Yet Another Pseudorandom Number Generator

Borislav Stoyanov, Krzysztof Szczypiorski, and Krasimir Kordov

Abstract—We propose a novel pseudorandom number generator based on Rössler attractor and bent Boolean function. We estimated the output bits properties by number of statistical tests. The results of the cryptanalysis show that the new pseudorandom number generation scheme provides a high level of data security.

Keywords—Rössler attractor, bent Boolean function, pseudorandom number generator

I. INTRODUCTION

BOOLEAN and chaotic functions have been used extensively in the area of a pseudorandom number generations. Novel encryption scheme based on bent Boolean function and feedback with carry shift register is proposed in [32]. In [27], a cryptographic algorithm based on the Lorenz chaotic attractor and 32 bit bent Boolean function is presented.

In [8], a new chaotic system with good cryptographic properties, is proposed. Novel pseudorandom generation algorithm based on Chebyshev polynomial and Tinkerbell map, is provided in [28]. Pseudorandom bit generators, based on the Chebyshev map and rotation equation, are proposed in [29], [33], and [34]. The presented schemes exhibit high level of security. The proposed scheme shows that the output stream possesses suitable properties for security-demanding applications. A modified pseudorandom bit generator, based on Tinkerbell map, is presented in [16]. Pseudorandom zero-one generation algorithm based on two chaotic Circle maps and XOR function is designed in [13]. In [14], pseudorandom number generation scheme, based on Signature attractor is presented.

Pseudorandom bit generation algorithms, based on Circle map and based on Chirikov map are proposed in [30] and in [31].

Several scientific papers present cryptographic primitives build from Rössler attractor.

An algorithm for secure data transmission with Rössler function protection of the information signal is presented in [6]. In [10], an improvement to an existing algorithm used in security data sending by modifying the structure of the Rössler map behaviour, is designed.

A fingerprint image encryption method based on hyperchaotic Rössler map is provided in [1]. The statistical analysis is presented to prove the secrecy of the biometric trait by using novel scheme.

This work is partially supported by the Scientific research fund of Konstantin Plesslavsky University of Shumen under the grant No. RD-08-121/06.02.2017.

Borislav Stoyanov and Krasimir Kordov are with the Department of Computer Informatics, Konstantin Plesslavsky University of Shumen, Bulgaria (e-mails: borislav.stoyanov@shu.bg, krasimir.kordov@shu.bg).

Krzysztof Szczypiorski is with Warsaw University of Technology, Warsaw, Poland; Cryptomage SA, Wroclaw, Poland (e-mail: ksz@tele.pw.edu.pl).

A communication scheme to encrypted audio and image information transmission, based on hyperchaotic generalized Hénon and Rössler maps is designed in [2]. The scheme is suitable in master-slave configuration. Another communication scheme with high stability in the recovered signal, based on Rössler circuit, is presented in [11].

The stability of impulsive synchronization of chaotic and Rössler hyperchaotic systems by using Lyapunov exponent of the variational synchronization error systems are studied in [12].

An algorithm to image encryption by using a Rössler chaotic function is presented in [18]. The approach consists of two substitution methods and two scrambling methods to change the value of the pixels and the location of the pixels, respectively.

A chaotic permutation for physical layer security in OFDM-PON is presented in [15]. An encryption algorithm based on improved hyperchaotic Rössler map is proposed in [20]. The proposed scheme is attractive for applications in private communication systems.

In [25], design and simulation of synchronization between two identical coupled Rössler circuits, are proposed.

In [4], based on the Rössler attractor, random sequences generator is proposed. The algorithm is SIMULINK modelled and is tested using statistical tests. Random number generator from Rössler attractor also is presented in [5]. The output chaotic signal proposes a negligible value of an autocorrelation function.

A pseudorandom number generator is proposed in this paper. The novel algorithm is based on chaotic function and bent Boolean function. The novelty of our approach lies in the combination of the Rössler attractor and Maiorana function.

