Entropy Computation for Oscillator-based Physical Random Number Generators

David Lubicz, Viktor Fischer

Entropy Computation for Oscillator-based Physical Random Number Generators

David Lubicz[1,2] and Viktor Fischer[3,4]

[1]Direction Générale pour l'Armement, Bruz, France
[2]Institut de Recherche Mathématiques de Rennes, France
[3]Laboratoire Hubert Curien, Université Jean Monnet, Member of Université de Lyon, Saint-Etienne, France
[4]Department of Information Security, Faculty of Information Technology, Czech Technical University in Prague, Czech republic

January 4, 2024

Abstract

In this paper, we provide a complete set of algorithms aimed at the design and security evaluation of oscillator-based True Random Number Generators (TRNG). While depending on some TRNG design assumptions, the proposed algorithms use as inputs the statistical parameters of the underlying random physical process such as the clock jitter originating from the thermal noise and give a lower bound of the entropy rate of the generated bit stream as output. We describe the general structure of a TRNG composed of multiple free-running oscillators and samplers, the outputs of which are post-processed by an entropy conditioner. Depending on the specification of the entropy conditioner, which can usually be any Boolean function, we describe several algorithmic optimizations. We then explain how to compute and efficiently manage the entropy rate at the output of such a post-processing block and at the output of the generator as a whole.

Keywords:

Hardware random number generators, free-running oscillators, stochastic models, entropy, dedicated statistical tests.

1 Introduction

Random numbers play a crucial role in modern cryptography as confidential keys, initialization vectors, padding values, nonces, and also as random masks in side-channel attack countermeasures. A recent implementation of a random number generator targeting cryptography applications is composed of two stages: a non deterministic stage,

called True Random Number Generator (TRNG) and a deterministic stage, called Deterministic Random Number Generator (DRNG).

The TRNG serves as an entropy source – it should guarantee unpredictability of generated numbers even for an attacker with an unlimited computational power. The DRNG, which uses TRNG output as a seed, guarantees the output of the random number generator is unpredictable for an attacker with limited computational power even if the TRNG partially or temporarily fails. DRNGs should be designed as cryptographic modes with a model of security (see [2]) and associated proof by reducing the difficulty involved in breaking an underlying cryptographic primitive (for instance, the AES cipher).

This paper deals with a general problem in designing a TRNG for cryptography – entropy extraction from several low entropy sources and entropy management in the resulting combined bit stream. Designing a TRNG with a proven security level that can be easily implemented in logic devices is still a challenge. The security level reached is assessed by the entropy rate per output bit or vector of bits. For practical reasons, Shannon entropy [21] is usually used.

To rigorously estimate the entropy rate, it is necessary to go through a general TRNG design process in several steps, some of which may be difficult to master and may also be limited by constraints determined by the available technology.
The designer should proceed in the following steps:

1. *Identify the physical process that causes the random behavior of the generator.* The causes of such a behavior in electronic equipment including logic devices are essentially random analog electric noises: a random noise is by definition an unpredictable phenomenon that cannot be manipulated by an attacker. Unfortunately, analog noises are not directly exploitable in logic devices. Consequently, designers have to find a way of converting the contribution of random analog noises into random behavior in the digital domain. This may be a process that converts analog noises into the clock jitter in the free-running oscillators and self-timed rings [22, 6], or into an uncertainty of the resumed state after a metastability event in metastable flip-flops [25], or into a number of oscillations caused by the oscillatory metastability [24], or to the selection of the final state after write collisions in dual-port RAMs [11] or after a startup of a bi-stable circuit or memory element [23].

2. *Build a stochastic model (or select an existing one) of the exploited random physical process, i.e. of a source of randomness.* The stochastic model of the random physical process gives the (statistical) law of evolution of its state in time, depending on certain input physical parameters.

3. *Design a TRNG that converts the output of the random physical process into a stream of bits or bit vectors and build a statistical model of the whole TRNG.* The stochastic model of the TRNG gives a distribution of probability of bits or vectors of bits depending on the input noise, the way it is converted into the digital domain, and the way the entropy is extracted from the physical process, depending on the input parameters of the physical process. Some input parameters of the TRNG are adjustable by design, e.g. the sampling period or accumulation time in oscillator

based TRNGs, whereas some depend only on the technology, e.g. jitter originated from thermal or flicker noise.

4. *Design a method (or select an existing one) to measure the input parameters of the stochastic model built in Step 2 and/or the model built in Step 3.* The measurement can be taken inside or outside the device. Embedded measurement methods like those presented in [10, 9] for TRNGs using clock generators are preferable, since they do not include contribution of external noise sources.

5. *Compute the entropy rate at the TRNG output and adjust the TRNG design parameters.* The TRNG design space is explored by using the model and its input parameters to obtain the desired entropy rate in a process called entropy management.

If any of the above steps in the TRNG design are omitted, the generator should not or even must not be used in high-security cryptographic applications, but only a very limited number of papers in which all these steps are addressed, have been published up to now. We cite for instance a free-running oscillator based TRNG [3, 10] or a phase-locked loop (PLL) based TRNG [9].

Building a stochastic model is not straightforward, since it requires a high level of expertise in both physics (electronics) and mathematics (statistics). Note that in the above mentioned TRNG design workflow, three levels of models could (and preferably should) exist:

- Model of the origin of random behavior – obviously the model of the random electric noise(s);
- Model of the random physical process including the underlying model of the electric noise(s) and of its conversion to the digital domain, i.e. a kind of an analog-to-digital conversion;
- Model of the TRNG including the model of the random physical process and the entropy extraction method (sampling of random signals, counting of random events, etc.).

Many different TRNG designs have been published in the last few years, but only a few include a statistical model. Some existing models are used to model dependence of the clock period of free-running oscillators (or clock generators in general) on random noises and especially on thermal noise [6, 3, 5], others estimate entropy using a simple urn model [22] or a general model based on sampling of a noisy signal [13].

But even in the case of oscillator based TRNG, which is arguably one of the simplest and best understood TRNG principles, a complete solution for the last step of the aforementioned TRNG design workflow has not yet been published. By a complete solution, we mean a description of a generic algorithmic suite that is efficient and easily adaptable to TRNGs using differences in clock phases as a source of randomness and that can be used to compute and guarantee a precise lower bound of the entropy rate.

For instance, in practice, the formula [3, Corollary 1] cannot be used to compute the entropy rate in most implementations of the oscillator based TRNG. The first problem is

that the formula is based on the assumption that the duty cycle is equal to 0.5, which is almost never the case in standard implementations. Certainly, one can easily tweak the formula [3, Corollary 1] in order to take a biased duty cycle into account. But a more serious concern is that this equation originates from a truncated Fourier series and as such, is an approximation that is only valid if the quality factor is sufficiently high (for a definition of the quality factor in this context see [3, Section 2.4]). For quality factors in the range of those used in practical applications, in Section 7, we show that the results of [3] do not apply.

Moreover, an oscillator-based TRNG is usually a complex structure composed of several free-running oscillators (e.g. ring oscillators) and samplers, whose outputs are somehow combined and post-processed using an entropy extraction algorithm. It is clear that this complex entropy extraction process has to be taken into account when estimating entropy rate.

