A novel S-Box based post-processing method for true random number generation

Erdinç AVAROĞLU 1*, Taner TUNCER 2

1Department of Computer Engineering, Faculty of Engineering, Mersin University, Mersin, Turkey,
ORCID iD: https://orcid.org/0000-0003-1976-2526
2Department of Computer Engineering, Faculty of Engineering, Fırat University, Elazığ, Turkey,
ORCID iD: https://orcid.org/0000-0003-0526-4526

Received: .201 • Accepted/Published Online: .201 • Final Version: .201

Abstract: The quality of randomness in numbers generated by True Random Number Generators (TRNGs) depends on the source of entropy. However, in TRNG, sources of entropy are affected by environmental changes and this creates a correlation between the generated bit sequences. Post-processing is required to remove the problem created by this correlation in TRNG. In this study, the S-Box based post-processing structure is proposed as an alternative to the post-processing structures seen in published literature. A Ring Oscillator (RO)-based TRNG is used to demonstrate the use of S-Box for post-processing and the removal of correlations between number sequences. The statistical properties of the numbers generated through post-processing are obtained according to the entropy, autocorrelation, statistical complexity measure and the NIST 800.22 test suite. According to the results, the post-processing was shown to have successfully removed the correlation. Also, the data rate of the bit sequence generated by the proposed post-processing is reduced to 2/3 of its original value at the output.

Key words: Ring Oscillator, Random Number, Post Processing, S-Box, Statistical Test

1. Introduction

Randomness is the absence of a specific pattern or predictability in phenomena. Systems that are not random are deterministic. Random numbers are defined as numbers that have no relation to each other, whose occurrence probabilities are equal, and that are defined over a specific interval. Random numbers are used in many areas, such as simulation, sampling, numerical analysis, entertainment and cryptography. Random numbers are especially important in cryptographic systems. In cryptography, random numbers are needed for key generation and distribution, the generation of starting vectors, for identity verification protocols, for prime numbers and for password generation. Security of a cryptographic system relies on the true randomness of the numbers obtained. For this reason, random number generators form the most critical aspect of cryptography [1]. Random number generators are divided into 2 classes, Pseudo Random Number Generators (PRNGs) and True Random Number Generators (TRNGs). Pseudo Random Number Generators are algorithms that create number sequences with characteristics similar to random numbers. These number sequences are calculated deterministically, using a initial value called the seed. No PRNG generates true random numbers because all PRNGs use a deterministic algorithm. For this reason, PRNGs are not suitable for cryptography applications. In TRNGs, a non-deterministic source of entropy is used to generate random numbers. In TRNG designs, sampling is achieved by digitizing the signals obtained from a source of entropy. After the sampling process,
generated bit sequences are post-processed in order to make the bit sequences independent of each other and remove any correlation between them. As shown in Figure 1, a TRNG is formed by 3 basic units. These are, in order, the source of entropy, the sampler and the post-processing [1, 2].

The true randomness of TRNGs is completely dependent on the source of entropy. If the source of entropy is of high quality, numbers generated by the TRNG have good statistical properties. Human-driven interaction and physical processes such as noise based on thermal, shot and atmospheric sources, and metastability-containing circuits, jitter and Brownian motion are used as sources of entropy in published literature [1]. In the sampling unit, random signals generated by the source of entropy are transformed into digital signals. For the sampling unit, periodic sampling, mixing, or aperiodic sampling circuits are used in published literature [1]. Post-processing is usually used to remove correlation to the source of entropy and increase the randomness of the signal. Another advantage of post-processing is that it makes the system more robust against environmental changes and side-channel attacks. Security of the generator is increased by removing correlation due to the post-processing algorithm. However, post-processing decreases the data rate of generated numbers as a result of removing correlation in the bit sequences. Post-processing is used to remove correlation in the source of entropy in TRNG systems. Most of the post-processing algorithms proposed in published literature reduce the data rate of the TRNG. For example, the data rate of an XOR corrector, Von Neumann corrector, H function and resilient function are reduced by a 1/2 $^{1}$, a 1/4 (approximate) [4], a 1/2 [5], and by 1/16, respectively. In this study, a novel post-processing algorithm is proposed that removes the statistical weaknesses and reduces the data rate of TRNG to 2/3 of the original. The proposed post-processing is performed by Substitution Boxes (S-box), which are used in the safe design of block encryption systems. S-boxes are non-linear bijective (one-to-one and surjective) transformations that change an n-bit input value to an m-bit output value. In this study, an RO-based TRNG system is defined and S-box post-processing is used to remove correlation between the sequences of numbers generated by this number generator. The RO-based TRNG uses the Sunar structure [6]. Statistical tests on the numbers are obtained according to the NIST 800.22 test suite. The rest of this study is organized as follows: Section 2 describes related work about post-processing in published literature. The proposed post-processing algorithms are explained in Section 3. Section 4 outlines the proposed use of S-box for post-processing. In Section 5, statistical test results, Scale Index and autocorrelation results are produced by the NIST 800.22 test suite in order to demonstrate the usability of random numbers obtained in real-time for cryptographic systems. In the conclusion Section, interpretation of the statistical tests of the proposed system is discussed by mentioning the advantages and disadvantages of the proposed system.

