A new Stream cipher based on Nonlinear dynamic system

1st O. Mannai
Risc Laboratory
National Engineering School of Tunis
Tunis El Manar University
1002, Tunis, Tunisia
email address: becheikh.rabei@gmail.com

2nd R. Becheikh
Risc Laboratory
National Engineering School of Tunis
Tunis El Manar University
1002, Tunis, Tunisia

3rd R. Rhouma
and S. Belghith
Risc Laboratory
National Engineering School of Tunis
Tunis El Manar University
1002, Tunis, Tunisia

***Abstract*—In this paper, we introduce a new synchronous stream cipher. The core of the cipher is the Ikeda system which can be seen as a Nonlinear Feedback Shift Register (NLFSR) of length 7 plus one memory. The cipher takes 256-bit key as input and generates in each iteration an output of 16-bits. A single key is allowed to generate up to 2^{64} output bits. A security analysis has been carried out and it has been showed that the output sequence produced by the scheme is pseudorandom in the sense that they cannot be distinguished from truly random sequence and resist to well-known stream cipher attacks.**

***Keywords*— Stream ciphers; NLFSR; Distinguishing attack; diffusion; confusion.**

I. Introduction

In cryptography, stream and block ciphers are known as symmetric cryptographic primitives used to guarantee data privacy over a communication channel. Block ciphers offer the possibility to transform a fixed block of symbols to blocks of ciphertext using a fixed encryption transformation [1], [2]. By contrast, stream ciphers encrypt each character of a plaintext bit by bit or word by word, using an encryption transformation which varies with time [3], [4].
Stream ciphers are characterized by limited error propagation. Moreover, they are generally faster than block ciphers, and they can be effectively implemented with low cost and can achieve the same security level as block ciphers using limited resources. For these significant advantages, stream ciphers are widely used in telecommunication applications like SSL, IPsec, RFID, Bluetooth, GSM, UMTS, online encryption of big data and in military communication systems [4], [5]. Nevertheless, the security of stream ciphers has not been studied sufficiently. So, many cryptographers are interested in developing and analyzing stream ciphers to produce cryptographic primitives that generate random-looking sequences, that are as "indistinguishable" as possible from truly random sequence. The easy way to build this kind of systems is by using pseudorandom number generators (PRNG) [6], [7].
Actually, a keystream generator (KSG) that generates a long pseudorandom sequence represents one common way to build stream ciphers. The principal task of a KSG is to produce a keystream with certain basic properties. These include a very large linear complexity, large period and white-noise statistics. It is on this basis that, the eSTREAM candidates were launched [8] to determine secure and efficient stream ciphers that might become useful for widespread adoption.

In this paper, we propose a new synchronous stream cipher (SSC) where the keystream is generated independently from the plaintext. The design of the SSC is based on the Ikeda system. The specific properties of this chaotic system as the extremely complex chaotic behavior introduces a nonlinearity to the cipher and guarantees the unpredictability of the output sequence.

The organization of the paper is as follows: The specification of the proposed cipher is described in section II. The security analysis of the cryptosystem is discussed in section III. Finally the conclusion is given in section IV.

II. Specifications of the cipher

Nonlinear delay differential systems are known as systems of infinite dimension [11], [12]. Hence, they have shown an increasing interest [13], [14]. Among these systems we have chosen the discrete model of Ikeda system, which can be defined as follows [15]:

$$\begin{aligned} a_i^{t+1} &= a_{i+1}^t \qquad i = 0, \ldots, N-2 \\ a_{N-1}^{t+1} &= a_{N-1}^t + \alpha \cdot (-\beta \cdot a_{N-1}^t + m \cdot \sin(a_0^t)) \end{aligned} \tag{1}$$

where a_i^0 represents the initial condition of the system and α, β and m are the control parameters. Whereas a_i^t represent the value of the variable a_i at time t.

The proposed cryptographic primitive is a synchronous stream cipher that uses a key K of length 256-bits and outputs 16-bits in each iteration which in turn is combined with a plaintext symbol and a ciphertext symbol will be computed. The main core of the cipher is Ikeda system. This system can be used to generate a keystream with a long period and good statistical properties to provide security. Moreover, it can be implemented with low cost. Actually, Ikeda system can be regarded as a nonlinear feedback shift register (NLFSR) with memory. In fact, an NLFSR is a finite state machine with a

ISBN 978-90-827970-1-5

register. In each iteration, the state variables are updated by shifting them to the left, except the most left state variable which is updated in a nonlinear way by using the feedback function. The system outputs the most right state variable in each clock cycle.
In this context, the state variable of Ikeda system can be viewed as a state variable of an NLFSR, and the most left state variables of the Ikeda are updated by involving the control parameters α, β and m. The model design presenting Ikeda system as an NLFSR with N registers is illustrated in Fig. 1.