The authors of [26] deal with the case when the entropy conditioner is an exclusive or (XOR) function. Their computations of a lower bound of the entropy rate of the TRNG are based on the results of [3, Corollary 1] and rely on the assumption that each time the TRNG produces a bit, its internal state (or to be more precise, its phase) is known by the attacker. This hypothesis, although very convenient for entropy computation, is not realistic, because the only knowledge the attacker gets from the TRNG when it outputs a bit, is the bit itself.

We will see that knowledge of the internal state, although not really discussed in [3, 26], leads to significant underestimation of the entropy rate of the TRNG. To assess the lower bound of the TRNG output entropy rate more precisely, we consider a more realistic security assumption, in which the attacker only knows the most recent output bits of the TRNG. Our solution is also more general since it can be used to model the operation of the entropy conditioner based on any Boolean function.

In this paper, we provide all the algorithms needed to compute a lower bound of the entropy rate produced by the thermal noise in a TRNG, in which multiple oscillators generate jittered clocks. These algorithms can easily be adapted to a wide range of designs of of TRNGs using differences in clock phases as a source of randomness. We can consider various versions of elementary oscillator-based TRNG (EO-TRNG)[3], multiple oscillator-based TRNG (MURO-TRNG) [22, 26], TRNG using self-timed rings (STR-TRNG) [6] or even coherent sampling oscillator-based TRNG (COSO-TRNG) [5] as suitable candidates, since in all these generators, a jittered clock signal is sampled on the edges of a reference clock signal. Note that the proposed algorithms are not suitable for evaluation of the transition effect ring oscillator-based TRNG (TERO-TRNG) [24], in which the randomness does not originate from differences in clock phases, but from the timing of collisions of two events circulating in the ring.

Furthermore, we discuss the complexity of the algorithms and potential difficulties with their application. All the algorithms have been implemented in Python and are freely available on a public server.

The paper is organized as follows: In Section 2, we describe the general scheme of oscillator based TRNGs we consider in the paper: Elementary Oscillator-based TRNG (EO-TRNG) and Multi-oscillator-based TRNG (MO-TRNG). In Section 3, we provide an overview of the entropy estimation algorithms and explain the main design choices we made. In Section 4, we explain how to model EO-TRNGs using a Markov chain,

and in Section 5, we clarify how to model MO-TRNGs including digitization and post-processing. In Section 6, we validate the models in simulations and illustrate the results using two security models: model A and B. In Section 7, we validate the new entropy estimation method in hardware experiments, discuss results we obtained and explain how the proposed algorithms can be used to manage the entropy rate at the MO-TRNG output. In Section 8, we conclude the paper and consider possible avenues for further study.

2 General scheme of oscillator-based TRNGs

In this section, we first describe the general structure of the simplest oscillator-based TRNG – Elementary Oscillator-based TRNG (EO-TRNG). This will help determine notations and specify inputs and outputs of the algorithms. We then generalize the same approach to construct what we call Multi-oscillator-based TRNG (MO-TRNG), variants of which can be found in the scientific literature. Indeed, our generalized theory can be adapted to many TRNG designs based on free-running oscillators, in which jittered clock signals are sampled on the edges of the reference clock signal.

A free-running oscillator, such as a ring oscillator or self-timed ring, produces a clock signal, the phase of which tends to drift because of the instability of the propagation time of electric signals across logic gates, termed phase jitter. The output signal $s(t)$ of a free-running oscillator O can be modeled by a periodic function of time t having the form:

$$s(t) = f_\alpha(\omega(t + \xi(t))), \tag{1}$$

where f_α is a given real valued 1-periodic function [3] such that $f_\alpha(x) = 1$ for $0 < x < \alpha$, $f_\alpha(x) = 0$ for $\alpha < x < 1$, and $f_\alpha(0) = f_\alpha(\alpha) = 1/2$. Here α is the duty cycle of the sampled oscillator. The term $\phi(t) = \omega(t + \xi(t))$ is the total phase of the oscillator where ω is its mean frequency and ξ accounts for the phase jitter.

The phase jitter ξ has different components, some are deterministic, e.g. periodic jitter or data dependent jitter, and others are non-deterministic. Only the non-deterministic (i.e. random) component of the phase jitter contributes to the output entropy of the TRNG. The non-deterministic component of the phase jitter may be a consequence of various physical phenomena, each of which may have different statistical properties, for instance, thermal noise, shot noise or flicker noise.

Thermal noise is the most frequently exploited noise in TRNGs, since it has been characterized, is well understood and modeled. Since thermal noise is statistically independent of other kinds of noises, its contribution to the entropy rate of a TRNG is added to contributions from other physical phenomena. As a consequence, only accounting for the contribution of thermal noise to the phase jitter is sufficient to determine a lower bound of the entropy rate at the TRNG output and to guarantee the security of the generator. In the following, we thus only consider the thermal noise component of the phase jitter.

Exactly like in [3], we (citation) "model the evolution of the total phase $\phi(t) = \omega(t + \xi(t))$ from Eq. (1), i.e. the phase of a ring oscillator depending on the thermal noise, as a Wiener stochastic process $\Phi(t)$ with drift $\mu > 0$ and volatility $\sigma^2 > 0$. In other words, for any time $t \geq t_0$, the phase $\Phi(t)$ conditioned by the value $\Phi(t_0) = \phi(t_0)$ follows

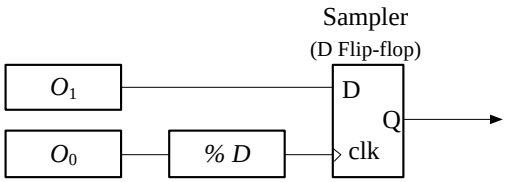

Figure 1: Schematic diagram of an elementary oscillator-based TRNG (EO-TRNG).

a Gaussian distribution of mean $\phi(t_0) + \mu(t - t_0)$ and variance $\sigma^2(t - t_0)$". Equivalently, in terms of conditional probability density, we have for all $t > t_0$, x, x_0,

$$\frac{d}{dx}\mathbb{P}\{\Phi(t) \leq x \mid \Phi(t_0) = x_0\} = \frac{1}{\sigma\sqrt{2\pi(t - t_0)}} \exp\left(\frac{-(x - x_0 - \mu(t - t_0))^2}{2\sigma^2(t - t_0)}\right). \quad (2)$$

Comparing the definition of μ with Eq. (1), we can see that $\mu = \omega$. Consequently, the parameters needed to model the probabilistic evolution of the phase jitter component caused by the thermal noise in oscillator O_i are:

- α_i – the duty cycle of the clock signal generated by the oscillator;
- T_i – its mean period;
- σ_i^2 – the volatility of the associated Wiener process.

In the following, we call the triplet $(\alpha_i, T_i, \sigma_i^2)$, the statistical parameters of oscillator O_i.

Once we have presented the background of a free-running oscillator, we introduce a RNG that exploits the phase jitter to produce random bits – an oscillator-based TRNG. In the simplest version of this kind of generator, which we call Elementary Oscillator-based TRNG (EO-TRNG) according to [3], the output of a free-running oscillator is sampled by a sampling operator at the time intervals defined by the reference clock. The sampling operator may be a synchronous D flip-flop, an asynchronous D-latch, or some more advanced structure like a synchronous or asynchronous counter.

