INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2015, VOL. 61, NO. 2, PP. 199–204
Manuscript received March 20, 2015; revised June 2015. DOI: 10.1515/eletel-2015-0026

A Random Number Generator Using Ring Oscillators and SHA-256 as Post-Processing

Szymon Łoza, Łukasz Matuszewski, and Mieczysław Jessa

Abstract—Today, cryptographic security depends primarily on having strong keys and keeping them secret. The keys should be produced by a reliable and robust to external manipulations generators of random numbers. To hamper different attacks, the generators should be implemented in the same chip as a cryptographic system using random numbers. It forces a designer to create a random number generator purely digitally. Unfortunately, the obtained sequences are biased and do not pass many statistical tests. Therefore an output of the random number generator has to be subjected to a transformation called post-processing. In this paper the hash function SHA-256 as post-processing of bits produced by a combined random bit generator using jitter observed in ring oscillators (ROs) is proposed. All components – the random number generator and the SHA-256, are implemented in a single Field Programmable Gate Array (FPGA). We expect that the proposed solution, implemented in the same FPGA together with a cryptographic system, is more attack-resistant owing to many sources of randomness with significantly different nominal frequencies.

***Keywords*—random numbers, cryptography, ring oscillators, hash functions, field-programmable gate arrays**

I. Introduction

As it was noted in [1] “True randomness can’t be left to chance”. This sentence reflects the importance of randomness for cryptography. Currently, there exist several technically useful sources of randomness. They are: noise generated by a physical system [1]-[5], metastable states [7]-[11], the chaos phenomenon [12]-[20] or jitter produced by ring oscillators [21]-[30]. Mixed solutions that combine various properties of these basic techniques also exist. Such sources are known as random bit generators (RBGs). They produce bits with bit rate of the order of several Mbit/s and they are not resistant to external attacks. Due to this issue, a good solution of a RBG needs to have an additional circuit or devices, dedicated to detect and disable a potential attack or simply shut off a random source after detecting an attack. The second main problem is the lack of possibility to integrate an analog random number generator in one microchip in order to be used in encryption/decryption process in dedicated solutions. Most of cryptographic systems are digital constructions. Therefore, it is expected that random number generators should be purely digital constructions, simply integrated in one chip. Nowadays there is a trend to find in digital circuits some behaviors or methods that will give possibility to produce random bit sequences “on demand”, with high bit rate, without any possibility to having access to elements of these sequences. It is proposed to use generators with jitter, constructed by using reprogrammable digital circuits or constructions based on meta-stability [31], [32]. Because the latter phenomenon, although interesting, is rather impractical for producing random bits in contemporary FPGAs [33], the most significant are concepts using ring oscillators or Galois Ring Oscillators (GARO). In both approaches jitter is used for signal generation [27], [31]. Random bit sequence is obtained by sampling signal generated by RO or GARO with rectangular wave with lower frequency. To obtain unbiased sequence that pass all known statistical tests for random sequences, e.g. NIST 800-22 test suite, Diehard, TestU01 or UC1, we need to combine bit streams produced by many RO-based random bit generators [34]-[39]. The ring oscillators must also have significantly different nominal frequencies to prevent the injection attack [40]. It forces to use delay lines built into FPGAs instead of inverters or latches [39].

To decrease the number of RO-based random bit generators necessary to pass all statistical tests, it is proposed in this paper to use SHA-256 hash function as post-processing. Both elements – RBG and SHA-256, were implemented in the same Virtex 5 FPGA (XL5VLX50T). Through experiments it has been shown that the minimal number of ROs that should be used for building a random bit generator with SHA-256 as post-processing is equal to eight.

The paper is organized as follows. The idea of producing random bits with a combined RBG with SHA-256 as post-processing is presented in Section II. The quality of sequences produced with the proposed generator is discussed in Section III. The last Section are conclusions.

II. Post-Processing with SHA-256 Hash Function

A. A Combined RO-Based Random Bit Generator

The simplest RBG that can be completely integrated with any digital system in the same FPGA uses a ring oscillator which output signal is sampled with a D flip-flop. Such kind of RBG is shown in Figure 1.

