2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) | 978-1-6654-0476-1/20/$31.00 ©2021 IEEE | DOI: 10.1109/ElConRus51938.2021.9396395

Application of a Hyperbolic Tangent Chaotic Map to Random Bit Generation and Image Encryption

Lazaros Moysis
Aristotle University of Thessaloniki
Thessaloniki, Greece
lmousis@physics.auth.gr

Ioannis Kafetzis
Aristotle University of Thessaloniki
Thessaloniki, Greece
kafetzii@math.auth.gr

Christos Volos
Aristotle University of Thessaloniki
Thessaloniki, Greece
volos@physics.auth.gr

Aleksandra V. Tutueva
Youth Research Institute
Saint Petersburg Electrotechnical University "LETI"
St. Petersburg, Russia
avtutueva@etu.ru,

Denis Butusov
Youth Research Institute
Saint Petersburg Electrotechnical University "LETI"
St. Petersburg, Russia
dnbutusov@etu.ru

***Abstract*—In this paper, a two-parameter one-dimensional chaotic map with hyperbolic tangent and two nested sinusoidal terms is proposed. The reported map exhibits rich chaotic behavior including such phenomena as the period-doubling route to chaos, crisis, and antimonotonicity appearing. The proposed map is applied for the pseudo-random bit generation and image encryption. To generate bit sequences, a simple rule is used. The generated sequences are verified using NIST statistical tests and cross-correlation analysis. Image encryption is performed using two rounds of shuffling of image pixels and XOR operation. We explicitly show the suitability of the proposed algorithm through histogram, correlation, and entropy analysis for the sample grayscale image.**

Keywords—Chaos-based cryptography; PRBG; NIST; encryption

I. Introduction

Chaotic systems have long been integrated into practical applications related to encryption and security due to their high unpredictability, sensitivity to initial conditions and parameter changes [1]. It was recently shown that chaos-based stream encryption schemes show high performance compared to conventional algorithms [2]. For this reason, the development of new simple discrete maps with chaotic behavior and complex dynamics is of interest.

In this paper, a one-dimensional chaotic map with two parameters and trigonometric terms is proposed and studied. The bifurcation diagrams and Lyapunov exponents (LE) of the proposed map show that the one-dimensional map can exhibit complex behavior. After the dynamical analysis, the proposed map is applied to cryptography applications. The first problem considered is the design of a random bit generator (PRBG) [3–5]. Chaos-based RBGs are commonly based on a simple rule to generate the bits, e.g. the modulo operation. The proposed generator generally follows this idea. We showed that the produced bitstreams are pseudo-random, which is verified through a series of NIST statistical tests [6]. The PRBG also shows good correlation characteristics.

The second problem under consideration is image encryption. Here two different rounds of shuffling are performed for the original image, followed by an XOR operation to encrypt the pixel values. As a simulation example, the classic "Peppers" image is encrypted and then tested through histogram, correlation and entropy analysis.

The rest of the paper is organized as follows: In Section II, the proposed map is studied. In Section III, the problem of random bit generation is considered. In Section IV, the application of image encryption is presented. Section V concludes the work with a discussion on future topics of interest.

II. The Proposed Chaotic Map

The proposed map is given by the following equation

$$x_i = a\tanh\left(\sin\left(\pi\sin\left(\pi x_{i-1}\right)\right)\right)+b \tag{1}$$

where a and b are parameters. Other examples of such maps with the hyperbolic tangent function as a source of chaotic behavior can be found in [7, 8].

