**Szymon ŁOZA**, Łukasz MATUSZEWSKI, Mieczysław JESSA, Paweł KUBCZAK
POZNAN UNIVERSITY OF TECHNOLOGY, FACULTY OF ELECTRONICS AND TELECOMMUNICATIONS
3 Polanka St., 61-131 Poznań

# A random number generator using ring oscillators and the Keccak as post-processing

**Abstract**

In cryptography, sequences of numbers with unpredictable elements are often required. Such sequences should pass all known statistical tests for random sequences. Because sequences produced in real circuits are biased, they do not pass many statistical tests, e.g., the distribution of numbers is not uniform. Such random number sequences should be subjected to a transformation called post-processing. In this paper, a true random number generator is considered. It uses ring oscillators and the Keccak hash function as post-processing. This paper presents only simulation conditions for this approach since the post-processing part was done using x86 architecture on a PC.

**Keywords**: true random number generator, ring oscillator, cryptography, field programmable gate array, hash function.

## 1. Introduction

Nowadays, random number generators (RNG) are widely used in many applications in cryptography [1]. There are two types of RNGs: pseudo-random number generators (PRNGs) which can be described with a deterministic algorithm and true random number generators (TRNGs) which produce non-deterministic sequences even if the structure of the generator is known. Most of cryptographic systems are digital constructions, therefore it is expected that true random number generators are digital constructions, simply integrated in one chip with a cryptographic system. The most popular solutions are based on jitter in ring oscillators (ROs) and metastable states [2-14]. Both constructions are well implemented in FPGAs (Field Programmable Gate Arrays) and ASICs (Application Specific Circuits). They provide single bits at the output and are called in the literature true random bit generators (TRBGs) [3-14]. In a RO-based TRBG the signal from RO is sampled in flip-flop with the use of another signal with significantly lower frequency. The result is the sequence of random bits. However, the generator produces bits that are biased and, consequently, random sequences do not pass many statistical tests. Such random number sequences should be subjected to a transformation called post-processing. In this paper the use of the Keccak hash function as a post-processor for the RO-based TRBG is proposed. The Keccak is a cryptographic hash function designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche [15].

## 2. Post-processing with the Keccak hash function

Hash functions are used in cryptography mainly to check integrity and in digital signature schemes. The definition of a 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" [1]. The input can consist of such data like text file, binary file, message, data block etc. In general, the length of the input is not limited.

The main goal of this paper is to verify the usability of the Keccak algorithm as post-processing. The scheme of a combined TRBG (CRBG) with the Keccak as post-processing is shown in Figure 1. Random bits from the combined TRBG are collected in blocks of 8 bits. Each data byte is sent via USB interface. The acquired file is spitted into 1088 bit blocks and concatenated with the previous ones. The Keccak algorithm is based on sponge construction, as it can be seen in Figure 2 [15, 16].

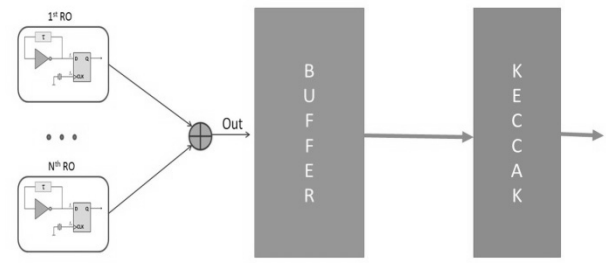


Fig. 1. A combined random bit generator (CRBG) with the Keccak as post-processing

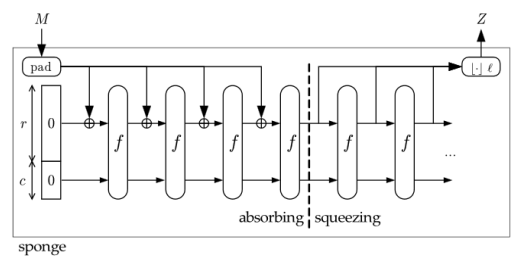


Fig. 2. Sponge structure [15, 16]

There are two main phases in the Keccak algorithm: absorbing and squeezing. In the absorbing phase, we call $b$-bit permutations $f$ with $b = r + c$, were $r$ is called rate and $c$ is called capacity; $c$ can increase the system security because those bits are not XORed with the input. In the squeezing phase, the output data is archived, the length of the output can be changed. There are seven sizes of Keccak-$f$ permutation; $b \in \{25, 50, 100, 200, 400, 800, 1600\}$. We can make security - speed trade-offs using the same permutation; more $c$-bits more security, more $r$-bits faster generation. The bits are hold in the state array of $5 \times 5 \times 2l$ bits (Fig. 3) [15, 16]. In this equation $l \in \{0, 1, 2, 3, 4, 5, 6\}$, so there are 1, 2, 4, 8, 16, 31 or 64 slices (5×5 arrays).