The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task 08/83/DSPB/4707 in 2014.

Szymon Łoza is with Poznan University of Technology, Faculty of Electronics and Telecommunications, ul. Polanka 3, 61-131 Poznan, Poland (Correspondence: e-mail: szymon.piotr.loza@gmail.com; tel: 695 564 375).

Lukasz Matuszewski is with Poznan University of Technology, Faculty of Electronics and Telecommunications, ul. Polanka 3, 61-131 Poznan, Poland (e-mail: lukasz.matuszewski@et.put.poznan.pl).

Mieczyslaw Jessa is with Poznan University of Technology, Faculty of Electronics and Telecommunications, ul. Polanka 3, 61-131 Poznan, Poland (e-mail: mjessa@ et.put.poznan.pl)

Generator uses jitter and frequency drift found in CMOS ring oscillator for random bit generation. A D-type flip-flop is triggered by a quartz oscillator signal which establishes the bit rate. Frequency f_H is greater than quartz oscillator frequency f_L. The f_H of a single ring oscillator is equal to

$$f_H = \frac{1}{2}\sum_k \frac{1}{d_k}, \tag{1}$$

where d_k is a delay of the k-th component of RO. The expression is true if all components are ideal and delays related with interconnections are ignored. In a real circuit delays caused by inner connections cannot be ignored [36]. Moreover, propagation delays in all circuit paths and gates vary in time, because of shot noise, thermal noise and supply voltage instability [28], [41]. Taking into account these factors the more realistic formula for f_H is the following [28], [39]:

$$f_H = \frac{1}{2}\sum_k \frac{1}{d_k + di_k + \Delta J_a}, \tag{2}$$

where

$$\Delta J_a = \sum_k \Delta dt_k + \Delta dv_k + \Delta dr_k \tag{3}$$

is an accumulated jitter during previous half period of signal with frequency of f_H and k delay elements. Parameter di_k is the delay of the k-th interconnections in ring oscillator, Δdt_k is the variation of the delay in the k-th component and interconnection caused by variation of temperature, Δdv_k represents the variation of the delay in the k-th component and interconnection caused by supply voltage instability and Δdr_k define others random delays in the k-th element and path in the ring oscillator, e.g., transition spacing or crosstalk's. The accumulated jitter can be divided into deterministic component and nondeterministic one [28], [39]:

$$\Delta J_a = \Delta J_{an} + \Delta J_{ad}, \tag{4}$$

where

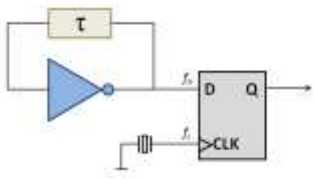
$$\Delta J_{an} = \sum_k \Delta t_{nk} + \Delta dv_{nk} + \Delta dr_{nk} \tag{5}$$

denotes an accumulated nondeterministic jitter and

$$\Delta J_{ad} = p \cdot \sum_k \Delta t_{dk} + \Delta v_{dk} + \Delta dr_{dk}$$

represents an accumulated deterministic jitter with proportion factor of p.

Realization of the delay element τ can be done with even number of inverters, a latches chain or a delay line that is built-in many FPGAs. The greater delay τ, the lower frequency f_H is obtained. Due to insufficient nondeterministic factor in a single RO, it is necessary to combine XOR many independent sources of randomness [31], [34], [37]. The combined RBG (CRBG) is shown in Figure 2.



(6)

Fig. 1. Uniformly sampled ring oscillator (RO) as a RBG

When all rings are built in the same way, they have similar frequencies and the RO-based RBG is sensitive to injection attack [40]. To ensure the robustness to this attack, we have to construct ROs with significantly different nominal frequencies. We can choose an even number of inverters, a chain of laches or delay lines available in FPGAs. The comparison of nominal frequencies of ROs using different types of delays is available in [39]. In this article, the chain of latches was used as τ. In the first RO – one latch, in the second RO – two latches, and in the N^{th} - N latches.

B. SHA-256

The output bits from the combined RBG may still be biased and correlated for small N [34], [37]. To overcome this problem we can use a post-processing [44]. The scheme of CRBG with SHA-256 as post-processing is shown in Figure 3.