To study the dynamical behavior of the map, we plot its bifurcation diagrams and calculate LE values with respect to each parameter. Fig. 1 and 2 represent the bifurcation diagram and the LE curve for a parameter, while $b = 0$ and $x_0 = 0.1$. It can be seen that the system traverses to chaotic behavior following a period doubling route to chaos. The chaotic behavior remains steady for a wide range of parameter values, interrupted by small periodic windows. Crisis phenomena also emerge, where the system abruptly exits chaos, and then re-enters it following a period doubling route. Fig. 3 shows multiple bifurcation diagrams with respect to parameter a, for various

This study was supported by the grant of the Russian Science Foundation (RSF), project 20-79-10334.

values of b. It is observed that for the integer values $b = 0, 1, 2, 3, 4, 5$, the considered behavior remains roughly the same, with the diagram moving vertically in the state space. In addition, minor changes occur for values around $a = 1.5$ and odd values of b. However, one can note that LE for the system with respect to a is not significantly affected for the values of parameter b that are considered (except from small changes around $a = 0.5$), but rather remains almost identical to the values seen in Fig. 2.

The bifurcation diagram for the system (1) with respect to parameter b is shown in Fig. 4. The investigated map exhibits chaotic behavior for certain ranges of the parameter value, interrupted by crisis phenomena. Moreover, the phenomenon of antimonotonicity appears, where the system enters chaos following a period doubling route, and as the bifurcation parameter increases, exits chaos following the reverse period halving route. This can be observed for example for values of b in the range of 1.5 to 1.9.

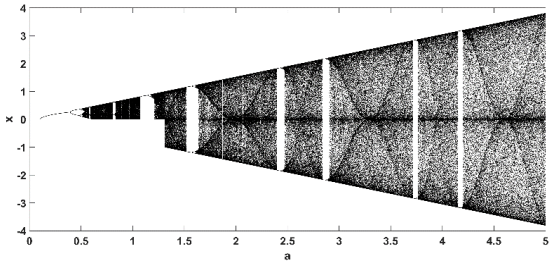

Fig. 1. Bifurcation diagram of (1) with respect to parameter a, for $b = 0$.

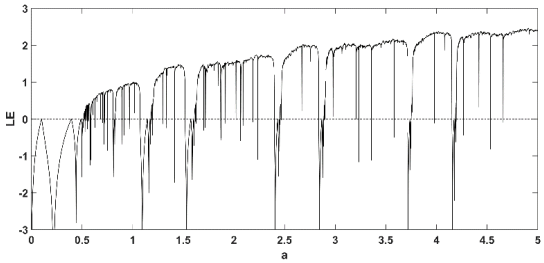

Fig. 2. Lyapunov exponent diagram of (1) with respect to parameter a, for $b = 0$.

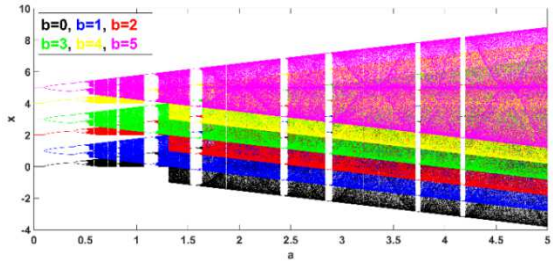

Fig. 3. Bifurcation diagram of (1) with respect to parameter a, for various b values.

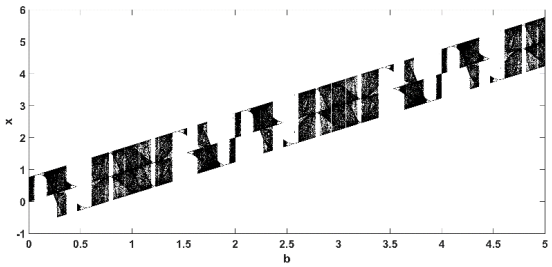

Fig. 4. Bifurcation diagram of (1) with respect to parameter b, for $a = 1$.

In Fig. 5, multiple bifurcation diagrams are depicted with respect to parameter b while a was varied. It is seen that similar chaotic phenomena are observed, with chaotic behavior appearing and disappearing at different parameter values for each a.

All of the above phenomena are verified in Fig. 6, where the LE diagram is depicted with respect to b, for different values of a. In contrast to the previous case, it can be seen that the value of a affects the largest value that LE can achieve.