---

$^{1}$Davies RB.(2002) Exclusive OR (XOR) and hardware random number generators [online]. Website http://www.robertnz.net/pdf/xor2.pdf [accessed 14 April 2019]
2. Related Work

Various post-processing algorithms have been proposed to mitigate the statistical shortcomings of numbers generated by TRNG systems. Some of these are the XOR corrector \(^1\), Von Neumann corrector \(^4\), H function \(^5\), SHA-1 \(^2\) and resilient functions \(^6\). The SHA-1 post-processing algorithm, together with Von Neumann, was used in an Intel TRNG, one of the very first TRNG designs. A thermal noise source was used as the source of entropy in the Intel TRNG. The statistical properties of numbers obtained by sampling from the system were not good. For this reason, the randomness properties of these numbers were improved through processing the numbers first with Von Neumann post-processing and then the SHA-1 algorithm. Numbers obtained from this system successfully passed the FIPS 140-1 test. In another study, a resilient function was used for post-processing in an RO-based TRNG system proposed by Sunar \(^6\). In this TRNG design 114 Ring Oscillators (RO), each having 13 inverters, were used as the source of entropy. Using too many RO circuits in the system caused various problems. These problems included high energy consumption, a reduction in the quality of randomness, and correlation in the output because the ring oscillators were not independent of each other. Therefore, the obtained bit sequence was post-processed using the resilient function. The data rate of the TRNG was reduced to 1/16 due to the resilient post-processing used in the system. There have been RO-based TRNG designs – used as a source of entropy – and various post-processing methods have been used in these designs. In the TRNG system developed by \(^7\), an oscillator ring with two transparent latches, a buffer and an inverter was used. The output of the TRNG was post-processed using an XOR function. For this reason, the data rate was low, and the obtained rate was less than 1Mbit/s. In \(^8\), a TRNG that used a Galois Ring Oscillator (GRO) and Fibonacci Ring Oscillator (FRO) was proposed. Outputs obtained from the FRO and GRO were unified using XOR, and random numbers were generated by sampling the XOR output with a D type flip-flop. The output bit sequence was post-processed by Linear Feedback Shift Register (LFSR)-based post-processing. A bit rate of 12.5 Mbit/s was achieved with this design. In \(^9\), the design described in \(^6\) was realized on a Xilinx Virtex II Pro FPGA (Field Programmable Data Array) using an RO and the inverter numbers (110, 3), (110, 13) and (210, 3). The strongest source of entropy was identified as (210, 3) during tests. The outputs obtained were post-processed using a resilient function and the final output rate was 2 Mbps. In \(^10\), the logistics map is used to improve the statistical properties of numbers that are obtained using an RO-based TRNG. There is no bit rate reduction in this post-processing. Also, successful results were obtained in the NIST tests with the resulting chaotic map. However, the biggest disadvantage of this system is the high energy consumption and low data rate. In \(^11\), performance differences between conventional method of TRNG that used chaotic system and recently designed FPGA based chaotic systems have been compared. In \(^12\), Authors are proposed a novel type of high-speed TRNG based on chaos and ANN implemented in (FPGA) chip. The generated random numbers have been tested with NIST 800.22 and FIPS 140-1 test suites. The high quality of generated numbers have been confirmed by passing all randomness tests. In \(^13\), authors are proposed a new dual entropy core discrete time chaos-based TRNG structure has been implemented on FPGA. The System have been synthesized in FPGA chip. The data rates of the designed dual entropy core TRNG units range between 390-464 Mbps. In \(^14\), Sundarapandian-Pehlivan chaotic system was modeled and simulated to generate random numbers. The frequency of the proposed system is 293.815 MHz and the system can calculate 1,000,000 data in 0.201 s. The proposed system is successful from the FIPS-140-1 and NIST-800-22 test suites.

---

\(^1\) Davies R B. (2002) Exclusive OR (XOR) and hardware random number generators [online]. Website http://www.robertnz.net/pdf/xor2.pdf [accessed 14 April 2019]

\(^2\) Kocher P B. (1999) The intel random number generator [online]. Website https://www.rambus.com/wp-content/uploads/2015/08/IntelRNG.pdf [accessed 30 April 2019]
In [15], a true random number generator is formed by uniformly sampled ring oscillators and using the Hash function for post-processing. Both the generator and SHA-256 are realized on a 5-core Virtex FPGA. As a result of the experiments, it was shown that at least 8 ring oscillators are required to make the system pass all statistical tests. In [16], the true random bit generation is achieved by using the random environmental noise signals coming from the microphone port on a computer sound card. As a post-processing method, a novel procedure for the distribution of bits, “Mixing Bits in Steps and XORing of Adjacent Bits” (MiBiS&XOR), was proposed. The proposed procedure divides input bits that are consecutive and specifically correlated by separating them in a simple and efficient manner. This procedure decreases the cumulative autocorrelation. Experimental statistical randomness tests performed on the bit sequences obtained by this procedure validates the perfect quality of TRNG outputs. In [17], an improvement by applying the N-bit Von Neumann ($V_{N}$) post-processing technique is shown. A general algorithm that explains N-bit $V_{N}$ and the application of 4-bit $V_{4}$ at the circuit level is shown. With $V_{4}$, an output rate of 40.6% was achieved. A new waiting strategy was proposed to further improve the output rate. $V_{4}$+ waiting and $V_{8}$+ waiting achieved 46.9% and 62.5% rates, respectively. When compared with the original Von Neumann ($V_{2}$) technique, they gave results improved by 1.88 and 2.50 times, respectively.