Random bits from the combined RBG are collected in blocks of 8 bits. Afterwards, each byte is stored in FIFO buffer which is 64 byte width. This is made to prepare 512 random bits that are processed by SHA-256. Hash functions are used in cryptography mainly to check integrity and in digital signature schemes. The definition of hash function says that "A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values" [45]. Input can consist of such data like text file, binary file, message, data block etc. In general, the length of input is not limited. A general schema that illustrates how does a hash function works is shown in Figure 4.

A family of hash functions SHA-2 includes SHA-256, SHA-384 and SHA-512. In this paper it was used the SHA-256. The algorithm comes from paper [46]. In its first step, input is processing by adding bit 1, next to the last significant bit and any number of bits 0 that $L \oplus 512 = 488$, where L is length of the message. A family of hash functions SHA-2 includes SHA-256, SHA-384 and SHA-512. In this paper it was used the SHA-256. The algorithm comes from paper [46]. In its first step, input is processing by adding bit 1, next to the last significant bit and any number of bits 0 that $L \oplus 512 = 488$, where L is length of the message. After that L is added as 64-bits big-edian representation. Next step is that 512 blocks split into smaller 32-bits blocks $M_j^{(i)}$ where $j = 0, 1, 63$, and $i = 0, 1, N$, where N is a number of divided massage block. After splitting, the SHA-256 algorithm prepares initial values for $H^{(0)}$ – sequences of 32 bits which

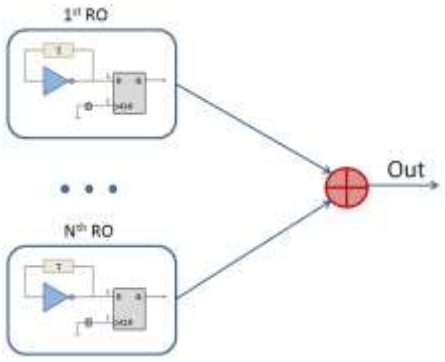


Fig. 2. Combined RBG block diagram.

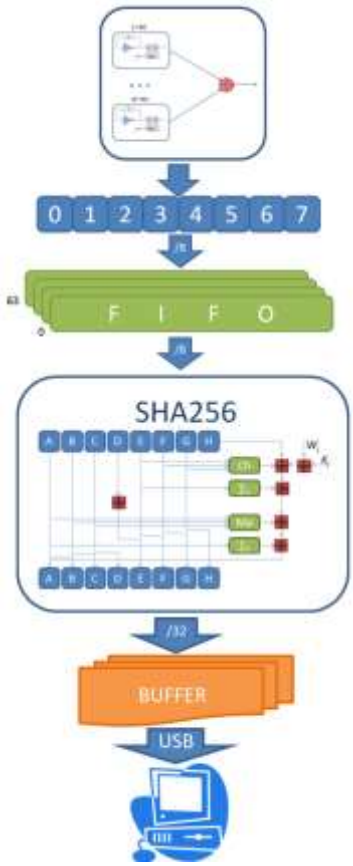


Fig. 3. A combined random bit generator with SHA-256 as post-processing

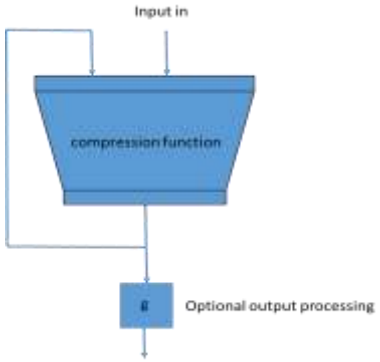


Fig. 4. A concept of a hash function.

were obtained as fractional parts of the square roots of the first eight primes. $H^{(0)}$ values are the following:

$$\begin{aligned} H^{(0)}_1 &= 6a09e667,\\ H^{(0)}_2 &= bb67ae85,\\ H^{(0)}_3 &= 3c6ef372,\\ H^{(0)}_4 &= a54ff53a,\\ H^{(0)}_5 &= 510e527f,\\ H^{(0)}_6 &= 9b05688c,\\ H^{(0)}_7 &= 1f83d9ab,\\ H^{(0)}_8 &= 5be0cd19. \end{aligned} \quad (1)$$