In Section 2, we propose novel pseudorandom number generator and its security analysis. Finally, the last section concludes the article.

II. PSEUDO-RANDOM BIT GENERATOR FROM RÖSSLER ATTRACTOR

A. Rössler chaotic attractor

The famous Rössler attractor is presented in [23], Eq. (1):

\[
\begin{align*}
    \frac{dx}{dt} &= -y - z \\
    \frac{dy}{dt} &= x + ay \\
    \frac{dz}{dt} &= b + z(x - c)
\end{align*}
\]

where the parameters \(a\), \(b\), and \(c\) are positive real numbers. The system is chaotic when \(a = 0.2\), \(b = 0.2\), and \(c = 5.7\). Three-Dimensional model of Rössler Attractor is illustrated in Fig. 1. Fig. 2 shows 2-Dimensional plot of Rössler Attractor. Sensitivity to initial conditions are shown in Fig. 3.
B. Bent Boolean Functions

In this subsection we refer to works of Cusick and Stǎnicǎ [7], Neumann [19], Pommerening [21], and Rothaus [22].

Definition 1: A Boolean function \( f \) in \( n \) variables is map from \( \mathbb{V}_n \) (the vector space on \( n \) dimension) to the two-element Galois field \( \mathbb{F}_2 \). The \((0,1)\)-sequence defined by 
\[
(f(x_0), f(x_1), \ldots, f(x_{2^n-1}))
\]
where \( v_0 = (0, \ldots, 0) \), \( v_1 = (0, \ldots, 0, 1) \), \( \ldots \), \( v_{2^n-1} = (1, \ldots, 1, 1) \), ordered by lexicographical order.

To each Boolean function \( f : \mathbb{V}_n \rightarrow \mathbb{F}_2 \) we associate sign function, denoted by \( \hat{f} : \mathbb{V}_n \rightarrow \mathbb{R}^* \subseteq \mathbb{C}^* \) and defined by 
\[
\hat{f}(x) = (-1)^{f(x)}.
\]

Definition 2: The Walsh transform of a function \( f \) on \( \mathbb{V}_n \) (with the values of \( f \) taken to be real numbers 0 and 1) is the map \( W(f) : \mathbb{V}_n \rightarrow \mathbb{R} \), defined by 
\[
W(f)(w) = \sum_{x \in \mathbb{V}_n} f(x)(-1)^{w \cdot x}
\]
which defines the coefficients of \( f \) with respect to the orthonormal basis of the group characters \( Q_x(w) = (-1)^{w \cdot x} \) (where \( w \cdot x \) is the scalar product); \( f \) can be recovered by the inverse Walsh transform
\[
f(x) = 2^{-n} \sum_{w \in \mathbb{V}_n} W(f)(w)(-1)^{w \cdot x}
\]

Definition 3: A Boolean function \( f \) in \( n \) variables is called bent if and only if the Walsh transform coefficients of \( \hat{f} \) are all \( \pm 2^{n/2} \), that is, \( W(f)^2 \) is constant.

Maorana construction provides us with the following bent function 
\[
x_0 y_0 \oplus \ldots \oplus x_{m-1} y_{m-1} \oplus x_0 x_1 \cdots x_{m-1},
\]
where the Boolean function is presents by the polynomial 
\[
P(x_0, \ldots, x_{m-1}, y_0, \ldots, y_{m-1}) = R(x_0, \ldots, x_{m-1}) \oplus x_0 y_0 \oplus \ldots \oplus x_{m-1} y_{m-1} \text{ and its dual polynomial}
P^*(x_0, \ldots, x_{m-1}, y_0, \ldots, y_{m-1}) = R(x_0, \ldots, x_{m-1}) \oplus x_0 y_0 \oplus \ldots \oplus x_{m-1} y_{m-1},
\]
where \( R = \mathbb{R}(m) \) is an arbitrary polynomials in \( m \) variables.
C. Proposed Bit Generator

The novel algorithm is based on the following steps:

Step 1: The initial values $x_0$, $y_0$, and $z_0$ from Eq. (1) are determined.