3. Post processing

Post-processing is usually used to increase the randomness in the signal. Post-processed signal values are uniform compared to the original signal. The second purpose of the post-processing – which has gained much importance as a countermeasure to combat side-channel attacks – is to make number generators more robust against tampering and environmental changes. The security of the generator will increase depending on the post-processing algorithm. In published literature, various post-processing algorithms such as XOR corrector, Von Neumann corrector, extractor function, H function and resilient functions are used [4–6]. These post-processing methods are described below.

3.1. XOR Corrector

In the XOR corrector shown in Figure 2. The bit sequence obtained is separated into n-bit ($n = 2$) blocks in order to produce an output bit.

![Figure 2](image-url).
Each separated block is processed on its own with the XOR function. This procedure causes the output bit efficiency to decrease $1/n$ times while it removes bias in the output bit. However, bias in the output bit sequence will only decrease on the condition that the input bits are independent. The advantage of this corrector is that it is simple and can achieve a constant output bit rate.

### 3.2. Von Neumann Corrector

This is the oldest and simplest method to remove irregularities in the bit sequence. As shown in Table 1, Von Neumann generates uniform 0 and 1 bits. It takes into consideration the synchronous pairs coming from the TRNG. If the bit sequence is $(1, 0)$, it produces a 1. If the bit sequence is $(0, 1)$, it produces a 0. Bit sequences $(0, 0)$ and $(1, 1)$ are discarded. The change in the bit rate is shown in Table 2. This corrector makes the entropy close to the ideal value of 1 and hence contributes to its improvement. However, since some bit sequences coming from the TRNG are discarded, the Von Neumann output bit rate relies on the TRNG output and is therefore not constant. The bit rate is reduced to $1/4$ of the input bit rate.

| Input Bit Pairs | Von Neumann Output |
|-----------------|--------------------|
| 00              | No output          |
| 01              | 0                  |
| 10              | 1                  |
| 11              | No output          |

### 3.3. H Function

The H post-processing function shown in Figure 3 is proposed by Dichtl to prevent bias in a TRNG. It is based on the quasigroup sequence transformation. The post-processing algorithm uses 16 bits of the source randomness to obtain an 8-bit output. The input bits for the post-processing are $a_0, a_1, \ldots, a_{15}$. The 8-bit $b_0, b_1, \ldots, b_7$ is defined as $b_i = a_i \oplus a_i + 1 \mod 8$. There are only 128 possible values for $b_0, b_1, \ldots, b_7$. When the input is a perfect random number generator, one bit of the entropy is removed. The output of the post-processing function $c_0, c_1, \ldots, c_7$ is defined as $c_i = b_i \oplus a_i + 8$. This function is called the H function. The 16-bit sequence in the input is divided into two: $A_1 = a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$ and $A_2 = a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}$. The $A_1$ byte is shifted left 1 bit by $rotateleft(A_1, 1)$.

\[ b_i = a_i \oplus a_i + 1 \mod 8 = A_i \oplus rotateleft(A_1, 1) \]  \hspace{1cm} (1)

\[ c_i = H(A_1, A_2) = A_1 \oplus RL(A_1, 1) \oplus A_2 \]  \hspace{1cm} (2)

---

1 DAVIES R B. (2002) Exclusive OR (XOR) and hardware random number generators [online]. Website http://www.robertnz.net/pdf/xor2.pdf [accessed 14 April 2019]
3.4. Extractor Function

The extractor function was proposed by Barak, Shaltiel and Tomer [18] to make the TRNG more robust against changing environmental conditions. This is a very strong function with measurable properties. This strong stateless function with measurable properties was first developed as a tool for complexity theory. In [18], a mathematical model was proposed to capture the impact of an attacker on the randomness source. This provides an open structure based on the correlation of the summary functions proven for the output properties with the input through unknown causes – should such a thing exist. The definition of an extractor is provided below [18]:

An extractor is a function $E : \{0, 1\}^n \times \mathcal{S} \rightarrow \{0, 1\}^m$ for some data set $\mathcal{S}$. Denoted by $E^\pi(\cdot) = E(\cdot, \pi)$ a single-input function that is the result of fixing the parameter $\pi$ to the extractor $E$. It is desirable for the output $E(X, \pi) = E^\pi(X)$ to be (close to) uniformly distributed, where $X \in \{0, 1\}^n$ is the output of the high-entropy source and $\pi \in \mathcal{S}$ is the public attribute.

Defining security, we consider the following ideal setting:

1. An adversary chooses $2^t$ distributions $D_1, \ldots, D_{2^t}$ over $\{0, 1\}^n$, such that $\min - \text{Ent}(D_i) > k$ for all $i = 1, \ldots, 2^t$.