The reference clock signal that determines the sampling periods can be produced by another free-running oscillator or by a quartz oscillator. However, note that the use of a free-running oscillator is preferable, because the use of two identical free-running oscillators (which generate both sampled and sampling clocks) significantly reduces the impact of global noises which can be manipulated. The frequency of the reference clock signal is divided by D. The value of D determines the jitter accumulation time and hence the entropy rate at the TRNG output (see [3]): if D increases, the output bit rate of the EO-TRNG decreases, but the entropy per bit increases and converges to one. Figure 1 represents a typical EO-TRNG that we consider in this paper: the sampling operator is

a D flip-flop and the sampled clock signal and reference clock signal are produced by two ring oscillators.

In [3, Appendix C], the output of oscillator O_1 with statistical parameters $(\alpha_1, T_1, \sigma_1^2)$, sampled at time intervals determined by oscillator O_0 featuring statistical parameters $(\alpha_0, T_0, \sigma_0^2)$, is shown to produce the same distribution of output bits as a stable clock signal (a jitter-free signal) with duty cycle 0.5 and period T_0 sampling an oscillator O_1 with statistical parameters $(\alpha_1, T_1, {\sigma'}_1^2)$ where

$$ {\sigma'}_1^2 = \left(\frac{T_0}{T_1}\right)^2 \sigma_0^2 + \sigma_1^2. \tag{3} $$

We have selected 0.5 arbitrarily for the duty cycle of the sampling signal as it does not alter the distribution of the output bits, since only the rising edge of O_0 affects the output of the D-flip-flop. Taking the above mentioned facts into account, we assume in the following that in an EO-TRNG, the sampled oscillator is the only oscillator subject to the phase jitter and that the sampling oscillator outputs a jitter-free clock signal.

We fix this setting in order to clarify the presentation, but it should be noted that all the TRNG variants we have mentioned up to now and possibly others can be taken into account in our algorithms with only minor modifications.

The sampling oscillator O_0 produces a stable clock signal, which is thus described by its triplet $(0.5, T_0, 0)$. On the other hand, the statistical parameters of the sampled oscillator O_1 are $(\alpha_1, T_1, {\sigma'}_1^2)$. By scaling the time by T_1, the parameters of the sampling oscillator (resp. the sampled oscillator) become $(0.5, T_0/T_1, 0)$ (resp. $(\alpha_1, 1, ({\sigma'}_1/T_1)^2)$). *In fine*, we see that the statistical properties of an EO-TRNG depend on three parameters.

Definition 1. *The statistical parameters of an EO-TRNG are:*

- $\alpha = \alpha_1$ – *the duty cycle of the sampled clock signal;*
- $\mu = T_0/T_1$ – *the drift of the Wiener process described in [3];*
- $\sigma^2 = ({\sigma'}_1/T_1)^2$ – *the volatility of the Wiener process described in [3].*

Note that in the preceding definition, knowing that the sampled signal is 1-periodic and assuming that the distribution of the output bits of the EO-TRNG remains unchanged, we can replace μ by $\mu' = T_0/T_1 \mod 1$ if $\mu' \neq 0$, otherwise $\mu' = 1$, and replace σ^2 by ${\sigma'}^2 = \sigma^2 \frac{\mu}{\mu'}$. In other words, if the volatility does not change then $\mu \in]0, 1]$.

The bit rate of a single EO-TRNG featuring a sufficient entropy rate is usually not high enough for practical applications. Moreover, it is preferable to have some security margin in the entropy rate obtained at the TRNG output, in order to account for possible flaws, aging and malfunctions. This is why an oscillator based TRNG is generally made of several EO-TRNGs working in parallel, the outputs of which are processed together by an entropy conditioning algorithm – a deterministic algorithm that combines (typically via a modulo 2 operation) several (non-deterministic) inputs, and outputs a high entropy bit stream.

Figure 2 shows the general scheme of a multi-oscillator-based TRNG (MO-TRNG) we consider in this paper. It is composed of ℓ elementary oscillators-based TRNGs,

EO-TRNG$_i$ for $i = 1, \ldots, \ell$ with statistical parameters $(\alpha_i, \mu_i, \sigma_i^2)$ that share the same reference clock generator (oscillator O_0). The outputs of *EO-TRNG*$_i$ are gathered in an entropy conditioner given by a function $\zeta_\ell(x_1, \ldots, x_\ell)$ of oscillators' output bits.

We denote by $(\alpha_i, \mu_i, \sigma_i^2)$ for $i = 0, \ldots, \ell$ the respective statistical parameters of O_0 (the sampling oscillator) and O_i for $i = 1, \ldots, \ell$ (the sampled oscillator(s)). We have to take care of the fact that, as EO-TRNG$_i$ share the same clock signal, we can not use Eq. (3) to compute ${\sigma'}_i^2$, $i = 1, \ldots, \ell$ for all sampled clocks. Instead, to avoid entropy overestimation, we should consider that the MO-TRNG is made of EO-TRNG$_i$, the phase jitter of which only comes from the sampled oscillator and thus take ${\sigma'}_i = \sigma_i$ (simple conservative approach). Alternatively, it is also possible to take into account the entropy coming from the sampling oscillator O_0 by taking ${\sigma'}_1^2 = (T_0/T_1)^2 \sigma_0^2 + \sigma_1^2$ just for one couple of rings and use ${\sigma'}_i^2 = \sigma_i^2$ for $i \geq 2$ (more precise solution).

Remark 1. *It would also have been possible to consider a MO-TRNG made of several EO-TRNGs each with their own clock reference. The drawback to such a design is that it might cause significant synchronization problems at the level of the entropy conditioner. It is nonetheless possible to simulate such a design with the techniques that we describe in the present paper.*

In this paper, we describe an efficient algorithm that takes $(\alpha_i, \mu_i, \sigma_i^2)$ as inputs and outputs the entropy rate per bit of the TRNG using the entropy conditioning function ζ_ℓ, while accounting for the thermal noise component of the phase jitter. As explained above, this entropy rate per bit guarantees the security of the TRNG aimed at cryptographic applications. We explain how to compute the Shannon entropy rate since it is one of important evaluation criteria in widely used evaluation methodology [20]. However, since our method is more general, it can be used to compute the full spectrum of Rényi entropies.

3 From Markov chains to the new security model

In [3], it is shown that the output bits of an oscillator-based TRNG are not independent so that its modeling by a non-trivial Markov chain is essential in the evaluation of entropy rate. In this section, we develop algorithms using Markov chains for that purpose. First, we recall the definition of a Markov chain, which is slightly adapted to our needs:

Definition 2. *Let* **bits** *be an alphabet (most of the time in the following* **bits** $= \{0, 1\}$*), k a positive integer and S_k the set of finite sequences of length k with value in* **bits***. We denote by $\mathcal{R}(S_k)$ the set of random variables with value in S_k. Let $\mathfrak{P}(S_k)$ be the set of subsets of S_k. We define the map* succ $: S_k \to \mathfrak{P}(S_k)$, $s = b_1 \ldots b_k \mapsto \{b_2 \ldots b_k b | b \in$ **bits**$\}$.