After preparation initial register values, the algorithm updates registers: *a, b, c, d, e, f, g, and h*. This update is calculated in 64 steps from $j = 0$ to $j = 63$ and it goes as following:

$$\begin{aligned} T_1 &\leftarrow h + \sum_1(e) + Ch(e, f, g) + K_j + W_j\\ T_2 &\leftarrow \sum_0(a) + Maj(a, b, c)\\ h &\leftarrow g\\ g &\leftarrow f\\ f &\leftarrow e\\ e &\leftarrow d + T_1\\ d &\leftarrow c\\ c &\leftarrow b\\ b &\leftarrow a\\ a &\leftarrow T_1 + T_2 \end{aligned} \quad (2)$$

where:

$$\begin{aligned} Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z)\\ Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)\\ \sum_0(x) &= S2(x) \oplus S13(x) \oplus S22(x)\\ \sum_1(x) &= S6(x) \oplus S11(x) \oplus S25(x)\\ \sigma_0(x) &= S7(x) \oplus S18(x) \oplus R3(x)\\ \sigma_1(x) &= S17(x) \oplus S19(x) \oplus R10(x) \end{aligned} \quad (3)$$

Sn – right n-bit shift

Rn – right n-bit rotation.

W_j – message blocks are determined as follows:

1. for first 16 blocks: $W_j = M_j^{(i)}$

2. for rest of blocks: $W_j = \boldsymbol{\sigma}_1(W_{j-2}) + W_{j-7} + \boldsymbol{\sigma}_0(W_{j-15}) + W_{j-16}$

K_j - 32-bit words determined as fractional parts of the cube roots of the first sixty four primes.

$\oplus$	bitwise XOR
$\wedge$	bitwise AND
$\neg$	bitwise complement
+	mod 2^{32} addition
Sn	right shift by n bits
Rn	right rotation by n bits

The next step is a calculation of intermediate hash *value* $H^{(i)}$:

$$\begin{aligned} H^{(i)}_1 &\leftarrow a + H_1^{(i-1)}\\ H^{(i)}_2 &\leftarrow a + H_2^{(i-1)}\\ &\cdot\\ &\cdot\\ &\cdot\\ H^{(i)}_8 &\leftarrow a + H_8^{(i-1)} \end{aligned} \quad (4)$$

As an output, we obtain hash value $H^{(N)}$ of message M generated as:

$H^{(N)} = \{H^{(N)}_1, H^{(N)}_2, H^{(N)}_3, H^{(N)}_4, H^{(N)}_5, H^{(N)}_6, H^{(N)}_7, H^{(N)}_8\}$.

Hash function returns eight 32-bit words. The bits are sent via buffer and USB interface to a personal computer (PC). In PC the quality of generated sequence is assessed using statistical tests and the restarts mechanism [37], [38]. In all experiments the sampling frequency f_L is equal to 100 MHz.

III. The Statistical Properties of Bit Sequences Produced by a CRBG using ROs and the SHA-256

To assess the minimal number of source generators of CRBG with SHA-256 as post-processing "A statistical Test Suite for Random and Pseudo-Random Number Generators for Cryptographic Applications", document 800-22 prepared by the National Institute of Standards and Technology (NIST) was used [47]. These tests are often referred to as the NIST 800-22 statistical test suite or, simply, the NIST 800-22 tests. During testing, we applied two approaches proposed by NIST: (1) we examined the proportion R_β of sequences that passed a statistical test, and (2) we examined the distribution of

P – values computed by the software; that is, we examined the value of P_T [47].

In the first step only one RO was connected to SHA-256 block. The sequence of 1 Gbit length was collected and examined with the NIST 800-22 test suite. The results of experiment were unsatisfactory because most of the tests were failed (Table I).