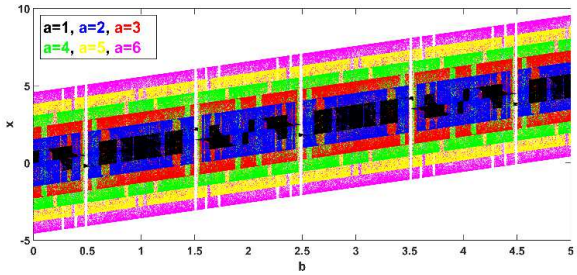

Fig. 5. Bifurcation diagram of (1) with respect to parameter b, for different values of a.

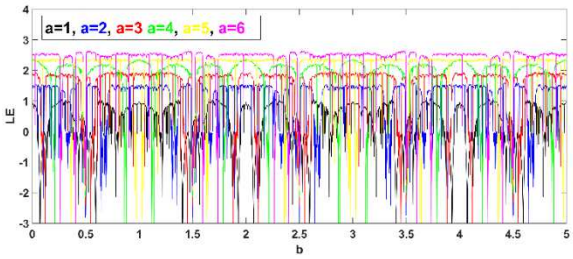

Fig. 6. Lyapunov exponent diagram of (1) with respect to parameter b, for different values of a.

Finally, Fig. 7 depicts the pairs of parameter values *(a, b)* that give a positive LE value, and hence chaos, plotted in black, and the ones yielding a negative or zero exponent, plotted in red.

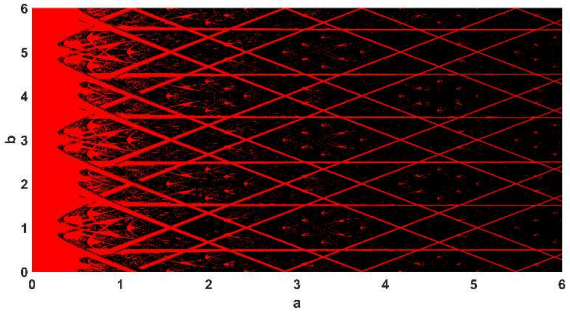

Fig. 7. Diagram depicting the parameter pairs (a, b) giving a positive Lyapunov exponent (black) and a zero or negative (red).

III. Application To Random Bit Generation

To design the PRBG, we will consider a simple rule based on the values of the map in each iteration. So, given the key values of the map a, b, x_0, a bit B_i is generated in each iteration as follows:

$$B_i = \lfloor mod(x_i x_{i-1}, 2) \rfloor \quad (2)$$

where $\lfloor\ \rfloor$ denotes the integer part of the argument. Note that the proposed PRBG utilizes the last two values of the chaotic map, rather than just the current value x_i. It can be assumed that the amount of additional memory required to apply such a technique is increased insignificantly, even for small microcontroller devices that may be used to implement this PRBG.

To verify the pseudo-random properties generated by the proposed PRBG, a set of 50×10^6 bitstreams is generated and tested through the NIST suite (Table 1). The bitstream was generated for parameter values $a = 300$, $b = 1$ and $x_0 = 0.1$. As one can see, the proposed PRBG passes all statistical tests.

TABLE I. NIST Test Results

No.	*Test*	*P-Value*	*Rate*
1	Frequency	0.419021	50/50
2	BlockFrequency	0.236810	50/50
3	CumulativeSums	0.816537	50/50
4	Runs	0.289667	50/50
5	LongestRun	0.739918	50/50
6	Rank	0.455937	50/50
7	FFT	0.574903	50/50
8	NonOverlappingTemplate	0.616305	49/50
9	OverlappingTemplate	0.534146	50/50
10	Universal	0.023545	50/50
11	ApproximateEntropy	0.883171	49/50
12	RandomExcursions	0.739918	26/26
13	RandomExcursionsVariant	0.213309	26/26
14	Serial	0.262249	49/50
15	LinearComplexity	0.455937	50/50

Besides, a random bitstream should have correlation properties similar to a delta function, and the cross-correlation should be close to zero [3]. This is verified using auto-correlation and cross-correlation plots shown in Fig. 8, computed for a bitstream of length 10^4, with $a = 300$, $b = 1$. For the cross-correlation, two different bitstreams were considered with initial conditions $x_0 = 0.1$ and $\hat{x_0} = 0.1 + 10^{-16}$.