2. A public attribute $\pi$ is chosen at random and is independent of the choices for $D_i$.

3. The adversary is given $\pi$, and selects $i \in \{1, \ldots, 2^t\}$.

4. The user computes $E^\pi(X)$, where $X$ is drawn from $D_i$.

3.5. Resilient Function

A resilient function involves the filtering of deterministic bits. In a study by Sunar, Martin, and Stinson, the affected deterministic bits were used to investigate the tolerance of the properties of the resilient function [6].

The high degree of flexibility and the expected number of deterministic bits in the sample determines the amount of tolerance of the TRNG of an active enemy. The $(n,m,t)$—resilient function is defined as follows:

---

**Figure 3.** H Post-Processing Function [17]
Definition 1 An \((n, m, t)\)-resilient function is a function of the type \(F(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)\) For \(Z_2^n\) to \(Z_2^m\), it has the property that, for any \(t\) coordinates \(i_1, \ldots, i_t\), for any constants \(a_1, \ldots, a_t\) from \(Z_2\) and any element \(y\) of the codomain: \(\text{Prob}[F(x) = y|\{i_1 = a_1, \ldots, i_t = a_t\}] = 1/2^m\) In computing this probability, all \(x_i\) for \(i \notin \{i_1, \ldots, i_t\}\) are viewed as independent random variables, each of which takes on the value 0 or 1 with a probability of 0.5.

4. Proposed Post-Processing

The name S-Box comes from the English word substitution. S-Box usually appears in methods that use one symmetric encryption variant called block encryption. The goal is to determine the topology of a table using the table. S-Box is the only non-linear component in block encryption systems such as DES and AES and provides the property of confusion. That is why, to be able to design strong encryption systems, S-boxes with good cryptological properties must be developed. Regardless of the S-box type or category created, an S-box must display specific attributes in order to be effective. It is not possible to design a good S-box by combining any arbitrary substitution schemes. One way to generate these inputs in S-boxes is to create a non-linear Boolean function which maps the \(n\)-bit input to an \(m\)-bit output. A special Boolean function set, called bent functions, can be used to achieve maximum non-linearity.

In an \(n \times m\) S-box, \(S\) is a mapping where \(S: \{0, 1\}^n \rightarrow \{0, 1\}^m\). \(S\) can be represented as \(2^n \times m\)–bit numbers, denoted \(r_0, \ldots, r_{2^n-1}\), in which case \(S(x) = r_x, 0 \leq x \leq 2^n\) and \(r_i\) are the rows of the S-box. Alternatively, \(S(x) = [c_{m-1}(x), c_{m-2}(x), \ldots, c_0(x)]\) where \(c_i\) are fixed Boolean functions \(c_i: \{0, 1\}^n \rightarrow \{0, 1\} \forall i\); these are the columns of the S-box. Finally, \(S\) can be represented by a \(2^n \times m\) binary matrix \(M\) where the \(i, j\) entry is bit \(j\) of row \(i\). A linear combination of two functions \(f, g: \{0, 1\}^n \rightarrow \{0, 1\}\) is defined to be: \((f \oplus g)(x) = f(x) \oplus g(x)\)

Where \(\oplus\) denotes modulo 2 addition. Let \(V_n\) denote the set of functions mapping \(\{0, 1\}^n \rightarrow \{0, 1\}\). Let \(L_n\) denote the set of linear functions mapping \(\{0, 1\}^n \rightarrow \{0, 1\}\). Let \(An\) denote the set of affine functions mapping \(\{0, 1\}^n \rightarrow \{0, 1\}\). There are other criteria that must be met in the design of an S-box. These criteria are confusion and diffusion. Generally, the design criteria below must be met for a good S-box.

A. Strict Avalanche Criterion

The Strict Avalanche Criterion was first published by Webster and Tavares. A function satisfying the strict avalanche criterion means that changing a single input bit has a probability of changing half of the output bits.

B. Bit Independence Criterion (BIC)

The Bit Independence Criterion requires the output bits to move independently. In other words, a statistical model or dependency between output vectors and output bits must not exist.

For a function \(f: \{0, 1\}^n \rightarrow \{0, 1\}^n\) where \(i, j, k \in \{1, 2, \ldots, n\}\) and \(i \neq k\), for all \(i, j, k\) values, when the input bit \(i\) is inverted, if the output bits \(j\) and \(k\) can be changed independently, BIC is satisfied.

C. Non-linearity Criterion

S-box is not a linear mapping from input to output. If it was, that would make the cryptosystem more vulnerable.
to attacks. When an S-box is formed by Boolean functions with maximum non-linearity, it will be incompatible with linear functions and therefore it becomes harder to break the cryptosystem.