TABLE I
THE RESULTS OF THE NIST 800-22 TESTS FOR THE CRBG WITHOUT SHA-256 AND WITH SHA-256

Type of test	***CRBG-1***		***CRBG-1 + SHA-256***	
	R_β	P_T	R_β	P_T
Frequency	**0.000**	**0.000**	**0.000**	0.980
Block Frequency	**0.000**	**0.056**	**0.000**	**0.491**
Cumulative Sums*	**0.000**	**0.000**	**0.000**	**0.976**
Runs	**0.000**	**0.000**	0.061	0.982
Longest Run of Ones	**0.000**	**0.000**	**0.000**	**0.977**
Rank	**0.000**	**0.000**	**0.000**	**0.000**
Spectral DFT	**0.000**	**0.000**	**0.000**	**0.000**
Non-overlapping Temp.*	**0.000**	**0.000**	**0.000**	**0.954**
Overlapping Templates	**0.000**	**0.000**	**0.000**	0.757
Universal	**0.000**	**0.000**	0.311	0.986
Approximate Entropy	**0.000**	**0.000**	**0.000**	**0.064**
Random Excursions*	**1.000**	---	0.242	0.564
Random Exc. Var.**	**1.000**	---	0.097	0.565
Serial*	**0.000**	**0.000**	**0.000**	**0.000**
Linear Complexity	0.236	0.995	0.651	0.994

TABLE II
FREQUENCIES OF ROS IN THE COMBINED RBG

RO number	Frequency [MHz]
1.	702
2.	555
3.	312
4.	220
5.	202
6.	164
7.	145
8.	111

TABLE III
THE RESULTS OF THE NIST 800-22 TESTS FOR THE CRBG WITHOUT SHA-256 AND WITH SHA-256

Type of test	***CRBG-8***		***CRBG-8 + SHA-256***	
	R_β	P_T	R_β	P_T
Frequency	**0.792**	0.993	0.989	0.848027
Block Frequency	**0.000**	0.949	0.991	0.630872
Cumulative Sums*	0.994	0.058	0.988	0.653773
Runs	**0.000**	0.803	0.991	0.680755
Longest Run of Ones	**0.000**	0.906	0.992	0.908760
Rank	**0.781**	0.989	0.995	0.699313
Spectral DFT	**0.000**	0.885	0.988	0.216713
Non-overlapping Temp.*	**0.000**	0.584	0.982	0.021554
Overlapping Templates	**0.000**	0.299	0.986	0.530120
Universal	**0.000**	0.937	0.987	0.649612
Approximate Entropy	**0.000**	**0.000**	0.988	0.446556
Random Excursions*	**0.595**	0.976	0.986	0.199785
Random Exc. Var.**	**0.863**	0.984	0.984	0.238697
Serial*	**0.000**	**0.000**	0.989	0.308561
Linear Complexity	**0.147**	0.989	0.922	0.431754

TABLE IV
THE RESULTS OF THE RESTARTS FOR THE CRBG WITHOUT SHA-256 AND WITH SHA-256

	CRBG-8	***CRBG-8 + SHA-256***
m_{min}	36	1

TABLE V
IMPLEMENTATION ISSUES

Number of Slice Registers	2540
Number of Slice LUTs	2467
Number of LUT Flip Flops pairs	2824
Max Clock Frequency	263 MHz

The experiment was repeated for CRBGs that uses two, three, etc. source bit streams, till the all tests from NIST 800-22 were passed. Each source bit stream was produced by a single RO-based RBG. The source generators differed only the delay τ in the ROs. During analysis, the final report file from NIST 800-22 package were used. The tests passed a combined RBG using eight or more RO-based source generators. The frequencies of eight ROs are shown in Table II.

During testing the standard set of parameters proposed by NIST in v. 2.1.1 was assumed. The significance level was $\beta = 0.01$. The minimum passing value for the standard set of parameters was approximately 0.9805. The minimum P_T value was 0.0001. An asterisk * denotes that this test consists of several subtests and that the worst result is shown. For tests marked with **, the minimum passing value for the standard set of parameters was approximately 0.9777. The results of the NIST statistical tests are shown in Table III.

In the next step of the experiment it was performed a test based on restarts mechanism [37], [38]. This test is based on multiple restarts of the combined generator with the same initial conditions. It helps to assess the amount of randomness and pseudo-randomness in generated sequences. If during producing bits the amount of deterministic factor is prevalent, the sequences will be almost the same or exactly the same. When the non-deterministic phenomena prevails, the generated sequences will vary.