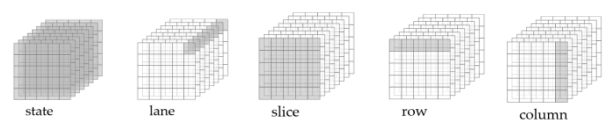


Fig. 3. Organization of state bits [15]

The algorithm has a variable number of rounds $n = 12 + 2l$, e.g., Keccak-$f$ (25) has 12 rounds and Keccak-$f$ (1600) has 24 rounds. Each round function consists of 5 sub-rounds; $R = l \circ \chi \circ \pi \circ \rho \circ \theta$. Each sub-round is very simple:

- $\chi$ Flip bits if neighbors in a row exhibit 01 pattern. This is the only non-linear operation in the Keccak.
- $\rho$ Cyclic shift of lanes with offsets.
- $l$ Exclusive-or a round constant into one word of the state. Without l, all rounds would be the same.
- $\pi$ Permute the 25 words (slice) in a fixed pattern. This disturbing horizontal and vertical alignment.
- $\Theta$ Compute the parity of each of the $5w$ (320, when $w = 64$) 5-bit columns, and exclusive-or that into two nearby columns in a regular pattern.

## 3. Statistical properties of bit sequences produced by a TRBG using ROs and the Keccak hash function

The main goal of the experiment was to determine if the Keccak hash function could be used as post-processing and how much it could change statistical parameters of the acquired random number sequence produced by an RO-based CRBG. To verify this, "A statistical Test Suite for Random and Pseudo-Random Number Generators for Cryptographic Applications", the document 800-22 prepared by the National Institute of Standards and Technology (NIST), was used [17]. During testing, there were applied two approaches proposed by the NIST: (1) examination of the proportion $R_\beta$ of sequences that passed a statistical test, and (2) examination of the distribution of $P$-values computed by the software; that is, the value of $P_T$ was examined [17].

The sequence of 1 Gbit length was collected, divided into 1000 subsequences with $10^6$ bits each, and examined with the NIST 800-22 test suite. The results of the tests for the standard set of parameters proposed by the NIST in v. 2.1.1 are shown in Table 1. 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 the tests marked with **, the minimum passing value for the standard set of parameters was approximately 0.9777.

The experiment was repeated for the RO-based combined TRBG that used two (CRBG-2), three (CRBG-3), etc. source bit streams. Each source bit stream was produced by a single RO-based TRBG. The source generators differed only in a delay $\tau$ in the ROs. During the analysis, the final report file from the NIST 800-22 package was used. The combined TRBG using eight or more RO-based source generators passed the tests.

Tables 1 and 2 show that when the output of the CRBG was post processed with the Keccak hash function the statistical properties were significantly improved. The improvement can be seen for the combined TRBG composed of only two source generators (Tab.1.) However, the generator did not pass two statistical tests. This situation repeats for CRBG-3, CRBG-4,..,CRBG-8 using 2ROs, 3ROs,…, 8ROs, respectively. Using SHA-256 as post-processing, all the tests were satisfied for the CRBG exploiting eight or more ROs (Tab.3) [14].