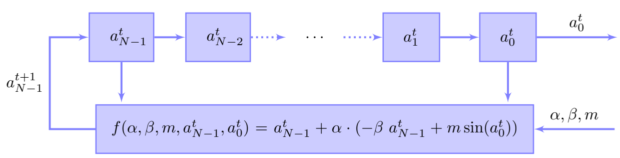

Fig. 1. Diagram of Ikeda system as an NLFSR

From here on, we view Ikeda system as an NLFSR with memory. According to the previous work in [15], Ikeda system ensures a chaotic behavior, for a specific interval of the parameters α, β and m. For instance, for $\alpha = 0.5, \beta = 1, N = 7$ and $m \in [6, 30]$, the behavior of the system is ergodic. Therefore, to design a good cryptosystem based on Ikeda system, the aforementioned parameters must be respected. In this context, the proposed stream cipher is based on an NLFSR of length 7 with memory which holds the value of the parameter m while the parameters α and β are fixed and regarded as known non-secret control parameters and have as value respectively 0.5 and 1.
In the following, we introduce the setup and keystream generation phase.

A. Setup phases

The setup phase is an important phase that must be well designed in order to prevent certain attacks such as resynchronization attack, re-keying attack, and divide and conquer attack. In this phase, the key is used to initialize the state variable of the NLFSR as well as the memory.
The main key K of length 256-bits is divided into 8 subkeys of length 32-bits labeled as $k_1 = K^{[0..32]}, k_2 = K^{[33..64]}, \ldots, k_8 = K^{[225..256]}$. These subkeys are loaded into 8 new variables denoted $X_{i,0}$ as follows:

$$X_{i,0} = k_{i+1}\, for\, 0 \leqslant i \leqslant 7$$

These variables $X_{i,0}$ are updated to $X_{i,1}$ as follows:

$$X_{i,1} = X_{i,0} \oplus (X_{(i+2)\, mod\, 8,0} \ggg 24) \oplus (X_{(i+3)\, mod\, 8,0} \ggg 16)$$

for $0 \leqslant i \leqslant 7$ and where $\ggg$ represents a right shift bit rotation.
In order to make each variable depends of the entire key bits, the above equation is iterated 3 times. Actually, in the first iteration, for instance the variable $X_{0,1}$ depends on $X_{0,0}, X_{2,0}$ and $X_{3,0}$ i.e depends on k_1, k_3 and k_4. In the second iteration, the variable $X_{0,2}$ depends on $X_{0,1}$ and the new value of $X_{2,1}$ which in turn depends on $X_{4,0}$ and $X_{5,0}$, and the value of $X_{3,1}$ which in turn depends on $X_{5,0}$ and $X_{6,0}$, which mean that the variable $X_{0,2}$ depends on the subkeys k_1, k_3, k_4, k_5, k_6 and k_7. By iterating the equation one more time, the new variable $X_{0,3}$ depends on $X_{2,2}$ and $X_{3,2}$ where the variable $X_{2,2}$ depends on $X_{7,0}$ and $X_{0,0}$ and the variable $X_{3,2}$ depends on $X_{0,0}$ and $X_{1,0}$ i.e $X_{3,0}$ depends of all the subkeys. Once these variables are computing the initial states variables denoted a^0_i of the NLFSR as well as the memory m will be initialized in a way that all of them depend on the whole key bits. These variables are computed as follows:

$$\begin{cases} a^0_i = 0.1 + \frac{X_{i,3}}{2^{28}}\, for\, 0 \leqslant i \leqslant 6 \\ m = 7 + \frac{X_{7,3}}{2^{28}}. \end{cases} \quad (2)$$

The form of computing each variable state a^0_i is chosen in a way to make each key valid i.e all the keys can be used to generate the keystream. In fact, in the case where all the state variable is zeros, the system does not work i.e the system will always output zero. This state could not hold regarding the adding value 0.1. On the other hand, the variable $X_{7,3}$ can take as maximum value 2^{32}, then the maximum value of the fraction $X_{7,3}/2^{28}$ is $2^4 = 16$ which lead to the conclusion that the value of m is bounded by 7 in the case where $X_{7,3} = 0$ and 21. Then chaotic behaviors are guaranteed.