D. Balance

This means that every Boolean vector which is responsible for the S-box contains the same amount of 0s and 1s. It is not possible to satisfy all these criteria simultaneously. A compromise must be made for some of the criteria. For example, the Bit Independence Criterion conflicts with the Non-linearity Criterion, while the Non-linearity Criterion conflicts with the Balance Criterion at the same time. The S-box which is designed for DES is listed in Figure 4.

| S1       |
|----------|
| 14  4  13  1  2  15  11  8  3  10  6  12  5  9  0  7 |
| 0  15  7  4  14  2  13  1  10  6  12  11  9  5  3  8 |
| 4  1  14  8  13  6  2  11  15  12  9  7  3  10  5  0 |
| 15 12  8  2  4  9  1  7  5  11  3  14  10  0  6  13 |
| S2       |
| 15  1  8  14  6  11  3  4  9  7  2  13  12  0  5  10 |
| 3  13  4  7  15  2  8  14  12  0  1  10  6  9  11  5 |
| 0  14  7  11  10  4  13  1  5  8  12  6  9  3  2  15 |
| 13  8  10  1  3  15  4  2  11  6  7  12  0  5  14  9 |
| S3       |
| 10  0  9  14  6  3  15  5  1  13  12  7  11  4  2  8 |
| 13  7  0  9  3  4  6  10  2  8  5  14  12  11  15  1 |
| 13  6  4  9  8  15  3  0  11  1  2  12  5  10  14  7 |
| 1  10  13  0  6  9  8  7  4  15  4  3  11  5  2  12 |
| S4       |
| 7  13  14  3  0  6  9  10  1  2  8  5  11  12  4  15 |
| 13  8  11  5  6  15  0  3  4  7  2  12  1  10  14  9 |
| 10  6  9  0  12  11  7  13  15  1  3  14  5  2  8  4 |
| 3  15  0  6  10  1  13  8  9  4  5  11  12  7  2  14 |
| S5       |
| 7  12  4  1  7  10  11  6  8  5  3  15  13  0  14  9 |
| 14  11  2  12  4  7  13  1  5  0  15  10  3  9  8  6 |
| 4  2  1  11  10  13  7  8  15  9  12  5  6  3  0  14 |
| 11  8  12  7  1  14  2  13  6  15  0  9  10  4  5  3 |
| S6       |
| 12  1  10  15  9  2  6  8  0  13  3  4  14  7  5  11 |
| 10  15  4  2  7  12  9  5  6  1  13  14  0  11  3  8 |
| 9  14  15  5  2  8  12  3  7  0  4  10  1  13  11  6 |
| 4  3  2  12  9  5  15  10  11  14  1  7  6  0  8  13 |
| S7       |
| 4  11  2  14  15  0  8  13  3  12  9  7  5  10  6  1 |
| 13  0  11  7  4  9  1  10  14  3  5  12  2  15  8  6 |
| 1  4  11  13  12  3  7  14  10  15  6  8  0  5  9  2 |
| 6  11  13  8  1  4  10  7  9  5  0  15  14  2  3  12 |
| S8       |
| 12  2  8  4  6  15  11  1  10  9  3  14  5  0  12  7 |
| 1  15  13  8  10  3  7  4  12  5  6  11  0  14  9  2 |
| 7  11  4  1  9  12  14  2  0  6  10  13  15  3  5  8 |
| 2  1  14  7  4  10  8  13  15  12  9  0  3  5  6  11 |

Figure 4. S-boxes used for DES [20]

The S-box designed for DES uses a 6-bit input and a 4-bit output. As shown in Figure 5, the most and the least significant bits of the 6-bit input determine the rows, and the other bits determine the columns. The integer at the chosen row and column is used as the output. In this study, the binary equivalent of this integer is used as the random number generated [20].
For example, suppose that the bit sequence 01100010001011110... is produced by the number generator. The first 6 bits of this sequence are 011000 and let the S-box that is going to be used for S1 be like Figure 5. The most and the least significant bits, 00, will be used for determining the row, and the other bits 1100, will be used for determining the column. As shown in Table 3, the determined output for the row and column chosen according to the S-box S1 is the integer 5. The binary representation of this number is 0101. As a result, the number 0101 will be produced by the generator using S-box post-processing for the 6-bit number 011000. An example bit sequence generated via S-box and the pseudocode is given below.

**Pseudocode:**

**Step1:** 48 bits are taken from the generated random bit sequence.

011000 010001 011110 111010 100001 100110 010100 100111

**Step2:** The 48-bit block is divided into 8 separate parts, each having 6 bits. (S1, S2, S3, S4, S5, S6, S7, S8)

011000 010001 011110 111010 100001 100110 010100 100111

|       | S1   | S2   | S3   | S4   | S5   | S6   | S7   | S8   |
|-------|------|------|------|------|------|------|------|------|
| 00    | 14   | 4    | 13   | 1    | 2    | 15   | 11   | 8    |
| 01    | 0    | 15   | 7    | 4    | 14   | 2    | 13   | 1    |
| 10    | 4    | 1    | 14   | 8    | 13   | 6    | 2    | 11   |
| 11    | 15   | 12   | 8    | 2    | 4    | 9    | 1    | 7    |

**Step3:** The most and the least significant (the first and the last) bits of S1 are taken and used to determine the row.

