Hardware-Efficient Post-processing Architectures for True Random Number Generators

Vladimir Rožić, and Ingrid Verbauwhede *Fellow, IEEE*

***Abstract*—In this brief we present novel post-processing modules for use in True Random Number Generators. These modules are based on mathematical constructs called *strong blenders*, which provide theoretical guarantees for the randomness of the output. We start by pointing out problems with current post-processing methods used in state-of-the-art TRNG designs. We present three novel hardware-efficient architectures and provide guidelines for choosing the design parameters.**

***Index Terms*—Cryptography, Entropy, Random Number Generation**

I. Introduction

Hardware True Random Number Generators (TRNGs) are used in all devices that require secure communication, device authentication or data encryption. Applications include smart cards, RFID tags and IoT devices.

TRNGs used in cryptography are subject to strict certification procedure. In the past, the security of TRNG designs was evaluated by running a set of statistical tests such as NIST 800–22 [1] and DIEHARD [2]. However, as pointed out in [3], the statistical features exploited by future cryptanalysis techniques cannot be foreseen in advance. Therefore, it is a risky practice to rely only on a finite set of statistical tests to verify the security of a random number generator. A notable incident happened in 2003 when the Motorola TRNG [4] was attacked [5] only one year after the details of the design were disclosed.

Today's certification authorities [6], [7] require a theoretical explanation for the unpredictability of generated data. Based on the theoretical model of the digital noise source (DNS), a designer has to make an *entropy claim* – i.e. a lower bound of the generated entropy. Once this bound is determined, an appropriate digital post-processing method is used to compress the sequence of raw numbers into a shorter sequence of full-entropy random numbers that could be used by the application.

While TRNGs presented in open literature often achieve impressive results in terms of throughput, energy and hardware area, they rarely follow all necessary requirements for use in cryptography. A common mistake is the wrong choice of the post-processing algorithm. For example, designs presented in [8], [9] use Von Neumann's post processing [10]. This method works only when the probability of generating the output bit 1 doesn't change over time and when there is no correlation between the generated bits. Unfortunately, these conditions are never met in practice. TRNGs presented in [11], [12] use a parity filter for post-processing, while a design from [13] uses an xor gate to combine the outputs of two independent physical sources of randomness. In some specific cases, these methods increase the min-entropy of the output, but they don't provide general-case theoretical guarantees. TRNG designs presented in [14]–[16] use the LFSR-based whitening schemes instead of post-processing. Such schemes don't compress the data and thus don't increase the entropy per bit. In addition, many designs [17]–[19] don't use any post-processing because the raw bits pass NIST 800–22 statistical tests. As illustrated by the attack [5] on the Motorola TRNG, this is not a good practice. We stress that the Motorola TRNG was also able to pass all statistical tests from the DIEHARD suite [2].

NIST special publication 800–90B [7] recommends using one of the vetted post-processing methods based on cryptographically secure primitives such as block ciphers or cryptographic hashes. However, these methods [20], [21] have a high cost in terms of area, energy and throughput which makes them unsuitable for lightweight applications. To the best of our knowledge, the only attempt to implement a mathematically secure, hardware-efficient post-processing method was made by Intel in their μRNG design for IoT applications [22]. This method was based on a single finite field addition and multiplication, a construct that was proposed and theoretically analyzed in [23]. Unfortunately, the implementation presented in [22] uses the wrong choice of the finite field and the design parameters, and thus fails to provide security guarantees.

Motivated by the current lack of mathematically-secure post-processing modules in the TRNG state-of-the-art, we propose three hardware-efficient post-processing architectures suitable for compact implementations. These architectures are based on mathematical constructs called *strong blenders* [24], which provide theoretically proved guarantees for the statistical quality and unpredictability of the output. The only requirement imposed on the digital noise source is that it produces sufficient amount of min-entropy, which makes these post-processing methods compatible with all physical sources of randomness. For all three architectures we provide a method for selecting design parameters given the min-entropy of the digital noise source.