Step 2: With using the third-order Runge-Kutta method [9], we numerically compute the attractor from Eq. (1) with step size $h = 0.01$. It is iterated for $L_1$ times.

Step 3: The iteration of the Eq. (1) continues, and as a result, two real fractions $x_i$ and $y_i$, are generated and post-processed as follows:

$$
\begin{align*}
    s_0 &= \text{abs}(\text{integer}(x_i \times 10^7)) \\
    s_1 &= \text{abs}(\text{integer}(y_i \times 10^7)),
\end{align*}
$$

where $\text{integer}(x)$ returns the integer part of $x$, truncating the value at the decimal point and $\text{abs}(x)$ returns the absolute value of $x$.

Step 4: Generate a numeric vector $V = (s_0[0], ..., s_p[0], s_1[0], ..., s_1[p])$ that contains the digits of the numbers $s_0$ and $s_1$.

Step 5: Apply the vector $V$ to Maiorana function from Eq. (2) to get a single output bit.

Step 6: Return to Step 3 until the bit stream limit is reached.

The proposed bit generator is implemented in C++, using the following initial values: $x_0 = 0.1$, $y_0 = 0.15$, $z_0 = 0.01$, and $L_1 = 2000$.

D. Key space evaluation

The secret key space is composed by the four secret keys $x_0$, $y_0$, $z_0$, and $L_1$. With number of about 15 decimal digits precision in IEEE double precision [36] the proposed key space is more than $2^{126}$, which is good enough against exhaustive key search [3].

E. Statistical tests

Three software test programs are used in order to measure the behaviour of the output binary streams.

The DIEHARD package [17] includes 19 statistical tests: Birthday spacings, Overlapping 5-permutations, Binary rank (31 x 31), Binary rank (32 x 32), Binary rank (6 x 8), Bitstream, Overlapping-Pairs-Sparse-Occupancy, Overlapping-Quadruples-Sparse-Occupancy, DNA, Stream count-the-ones, Byte-count-the-ones, Parking lot, Minimum distance, 3D spheres, Squeeze, Overlapping sums, Runs (up and down), and Craps. The tests return $P-values$, which should be uniform in [0,1], if the input file contains pseudorandom numbers. The $P-values$ are obtained by $p = F(y)$, where $F$ is the assumed distribution of the sample random variable $y$, often the normal distribution.

The NIST software application [24] is a set of 15 statistical tests: Frequency (monobit), Block-frequency, Cumulative sums (forward and reverse), Runs, Longest run of ones, Rank, Fast Fourier Transform (spectral), Non-overlapping templates, Overlapping templates, Maurers ”Universal Statistical”, Approximate entropy, Random excursion, Random-exursion variant, Serial, and Linear complexity.

The testing process consists of the following steps:

Step 1: State the null hypothesis. Assume that the zero/one sequence is random.

Step 2: Compute a sequence test statistic. Testing is carried out at the bit level.

Step 3: Compute the $P-value$, $P-value \in [0,1]$.

Step 4: Fix $\alpha$, where $\alpha \in [0.0001,0.01]$. Compare the $P-value$ to $\alpha$. Success is declared whenever $P-value \geq \alpha$; otherwise, failure is declared.

The NIST package calculates the proportion of sequences that pass the particular tests. The range of acceptable proportion is determined using the confidence interval defined as:

$$
\hat{p} \pm 3 \sqrt{\frac{\hat{p}(1-\hat{p})}{m}},
$$

where $\hat{p} = 1 - \alpha$, and $m$ is the number of binary tested sequences. NIST recommends that, for these tests, the user should have at least 1000 sequences of 1000000 bits each. In our setup $m = 1000$. Thus the confidence interval is

$$
0.99 \pm 3 \sqrt{\frac{0.99(0.01)}{1000}} = 0.99 \pm 0.0094392.
$$

The proportion should lie above 0.9805607 with exception of Random excursion and Random excursion variant tests. These two tests only apply whenever the number of cycles in a sequence exceeds 500. Thus the sample size and minimum pass rate are dynamically reduced taking into account the tested sequences.