**Step4:** The remaining bits determine the column

**Step5:** The new value is taken by looking up the specified row and column of the S-box.

|                     | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 00                  | 14   | 15   | 14   | 13   | 1    | 2    | 15   | 11   | 8    | 17   | 13   | 15   | 12   | 9    | 7    | 10   |
| 01                  | 0    | 15   | 7    | 4    | 14   | 2    | 13   | 1    | 10   | 6    | 12   | 11   | 9    | 5    | 3    | 8    |
| 10                  | 4    | 1    | 14   | 8    | 13   | 6    | 2    | 11   | 15   | 12   | 9    | 7    | 3    | 10   | 5    | 0    |
| 11                  | 15   | 12   | 8    | 2    | 4    | 9    | 1    | 7    | 5    | 11   | 3    | 14   | 10   | 0    | 6    | 13   |
Step 6. The new bit sequence is obtained by writing the new value in a binary representation.

Step 7. Go to Step 2 unless S8 has been reached.

Step 8. A 32-bit block is generated by S1, S2, S3, S4, S5, S6, S7 and S8.

\[
\begin{array}{cccccccc}
011000 & 010001 & 011110 & 111010 & 100001 & 100110 & 010100 & 100111 \\
0101 & 1100 & 1000 & 0010 & 1011 & 0010 & 1001 & 0111
\end{array}
\]

Step 9. The End

In this study, in order to demonstrate the use of S-boxes as a post-processing method, random bit sequences obtained from the RO-based TRNG system were proposed as a source of entropy by Sunar. In published literature, Sunar performed random number generation and obtained successful results with TRNG [10, 22].

In [10], bit sequences obtained without post-processing are used. According to the results obtained from [10], RO-based TRNG does not pass the NIST test without applying post-processing. The TRNG system shown in Figure 5 has 114 ROs and 13 inverters in each RO. Random signals obtained from RO are unified by the XOR function. After the XOR function, the signals obtained are sampled using a D type flip-flop. Sunar used the resilient function in post-processing to mitigate the statistical weaknesses occurring in this design [6]. In this study, instead of the resilient function, the S-box that is proposed for post-processing is shown in Figure 6.

**Figure 6.** The Post-processing Model based on S-Box (Ring Oscillator)

As shown in Figure 6, an FPGA application is realized in order to apply S-box post-processing on bits obtained from the RO-based TRNG. Figure 7 shows the hardware used for the S1 box. The Rom data module shows the memory unit that stores the raw bits obtained from the RO-based random number generator. The most and the least significant bits of the 6-bit number that will be read from the first address are given to the Dec decoder module as input. The value 00 ensures the 0th row, 01 the 1st row, 10 the 2nd row, and finally 11 the 3rd row of the S1 box is chosen, each of these values being input to this module. The other 4 bits choose
the column for the chosen row. Memory modules $S_{box11}$, $S_{box12}$, $S_{box13}$, and $S_{box14}$ are used. The outputs of these 4 memory units are the random numbers generated and they are recorded into a Ram module with the help of a $Mux$ in order to perform statistical tests. $C_{data}$ and Dec1 modules are only used to perform type conversions. For example, in the $C_{data}$ module, the 6 bit vector is converted to a 4 bit vector and $2 \times 1$ bit output.

![Figure 7. Realization of the S-box S1 on an FPGA.](image)

5. Results Obtained from S-Box based post-processing

Bit sequences generated by random number generators may show statistical weaknesses. The generated bit sequence is therefore subject to post-processing in order to remove these weaknesses. Post-processing applications cause a reduction in the bit-rate while removing these weaknesses. Table 3 provides a comparison between the proposed S-box based post-processing and other post-processing methods in published literature.

| Post Processing                  | Bit rate   |
|----------------------------------|------------|
| Van Neumann                      | About 1/4  |
| XOR                              | 1/2        |
| H function                       | 1/2        |
| Resilient/Extractor              | 1/16       |
| Proposed S-box post-processing   | 2/3        |
The proposed S-box based post-processing algorithm in this study reduces the data rate of the TRNG to 2/3 of the original, so using S-box causes less data rate reduction than other post-processing methods. In order to use the developed system in cryptographic applications in a secure manner, randomness tests must be applied to it. Many test suites have been developed to analyze the randomness of generated numbers. One of these tests is the statistical NIST 800.22 test suite. In the NIST 800.22 test suite, there are 16 tests in total and the parameters of each test are explained in detail [23]. According to the NIST 800.22 test suite, statistical tests on the generated numbers by the system are performed by the developed software [24]. The statistical test results performed by the NIST 800.22 test suite are given in Table 4 and Table 5 below. Table 4 shows the statistical test results of the S-box based post-processed bit sequence obtained from the Sunar TRNG system. Successful results are achieved by post-processing the source of entropy. Table 5 shows a comparison between the proposed S-box post-processing method and the other post-processing algorithms.

While the entropy of the pure bits generated by the RO-based TRNG is 0.989, with the proposed S-box post-processing, it is increased to 1.0. Entropy values of both the S-box and the other post-processing algorithms are given in Table 6. These results, together with the NIST test results show that S-box post-processing can be used in TRNG systems because it can produce bit sequences with no statistical correlation between the sequences.