Tab. 1. The results of NIST 800-22 tests for CRGB2

| Type of test | *CRBG-2* | | *CRBG-2 + Keccak 256* | |
|---|---|---|---|---|
| | $R_\beta$ | $P_T$ | $R_\beta$ | $P_T$ |
| Frequency | 0 | 1 | 0.001967 | 0.992 |
| Block Frequency | 0 | 1 | 0.201189 | 0.992 |
| Cumulative Sums* | 0 | 1 | 0.018540 | 0.995 |
| Runs | 0 | 0 | 0.055714 | 0.992 |
| Longest Run of Ones | 0 | 0 | 0.132640 | 0.995 |
| Rank | 0 | 0.539 | 0.258307 | 0.994 |
| Spectral DFT | 0 | 0.042 | 0.067722 | 0.987 |
| Non-overlapping Temp.* | 0 | 0.101 | 0.000009 | 0.988 |
| Overlapping Templates | 0 | 0 | 0.0013336 | 0.99 |
| Universal | 0 | 0 | 0 | 0 |
| Approximate Entropy | 0 | 0 | 0.008207 | 0.994 |
| Random Excursions* | 0 | 0.373 | 0.073701 | 0.978 |
| Random Exc. Var.** | 0 | 0.865 | 0.052778 | 0,99 |
| Serial* | 0 | 0 | 0.025882 | 0.988 |
| Linear Complexity | 0.794 | 0.986 | 0.001730 | 0.986 |

Tab. 2. The results of NIST 800-22 tests for CRBG8

| Type of test | *CRBG-8* | | *CRBG-8+ Keccak 256* | |
|---|---|---|---|---|
| | $R_\beta$ | $P_T$ | $R_\beta$ | $P_T$ |
| Frequency | 0.008 | 0.994 | 0.420 | 0.986 |
| Block Frequency | 0 | 0.993 | 0.717 | 0.990 |
| Cumulative Sums* | 0 | 0.995 | 0.836 | 0.987 |
| Runs | 0 | 0.901 | 0.287 | 0.989 |
| Longest Run of Ones | 0 | 0 | 0.740 | 0.993 |
| Rank | 0 | 0.987 | 0.284 | 0.99 |
| Spectral DFT | 0 | 0 | 0.225 | 0.987 |
| Non-overlapping Temp.* | 0 | 0.986 | 0.0004 | 0.991 |
| Overlapping Templates | 0 | 0 | 0.762 | 0.991 |
| Universal | 0 | 0.759 | 0 | 0 |
| Approximate Entropy | 0 | 0 | 0.554 | 0.995 |
| Random Excursions* | 0.814 | 0.979 | 0.042 | 0.997 |
| Random Exc. Var.** | 0.685 | 0.991 | 0.052 | 0.997 |
| Serial* | 0 | 0 | 0.119 | 0.993 |
| Linear Complexity | 0.629 | 0.995 | 0.982 | 0.99 |

Comparing Tables 1, 2, and 3, it can be noted that both secure hash algorithms improve the statistical properties of raw sequences. The use of SHA-256 provides better results because the CRBG needs only 8 source RO-based TRBGs to pass all statistical tests. On the other hand, it has been proven in [18] that Keccak hash function performance is better than SHA-2 hash functions family. This argument makes the Keccak to be worth considering in post-processing when the performance is a key element and the resources are not critical.

Tab. 3. The results of NIST 800-22 tests for the RO-based combined TRBG with SHA-256 as post-processing [14]

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

The resources used to produce random bits with RO-based CRBG were about 5% of all the resources available in Virtex-5 (XL5VLX50T) FPGA. The remaining 95% 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.

## 4. Conclusion

In this paper, the utility of the Keccak hash function for post-processing of raw bits produced by a combined TRBG using ring oscillators has been considered. To the best authors' knowledge, it is the first paper that discusses this problem for CRBGs. The main conclusion of the paper is the statement that the use of SHA-256 provides better results but the Keccak is faster when a computer is used in post-processing. The next step is the implementation of the RO-based TRBG and the Keccak hash function in the same FPGA to compare the hardware efficiency of post-processing with SHA-256 and the Keccak.

*The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task 08/83/DSPB/4712 in 2015.*