The distribution of $P-values$ is examined to ensure uniformity. The interval between 0 and 1 is divided into 10 subintervals. The $P-values$ that lie within each subinterval are counted. Uniformity may also be specified through an application of a $\chi^2$ test and the determination of a $P-value$ corresponding to the goodness-of-fit distributional test on the $P-values$ obtained for an arbitrary statistical test, $P-value$ of the $P-values$. This is implemented by calculating

$$
\chi^2 = \sum_{i=1}^{10} \frac{(F_i - s/10)^2}{s/10},
$$

where $F_i$ is the number of $P-values$ in subinterval $i$ and $s$ is the sample size. A $P-value$ is computed such that $P-value_F = IGAMC(9/2, \chi^2/2)$, where $IGAMC$ is the complemented incomplete gamma statistical function. If $P-value_F \geq 0.0001$, then the sequences can be considered to be uniformly distributed.

The ENT package [35] performs 6 tests to sequences. They are Entropy, Optimum compression, $\chi^2$ distribution, Arithmetic Mean value, Monte Carlo Value for $\pi$, and Serial Correlation Coefficient. The sequences of bytes are stored in files. The suite outputs the results of those tests. We tested output stream of 125000000 bytes of the novel pseudorandom number generator.

The test results are given in Table I, Table II, and Table III, respectively. All of statistical tests are passed successfully.

III. Conclusions and Future Work

A novel pseudorandom number generator based on a chaotic map is proposed in this article. The proposed algorithm combines Rössler attractor and Maiorana bent Boolean function.
### Table I

**DIEHARD Statistical Test Results for Two 80 Million Bits Sequences Generated by the Proposed Generator**

<table>
<thead>
<tr>
<th>DIEHARD Statistical Test</th>
<th>Proposed Generator</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birthday spacings</td>
<td></td>
<td>0.593701</td>
</tr>
<tr>
<td>Overlapping S-permutation</td>
<td></td>
<td>0.395409</td>
</tr>
<tr>
<td>Binary rank (31 x 31)</td>
<td></td>
<td>0.746323</td>
</tr>
<tr>
<td>Binary rank (32 x 32)</td>
<td></td>
<td>0.955445</td>
</tr>
<tr>
<td>Binary rank (6 x 8)</td>
<td></td>
<td>0.479493</td>
</tr>
<tr>
<td>Ristream</td>
<td></td>
<td>0.922760</td>
</tr>
<tr>
<td>OQSO</td>
<td></td>
<td>0.569775</td>
</tr>
<tr>
<td>DNA</td>
<td></td>
<td>0.509277</td>
</tr>
<tr>
<td>Stream count-the-ones</td>
<td></td>
<td>0.213018</td>
</tr>
<tr>
<td>Byte count-the-ones</td>
<td></td>
<td>0.612866</td>
</tr>
<tr>
<td>Parking lot</td>
<td></td>
<td>0.428480</td>
</tr>
<tr>
<td>Minimum distance</td>
<td></td>
<td>0.479449</td>
</tr>
<tr>
<td>3D spheres</td>
<td></td>
<td>0.470514</td>
</tr>
<tr>
<td>Squeeze</td>
<td></td>
<td>0.935011</td>
</tr>
<tr>
<td>Overlapping sums</td>
<td></td>
<td>0.616515</td>
</tr>
<tr>
<td>Runs up</td>
<td></td>
<td>0.317489</td>
</tr>
<tr>
<td>Runs down</td>
<td></td>
<td>0.347915</td>
</tr>
<tr>
<td>Craps</td>
<td></td>
<td>0.587091</td>
</tr>
</tbody>
</table>