For m as a positive integer, a Markov chain X with memory $m \geq 1$ is a sequence $(X_i)_{i \geq 0}$ of elements of $\mathcal{R}(S_m)$, where S_m is the set of states, such that for all n, k positive integers, $n \geq k$:

1. $\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, \ldots X_{n-k} = x_{n-k}) = \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})$,
2. $\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})$ *does not depend on n.*

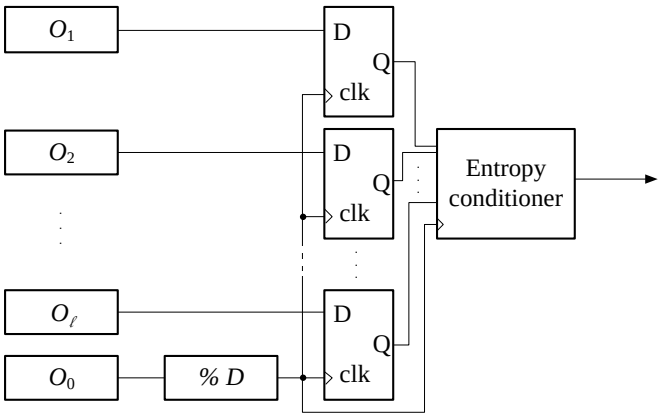

Figure 2: Schematic diagram of a multi-oscillator-based TRNG (MO-TRNG).

Moreover, we assume that if $x_n \notin \mathrm{succ}(x_{n-1})$ *then* $\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}) = 0$. *The matrix* $T(X) = (t_{x,y})$ *such that* $t_{x,y} = \mathbb{P}(X_n = x | X_{n-1} = y)$ *for* $x, y \in S_m$ *is called the transition matrix of* X. *We denote by* $\mathcal{M}(m, \mathbf{bits})$ *the set of Markov chains with memory* m, *over the alphabet* **bits**.

It is clear that succ maps any element $s \in S_m$ to the set of its possible successors. If $s' = b_1 \dots b_m \in \mathrm{succ}(s)$ then the transition from state s to s' is labeled by $s_m \in \mathbf{bits}$.

Definition 3. *We denote by* $\mathcal{T}(m)$ *the set of matrices* $T = (t_{x,y})_{x,y \in S_m}$ *such that* $t_{x,y} \in [0,1]$ *for all* $x, y \in S_m$, $t_{x,y} = 0$ *if* $x \notin \mathrm{succ}(y)$ *and* $\sum_{y \in S_m} t_{x,y} = 1$.

Note that a transition matrix of a Markov chain is in $\mathcal{T}(m)$. From our knowledge of a current state distribution $X_i \in \mathcal{R}(S_m)$ and transition matrix $T(X)$, we can compute the next state distribution $X_{i+1} \in \mathcal{R}(S_m)$ with:

$$\mathbb{P}(X_i = b_1 \dots b_m) = \sum_{b \in \mathbf{bits}} t_{b_1 \dots b_m, b\, b_1 \dots b_{m-1}} \mathbb{P}(X_{i-1} = b\, b_1 \dots b_{m-1}), \tag{4}$$

so that we can inductively recover $(X_i)_{i>0}$ from X_0. This motivates the following definition.

Lemma 1. *A Markov chain* $X = (X_i)_{i \geq 0}$ *is defined by the transition matrix* $T(X) = (t_{x,y}) \in \mathcal{T}(m)$ *and initial state* X_0 *in* $\mathcal{R}(S_m)$. *We denote such a Markov chain by* $X(X_0, T(X))$.

The representation of a Markov chain by a pair $(X_0, T(X))$ is not efficient in terms of time and space since most of the coefficients $t_{x,y}$ are 0. To obtain a representation that is better adapted to computations, note that from $X_0 \in \mathcal{R}(S_m)$ and $T(X) \in \mathcal{T}(m)$,

we can define inductively random variables, that we denote by $\hat{X}_0(k) \in \mathcal{R}(S_k)$ for all $k > m \geq 1$ by:

$$\begin{aligned}&\mathbb{P}(\hat{X}_0(k) = b_1 \dots b_k) \\ &\quad = \mathbb{P}(X_n = b_{k-m+1} \dots b_k | X_{n-1} = b_{k-m} \dots b_{k-1}) \mathbb{P}(\hat{X}_0(k-1) = b_1 \dots b_{k-1}). \end{aligned} \tag{5}$$

Note that $\hat{X}_0(m+1)$ makes it possible to recover $\hat{X}_0(m)$ using:

$$\mathbb{P}(\hat{X}_0(m) = b_1 \dots b_m) = \sum_{b \in \mathbf{bits}} \mathbb{P}(\hat{X}_0(m+1) = b_1 \dots b_m b). \tag{6}$$

But this means that $\hat{X}_0(m+1)$ makes it possible to recover the transition matrix $t_{x,y}$ for $x \in \mathrm{succ}(y)$ using Eq. (5) with $k = m+1$. Precisely, for $x = b_1 \dots b_m$ and $y = b_0 \dots b_{m-1}$ in S_m, we have:

$$t_{x,y} = \frac{\mathbb{P}(\hat{X}_0(m+1) = b_0 \dots b_m)}{\mathbb{P}(\hat{X}_0(m) = b_0 \dots b_{m-1})}. \tag{7}$$

We have the following lemma:

Lemma 2. *The map $\beta : \mathcal{R}(S_{m+1}) \to \mathcal{M}(m, \mathbf{bits})$, $\hat{R}(m+1) \mapsto X(X_0, T(X))$ where X_0 and $T(X)$ are obtained from $\hat{R}(m+1)$ respectively by Eq. (6) and (7), is bijective.*

Proof. It is enough to prove that the map $\beta_0 : \mathcal{R}(S_{m+1}) \to \mathcal{R}(S_m) \times \mathcal{T}(m)$ defined by Eq. (6) and (7) is bijective. Let us denote by $\gamma : \mathcal{R}(S_m) \times T(X) \to \mathcal{R}(S_{m+1})$ defined in (5). It is clear that μ is an inverse of β_0 and we are done. □

Definition 4. *The random variable $\hat{X}_0(m+1) \in \mathcal{R}(S_{m+1})$ representing $X(X_0, T(X))$ can be encoded by a function:*

$$\begin{aligned} f_{X(X_0,T(X))} : S_{m+1} &\to \mathbb{R}, \\ b_1 \dots b_{m+1} &\mapsto \mathbb{P}(\hat{X}_0(m+1) = b_1 \dots b_{m+1}). \end{aligned} \tag{8}$$

We say that $f_{X(X_0,T(X))}$ is the encoding function of $X(X_0, T(X))$.

This way of encoding any Markov chain in $\mathcal{M}(m, \mathbf{bits})$ takes $O(\log_2(|\mathbf{bits}|^{m+1}))$ memory bits and it is consequently optimal. Let $X = (X_i)_{i \geq 0} \in \mathcal{M}(m, \mathbf{bits})$ and suppose that $X = X(X_0, T(X))$ for $X_0 \in \mathcal{R}(S_m)$ and $T(X) \in \mathcal{T}(m)$. From the pair $(X_0, T(X))$, one can inductively compute $(X_j, T(X))$ using Eq. (4). Algorithm 1 does the same thing with the encoding function $f_{X(X_j,T(X))}$.