II. Definitions and Problem Statement

Table I summarizes the notation used in this paper. We will use concepts of min-entropy and Shannon entropy. Min-entropy is equal to the information content of *the most likely outcome* of a distribution. It is either smaller or equal to the

The authors are with COSIC, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium (e-mail: Vladimir.Rozic@esat.kuleuven.be; Ingrid.Verbauwhede@esat.kuleuven.be).

TABLE I: Notation.

Symbol	Description
x	Variables.
x_i	Possible outcomes of a variable x.
$\mathcal{X}$	Probability distributions.
$x \leftarrow \mathcal{X}$	Variable x is sampled from distribution $\mathcal{X}$.
$\Pr_{x \leftarrow \mathcal{X}}[event]$	Probability of an event given that $x \leftarrow \mathcal{X}$.
$\mathcal{U}_n$	Uniform distribution over the domain $\{0,1\}^n$.
$(\mathcal{X}, \mathcal{Y})$	Joint distribution of $\mathcal{X}$ and $\mathcal{Y}$.
$H_\infty(\mathcal{X})$	Min-entropy of a distribution $\mathcal{X}$. $H_\infty(\mathcal{X}) = -\log_2\left(\max Pr[\mathcal{X} = i]\right)$.
$H_1(\mathcal{X})$	Shannon entropy of a distribution $\mathcal{X}$. $H_1(\mathcal{X}) = -\sum_i Pr[\mathcal{A} = i] \cdot \log_2\left(Pr[\mathcal{A} = i]\right)$.
$f(x, y)$	Output of a function for input values x and y.
$f(\mathcal{X}, \mathcal{Y})$	Distribution of the function output for the input distributions $\mathcal{X}$ and $\mathcal{Y}$.

Shannon entropy which represents the *average* information content of a distribution. Equality holds only for the uniform distribution. Statistical distance is used to measure the proximity of two distributions over the same domain. This metric is often used to quantify the difference between the output distribution and the ideal (uniform) distribution.

Definition 1. *The **statistical distance** SD between two distributions $\mathcal{A}$ and $\mathcal{B}$ over the domain D is:*

$$SD(\mathcal{A}, \mathcal{B}) = \frac{1}{2} \sum_{i \in D} |Pr[\mathcal{A} = i] - Pr[\mathcal{B} = i]| \,. \tag{1}$$

The statistical distance always takes a value between zero and one, where zero is achieved only for identical distributions.

In this work, we will also consider a more strict metric called ε-robustness. Loosely speaking, a random variable is ε-robust if there is no outcome that is significantly more likely or significantly less likely than any other outcome. The degree of significance is quantified using parameter ε.

Definition 2. *A distribution of an n–bit variable is said to be ε–robust if the probability of any outcome $x_i \in \{0,1\}^n$ is within the following bounds:*

$$2^{-n} \cdot (1 - \varepsilon) \leq \Pr_{x \leftarrow \mathcal{X}}[x = x_i] \leq 2^{-n} \cdot (1 + \varepsilon) \,. \tag{2}$$

A. Entropy Extractors

The first method for entropy extraction from biased bit-sequences was proposed by Von Neumann in [10]. However, this method has limited applicability because it requires that the input bits are independent and identically distributed. A very general class of entropy sources, limited only by the level of min-entropy, was introduced by Chor and Goldreich in [25]. These randomness generators are called (l, b)-sources, and they produce l-bit strings with min-entropy of at least b – i.e. each of the 2^l possible outcomes is generated with a probability lower than 2^{-b}. This class is suitable for modeling all entropy sources from nature because it only requires that some amount of min-entropy is generated without making any additional assumption about the underlying distributions. The extractor theory deals with the problem of post-processing data from (l, b)-sources to obtain a uniformly distributed output. We note that the perfect extraction is not possible, but it is only possible to guarantee that the output of the extractor is statistically close to $\mathcal{U}$ up to an arbitrarily small constant δ. In addition, we note that it is impossible to construct a single-source entropy extractor–at least two independent sources have to be used [25].