During the restarts $N = 2084$ sequences were generated and $M = 19968$ bits were send to PC for each restart. Next, M=19968 chi-square tests were performed. If a single bit in the sequences was produced in a non-deterministic process then chi-square test is passed. A computer program searches the results of 19968 chi-square tests for the greatest m for which the sequences failed the chi-square test for three successive indices, i.e., m, m-*1* and m-*2*, where m=1,2,...,M. For $j > m$ and a given significance level of the test, there is no reason to reject the hypothesis that zeros and ones occur with the same probability in 19968 bit sequences. The smallest j is equal to $m+1$ and denoted by m_{min} [37]-[39]. The results of the restarts are shown in Table IV.

The described generator was implemented in Virtex-5 (XL5VLX50T). Used resources are specified in Table IV.

Those resources are about 9% of all resources available in Virtex-5 (XL5VLX50T) FPGA. The remaining 91% can be used for monitoring on-line the quality of random bits to detect any disturbances caused, e.g., by an attack and for implementing a cryptosystem that exploits random bits. The strings of bits can be produced on demand or in random instances, hampering cryptographic attacks.

IV. Conclusions

It is known from the literature that combining XOR bits produced at the same time by many independent random number generators is an efficient method for producing random sequences that pass every statistical test. This method requires relatively large resources, and excellent statistical properties can be observed for both deterministic and nondeterministic systems. The proposed true random number generator is able to provide random bits with average bit rate of 36 Mbit/s. The minimal number of RO that should be used when building the combined RBG with SHA-256 as post-processing is eight. For smaller number of RO-based source generators the combined RBG does not pass all NIST 800-22 tests. The use of SHA-256 function as post-processing enhances significantly the statistical properties of the output sequences and reduces the m_{min} value, but it works up to a certain level. When a generator produces sequences with very poor statistical properties, post-processing with SHA-256 does not improve sufficiently the statistical properties. The expected robustness to the injection attack results from significantly different frequencies of eight RO-based source generators. The use of ROs with significantly different frequencies hampers also mutual synchronization between ROs implemented in the same FPGA, preventing the quality degradation of RO-based combined RBGs implemented in various FPGAs.

References

[1] A. Vassilev and T. A. Hall, "The importance of entropy to information security,", IEEE Computer, pp. 78-81, February 2014.

[2] W. T. Holman, J. A. Connelly, and A. B. Downlatabadi, "An integrated analog/digital random noise source," *IEEE Trans. Circuits and Syst. I, Fundam. Theory Appl.*, vol. 44, pp. 521-528, June 1997.

[3] V. Bagini and M. Bucci, "A Design of reliable true random number generator for cryptographic applications," in *Proc. Workshop Cryptograph. Hardware Embed. Syst. CHES'1999*, Heidelberg, 1999, LNCS 1717, pp. 204-218.

[4] C. S. Petrie, J. A. Connelly, "The sampling of noise for random generation," in *Proceedings of the 50th International Symposium on Circuits and Systems ISCAS'1999,* vol. 6, pp. VI-26-VI-29, 1999.

[5] M. Bucci, L. Germani, R. Luzzi, P. Tommasimo, A. Trifiletti, and M. Varanonuovo, "A high-speed IC random-number source for smartcard microcontrollers," *IEEE Trans. Circuits and Syst. I, Fundam. Theory Appl.*, vol. 50, pp. 1373-1380, Nov. 2003.

[6] J. Holleman, S. Bridges, B. P. Otis, and Ch. Diorio "A 3 μW CMOS true random number generator with adaptive floating-gate offset cancellation, *IEEE J. of Solid-State Circuits,"* vol. 43, pp. 1324-1336, May 2008.

[7] M. J. Bellido *et al*., "A simple binary random number generator: New appoaches for CMOS VLSI," in *Proc. 35th Midwest Symp. Circuits Syst*., vol. 1, pp. 127-129, 1992.

