysis, L., Tutueva, A., Volos, C., and Butusov, D. A Chaos Based Pseudo-Random Bit Generator Using Multiple Digits Comparison. Chaos Theory and Applications, 2(2), 58-68.