IV. Application To Image Encryption

In this section, we apply the considered PRBG to the image encryption. The proposed scheme consists of two rounds of pixel shuffling, and one round of encryption using XOR. In each step, a chaotic map (1) with different parameters values is utilized, to increase the overall key space. The procedure is outlined in Fig. 9. The proposed design falls in the category of symmetric encryption schemes. In this case, the receiver should obtain information on the key values of the chaotic maps used, to generate the chaotic sequences applied in each shuffling and encryption step.

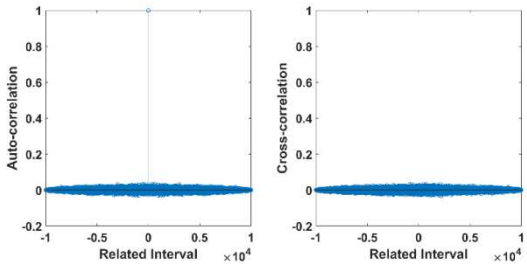

Fig. 8. Auto-correlation and Cross-correlation for a pseudo-random bit sequence of length 10^4.

A. First Round of Shuffling

The shuffling of the pixels in an image aims to reducing the correlation between adjacent pixels. Normally, in any image the correlation between adjacent pixels is high, since the image contains some sort of information. Thus, by shuffling the pixels of an image, the correlation is reduced and the resulting image carries less information, and is thus less susceptible. The procedure is similar to [4, 5]. Starting from an image A of dimensions $M \times N$, the first step is to shuffle rows and columns. First, the chaotic map is utilized with parameters a_1, b_1, x_0. For the shuffling of the rows, the parameters a_1, b_1 are chosen so that the maps values lie in an interval larger than $[-\max(M, N), \max(M, N)]$. In each iteration of the map, the value $r_j = |x_i| + 1$ is computed, until M discrete values $\{r_1, \ldots, r_M\}$ in the interval $\{1, \ldots, M\}$ are generated. These denote the rearrangement of the rows of A. So, the 1st row is moved to the r_1 row, the 2nd row to the r_2 row and so on. This procedure is equivalent to the multiplication of the matrix A from the left by an invertible matrix L, where each row has zero in its entries, and 1 in the position r_j.

After the above procedure is completed, we set x_0 as the final value of the last iteration of the map, and the procedure is repeated using $c_j = |x_i| + 1$, until N discrete values $\{c_1, \ldots, c_M\}$ in

the interval {1, ..., N} are generated. This set represents the rearrangement of the columns of A, and is equivalent by right multiplication by R, where R has zero entries in each column, and 1 in the position c_j.

The complete shuffling procedure is denoted by $A_1 = LAR$, and can easily be reversed to get the original matrix by taking $A = L^{-1}A_1R^{-1}$.

B. Second Round of Shuffling

The second round involves shuffling the entries of each row using a second chaotic map. For each row, the parameters a_2, b_2 are chosen so that the values of the map lie in an interval larger than $[-M, M]$ and the initial condition is taken as $y_0^j = y_0^{j-1} + 0.1$, where $j = 1, \ldots, M$ and y_0^1 is the initial value of the map chosen for the first row.