### Table II

**NIST Statistical Test Suite Results for 1000 Sequences of Size 10^6-bit Each Generated by the Proposed Generator**

<table>
<thead>
<tr>
<th>NIST Statistical Test</th>
<th>Proposed Generator</th>
<th>P-value</th>
<th>Pass Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (monobit)</td>
<td></td>
<td>0.896345</td>
<td>991/1000</td>
</tr>
<tr>
<td>Block-frequency</td>
<td></td>
<td>0.348869</td>
<td>988/1000</td>
</tr>
<tr>
<td>Cumulative sums (Forward)</td>
<td></td>
<td>0.095426</td>
<td>992/1000</td>
</tr>
<tr>
<td>Cumulative sums (Reverse)</td>
<td></td>
<td>0.632955</td>
<td>993/1000</td>
</tr>
<tr>
<td>Runs</td>
<td></td>
<td>0.195864</td>
<td>992/1000</td>
</tr>
<tr>
<td>Longest run of Ones</td>
<td></td>
<td>0.597620</td>
<td>990/1000</td>
</tr>
<tr>
<td>Rank</td>
<td></td>
<td>0.587274</td>
<td>992/1000</td>
</tr>
<tr>
<td>FFT</td>
<td></td>
<td>0.849708</td>
<td>987/1000</td>
</tr>
<tr>
<td>Non-overlapping templates</td>
<td></td>
<td>0.505628</td>
<td>990/1000</td>
</tr>
<tr>
<td>Overlapping templates</td>
<td></td>
<td>0.308561</td>
<td>986/1000</td>
</tr>
<tr>
<td>Universal</td>
<td></td>
<td>0.474986</td>
<td>988/1000</td>
</tr>
<tr>
<td>Approximate entropy</td>
<td></td>
<td>0.973718</td>
<td>987/1000</td>
</tr>
<tr>
<td>Random-excursions</td>
<td></td>
<td>0.476145</td>
<td>629/635</td>
</tr>
<tr>
<td>Random-excursions Variant</td>
<td></td>
<td>0.502145</td>
<td>630/635</td>
</tr>
<tr>
<td>Serial 1</td>
<td></td>
<td>0.707513</td>
<td>997/1000</td>
</tr>
<tr>
<td>Serial 2</td>
<td></td>
<td>0.729870</td>
<td>987/1000</td>
</tr>
<tr>
<td>Linear complexity</td>
<td></td>
<td>0.096578</td>
<td>993/1000</td>
</tr>
</tbody>
</table>

### Table III

**ENT Statistical Test Results for Two 80 Million Bits Sequences Generated by the Proposed Generator**

<table>
<thead>
<tr>
<th>ENT Statistical Test</th>
<th>Proposed Generator results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>7.999998 bits per byte</td>
</tr>
<tr>
<td>Optimum compression</td>
<td>OC would reduce the size of this 125000000 byte file by 0%. For 125000000 samples is 271.65, and randomly would exceed this value 22.62% of the time.</td>
</tr>
<tr>
<td>χ² distribution</td>
<td>127.4991 (127.5 = random)</td>
</tr>
<tr>
<td>Arithmetic mean value</td>
<td>3.141880562 (error 0.01%)</td>
</tr>
<tr>
<td>Monte Carlo π estim.</td>
<td>-0.000089 (totally uncorrelated = 0.0)</td>
</tr>
</tbody>
</table>

An accurate security analysis on the novel scheme is given. Based on the results, we can conclude that the proposed pseudorandom number generation algorithm is acceptable for the secure data encryption.

We intend to use this algorithm for Galois field based encryption [26].
ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for valuable and helpful comments.

The authors would like to thank Borislav Bedzhev and Stanimir Stanev for their comments and suggestion on earlier drafts of this paper.

REFERENCES