## 5. References

[1] Menezes A. J., van Oorschot P. C., and Vanstone S. C.: Handbook of Applied Cryptography. Boca Raton: CRC, 1997.
[2] Sunar B., Martin W. J., and Stinson D. R.: A provably secure true random number generator with built-in tolerance to active attacks. IEEE Trans., Comput., vol. 56, pp. 109-119, Jan. 2007.
[3] Wieczorek P., Golofit K.: Dual-Metastability Time-Competitive True Random Number Generator. IEEE Trans. On Circuits and Systems, v. 61, I. 1, pp. 134-145 (2014).
[4] Wold K. and Petrović S.: 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.
[5] Valtchanov B., Aubert A., Bernard F., and Fischer V.: 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.
[6] Güler Ü., Ergün S., and Dündar G.: 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.
[7] Jessa M. and Matuszewski L.: Producing random bits with delay-line-based ring oscillators. Int. Journal of Electronics and Telecommunications, vol. 59, No. 1, pp. 41-50, 2013.
[8] Sunar B., Martin W. J., and Stinson D. R.: A provably secure true random number generator with built-in tolerance to active attacks. IEEE Trans., Comput., vol. 56, pp. 109-119, Jan. 2007.
[9] Wold K. and Tan C. H.: Analysis and enhancement of random number generator in FPGA based on oscillator rings. Int. J. of Reconfiugurable Computing, vol. 2009, pp. 1-8, 2009.
[10] Jessa M. and Jaworski M.: 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.
[11] Markettos A. T. and Moore S. M.: 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.
[12] Jessa M.: On the Quality of Random Sequences Produced with a Combined Random Bit Generator. IEEE Transactions on Computers, Vol. 64, No. 3, March 2015, pp. 791-804.
[13] Bucci M. and Luzzi R.: Fully digital random bit generators for cryptographic applications. IEEE Trans. Circuits and Syst. I: Regular Papers, vol. 55, pp. 861-875, April 2008.
[14] Łoza Sz., Matuszewski Ł., Jessa M.: A Random Number Generator Using Ring Oscillators and SHA-256 as Post-Processing. Int. Journal of Electronics and Telecommunications, vol. 61, No. 2, pp. 199-204.
[15] Bertoni G., Daemen J., Peeters M., Van Assche G.: Keccak sponge function family main document. Available at: http://keccak.noekeon.org/Keccak-main-2.1.pdf.
[16] Bertoni G., Daemen J., Peeters M., Van Assche G.: Cryptographic sponge functions. Available at: http://sponge.noekeon.org/.
[17] Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., Leigh S., Levenson M., Vangel M., Banks D., Heckert A., Dray J., Vo S.: 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/.
[18] Krishna Dahal R., Bhatta J., Nath Dhamala T.L Performance Analysis of SHA-2 and SHA-3 finalists. International Journal on Cryptography and Information Security, Vol.3, No. 3, September 2013.

*Received: 02.04.2015 Paper reviewed Accepted: 02.06.2015*

**Szymon ŁOZA, MSc, eng.**

Graduate of Poznan University of Technology. He gained Bachelor of Engineering degree in specialization of Computer and Information Systems Security/Information Assurance within Computer Science studies at the Faculty of Electrical Engineering. He graduated master studies also in Computer Science at the Faculty of Computer Science, specializing in Embedded System and Mobile Devices. Currently a PhD student at the Faculty of Electronics and Telecommunications. Area of his interests are systems security/information assurance.

*e-mail: szymon.piotr.loza@gmail.com*

**Łukasz MATUSZEWSKI, MSc, eng.**

He received the M.S. degree from Poznan University of technology in 2010. Now he prepares his Ph.D. thesis. He works in the Chair of Telecommunication Systems and Optoelectronics of Poznan University of Technology as teaching assistant. His research interest include randomness in digital logic, reconfigurable systems and Field Programmable Gate Arrays.

*e-mail: lukasz.matuszewski@et.put.poznan.pl*

**Mieczysław JESSA, DSc, eng.**

He works in the Chair of Telecommunication Systems and Optoelectronics at Poznan University of Technology. Initially, his research interest included phase-locked loops and network synchronization. In the years 1995-1997 he was an expert of the Polish Ministry of Communications. His current research concerns the mathematical models of randomness and pseudo-randomness and the applications of the chaos phenomenon. He is the author or co-author of over one hundred journal and conference papers and fifteen patents.

*e-mail: mjessa@et.put.poznan.pl*

**Paweł KUBCZAK, MSc, eng.**

He received the M.S. degree from Poznan University of Technology in 2013. He continues studies at Faculty of Electronics and Telecommunications and prepares the Ph.D. thesis. His research interest include: digital measurement systems, time interval error measurement, randomness in digital logic, microcontrollers, and Field Programmable Gate Arrays.

*e-mail: szymon.piotr.loza@gmail.com*