Definition 3. *A (l, b, w, δ) two-source extractor $Ext : \{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}^w$ is a function such that for any two (l, b)-sources $\mathcal{X}$ and $\mathcal{Y}$, the output is statistically close to $\mathcal{U}_w$:*

$$SD(Ext(\mathcal{X}, \mathcal{Y}), \mathcal{U}_w) < \delta \,. \tag{3}$$

The concept of strong blenders was introduced in [24]. A strong blender is an entropy extractor for which the output is independent of one of the inputs.

Definition 4. *A (l, b, w, δ) two-source strong blender $Ble : \{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}^w$ is a function such that for any two (l, b)-sources $\mathcal{X}$ and $\mathcal{Y}$, the output is statistically close to $\mathcal{U}_w$ and independent of $\mathcal{Y}$:*

$$SD((Ble(\mathcal{X}, \mathcal{Y}), \mathcal{Y}), (\mathcal{U}_w, \mathcal{Y})) < \delta \,. \tag{4}$$

The following method for constructing strong blenders was proposed and proven in [26]:

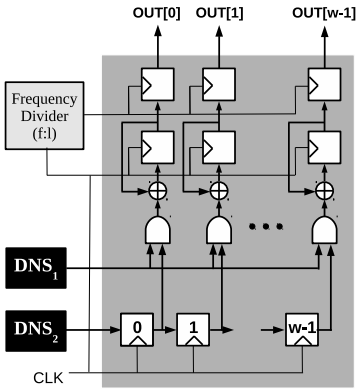
- Let $A_1, ..., A_w$ denote $l \times l$ binary matrices and let r denote an integer such that $r < l$. Further, let the matrices $A_1, ..., A_w$ have the following property:
$$rank\Big(\sum A_i\Big) \geqslant l - r \,, \tag{5}$$
where the summation is done over any non-empty subset of $\{1, ..., w\}$.
- A function $Ble : \{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}^w$ is defined as:
$$Ble(x, y) = ((xA_1) \cdot y, (xA_2) \cdot y, ...(xA_w) \cdot y) \,, \tag{6}$$
where $\cdot$ denotes the inner product and xA_i denotes the matrix multiplication.
- Function $Ble(x, y)$ is a (l, b, w, δ) two-source strong blender where:
$$\delta \leqslant 2^{-\frac{2b+2-(l+r+w)}{2}} \,. \tag{7}$$

B. Problem statement

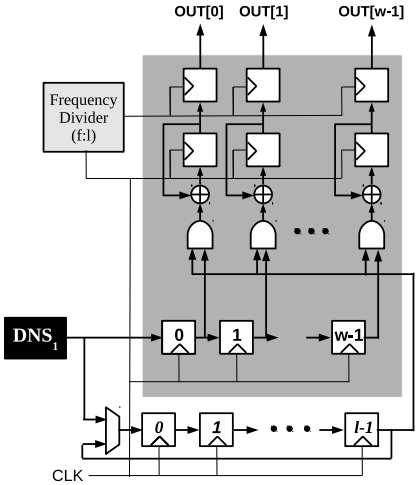
Our goal is to develop a digital post-processing method that provides theoretical guarantees for the quality of the output bits. We require this module to be universal in the sense that the guarantees are provided for any digital noise source with sufficient level of min-entropy. Since there is no known mathematical method which guarantees uniformity (full-entropy) for all such sources, we settle for a slightly weaker requirement; we require ε-robustness of the output for an arbitrarily small constant ε set by the designer. We further require that this post-processing module has a compact hardware implementation.