Definition 5. *An element X_{k_0} of the Markov chain $(X_i)_{i \geq 0}$ is stable, if for all $k \geq k_0$ condition $X_{k+1} = X_k$ is fulfilled.*

A Markov chain $(X_i)_{i \geq 0}$ with memory m verifies the ergodicity and irreducibility conditions, so that according to [1], it has a unique stable element, towards which it converges starting from any X_0. We denote this stable element in $\mathcal{R}(S_m)$ by X_∞. We have:

$$\lim_{i \to \infty} X_i = X_\infty. \tag{9}$$

Algorithm 1: Algorithm to compute $(X_{j+1}, T(X))$ from $(X_j, T(X))$.

notation: Let $X = (X_0, T(X))$ be the Markov chain with initial state X_0 and transition matrix $T(X)$.

We use the notations:

- for s= $b_1 \dots b_m \in S_m$ and $k = j, j+1$

$$\text{prob\_state}(k, b_1 \dots b_m) = \sum_{b \in \mathbf{bits}} f_{X(X_k, T(X)}(b_1 \dots b_m + b)$$

  where $+$ is the concatenation (see formula (6)) ;

- for s= $b_1 \dots b_m \in S_m$, $b \in \mathbf{bits}$:

$$\text{prob\_trans}(b_1 \dots b_m, b) = f_{X(X_j, T(X))}(b_1 \dots b_m + b)/\text{prob\_state}(j, b_1 \dots b_m),$$

  (see formula (7)).

input : $f_{X(X_j, T(X))}$ the encoding function of $X(X_j, T(X))$ the Markov chain with initial state X_j and transition matrix $T(X)$.

output : $f_{X(X_{j+1}, T(X))}$ the same notation as before, but with $j+1$.

for $s \in S_m$ **do**
  Write $s = b_1 \dots b_m$, $b_i \in \mathbf{bits}$;
  $ps0 \leftarrow' 0' + b_1 \dots b_{m-1}$;
  $ps1 \leftarrow' 1' + b_1 \dots b_{m-1}$;
  `/* Comment:` $ps0$ `and` $ps1$ `are the previous states of s */`
  p_next $\leftarrow$ prob_state$(j, ps0) *$ prob_trans$(ps0, b_m)$ + prob_state$(j, ps1) *$ prob_trans$(ps1, b_m)$;
  `/* Comment: this is the probability of state` s `at time` $j+1$ `*/`
  $f_{X(X_{j+1}, T(X))}(s +' 0') \leftarrow$ p_next $*$ prob_trans$(b_1 \dots b_m, 0)$;
  $f_{X(X_{j+1}, T(X))}(s +' 1') \leftarrow$ p_next $*$ prob_trans$(b_1 \dots b_m, 1)$;
end
return $f_{X(X_{j+1}, T(X))}$;

Algorithm 2: Algorithm to compute the stable state of a Markov chain $X = (X_i)_{i\geq 0}$.

input :

- $f_{X(X_0,T(X))}$ the encoding function of $X = X(X_0, T(X))$;
- $\epsilon > 0$ the precision of the computation.

output: $f_{X(X_\infty,T(X))}$ the encoding function of $X(X_\infty, T(X))$ where $X_\infty \in \mathcal{R}(S_m)$ is the stable element of X.

$k \leftarrow j$;
repeat
  Call Algorithm 1 to compute $f_{X(X_{k+1},T(X))}$ from $f_{X(X_k,T(X))}$;
until $\forall s \in S_{m+1}$, $|f_{X(X_{k+1},T(X))}(s) - f_{X(X_k,T(X))}(s)| < \epsilon$;
return $f_{X(X_{k+1},T(X))}$;

Algorithm 2 uses this last limit to compute an approximation of the stable element. In practice, the convergence is rapid: to obtain a precision of 10^{-3}, less than 100 iterations are usually required.

The link between the Markov chain and the Shannon entropy as a measure of security of the TRNG we want to compute is given by Definition 6.

Definition 6 ([1])**.** *The Shannon entropy of a memory m Markov chain $X = (X_i)_{i\geq 0}$, with stable element X_∞ and transition matrix $T = (t_{x,y})$ can be computed as:*

$$\mathcal{E}_m(X) = \sum_{x\in S_m} \sum_{y\in S_m} \mathbb{P}(\hat{X}_\infty(m) = y)(-t_{x,y} \log_2 t_{x,y}) \tag{10}$$

From the definition, we immediately deduce Algorithm 3 to compute the Shannon entropy of a Markov chain. Note that our definition of the Shannon entropy depends on parameter m. This is because we approximated the statistical behavior of the TRNG using an information source with memory m. When m tends to infinity, under general assumptions on the TRNG, which are fulfilled according to [3], this source of information converges towards a perfect statistical model of the TRNG. Thus the Shannon entropy of the TRNG is given by:

$$\lim_{m\to\infty} \mathcal{E}_m(X^m), \tag{11}$$

where $(X^m)^{m\geq 0}$ is a sequence of Markov chains with memory m approximating the output distribution of the TRNG.

This definition of the entropy rate of the TRNG may seem to be theoretical, but as we will explain shortly, this sequence converges rapidly, so that in practice $m = 10$ is usually sufficient to obtain a very good approximation of the entropy rate of the TRNG.

Before explaining the main ingredients of the algorithm, we specify the security model of the TRNG. According to the well known Kerckhoffs principle, we assume that the attacker has complete knowledge of the TRNG including its initial state. Moreover, we

Algorithm 3: Algorithm to compute the Shannon entropy of a Markov chain.

input :

- $f_{X(X_\infty,T(X))}$ the encoding function of $X = X(X_\infty, T(X))$.

output: The Shannon entropy of X.

$S \leftarrow 0$;
for $s \in S_m$ **do**
  $pr = f_{X(X_\infty,T(X))}(s +' 0') + f_{X(X_\infty,T(X)}(s +' 1')$;
  `/* Comment:` pr `is the probability of the state` s `*/`
  $tr = f_{X(X_\infty,T(X)}(s +' 0')/pr$;
  `/* Comment:` tr `is the probability of transition to the next states of` s `labelled by 0 */`
  $S \leftarrow S + pr.(-tr\log(tr) - (1-tr)\log(1-tr)$;
end
return S ;

assume that the attacker has infinite computational power and is able to observe the preceding TRNG output bits. The attacker aims to predict the next bit of the TRNG from this knowledge.

In this context, we can interpret the entropy rate of the TRNG as the quantity of additional information the attacker needs, to be able to perfectly predict one output bit of the TRNG. For instance, if the entropy rate is 1, the attacker would need full information contents of the following bit, which is therefore unpredictable. The general idea behind the algorithms presented in this paper is to compute an approximation of the statistical distribution of the TRNG output by modeling the attacker's knowledge using a Markov chain with memory m, for an m that is big enough to approximate the entropy rate of the TRNG via Eq. (10) with sufficient precision.

Using the assumptions we made on the EO-TRNG in Section 2, the state of an EO-TRNG at time t is given by the total phase of the sampling oscillator $\omega(t+\xi(t))$. Indeed, from this knowledge, according to Eq. (1), the attacker can predict the output of the EO-TRNG at time t via a deterministic function f_α representing a period of the sampled oscillator featuring duty cycle α.