Autocorrelation test results are given in Table 7. Correlation shows the linear relationship between two or more variables. It takes a value between +1 and −1. If it is 0 or close to 0, there is no linear relationship between these variables. As shown in Table 7, for values 8 and 15 for D, successful results are achieved by the S-box post-processing [25]. In Table 8, the statistical complexity measure is shown. For aperiodic sequences, the statistical complexity measure must be zero or close to zero. Table 8 shows that successful results are achieved for the S-box method [25].

| Test                                      | Sunar Design without post-processing | Sunar Design with S-box post-processing |
|-------------------------------------------|--------------------------------------|----------------------------------------|
| Frequency (Monobit) Test                  | P- Value: 0.639, Result: Passed      | P- Value: 0.108, Result: Passed         |
| Frequency Test within a Block             | P- Value: 0.177, Result: Passed       |                                         |
| Runs Test                                 | P- Value: 0.123, Result: Passed       |                                         |
| Test for the Longest Run of Ones in a Block | P- Value: 0.424, Result: Passed       | P- Value: 0.564, Result: Passed         |
| Binary Matrix Rank Test                   | P- Value: 0.430, Result: Passed       | P- Value: 0.879, Result: Passed         |
| Discrete Fourier Transform Test           | P- Value: 0.114, Result: Passed       |                                         |
| Non-overlapping Template Matching Test    | P- Value: 0.413, Result: Passed       |                                         |
| Overlapping Template Matching Test        | P- Value: 0.525, Result: Passed       |                                         |
| Maurer’s Universal Statistical Test       | P- Value: 0.615, Result: Passed       | P- Value: 0.265, Result: Passed         |
| Linear Complexity Test                    | P- Value: 0.879, Result: Passed       |                                         |
| Serial Test                               | P- Value: 0.318, Result: Passed       |                                         |
| Approximate Entropy Test                  | P- Value: 0.430, Result: Passed       |                                         |
| Cumulative Sums Test                      | P- Value: 0.709, Result: Passed       |                                         |
Table 5. Statistical test results of the Sunar system processed by the other post-processing methods.

| Test                                           | Sunar Design without post-processing | Von Neumann | XOR | H Function |
|-----------------------------------------------|-------------------------------------|-------------|-----|-----------|
| Frequency (Monobit) Test                      | -                                   | 0.365       | -   | 0.241     |
| Frequency Test within a Block                 | -                                   | 0.596       | -   | 0.924     |
| Runs Test                                     | -                                   | 0.083       | -   | 0.302     |
| Test for the Longest Run of Ones in a Block   | -                                   | 0.382       | -   | 0.038     |
| Binary Matrix Rank Test                       | 0.424                               | 0.980       | 0.795 | 0.332     |
| Discrete Fourier Transform Test               | -                                   | 0.021       | 0.184 | 0.692     |
| Non-overlapping Template Matching Test        | -                                   | 0.013       | -   | -         |
| Overlapping Template Matching Test            | -                                   | 0.322       | 0.272 | 0.156     |
| Maurer’s Universal Statistical Test           | -                                   | 0.101       | 0.735 | 0.240     |
| Linear Complexity Test                        | 0.615                               | 0.970       | 0.567 | 0.337     |
| Serial Test                                   | -                                   | 0.435       | -   | 0.634     |
| Approximate Entropy Test                      | -                                   | 0.154       | 0.012 | 0.126     |
| Cumulative Sums Test                          | -                                   | 0.280       | -   | 0.192     |

Table 6. The change in the entropy of post-processing methods.

| Method                                | Entropy |
|---------------------------------------|---------|
| Pure bit                              | 0.989   |
| XOR                                   | 0.992   |
| H Function                            | 1.0     |
| Von Neuman                            | 1.0     |
| Proposed Sbox Post-Processing         | 1.0     |

Table 7. Autocorrelation Test Results.

| D Value | Pure Bit | Von Neumann | XOR  | H Function | S-box |
|---------|----------|-------------|------|------------|-------|
| Autocorrelation | 8     | -2.875      | 0.208 | 1.193      | 0.088 | 0.526 |
|          | 15      | -3.633      | 1.491 | 1.569      | 0.187 | 0.698 |

Table 8. Statistical Complexity measure result.

| Statistical Complexity Measure | Pure Bit | Von Neumann | XOR | H Function | S-box |
|--------------------------------|----------|-------------|-----|------------|-------|
|                                 | 0.013    | 0.305       | 0.217 | 0.204      | 0.102 |
6. Conclusions
The randomness of the numbers obtained from TRNGs depends on the source of entropy. There is a correlation in the generated bit sequences due to the sources of entropy being affected by environmental changes. In this study, in order to remove the correlation problem, an S-box based post-processing method was proposed. The proposed S-box post-processing algorithm was applied to the pure bits obtained from the RO-based TRNG system. Pure number sequences generated by the RO-based TRNG did not pass statistical tests. As a result of applying S-box to the pure bits, successful results were achieved with the NIST tests. Also, successful results were achieved with autocorrelation and complexity measures. The proposed S-box post-processing, when compared to the other post-processing algorithms, achieved an entropy value of 1, as for the H function and Von Neumann methods. Another advantage of S-box post-processing is that it can achieve 2/3 of the original data rate for the generated bit sequences. Random bit sequences generated according to these results will be suitable for many applications, such as cryptography.