We assume that the DNS has a constant data rate of one raw bit per clock cycle. In addition, we assume that the DNS can be modeled as the (l, b) source, i.e. the output entropy is always higher than some value b that can be estimated at design time. Both assumptions are in line with most TRNG designs available in literature.

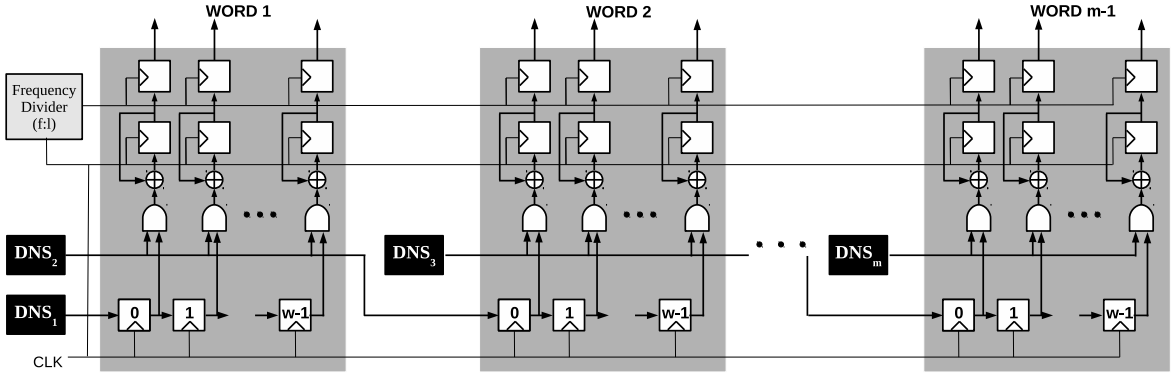
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2018.2881559, IEEE Transactions on Circuits and Systems II: Express Briefs



(a) The standard extractor architecture using two independent Digital Noise Sources.



(b) A modified architecture with entropy recycling using a single Digital Noise Source.



(c) A modified architecture with entropy sharing. The architecture uses m Digital Noise Sources to generate $m-1$ output words.

Fig. 1: Architectures of the post-processing modules based on strong blenders.

TABLE II: The minimal value of l needed to guarantee ε-robustness.

ε	2^{-64}				2^{-32}				2^{-16}			
α	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
$w=1$	651	326	217	163	331	166	110	83	171	86	57	43
$w=8$	791	396	264	198	471	236	157	118	311	156	104	78
$w=16$	951	476	317	238	631	316	210	158	471	236	157	118
$w=32$	1271	636	424	318	951	476	317	238	791	396	264	198
$w=64$	1911	956	637	478	1591	796	530	398	1431	716	477	358
$w=128$	3191	1596	1064	798	2871	1436	957	718	2711	1356	904	678
$w=256$	5751	2876	1917	1438	5431	2716	1810	1358	5271	2636	1757	1318

III. Proposed Architectures

We propose three hardware-efficient architectures for post-processing modules. These architectures are based on a two-source strong blender [26] which consumes l bits from each source and produces a single w-bit word. Such blender can be constructed using any set of $l \times l$ matrices with a property given in Equation (5). We opt for the right-shift matrices because of their efficient hardware implementation and their compatibility with bit-serial DNS architectures. These are superdiagonal matrices given by:

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \qquad A_{i+1} = A_1 A_i \,. \tag{8}$$

A bit vector multiplied by A_i results in the same bit-vector shifted by i positions to the right. In bit-serial architectures this multiplication is implemented by simply delaying the bit-sequence for i clock cycles. A sum of any subset of matrices $A_1, \dots A_w$ results in a matrix with a rank of at least $l-w$. Thus,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2018.2881559, IEEE Transactions on Circuits and Systems II: Express Briefs

by equation (5) $r = w$, and by equation (7), the statistical distance of the output from $\mathcal{U}_w$ is limited to:

$$\delta < 2^{-(b+1-w-l/2)}\,. \tag{9}$$

A. Hardware Architectures

Fig. 1 shows the architectures of the proposed post-processing modules. A straightforward implementation using two DNSs is shown in Fig. 1a. In this architecture the strong-blender is used as a two-source extractor. Multiplications $A_i x$ are implemented by delaying an input bit-stream by i clock cycles. Inner products are implemented using an AND gate, an XOR gate and a flip-flop for storing intermediate results. The computation is performed in l-clock cycles while the sources generate raw bits, after which the result is stored in the w-bit output register.