[8] M. Epstein, L. Hars, R. Krasinski, M. Rosner, and H. Zheng, "Design and implementation of a true random number generator based on digital circuit artifacts," in *Proc. Workshop Cryptograph. Hardware Embed. Syst. CHES'2003*, LNCS 2779, pp. 152-165, 2003.

[9] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy "Fast digital RBG based on metastable ring oscillator," in *Proc. Workshop Cryptograph. Hardware Embed. Syst. CHES'2008*, LNCS 5154, pp. 164-180, 2008.

[10] S. Srinivasan, S. Mathew, V. Erraguntla, and Krishnamurthy, "A 4Gbps 0.57pJ/bit Process-Voltage-Temperature Variation Tolerant All-Digital True Random Number Generator in 45nm CMOS," in *Proc. 22nd International Conference on VLSI Design*, pp. 301-306, 2009.

[11] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generators," in *Proc. Workshop Cryptograph. Hardware Embed. Syst. CHES'2010*, LNCS 6225, pp. 351-365, 2010.

[12] G. Bernstein and M. A. Lieberman, "Secure random number generation using chaotic circuits," *IEEE Trans. Circuits and Syst.* vol. 37, pp. 1157-1164, Sept. 1990.

[13] R. Bernardini and G. Cortelazzo, "Tools for designing chaotic systems for secure random number generation," *IEEE Trans. Circuits and Syst. I, Fundam. Theory Appl.*, vol. 48, pp. 552-564, May 2001.

[14] T. Stojanovski and L. Kocarev, "Chaos-based random number generators – Part I: Analysis," *IEEE Trans. Circuits and Syst. I, Fundam. Theory Appl.*, vol. 48, pp. 281-288, March 2001.

[15] T. Stojanovski, J. Pihl, and L. Kocarev, "Chaos-based random number generators – Part II: Practical realization," *IEEE Trans. Circuits and Syst. I, Fundam. Theory Appl.*, vol. 48, pp. 382-385, March 2001.

[16] M. E. Yalçin, J. A. K. Suykens, and J. Vandewalle, "True bit generation from a double-scroll attractor," *IEEE Trans. Circuits and Syst. I, Regular Papers.*, vol. 51, pp. 1395-1404, July 2004.

[17] S. Callegari, R. Rovatti, and G. Setti, "Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos," *IEEE Trans. Signal Processing*, vol. 53, pp. 793-805, Feb. 2005.

[18] S. Callegari, R. Rovatti, and G. Setti, "First direct implementation of true random source on programmable hardware," *Int. J. Circ. Theor. Appl*. vol. 33, pp. 1-16, 2005.

[19] M. Drutarovsky and P. Galajda, "Chaos-based true random number generator embedded in a mixed-signal reconfigurable hardware," *J. Electrical Engineering*, vol. 57, pp. 218-225, April 2006.

[20] S. Ergün and S. Özog, "Truly random number generator based on a non-autonomous chaotic oscillator," *Int. J. Electronics and Commun*., vol. 61, pp. 235-242, April 2007.

[21] C. S. Petrie, J. A. Connelly, "Modelling and simulation of oscillator-based random number generators," in *Proceedings of the 47th International Symposium on Circuits and Systems ISCAS'1996,* vol. 4, pp. 324-327, 1996.

[22] C. S. Petrie and J. L. Connelly, "A noise-based IC rndom number generator for applications in Cryptography," *IEEE Trans. Circuits and Syst. I, Fundam. Theory Appl.*, vol. 47, pp. 615-621, May 2000.

[23] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti M. Varnonuovo, "A high-speed oscillator-based truly random number source for cryptographic applications on a smartcard IC," *IEEE Trans. on Computers*, vol. 52, pp. 403-409, April 2003.

[24] B. Jun, B. Kocher, *"The Intel random number generator,"* Cryptography Research Inc., San Francisco, CA, white paper for Intel Corp., April 1999. Available at: http://www.cryptography.com/resources,whitepapers/IntelRNG.pdf.

[25] P. Kohlbrenner and K. Gay, "An embedded true random number generator for FPGAs," *in Proc. of the 2004 ACM/SIGDA 12th International Symposium on FPGAs, FPGA'04*, pp. 71-77, 2004.