To generate the shuffling rules, N values of the following map are generated

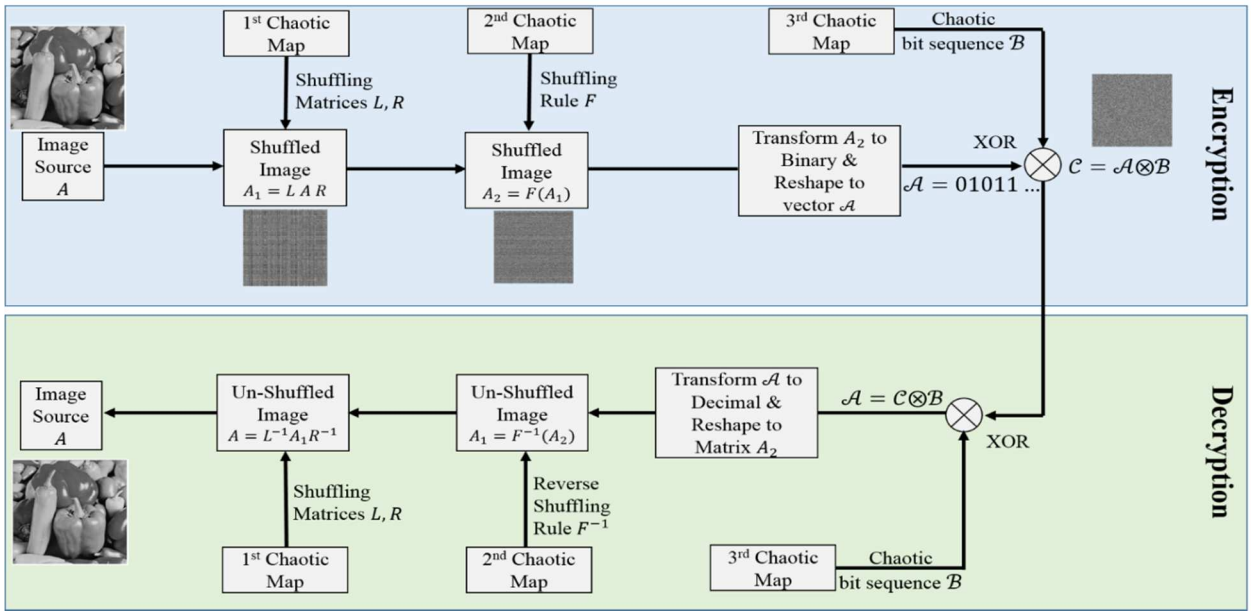

Fig. 9. Outline of the image encryption design

The parameter $(a_2 - b_2 - k) / \tanh(1)$ here is chosen after observing the bifurcation diagrams in Fig. 3, so that in each iteration, the values of the map lie between the interval $[-(a_2 - b_2 - k); a_2 + b_2 - k]$. Hence, we can easily choose the parameters so that the following shuffling rule generates acceptable values. Using the above sequence, the following finite sequence of indices w_k is defined as

$$w_k = \lfloor y_k \rfloor + 1 \tag{3}$$

At the k^{th} step, perform the element permutation $\sigma_k(w)$ defined by the indices of each element as

$$\begin{matrix} w \\ \sigma_k(w) \end{matrix} \begin{pmatrix} 1 & \cdots & w_{k-1} & w_k & w_{k+1} & \cdots & w_{N-1} & w_N \\ 1 & \cdots & w_{k-1} & w_{k+1} & w_{k+2} & \cdots & w_N & w_k \end{pmatrix}$$

It is known, that every permutation has an inverse, which is its transpose, in the sense of interchanging the two rows in the permutation matrix. Clearly, this will be used for the decryption of the message.

Thus, let r_j denote the j–th row of the shuffled matrix. Then, the column indices after shuffling this row with the proposed method are

$$r_j^{shuf} = \sigma_N\left(\sigma_{N-1}\left(\cdots\sigma_1\left(r_j\right)\right)\right) \tag{4}$$

The decryption method for this step is to take the coded message, and produce the same set of integers w_k, $k = 1, \ldots N$. Then starting from w_N and moving step by step until w_1, bring the last element of the list to the w_k position, that performs the permutation $\tau_k(w)$ which is

$$\begin{matrix} w \\ \tau_k(w) \end{matrix} \begin{pmatrix} 1 & \cdots & w_{k-1} & w_k & w_{k+1} & \cdots & w_{N-1} & w_N \\ 1 & \cdots & w_{k-1} & w_N & w_k & \cdots & w_{N-2} & w_{N-1} \end{pmatrix}$$

One can easily see that the matrix for τ_k is the transpose for σ_k and thus τ_k is the inverse of σ_k for $k = 1, \ldots, N$, that is $\tau_k\sigma_k = I$, where I denotes the identity permutation, that is $I(i) =$

i.