The attacker's knowledge of the phase of an EO-TRNG can be represented by a random variable $X_e(t)$ with values in $\mathbb{R}$. Then the distribution of the output bit at time t is given by the random variable $f_\alpha(X_e(t))$. Let $p_e(t,x) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be the statistical distribution of $X_e(t)$. By following the evolution of this distribution over time, one can compute a Markov chain with memory m associated with the EO-TRNG and from the Markov chain the corresponding entropy rate. The evolution of $p_e(t,x)$ depends on two parameters:

- Physical noises, which tend to decrease the attacker's knowledge of the state of the generator;
- The sequences of output bits, which allow the attacker to get some information about the internal state of the generator.

To compute a Markov chain with memory m associated with a multi-oscillator-based TRNG (MO-TRNG) composed of ℓ elementary oscillator-based TRNGs, $EO\text{-}TRNG_i$ for $i = 1, \ldots, \ell$, at least two approaches are possible:

1. One can consider the space of phases V of all EO-TRNGs, with the dimension of ℓ (since we have seen that the phase space of one EO-TRNG has the dimension of 1) and then represent the attacker's knowledge by a random variable $X(t)$ on V, compute the evolution of $X(t)$ and the distribution of the output bits by $f_{\alpha_1,\ldots,\alpha_n}(X(t))$, where $f_{\alpha_1,\ldots,\alpha_n}$ is a multi-dimensional sampling function.

2. One can compute for $i = 1, \ldots, \ell$, Markov chains $(X_j^i)_{j\geq 0}$ with memory m for $EO\text{-}TRNG_i$, and then compute a final Markov chain with memory m obtained by combining the outputs of Markov chains X^i using the entropy conditioning function.

The first approach is very inefficient since it involves carrying out numerical integration in a space of the dimension of ℓ. In this paper, instead, we develop the second approach that is efficient and works for a general entropy conditioning function.

In Section 4, we explain how to compute a Markov chain with memory m associated with an EO-TRNG, and in Section 5, we describe an algorithm to compute a Markov chain, when the entropy is conditioned from ℓ Markov chains.

4 Modeling the EO-TRNG using a Markov chain

In this section, we provide an algorithm to compute a Markov chain $X = (X_j)_{j\geq 0}$ representing the statistical distribution of output values of an EO-TRNG. Based on what we said above, we can and will assume that the EO-TRNG is only subject to thermal noise. We assume we have an EO-TRNG characterized by input parameters α (i.e. the duty cycle of the sampled oscillator), μ (i.e. the drift of the Wiener process described in [3]), σ^2 (i.e. the volatility of the Wiener process described in [3]) that outputs a Markov chain with memory m. In Section 2, we explained how to derive (α, μ, σ^2) from measures of the duty cycle, clock periods, and phase jitters of a couple of ring oscillators.

Remark 2. *Needless to say, in order to obtain a good approximation of the entropy rate of the EO-TRNG, the parameters (α, μ, σ^2) that serve as inputs in our algorithm have to be measured highly precisely. One of the main difficulties with this measurement is the need to distinguish between the contribution of the thermal noise to the phase jitter and contributions from other sources of noise. Indeed, when using the phase jitter measurement methods usually implemented in widely used oscilloscopes, the phase jitter distribution obtained will include not only the contribution of the thermal noise but also that of the flicker noise as well as of all the deterministic noises. Several techniques are described in the literature to evaluate the contribution of thermal noise (see for instance [10, 12]).*

For $n \in \mathbb{N}$, denote by $X_r(n)$ the random variable on S_n such that $\mathbb{P}(X_r(n) = s)$ for $s \in S_n$ is the empirical probability that the EO-TRNG outputs pattern s (we assume here that such an empirical probability exists). In other words, the probability distribution of

$X_r(n)$ is the same as that of the empirical distribution of the output bits of the EO-TRNG grouped in patterns of length n.

In Section 3, we saw that a Markov chain $X = X(X_0, T(X))$ with memory m determines a random variable $\hat{X}_0(m+1)$ on S_{m+1} and that reciprocally, $\hat{X}_0(m+1)$ encodes the initial state X_0 and transition matrix of X. Thus, the Markov chain with memory m, which best approximates the probability distribution of the EO-TRNG, is such that $\hat{X}_0(m+1)$ is the random variable $X_r(m+1)$. So we only need to explain how to compute $X_r(m+1)$ from knowledge of the statistical parameters (α, μ, σ^2) of the EO-TRNG.

Remark 3. *Let $X = X(X_0, T(X))$ be the Markov chain defined by $X_r(m+1)$. In general, the stable element of X will not be X_0 so that the evaluator will need to call on Algorithm 2 to compute X_∞, in order to compute the entropy associated with X.*

Let us denote by $\tilde{p}(x, t|\tilde{p}_0) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ the distribution of probability at time $t > t_0$ representing the attacker's knowledge of the phase of the EO-TRNG under the hypothesis that at time t_0 the attacker's knowledge is given by the distribution $\tilde{p}_0 = \tilde{p}(x, t_0)$. As the sampled signal is 1-periodic, the output of the EO-TRNG at time t depends only on the phase of the EO-TRNG $\mod 1$, so we can consider:

$$\begin{aligned} p : \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \\ (x, t) &\mapsto p(x, t|p_0) = \sum_{n \in \mathbb{Z}} \tilde{p}(x - n, t|\tilde{p}_0). \end{aligned} \tag{12}$$

Let $\mathcal{C}[0,1]$ be the set of 1-periodic real valued functions f and let $\mathcal{D}[0,1] \subset \mathcal{C}[0,1]$ be the set of integrable functions such that $\int_0^1 f(x)dx = 1$. It is clear that $p(x, t|p_0) \in \mathcal{D}[0,1]$. Let us denote by $\tilde{G}(\mu, \sigma)(x)$ the Gaussian distribution with mean μ and standard deviation σ and let

$$G(\mu, \sigma)(x) = \sum_{n \in \mathbb{Z}} \tilde{G}(\mu, \sigma)(x - n) \in \mathcal{D}[0,1].$$

Recall that α is the duty cycle of the sampled clock signal. We mark $f_\alpha^1 = f_\alpha$ and $f_\alpha^0 = 1 - f_\alpha$. With these notations, the following result gives the probability that the EO-TRNG outputs bit b at time t knowing $p(x, t|p_0)$ and shows how knowledge of b affects $p(x, t|p_0)$:

Proposition 1 ([3], A.1. Lemma 2)**.** *Let $b \in \{0, 1\}$, the probability $\mathbb{P}(X(t) = b)$ that the EO-TRNG with parameters (α, μ, σ^2) sampled at time t outputs bit b is:*

$$\mathbb{P}(X(t) = b|p(x, t|p_0)) = \int_0^1 p(x, t|p_0) f_\alpha^b(x) dx. \tag{13}$$

Moreover, knowing that the output of the EO-TRNG at time t is bit b, the distribution of probability $p(x, t|X(t) = b, p_0)$ is:

$$p(x, t|X(t) = b, p_0) = p(x, t|p_0) f_\alpha^b(x) / (\int_0^1 p(x, t|p_0) f_\alpha^b(x) dx). \tag{14}$$

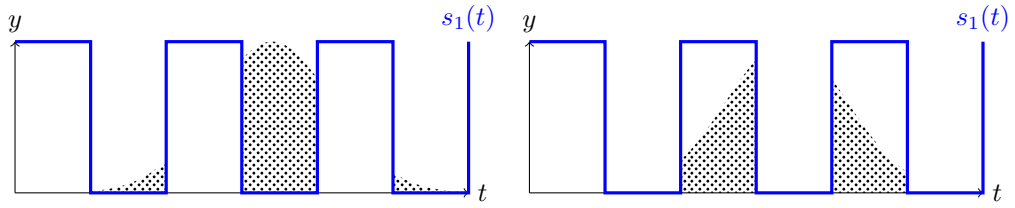

Figure 3: Effect of sampling the clock signal equal to 0 (left) and 1 (right) depending on the distribution $\tilde{p}(x,t)$, $s_1(t)$ is the sampled clock signal.