Once the state variables, as well as the memory, are initialized. The system is iterated 12 times without producing any output. The goal of this process is to prevent a certain attack such as Guess and determine attack which is based on the fact that some keystream outputs do not depend on the entire input.

B. keystream generation phase

The keystream generation phase consists on two functions: 1) The next update function which updates the state variables of NLFSR, 2) the output function which yields 16-bits as output. Fig. 2 illustrates the diagram of the keystream generator.

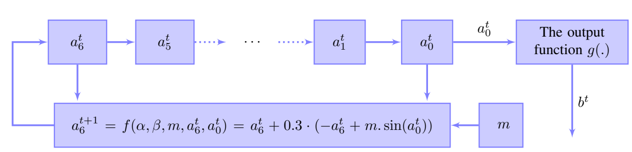

Fig. 2. General structure of the keystream generator

Next update function: In each iteration the state variables a^t_i of the NLFSR are updated to a^{t+1}_i, all these variables are shifted of one position to the right while the value by the most left register is updated in nonlinear way as follows :

$$a^{t+1}_6 = a^t_6 + 0.3 \cdot (-a^t_6 + m \cdot sin(a^t_0)) \quad (3)$$

To illustrate the concept, Let the state of the NLFSR at time t denoted by $s_t = (a_6^t, a_5^t, a_4^t, a_3^t, a_2^t, a_1^t, a_0^t)$. By iterating the system one time, the new state at $t+1$ will be $s_{t+1} = (a_6^t + 0.3(-a_6^t + m.\sin(a_0^t)), a_6^t, a_5^t, a_4^t, a_3^t, a_2^t, a_1^t)$.

Output function: At each time t, NLFSR outputs the value of the most right register i.e a_0^t. The output function takes a_0^t as input and outputs b^t which computed as follows:

$$b^t = e\, mod\, 2^{16},$$

where e represents the fraction part of a_0^t
The idea behind this function is to follow the aspect of a one-way function which means that for a given output, one is not able to derive the input despite that the function is known which is, in turn, increases the security level of the cipher. To give an example, assume that the NLFSR output $a_0^t = 7.1234567891$, so the fraction part of this output is $e = 1234567891$ and the output of $g()$ is $b^t = 723$.

III. Security analysis

In this section, we investigate the security level of the proposed cipher against well known cryptanalytic methods.

A. Statistical tests

The goal of the designer of a stream cipher is to design a keystream generator which produces a keystream sequence that should be indistinguishable from truly random sequences and should not leak any information about the secret key and the internal state of the cipher. Actually, if the keystream generated sequence does not have a random behavior then the generator is susceptible to distinguishable attack (and perhaps also to a key recovery attack). In practice, the randomness analysis relies extremely on the empirical tests of randomness. Each test evaluates the randomness from a specific viewpoint, by testing certain statistic characteristic. The majority of the empirical tests are based on hypothesis tests. Therefore each test is constructed to examine the null hypothesis, namely that the sequence being tested is random from the specific viewpoint of the test. The result of the statistical tests of randomness is described in term of p-value which represents the probability that a perfect random generator would produce a sequence with less randomness than the sequence being testing. In order to evaluate a test, a p-value is compared to a significant value α. If the p-value is greater than α, thus the null hypothesis is accepted otherwise it is rejected. The value of α is commonly set to a small value typically 0.01. Since the statistical randomness can be tested from several viewpoints the statistical tests can be classified into several tests suite. The well-known tests NIST Statistical Test Suite (STS) [16], Diehard [17] and TestU01 [18].
In this context, we use NIST STS to examine the randomness of the output sequence generated by the proposal keystream generator. The reason for choosing NIST STS as tools to evaluate the randomness is relevant to the fact that it has used to evaluate AES cipher and it is often used for formal certification or approvals. In the tests, 200 keystream sequences of length 1000000- bits generated by the keystream generator were empirically evaluated. Table I illustrate the result of the tests. Each row of the table gives the name of the test, the number of tests that was passed out of 200 sequences, the P-value which can be interpreted as probability that a perfect random generator would have produced a sequence less random as the target sequence, and the distribution of the 200 P-value for the individual tests. Results mentioned in the table did not show any deviation from a truly random sequence since all the value of the P-value is greater than the significant value α.