It is mentioned that for the decryption process, the inverse should be taken from last to first. The reason behind this is clear considering (4). Indeed, consider the shuffled row r_j^{shuf} and apply τ_N. Then

$$\tau_N\left(r_j^{shuf}\right)=\tau_N\left(\sigma_N\left(\sigma_{N-1}\left(\ldots\sigma_1\left(r_j\right)\right)\right)\right)=\sigma_{N-1}\left(\ldots\sigma_1\left(r_j\right)\right)$$

and therefore it can be easily verified that

$$\tau_1\left(\tau_2\left(\ldots\tau_N\left(r_j^{shuf}\right)\right)\right)=r_j$$

which is the original row.

To clarify the above, we present an illustrative example of the shuffling method introduced. For the sake of simplicity, the vector entries are denoted by letters and only four steps are performed.

Example 1: Suppose the row vector to be shuffled is

$$v_0 = (a\ b\ c\ d\ e\ f)$$

And that the shuffle indices are $w_1 = 4$, $w_2 = 2$, $w_3 = 1$, $w_4 = 1$. Then, performing the shuffling steps leads to the following vectors

$$\begin{matrix} \\ w_1=4 \\ w_2=2 \\ w_3=1 \\ w_4=1 \end{matrix}\begin{pmatrix} a & b & c & d & e & f \\ a & b & c & f & e & d \\ a & c & f & e & d & b \\ c & f & e & d & b & a \\ f & e & d & b & a & c \end{pmatrix}$$

and thus the shuffled row is $v_{shuff} = (f\ e\ d\ b\ a\ c)$. Consider now the unshuffling method starting from v_{shuff}

$$\begin{matrix} w_4=1 \\ w_3=1 \\ w_2=2 \\ w_1=4 \\ \\ \end{matrix}\begin{pmatrix} f & e & d & b & a & c \\ c & f & e & d & b & a \\ a & c & f & e & d & b \\ a & b & c & f & e & d \\ a & b & c & d & e & f \end{pmatrix}$$

and thus the decoded message is $(a\ b\ c\ d\ e\ f)$ which is actually v_0, that is, the original message.

The advantage of shuffling each row separately is twofold. First, although these permutations constitute a shuffling of the entries in each row, they cannot be represented as post-multiplication of the shuffled image by a non-singular matrix. This is because the columns of each row are shuffled independently from the elements of the respective columns in other rows. Secondly, the shuffling of different rows is independent. This means that the shuffling and unshuffling (in the encryption and decryption process, respectively) of different rows can be performed simultaneously, in parallel computation, which increases time efficiency significantly.

Note that the same procedure can be applied to shuffle the pixels in each individual column of the image, for further confusion of the pixel ordering.

Remark 1: The increment by 1 that appears in (3) where the sequence of indices w_k is produced, guarantees that $w_k \geq 1$; $k = 1, \ldots N$. The reason behind this is that, the implementation for these steps was done in MATLAB, where the index for the first row elements is 1. If the same code is to be implemented in another programming language (e.g. *Python*) where the first index in a list equals 0 then the +1 is to be omitted.

C. Encryption

Finally, after the second round of shuffling is performed, the resulting image matrix, noted as $A_2 = F(A_1)$, with values in [0; 255] is transformed to binary, and then reshaped to a single vector A of length 8 × M × N. This vector is then combined with a random bitstream B of equal length using the commonly used XOR operation [4, 5, 9]. The bitstream is generated using the proposed PRBG from the previous section, with parameters a_3; b_3; z_0. The resulting encrypted vector $C = A \otimes B$ can be safely transmitted from a channel to a receiver.

D. Decryption Process

At the receiver end, the encrypted vector can be transformed back to the original image by following the reverse procedure used in the encryption process.