Figure 3 shows a graphical representation of the effect of sampling the clock signal equal to 0 and 1 following a Gaussian distribution $\tilde{p}(x,t)$ (for clarity, in this figure, we use $\tilde{p}(x,t)$ rather than $p(x,t)$ as it is done in Proposition 1).

Remark 4. *Compared to [3] where the authors compute in the frequency domain, we chose to do our computations in the time domain. The two approaches are equivalent from an algorithmic point of view: we will see that we have to perform a product and a convolution product whether we represent $p(x,t|p_0)$ in the time or in the frequency domain, so that we will have to do a discrete Fourier transform. We think that the approach in the time domain makes it easier to control the loss of precision in the computations: even when using a very naive Riemann integration algorithm, we obtain the lower and upper bounds of the correct result.*

In the following, we define the sampling operator $S(b,\alpha)$ by:

$$\begin{aligned} S(b,\alpha) &: \mathcal{C}[0,1] \to \mathcal{C}[0,1] \\ S(b,\alpha)(p)(x) &= p(x) f_\alpha^b(x). \end{aligned} \tag{15}$$

The evolution of $p(x,t)$ over time is given by the following proposition:

Proposition 2 ([3], A.1. Lemma 1)**.** *Assuming the attacker's knowledge of the phase of the EO-TRNG, modeled by a Wiener process with parameters (α,μ,σ^2), at time t_0 is given by $p(x,t_0)$. Then for $t > t_0$, we have:*

$$p(x,t) = p(x,t_0) * G((t-t_0)\mu, \sqrt{(t-t_0)\sigma^2}), \tag{16}$$

where $*$ is the convolution product.

In the following, we denote by $E(\mu,\sigma,\Delta t) : \mathcal{D}[0,1] \to \mathcal{D}[0,1]$, the evolution operator such that:

$$E(\mu,\sigma,\Delta t)(p)(x) = p(x) * G(\Delta t\mu, \sqrt{\Delta t\sigma^2}).$$

Figure 4 shows a graphical representation of the effect of time evolution on the Gaussian distribution $\tilde{p}(x,t)$.

From the two preceding propositions, we immediately deduce the following:

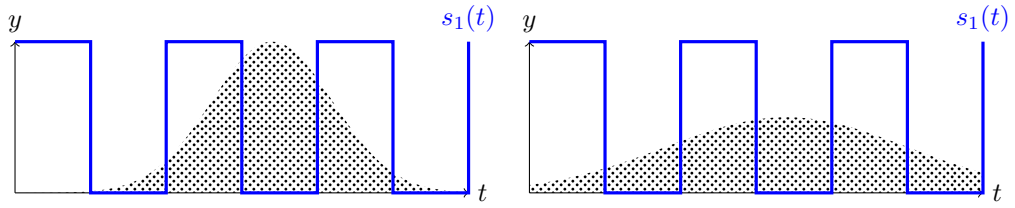

Figure 4: Effect of evolution of time on the distribution $\tilde{p}(x,t)$ for a short accumulation time (left panel) and long accumulation time (right panel), $s_1(t)$ is the sampled signal.

Proposition 3. *Let $b = b_1 \dots b_n \in S_n$ be an n-bit pattern. Let $p_0(x,t_0) \in \mathcal{D}[0,1]$ be the distribution of probability representing the attacker's knowledge about the phase of the EO-TRNG at time t_0. We model the EO-TRNG by a Wiener process with parameters (α,μ,σ^2). Let $t_0 < t_1 < \ldots < t_n$ be the sampling times.*

We denote by $T(\alpha,\mu,\sigma,(t_i),(b_i)) : \mathcal{C}[0,1] \to \mathcal{C}[0,1]$, the operator defined by

$$T(\alpha,\mu,\sigma,(t_i),(b_i)) = S(b_n,\alpha) \circ E(\mu,\sigma,t_n - t_{n-1}) \circ \ldots \circ S(b_1,\alpha) \circ E(\mu,\sigma,t_1 - t_0) \quad (17)$$

Let $\mathbb{P}(X(t_i) = b_i, i = 1,\dots,n | p_0(x,t_0))$ be the probability that the EO-TRNG sampled at times (t_i) outputs the bit pattern (b_i). Then, we have:

$$\mathbb{P}(X(t_i) = b_i, i = 1,\dots,n | p_0(x,t_0)) = \int_0^1 T(\alpha,\mu,\sigma,(t_i),(b_i)) p_0(x,t_0) dx. \quad (18)$$

Using Proposition 3, we obtain Algorithm 4, which can be used to compute the probability that the EO-TRNG outputs a given bit pattern $b_1 \dots b_n$.

Distribution $f \in \mathcal{D}[0,1]$ can be represented either in the time domain or in the frequency domain. In the time domain, f is given by a table of floating point numbers $[f(i/k)]_{i=0,\dots,k}$. In this case, $\int_0^1 f(x)dx$ can be computed via a Riemann integral, which gives upper and lower bounds of the correct result, making it possible to control the precision of the result. The computation of the integral in Step 3 for a fixed precision has a linear complexity in k. Step 4 has a linear complexity in k, too. The convolution product can be computed in the frequency domain in quasi-linear time $O(k\log(k))$ by using a discrete Fourier transform. So the whole Algorithm 4 has a quasi-linear complexity $O(k\log(k))$ in k measured in the number of floating point operations.

We end this section by addressing the choice of the distribution $p_0(x,t_0)$, which depends on the attacker's initial knowledge and memory m of the Markov chain. We discuss it in relation with the security model we use and the target precision of the entropy computation. The security model usually depends on the assumed power of the attacker. In our context, i.e. the attacker has infinite computation power, the difference between the security models is in the attacker's knowledge about the state of the TRNG:

Definition 7. *We define two security models for free-running oscillator based TRNGs according to the attacker's knowledge about the TRNG:*

- *Security model A: we assume the attacker has complete knowledge of the state of the TRNG at start-up and each time the TRNG outputs a bit;*

Algorithm 4: Algorithm to compute the probability of a bit pattern at the output of an EO-TRNG.

input :

- (α, μ, σ^2), the statistical parameters of the EO-TRNG;
- $p_0(x, t_0) \in \mathcal{D}[0, 1]$, representing the initial attacker's knowledge about the phase of the EO-TRNG;
- Δt the time interval between two samplings;
- n integers and $b_1 \dots b_n$ a bit pattern.

output: The joint probability $\mathbb{P}(X_r(i) = b_i, i = 1, \dots, n)$ that the EO-TRNG outputs bits b_i.