Higher utilization of available entropy can be achieved by exploiting the independence of the strong blender output from one of the inputs. In the architecture shown in Fig. 1b one of the input sequences is reused for generating multiple output words. This architecture uses the same computational core (shown in gray) as the two-source architecture, but it requires only one DNS. The operation consists of two phases: the setup phase and the entropy extraction phase. In the setup phase, an l-bit sequence is generated and stored in the circular-shift register. During the extraction phase, this sequence is rotated through this register providing one input for the computational core, while the DNS generates the data for the second input. Between these two phases, the DNS should be restarted in order to guarantee the independence of the two inputs. This way, we bypass the requirement for two distinct entropy sources.

Fig. 1c shows an architecture that is suitable for high-throughput designs. This architecture uses m sources and $m-1$ computational cores. Each generated bit-sequence is used at most twice, thereby avoiding the risk that a single corrupted sequence affects many output words.

The selection of the optimal architecture should be guided by the application requirements and the properties of the used digital noise source. The choice between the one-source and the two-source architecture comes down to the choice between an l-bit shift register and a DNS. In case that the selected DNS performs better in terms of the the metric of interest (area, power or energy) compared to the shift register, a two-source architecture should be used. Otherwise, a single source architecture is optimal. The multiple source architecture shown in Fig. 1c should be used in case when the throughput requirement cannot be achieved using the other two architectures.

B. Choosing Design Parameters

To guarantee the correct operation of the proposed post-processing modules, it is necessary to select sufficiently high input word length l. The minimal required l depends on the target output word length w, the target ε-robustness and the amount of min-entropy generated by the DNS. Typically, a designer has to make an estimation of the lower bound on the min-entropy generated by the source. We will quantify this estimation using parameter α such that $\alpha \leqslant b/l$. The estimation of this parameter should be derived from the theoretical model of the entropy source. To find the required input word length l, we rely on the following Lemma:

TABLE III: The minimal value of l needed to guarantee compatibility with AIS-31.

α	0.6	0.7	0.8	0.9
$w = 1$	100	50	34	25
$w = 8$	240	120	80	60
$w = 16$	400	200	134	100
$w = 32$	720	360	240	180
$w = 64$	1360	680	454	340
$w = 128$	2640	1320	880	660
$w = 256$	5200	2600	1734	1300

TABLE IV: Area (Gate equivalent) results and comparison with conventional solutions.

This Work (1-DNS)							
w	1	8	16	32	64	128	256
l	25	60	100	180	340	660	1300
Area (GE)	166.3	518.6	922	1728	3350	6592	13075
This Work (2-DNS)							
w	1	8	16	32	64	128	256
Area (GE)	22.7	176.7	353	706	1420	2839	5685
[21]							
	VN		100 XOR			PRESENT	
Area (GE)	20		267			1171	

Lemma 1. *Any w-bit random distribution $\mathcal{X}$ such that $SD(\mathcal{X},\mathcal{U}_w) < 2^{-w} \cdot \varepsilon$, is ε-robust.*

Proof. For any $x_i \in \{0,1\}^w$:

$$\Pr_{x \leftarrow \mathcal{X}}[x = x_i] \leqslant 2^{-w} + SD(\mathcal{X},\mathcal{U}_w) < 2^{-w}(1+\varepsilon)\,,$$

$$\Pr_{x \leftarrow \mathcal{X}}[x = x_i] \geqslant 2^{-w} - SD(\mathcal{X},\mathcal{U}_w) > 2^{-w}(1-\varepsilon)\,.$$