First, the encrypted vector C is XORed again with the same bitstream used in the encryption step, in order to obtain the initial vector $A = C \otimes B$. This binary vector is then transformed back to its decimal values and reshaped to matrix form A_2. The reverse second round of shuffling is performed as described in subsection IV-B, to obtain image A_1. It is then unshuffled again using the reverse procedure described in subsection IV-A to obtain the original image $A = L^{-1}A_1R^{-1}$.

E. Key Space

As for the computation of the key space, the design utilizes three chaotic maps: one chaotic map for the generation of the first round of shuffling, one chaotic map for the second round of shuffling, and one chaotic map for the generation of the bit sequence used in the XOR operation. Since map (1) is one-dimensional with two parameters, there are three key values for each map. Hence, the overall design utilizes 9 key values. So, assuming a 15-digit accuracy, an upper bound of the key space is given by $10^{15\times 9} = 10^{135} \approx 2^{450}$. This is higher than the bound 2^{100}

that is considered necessary to resist brute force attacks [10].

F. Simulation Performance

For the simulation of the above design, the well-known 512×512 grayscale image of "Peppers" is considered. After the encryption is completed, a series of tests are performed on the original, shuffled, and final encrypted images, to test the performance of the encryption. For the first round of shuffling, we choose $a_1 = 512 / \tanh(1)$, $b_1 = 0$, $x_0 = 0.1$. For the second round of shuffling, we chose $a_2 = 512, b_2 = 0, y_0^1 = 0.2$. For the XOR operation, the parameters are $a_3 = 300$, $b_2 = 1$, $z_0 = 1$. The resulting images are all shown in Fig.10.

The histograms of the original and encrypted images are shown in Fig. 11, where one can observe that the encrypted image has a uniform distribution of grey values as expected.

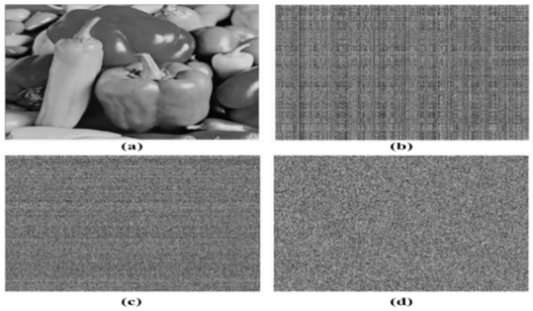

Fig. 10. (a) Original image, (b) image after first shuffling round, (c) image after second shuffling round, (d) final encrypted image after XOR.

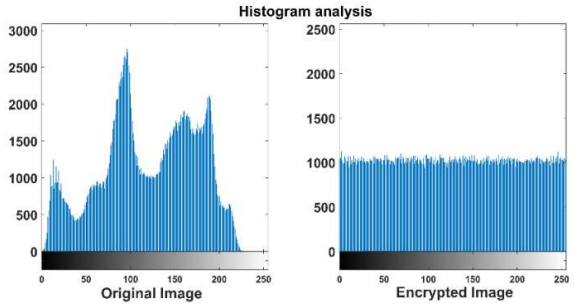

Fig. 11. Histogram of the original (left) and encrypted (right) images.

Regarding correlation, Table II shows the correlations between adjacent horizontal, vertical, and diagonal pixels for all the images. As we expected, the encrypted image has a significantly lower correlation among all the images, and also it is observed that the second round of shuffling reduces the correlation from the first round. As an illustration, Fig. 12 also plots the correlation between 10 000 randomly chosen pairs of diagonal pixels.

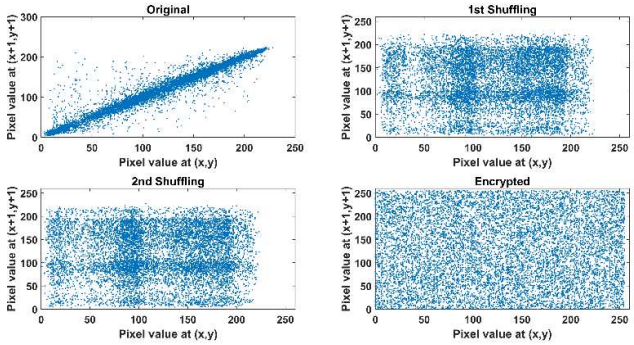

Fig. 12. Correlation between adjacent diagonal pixels.