B. Distinguishing attack

The randomness of the keystream is an important requirement for a stream cipher. A bias in the keystream could be exploited to distinguish a keystream from a truly random sequence. This kind of attack is known as distinguishing attack. The easy way to evaluate the randomness of the keystream generated is to use statistical tests. Despite that, the latter play an important role in analyzing the security of the cipher, they are still insufficient to demonstrate whether the cryptosystem is secure, due to the fact that they don't take into account the structure of the generator. For this reason, we introduce in the following, two tests to measure the effect of the key in the keystream generation process.

1) Correlation analysis: The aim of this test is to examine the correlation between the key and the first bits of keystream sequence. High correlation between them may allow an attacker to disclose the secret key or reduce the search key space. To illustrate, assume a KSG with a key of l-bits, during this experience, T random keys denoted $K_1, K_2, ..., K_T$ are chosen. l first keystream bits derived from each key K_i is computed and denoted KS_i. Then each key is XORed with the corresponding keystream bits to form the variable $XR_i = K_i \oplus KS_i$. Once the T variables XR_i are obtained, the weight of each XR_i denoted W_i is computed where the weight consist of counting the number of a bit 1. Then the obtained T weights W_i are classified into 5 categories. The distribution of the W_i is binomial i.e $\text{Bin}(l, 1/2)$. To assess this test a χ^2 test of Goodness of Fit is applied which has the following form:

$$\chi^2 = \sum_{i=1}^{5} (o_i - e_i)^2 / e_i$$

where o_i and e_i represent the observed and the expected frequency for a category i respectively. If the p-value which is relevant to the value of χ^2 is greater than the significant level α, so the test is passed.
In this context, the cipher uses a key of length 256-bits, then the 5 categories are chosen as (i) $0 - 121$, (ii) $122 - 125$, (iii) $126 - 130$, (iv) $131 - 134$ and (v) $135 - 256$. This test is applied for $T = 2^{10}$ and its corresponding p-value is 0.57 which is acceptable result since it is greater than the significant level $\alpha = 0.01$. To be more accurate we repeat this correlation test 100 times and the average of 100 p-values is 0.53 which emphasize that there is no correlation between the key and the generated keystream.

ISBN 978-90-827970-1-5

TABLE I
RESULTS OF NIST STATISTICAL TESTS SUITE ON THE CIPHER OVER 200 SEQUENCES

STATISTICAL TEST	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION
Frequency	21	16	21	22	23	19	19	15	30	14	0.375313	197/200
BlockFrequency	16	26	17	13	20	20	26	21	23	18	0.534146	199/200
CumulativeSums	22	18	19	20	22	18	22	23	19	17	0.991468	198/200
Runs	18	19	29	15	19	20	27	21	17	15	0.366918	198/200
LongestRun	21	17	16	19	20	24	25	14	17	27	0.524101	198/200
Rank	35	24	9	15	14	14	14	25	21	29	0.000422	199/200
FFT	22	20	24	22	20	10	23	20	17	22	0.605916	198/200
NonOverlappingTemplate	28	22	12	18	16	24	30	13	19	18	0.064822	199/200
OverlappingTemplate	15	30	16	18	13	21	24	25	16	22	0.171867	194/200
Universal	18	17	17	27	26	19	18	19	17	22	0.709558	199/200
ApproximateEntropy	25	25	20	28	19	12	20	16	22	13	0.191687	197/200
RandomExcursions	8	15	14	13	12	9	16	9	8	15	0.454224	119/119
RandomExcursionsVariant	10	16	12	8	16	15	10	10	12	10	0.599316	119/119
Serial	27	20	18	24	19	13	11	20	26	22	0.213309	198/200
LinearComplexity	16	18	18	17	22	25	29	15	23	17	0.410055	197/200

2) Diffusion analysis: Diffusion analysis for a stream cipher allows to determinate the sensitive of the output for a change in the input. In a stream cipher with a good diffusion property, if a single bit is flipped in the key, the outputs keystream changes in an unpredictable manner and every bit in the output keystream have the probability one half to be changed. This is defined as the Strict Avalanche Criterion-r (SAC-r) and Strict Avalanche Criterion-c (SAC-c).