References
[1] Koç CK. About cryptographic engineering. In: Koç CK (editor). Cryptographic engineering, US: Springer, 2009, pp. 5-16.
[2] Avaroğlu E, Koyuncu İ, Özer AB, Türk M. Hybrid pseudo-random number generator for cryptographic systems. Nonlinear Dynamics 2015; 82(1-2):239-248.
[3] Tuncer T. Implementation of duplicate TRNG on FPGA by using two different randomness source. Elektronika Ir Elektrotechnika 2015; 21(4):35-39. 10.5755/j01.eee.21.4.12779.
[4] Suresh VB, Burleson WP. Entropy extraction in metastability-based TRNG. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST); Anaheim, CA, USA: IEEE, 2010. pp. 135–140.
[5] Dichtl M. Bad and good ways of post-processing biased physical random numbers. In: Biryukov A. (editor). Fast Software Encryption. FSE 2007. Lecture Notes in Computer Science, vol 4593. Berlin, Heidelberg: Springer, 2007. pp. 137–152.
[6] Sunar B, Martin WJ, Stinson DR. A provably secure true random number generator with built in tolerance to active attacks. IEEE Transaction on Computers 2007; 56(1): 109–119.
[7] Kohlbrenner P, Gaj K. An embedded true random number generator for FPGAs. In: Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, New York, NY, USA; ACM, 2004. pp. 71–78.
[8] Golic JDJ. New methods for digital generation and post processing of random data. IEEE Transaction on Computers 2006; 55(10):1217–1229.
[9] Schellekens D, Preeneel B, Verbauwhede I. FPGA vendor agnostic true random number generator. In: International Conference on Field Programmable Logic and Applications; Madrid, Spain; IEEE, 2006. pp. 1–6.
[10] Avaroğlu E, Tuncer T, Özer AB, Ergen B, Türk M. A novel chaos-based post-processing for TRNG. Nonlinear Dynamics 2015; 81(1-2):1–11.
[11] Tuna M, Fidan CB. A study on the importance of chaotic oscillators based on FPGA for true random number generating (TRNG) and chaotic systems. Journal of the Faculty of Engineering and Architecture of Gazi University 2018; 33(2):469–486.
[12] Alçin M, Koyuncu İ, Tuna M, Varan M, Pehlivan İ. A novel high speed artificial neural network based chaotic true random number generator on field programmable gate array. International Journal of Circuit Theory and Applications 2019; 47(3):365-378.
[13] Tuna M, Fidan CB, Koyuncu İ. The chaos-based dual entropy core TRNG on FPGA: VHDL Codes of Chaotic Systems, Beau Bassin, Mauritius: LAMBERT Academic Publication (LAP), 2019.

[14] Koyuncu İ, Özerit AT. The design and realization of a new high speed FPGA-based chaotic true random number generator. Computers & Electrical Engineering 2017; 58:203-214.

[15] Loza S, Matuszewski L, Mieczysław J. A random number generator using ring oscillators and SHA-256 as post processing. International Journal of Electronics and Telecommunications 2015; 1(2):199-204.

[16] Nikolić S, Veinović M. Advancement of true random number generators based on sound cards through utilization of a new post-processing method. Wireless Personal Communications 2016; 91(2):603–622.

[17] Zhang R, Chen S, Wan C, Shinohara H. High-throughput Von Neumann post-processing for random number generator. In: International Symposium on VLSI Design, Automation and Test (VLSI-DAT) Hsinchu, Taiwan; IEEE, 2018. pp. 1–4.

[18] Barak B, Shaltiel R, Tromer E. True random number generators secure in a changing environment. In: Walter CD, Koç C, Paar C. (editors). Cryptographic Hardware and Embedded Systems(CHES), Berlin, Heidelberg: Springer, 2003, pp. 166–180.

[19] Webster AF, Tavares SE. On the design of S-Boxes. In: Advances in Cryptology — CRYPTO ’85 Proceedings, Berlin, Heidelberg: Springer, 1986, pp. 523–534.

[20] Paar C, Pelzl J. Understanding Cryptography: A Textbook for Students and Practitioners. Berlin, Heidelberg : Springer, 2010.

[21] Stallings W. Cryptography and Network Security Principles and Practices. Upper Saddle River, NJ, US: Pearson, 2005.

[22] Wold K, Tan CH. Analysis and enhancement of random number generator in fpga based on oscillator rings. In: International Conference on Reconfigurable Computing and FPGAs; Cancun, Mexico; IEEE, 2008, pp. 385–390.

[23] Lawrence EB, Andrew LR, Juan S, James RN, Miles ES et al. A statistical test suite for random and pseudo random number generators for cryptographic applications. Special Publication (NIST SP) - 800-22 Rev 1a, 2010.

[24] Avaroğlu E, Türk M. Hardware based realization of random number generator. PhD, Firat University, Elazığ, Turkey, 2014.

[25] Avaroğlu E. Pseudorandom number generator based on Arnold cat map and statistical analysis. Turkish Journal Of Electrical Engineering & Computer Sciences 2017, 25(1): pp. 633–643.