Finally, the entropy of the original image is 7.5937, while for the encrypted image is 7.9993, which is closer to 8, hence the encrypted image is similar to a random image. Thus, the image has a uniform histogram, low correlation, and information entropy close to 8. All of these properties indicate that it is similar to a noise-like image that carries no distinguishable information.

TABLE II. CORRELATION COEFFICIENTS FOR IMAGE ENCRYPTION.

	Original	***1st Shuffle***	***2nd Shuffle***	***Encrypted***
Horizontal	0.9768	0.0797	0.0645	-0.0001
Vertical	0.9792	0.0910	-0.0027	0.0026
Diagonal	0.9639	-0.0065	-0.0059	0.0001

V. CONCLUSIONS

In this work, a simple one-dimensional two-parameter chaotic map was proposed, having a hyperbolic tangent term. The map was analyzed by computation of its bifurcation diagrams and Lyapunov exponent diagrams and it was shown to possess chaotic behavior for a wide range of parameters, as well as other chaos related phenomena like period doubling route to chaos, crisis and antimonotonicity. The map was then applied to pseudo-random bit generation, yielding a statistically random PRBG. Using the proposed PRBG, an image encryption scheme was designed, using two rounds of pixel shuffling and one round of XOR encryption. The overall design yielded encrypted images with satisfying properties related to grayscale shade distribution, correlation and entropy. Future extensions of this work will consider fractional versions of the proposed chaotic map, the application to color images, the enhancement of the encryption scheme with a second round of encryption, and also the microcontroller implementation of the complete design.

REFERENCES

[1] S.N. Elaydi, Discrete chaos: with applications in science and engineering. CRC Press, p. 419, 2007.

[2] A.V. Tutueva, E.G. Nepomuceno, A.I. Karimov, V.S. Andreev, D.N. Butusov, Adaptive chaotic maps and their application to pseudo-random numbers generation. *Chaos, Solitons & Fractals*, 133, pp. 109615, 2020

[3] X. Huang, L. Liu, X. Li, M. Yu, Z. Wu, A new pseudorandom bit generator based on mixing three-dimensional Chen chaotic system with a chaotic tactics. *Complexity*, vol. 2019, p. 10, 2019.

[4] L. Moysis, A. Tutueva, K. Christos, D. Butusov, A chaos based pseudo-random bit generator using multiple digits comparison, *Chaos Theory and Applications*, vol. 2, no. 2, pp. 58–68, 2020.

[5] T. Gao, Z. Chen, A new image encryption algorithm based on hyper-chaos, *Physics Letters A*, vol. 372, no. 4, pp. 394–400, 2008.

[6] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, A statistical test suite for random and pseudorandom number generators for cryptographic applications, *Booz-Allen and Hamilton Inc Mclean Va*, Tech. Rep., p. 131, 2001.

[7] N. Jiteurtragool, T. Masayoshi, W. San-Um, Robustification of a one-dimensional generic sigmoidal chaotic map with application of true random bit generation, *Entropy*, vol. 20, no. 2, p. 136, 2018.

[8] G. Ablay, Chaotic map construction from common nonlinearities and microcontroller implementations, *International Journal of Bifurcation and Chaos*, vol. 26, no. 07, p. 1650121, 2016.

[9] F. Özkaynak, Brief review on application of nonlinear dynamics in image encryption, *Nonlinear Dynamics*, vol. 92, no. 2, pp. 305–313, 2018.

[10] G. Alvarez, S. Li, Some basic cryptographic requirements for chaos-based cryptosystems, *International journal of bifurcation and chaos*, vol. 16, no. 08, pp. 2129–2151, 2006.