SAC-r Diffusion test

The purpose of this test is to check whether one-half of keystream bits is changed for any flipped bit in the key. To illustrate this concept, assume a keystream generator used a key of length n, and let an error vector $e_i = (0, \ldots, 1, \ldots, 0)$ where 1 is located in the i^{th} position. This test can be performed as follows: a random key K is chosen, and the L bits of the associated keystream is generated. Then the first bit of the key is flipped by XORing the key and the error vector e_1 i.e $K = K \oplus e_1$ and used as input to the keystream generator to yield the associate keystream. The new keystream and the original one are XORed and stored in the first row r_1 in the matrix M_1. This process is repeated for $R-1$ random different keys and the derived result for each key is stored in a new row r_i $(i \neq 1)$ in M_1. Once M_1 is constructed, the weight of each row W_i^r is computed which mean the number of bits with value 1 i.e $W_i^r = \#_1(r_i)$. As result, R weights $W_1^r, \ldots, W_R^r$ are obtained which are then classified into 5 categories and the frequency of each category is computed. For a secure cryptosystem, the frequency value must follow a multinomial distribution with two parameters R and p_i where the later represents the probability of a value to belong to the i^{th} category and determined by using cumulative distribution. Finally, the χ^2 test of Goodness of Fit is applied and the p-value is calculated. This experience determines the sensitivity of the output by flipping the first bit of the key. So in order to evaluate the effect on the other input bits, on should repeat the previous test by applying the other considering error vector e_i where $i \neq 1$.

In this context, the proposed scheme used a key of length 256-bits, so we have applied the SAC-r diffusion test on the scheme for $n = 256, L = 2^{10}$ and $R = 2^{10}$. So the appropriate categories are chosen like this (i) $0-498$, (ii) $499-507$, (iii) $508-516$, (iv) $517-525$ and (v) $526-1023$. The result of this test which investigates the impact of all the key bits on the keystream outputs bits is illustrated in Table II. According to this table, we notice that the p-values corresponding to each input bit are greater than $\alpha = 0.01$. Therefore, the cipher pass this test.

IV. CONCLUSION

In this contribution, a word based stream cipher based on Ikeda system is introduced. This cipher is designed in a way that the well-known attacks are infeasible by well designing the setup phase which ensures the diffusion property and selecting the appropriate parameters of the system which ensure good statistical properties and long period.

REFERENCES

[1] P. Stavroulakis, *Chaos Applications in Telecommunications*. Boca Raton, USA: CRC Press, 2005.

[2] S. Behnia, A. Akhshani, H. Mahmodi, and A. Akhavan, “A novel algorithm for image encryption based on mixture of chaotic maps,” *Chaos solitons and fractals*, vol. 35, 2008.

[3] M. Hell, “On the design and analysis of stream ciphers,” Ph.D. dissertation, Lund University, 2007.

[4] G. Vidal, M. S. Baptista, and H. Mancini, “A fast and light stream cipher for smartphones,” *EPJ-ST*, vol. 8, 2014.

[5] J. Golic, “Cryptanalysis of alleged a5 stream cipher,” Ph.D. dissertation, School of Electrical Engineering, University of Belgrade, Yugoslavia, 1997.

[6] M. Hell, T. Johansson, and W. M. Grain, “A stream cipher for constrained environments,” in *Workshop on RFID and Light-Weight Crypto*, Graz, Austria, 2005.

[7] M. H. S. Khazaei and M. Kiaei, “Distinguishing attack on grain,” *posted on eSTREAM webpage. www.ecrypt.eu.org/stream/*, 2005.

[8] “estream project,” http://www.ecrypt.eu.org/stream/.

[9] P. Ekdahl and T. Johansson, “A new version of the stream cipher snow. selected areas in cryptography,” *Springer*, 2002.

[10] P. Hawkes and G. Rose., “Primitive specification for sober-128,” *IACR*, 2003.