[26] J. D. Golić, "New methods for digital generation and postprocessing of random data," *IEEE Trans., Comput.*, vol. 55, pp. 1217-1229, Oct. 2006.

[27] M. Dichtl and J. D. Golić, „High speed true random number generation with logic gates only, in *Proc. Workshop Cryptograph. Hardware Embed. Syst.* CHES'2007, LNCS 4727, pp. 45-62, 2007.

[28] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer, "Modeling and observing the jitter in ring oscillators implemented in FPGAs," in *Proc. of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, DDECS'08*, pp. 1-6, 2008.

[29] B. Valtchanov, V. Fischer, A. Aubert, and F. Bernard, "Characterization of randomness sources in ring oscillator-based true random number generators in FPGAs," in *Proc. of IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, DDECS'10*, pp. 48-53, 2010.

[30] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *J. Cryptology*, vol. 24, pp. 398-425, 2011.

[31] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Trans., Comput*., vol. 56, pp. 109-119, Jan. 2007.

[32] P. Wieczorek and K. Golofit, "Dual-Metastability Time-Competitive True Random Number Generator", *IEEE Trans. On Circuits and Systems*, vol. 61, I. No. 1, pp. 134-145, 2014.

[33] P. Kubczak, M. Jessa, and L. Matuszewski, "Random number generator exploiting metastability implemented in Xilinx FPGA," Mesurement Automation and Monitoring (PAK), vol. 60, No. 7, pp. 450-452, 2014.

[34] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," *Int. J. of Reconfiugurable Computing*, vol. 2009, pp. 1-8, 2009.

[35] K. Wold and S. Petrović, "Optimizing speed of a true random number generator in FPGA by spectral analysis," in *Proc. of Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT'09*, pp. 1105-1110, 2009.

[36] K. Wold and S. Petrović, "Security properties of oscillator rings in true random number generators," in *Proc. of 15th International Symposium on Components, Circuits, Devices and Systems*, pp. 145-150, 2012.

[37] M. Jessa and M. Jaworski, "Randomness of a combined RBG based on the ring oscillator sampling method," *Proc. of International Conference on Signals and Electronic Systems, ICSES'10*, pp. 323-326, 2010.

[38] M. Jessa and L. Matuszewski, "Enhancing the Randomness of a Combined True Random Number Generator Based on the Ring Oscillator Sampling Method," *Proc. of International Conference on ReConFigurable Computing and FPGAs*, *ReConFig'2011*, , 2011, pp. 274-279, 2011.

[39] M. Jessa and L. Matuszewski, "Producing random bits with delay-line-based ring oscillators," *Int. Journal of Electronics and Telecommunications*, vol. 59, No. 1, pp. 41-50, 2013.

[40] A. T. Markettos and S. M. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Proc. Workshop Cryptograph. Hardware Embed. Syst. CHES'2009*, Sept., 2009, LNCS 5747, pp. 317-331.

[41] Ü. Güler, S. Ergün, and G. Dündar, "A digital IC random number generator with logic gates only," *Proc. of 17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec. 2010, pp. 239-242.

[42] N. Bochard, F. Bernard, and V. Fischer, "Observing the randomness in RO-based RBG," in *Proc. of International Conference on Reconfigurable Computing and FPGAs, ReConFig 2009*, pp. 237-242, 2009.

[43] S Markovski, ,D. Gligoroski, and L. Kocarev, "Unbiased Random Sequences from Quasigroup String Transformations", *12th International Workshop*, FSE 2005, Paris, France, February 21-23, 2005.

[44] M. Bucci and R. Luzzi, "Fully digital random bit generators for cryptographic applications," *IEEE Trans. Circuits and Syst. I: Regular Papers*, vol. 55, pp. 861-875, April 2008.

[45] A. J. Menezes, P. C. van Oorschot, and S. C. Vanstone, Handbook of Applied Cryptography. Boca Raton: CRC, 1997.

[46] Descriptions of SHA-256, SHA-384, and SHA-512 http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf.

[47] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST special publication 800-22, Revised: April 2010, Available at: http://csrc.nist.gov/rng/.