□

From Lemma 1 and equation (9) follows that for any value of $\alpha \geqslant 0.5$ and any target values of output word length w and ε-robustness, the required length l of the bit sequence can be computed as:

$$l \geqslant \frac{2w - 1 + log_2(\frac{1}{\varepsilon})}{\alpha - \frac{1}{2}}\,. \tag{10}$$

Table II sums up the minimal required l for different values of w, α and ε. As expected, for higher levels of min-entropy α, fewer input bits are required. We note that extraction efficiency increases for higher output word length w, e.g. for $\alpha = 0.9$ extracting a single 2^{-64}-robust bit reduces the throughput by a factor of 163 while extracting 256-bit words under the same conditions and requirements reduces the throughput by a factor of less than six.

Sometimes, designer's goal is not to provide a high level of ε-robustness, but only to comply with the AIS-31 recommendations. AIS-31 requires that the Shannon entropy of each output bit is at least 0.997. To simplify the analysis we set a

more strict requirement that each generated bit should have min-entropy level of at least 0.997. When the output word has min-entropy of $w-0.003$ this requirement is certainly fulfilled. Thus, we can make the following conservative estimate for l:

$$l \geqslant \frac{2w - 1 - log_2(2^{0.003} - 1)}{\alpha - \frac{1}{2}}. \qquad (11)$$

Table III sums up the required l for different values of w and α. Again we see that the efficiency of the entropy extraction increases with min-entropy per bit α and with the size of the output word w. The proposed designs were synthesized using the open cell library NanGate 45 nm. The area results for selected parameter values are summed up in Table IV. For one-source architecture, the value of parameter l is chosen such that the output fulfills AIS-31 requirements for $\alpha = 0.9$. For sources with lower entropy, this post-processing is more area consuming because it requires a longer shift-register. The area of the two-source post-processing doesn't depend on l. For 32-bit or shorter output word, this post-processing consumes less than 1 kGE. These results are compared with three conventional types of post-processing (Von Neumann, xor with 100 stages and a lightweight block cipher PRESENT) [21]. We note that even though Von Neumann's and xor post processing methods have low area implementations they are not suitable for many entropy sources. PRESENT-based post-processing consumes more area than strong-blender implementations with short output words.

IV. Conclusion

Post-processing is an essential component of every TRNG because it compensates for non-ideal nature of physical entropy sources. However, this important aspect of TRNG design is often neglected, and existing solutions usually don't provide any theoretical guarantees. In this brief we have proposed post-processing modules based on strong-blenders. We developed lightweight hardware architectures that are compatible with bit-serial sources of randomness. Unlike previously used techniques such as Von Neumann filter or XOR post-processing, the proposed method is compatible with all physical sources of randomness that produce more than 0.5 bits of min-entropy.

Acknowledgment

This work was supported in part by the Research Council KU Leuven C16/15/058, by the Flemish Government, FWO G.0876.14N, by the Hercules Foundation AKUL/11/19, and by the European Commission through the Horizon 2020 research and innovation programme Cathedral ERC Advanced Grant 695305. Vladimir Rožić is a Postdoctoral Fellow of the Fund for Scientific Research – Flanders (FWO).

References

[1] A. Rukhin et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications." NIST Special Publication 800-22, August 2008.

[2] G. Marsaglia, "Diehard test suite," 1998.

[3] B. Barak, R. Shaltiel, and E. Tromer, "True Random Number Generators Secure in a Changing Environment," in *Cryptographic Hardware and Embedded Systems - CHES*, 2003, pp. 166–180.

[4] T. E. Tkacik, "A Hardware Random Number Generator," in *Cryptographic Hardware and Embedded Systems - CHES*, 2002, pp. 450–453.