ISBN 978-90-827970-1-5 © EURASIP 2018

TABLE II
SAC-R EXPERIMENTAL RESULT FOR ALL POSSIBLE FLIPPED BITS IN THE KEY

Flip bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p-value	0.12	0.81	0.87	0.90	0.23	0.82	0.59	0.45	0.33	0.15	0.07	0.29	0.07	0.78	0.96	0.65
Flip bit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
p-value	0.28	0.95	0.14	0.78	0.50	0.88	0.09	0.97	0.18	0.76	0.13	0.39	0.86	0.25	0.43	0.95
Flip bit	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
p-value	0.50	0.94	0.95	0.47	0.25	0.95	0.72	0.73	0.16	0.62	0.58	0.39	0.39	0.81	0.02	0.02
Flip bit	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
p-value	0.36	0.64	0.12	0.39	0.55	0.78	0.21	0.75	0.64	0.26	0.94	0.57	0.05	0.26	0.15	0.28
Flip bit	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
p-value	0.94	0.42	0.42	0.83	0.45	0.40	0.50	0.65	0.38	0.17	0.39	0.47	0.32	0.17	0.08	0.65
Flip bit	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
p-value	0.93	0.11	0.66	0.29	0.48	0.42	0.37	0.17	0.65	0.51	0.70	0.16	0.97	0.77	0.09	0.25
Flip bit	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
p-value	0.64	0.13	0.15	0.65	0.38	0.89	0.37	0.18	0.77	0.88	0.28	0.15	0.98	0.46	0.12	0.60
Flip bit	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
p-value	0.16	0.46	0.85	0.59	0.71	0.55	0.43	0.45	0.68	0.59	0.25	0.14	0.70	0.14	0.70	0.76
Flip bit	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
p-value	0.33	0.81	0.69	0.22	0.27	0.51	0.45	0.34	0.72	0.27	0.83	0.12	0.58	0.80	0.08	0.40
Flip bit	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
p-value	0.21	0.60	0.87	0.19	0.10	0.97	0.26	0.36	0.96	0.44	0.41	0.67	0.10	0.84	0.43	0.67
Flip bit	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
p-value	0.48	0.86	0.27	0.43	0.04	0.54	0.37	0.82	0.73	0.59	0.91	0.72	0.28	0.80	0.06	0.63
Flip bit	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
p-value	0.06	0.85	0.43	0.38	0.74	0.30	0.77	0.54	0.06	0.42	0.60	0.53	0.85	0.49	0.56	0.31
Bit flip	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
p-value	0.67	0.89	0.01	0.99	0.92	0.48	0.95	0.31	0.71	0.68	0.30	0.26	0.22	0.27	0.31	0.60
Flip bit	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
p-value	0.50	1.00	0.07	0.73	0.39	0.14	0.14	0.54	0.78	1.00	0.99	0.35	0.15	0.88	0.32	0.19
Flip bit	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
p-value	0.21	0.06	0.16	0.75	0.51	0.60	0.95	0.98	0.32	0.60	0.68	0.16	0.22	0.11	0.57	0.53
Flip bit	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256
p-value	0.05	0.96	0.28	0.60	0.07	0.02	0.18	0.04	0.35	0.03	0.93	0.39	0.56	0.13	0.82	0.91

[11] K. Ikeda, "Multilpe-valued stationnary state and its instability of the transmitted light by a ring cavity system," *Optics Communications*, vol. 3, pp. 257–261, 1979.

[12] Lakshmanan, Muthusamy, Senthilkumar, and D. Vijayan, *Dynamics of Nonlinear Time Delay Systems*. Springer, 2011.

[13] H. M. Gibbs, F. A. Hopf, D. L. Kaplan, and R. L. Shoemaker, "Observation of chaos in optical bistability," *PHYSICAL REVIEW LETTERS*, 1981.

[14] T.Aida, "Oscillation modes of laser diode pumped hybrid bistable system with large delay and application to dynamical memory," *Quantum Electronics*, 1992.

[15] O. Mannai, R. Bechikh, H. Hermassi, R. Rhouma, and S. Belghith, "A new image encryption scheme based on a simple first-order time-delay system with appropriate nonlinearity," *Nonlinear Dynamics*, vol. 82, 2015.

[16] A. RUKHIN, J. SOTO, J. NECHVATAL, M. SMID, E. BARKER, M. L. S. LEIGH, M. VANGEL, D. BANKS, A. HECKERT, J. DRAY, and S. VO, "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications, version sts- 2.1," http://csrc.nist.gov/ publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf, April, 2010.

[17] R. G. BROWN, "Dieharder: A random number test suite version 3.31.1," 2004.

[18] P. L'ECUYER and R. SIMARD, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, 200.