[5] M. Dichtl, "How to Predict the Output of a Hardware Random Number Generator," in *Cryptographic Hardware and Embedded Systems - CHES*, 2003, pp. 181–188.

[6] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators," ser. BSI, Bonn, 2011.

[7] M. S. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation." NIST Special Publication 800-90B, 2018.

[8] R. Brederlow, R. Prakash, C. Paulus, and R. Thewes, "A low-power true random number generator using random telegraph noise of single oxide-traps," in *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, Feb 2006, pp. 1666–1675.

[9] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadyay, "An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 452–456, April 2017.

[10] J. Von Neumann, "Various techniques used in connection with random digits," *National Bureau of Standards Applied Mathematics*, pp. 36–38, 1951.

[11] V. Rožić, B. Yang, W. Dehaene, and I. Verbauwhede, "Highly efficient entropy extraction for true random number generators on FPGAs," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, 2015, pp. 116:1–116:6.

[12] M. K. et. al., "A 82 nW Chaotic Map True Random Number Generator Based on a Sub-Ranging SAR ADC," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1953–1965, July 2017.

[13] V. von Kaenel and T. Takayanagi, "Dual True Random Number Generators for Cryptographic Applications Embedded on a 200 Million Device Dual CPU SoC," in *2007 IEEE Custom Integrated Circuits Conference*, Sept 2007, pp. 269–272.

[14] D. Liu, Z. Liu, L. Li, and X. Zou, "A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 6, pp. 608–612, June 2016.

[15] E. Kim, M. Lee, and J. J. Kim, "8Mb/s 28Mb/mJ robust true-random-number generator in 65nm CMOS based on differential ring oscillator with feedback resistors," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 144–145.

[16] A. T. Do and X. Liu, "25 fJ/bit, 5 Mb/s, 0.3 V True Random Number Generator With Capacitively-Coupled Chaos System and Dual-Edge Sampling Scheme," in *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov 2017, pp. 61–64.

[17] N. L. et al., "A True Random Number Generator using Time-dependent Dielectric Breakdown," in *2011 Symposium on VLSI Circuits - Digest of Technical Papers*, June 2011, pp. 216–217.

[18] S. Robson, B. Leung, and G. Gong, "Truly Random Number Generator Based on a Ring Oscillator Utilizing Last Passage Time," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 12, pp. 937–941, Dec 2014.

[19] H. Xu et al., "A16 × 16pixel post-processing free quantum random number generator based on spads," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 627–631, May 2018.

[20] C. P. Rentería-Mejía and J. Velasco-Medina, "High-throughput ring-lwe cryptoprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2332–2345, Aug 2017.

[21] V. B. Suresh and W. P. Burleson, "Entropy and energy bounds for metastability based trng with lightweight post-processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 7, pp. 1785–1793, July 2015.

[22] S. K. Mathew et al., "μRNG: A 300-950 mV, 323 Gbps/W All-Digital Full-Entropy True Random Number Generator in 14 nm FinFET CMOS," *J. Solid-State Circuits*, vol. 51, no. 7, pp. 1695–1704, 2016.

[23] B. Barak, R. Impagliazzo, and A. Wigderson, "Extracting randomness using few independent sources," *SIAM J. Comput.*, vol. 36, no. 4, pp. 1095–1118, 2006.

[24] Y. Dodis and R. Oliveira, "On Extracting Private Randomness over a Public Channel," in *APPROX–RANDOM 2003, Princeton, NJ, USA, August 24-26*, 2003, pp. 252–263.

[25] B. Chor and O. Goldreich, "Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity (Extended Abstract)," in *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October*, 1985, pp. 429–442.

[26] Y. Dodis, A. Elbaz, R. Oliveira, and R. Raz, "Improved Randomness Extraction from Two Independent Sources," in *APPROX–RANDOM 2004, Cambridge, MA, USA, August 22-24*, 2004, pp. 334–344.