$p(x, t_0) = p_0(x, t_0)$;
for $i \leftarrow 1$ **to** n **do**
  $p(x, t) \leftarrow p(x, t) * G(\mu\Delta t, \sqrt{\sigma^2(\Delta t)})$;
  $p(x, t) \leftarrow p(x, t) f_\alpha^{b_i}(x)$;
end
return $\int_0^1 p(x, t)dx$;

- *Security model B: each time the TRNG produces a bit, only the value of this bit is given to the attacker.*

In Security model A, what we mean by the internal state of the TRNG is the phase and duty cycle of each oscillator in the TRNG. Thus, in this security model, each time the TRNG outputs a bit, the attacker is given the internal state of the TRNG from which the value of the output bit can be recovered. Of course, Security model A gives more power to the attacker than Security model B, so it is more stringent. Nonetheless, in real world scenarios, the attacker does not have access to the internal state of the TRNG. Thus, Security model A is not realistic apart from the start-up of the TRNG, when the initial relative phase of the two oscillators in the EO-TRNG could always be the same and may thus be known to the attacker.

Nevertheless, this first model is useful because it greatly simplifies computations. Indeed, each time the TRNG outputs a bit, the attacker gets complete knowledge of its state, so he/she does not need to take the preceding bits into account to predict the next one. Thus, all the useful information is contained in the initial distribution $p_0(x, t_0)$, which is a Dirac distribution $\delta(x_0)$ with all its mass concentrated in x_0 (here t_0 is either the start-up time of the TRNG or each time at which it outputs a bit). In this case, we do not need a Markov chain to model the output stream of the EO-TRNG, so we can assume that the memory depth m is equal to 0. Note that the explicit expressions for entropy computation published in [3, 26] rely on this security assumption.

In Security model B, the attacker only obtains the value of the output bit each time the EO-TRNG outputs a bit. Taking into account continuous operation of the TRNG, this assumption may be viewed as realistic. Following Eq. (11), we should choose a sufficiently

m	1	2	3	4	5	6	7	8	9	10
$\mathcal{E}_D(m)$	0.111	0.306	0.424	0.466	0.476	0.475	0.471	0.468	0.466	0.465
$\mathcal{E}_U(m)$	0.499	0.474	0.467	0.465	0.464	0.464	0.464	0.464	0.464	0.464

Table 1: Values of entropy $\mathcal{E}_D(m)$ based on the Dirac initial distribution and of entropy $\mathcal{E}_U(m)$ based on the uniform initial distribution for memory $m = 1, \ldots, 10$.

large memory depth m of the Markov chain, which determines the distribution of output bits of the EO-TRNG, so that the events that occur before the last m bits do not affect the value of the following bit.

To find such an m, we can consider two very different initial distributions by taking for $p_0(x, t_0)$ either the Dirac distribution or the uniform distribution. Then, we can use Algorithm 3 to compute entropy $\mathcal{E}_D(m)$ (for the Dirac distribution) and $\mathcal{E}_U(m)$ (for the uniform distribution) depending on m. For m big enough, one can check that entropies $\mathcal{E}_D(m)$ and $\mathcal{E}_U(m)$ converge towards the same value. For any m, we can take the smallest integer such that $|\mathcal{E}_D(m) - \mathcal{E}_U(m)| < \epsilon$, ϵ being the desired precision of our computations. Table 1 gives values of $\mathcal{E}_D(m)$ and $\mathcal{E}_U(m)$ for an EO-TRNG with parameters $(\alpha, \mu, \sigma^2) = (0.5, 1, 4.9\,10^{-3})$ and $m = 1, \ldots, 10$. It shows that the convergence is rapid enough to achieve the required precision of 10^{-3} with $m = 10$. We checked experimentally that the convergence is quicker when σ is bigger. To compute Table 1, we chose $\sigma = 0.7$ (which is in the order of typical values that we measure in real hardware implementations) to be sure of making a realistic assessment of the convergence.

Remark 5. *A Dirac distribution is easy to implement in our algorithms since we only use it to compute a convolution product and we can therefore define it just as distribution $\delta(x_0)$ such that $(\delta(x_0) * f)(x) = f(x - x_0)$. As x_0 is unknown, we have to choose the value that minimizes the entropy rate of the generator. There is no obvious solution to this problem, but our experience confirms that the entropy rate of the generator with parameters (α, μ, σ^2) for $\mu \in]0, 1]$ (see the remark just after Definition 1) is minimal when $\mu = 1$. This is also obvious in the entropy formula of [3, Proposition 1]. Moreover, if $\mu = 1$, it is easy to choose the x_0 that minimizes the entropy rate: it is sufficient to take $x_0 = \alpha_0/2$.*

5 Modeling the MO-TRNG including entropy conditioner

In this section, we assume that the MO-TRNG is composed of ℓ EO-TRNGs, denoted EO-TRNG$_i$ for $i = 1, \ldots, \ell$ where EO-TRNG$_i$ is made of the sampling oscillator O_0 and the sampled oscillator O_i. Their outputs are post-processed by an entropy conditioner (see Figure 2). In the previous section, we explained how to compute Markov chains $(X_j^i)_{j \geq 0}$ that approximate the statistical distribution of the output bits of EO-TRNG$_i$. We are now interested in the problem of computing a Markov chain that describes the distribution at the output of the entropy conditioner.

The only general assumption we make is that the entropy conditioner has no memory so that it is defined by a function $\zeta_\ell : \mathbf{bits}^\ell \to \mathbf{bits}$, $(b^1, \ldots, b^\ell) \mapsto \zeta_\ell(b^1, \ldots, b^\ell)$ taking one output bit of each EO-TRNG$_i$ for $i = 1, \ldots, \ell$ to output one bit of the MO-TRNG. If $\mathbf{bits} = \{0, 1\}$, ζ_ℓ is a general Boolean function. For $m \geq 1$, we denote by $\zeta_\ell^m : S_m^\ell \to S_m$, the function obtained by the bit-wise operation of ζ_ℓ on S_m that is ${\zeta_\ell}^m(b_1^1 \ldots b_m^1, \ldots, b_1^\ell \ldots b_m^\ell) = (\zeta_\ell(b_1^1, \ldots, b_1^\ell) \ldots (\zeta_\ell(b_m^1, \ldots, b_m^\ell))$.

By combining using ζ_ℓ the outputs of ℓ Markov chains with memory m, $(X_j^i)_{j\geq 0}$, we obtain the sequence of random variables $(X^{\zeta_\ell})_{j\geq 0} = (\zeta_\ell^m(X_j^1, \ldots, X_j^\ell))_{j\geq 0}$. The following lemma confirms that this sequence is a Markov chain, too.

Lemma 3. *Let $(X_j^i)_{j\geq 0}$ for $i = 1, \ldots, \ell$ be Markov chains over S_m. The sequence $(X^{\zeta_\ell})_{j\geq 0} = (\zeta_\ell^m(X_j^1, \ldots, X_j^\ell))_{j\geq 0}$ of random variables over S_m is a Markov chain.*

Moreover, for all $k \geq m$, we have:

$$\hat{X}^{\zeta_\ell}(k) = \zeta_\ell^m(\hat{X}^1(k), \ldots, \hat{X}^\ell(k)) \tag{19}$$

Proof. To prove the first claim, we have to check the conditions of Definition 2. We can consider $(X_j^i)_{j\geq 0}^{i=1,\ldots,\ell}$ as a sequence of random variables over S_m^ℓ. Being the joint probability distribution of $(X_j^i)_{j\geq 0}$ for $i = 1, \ldots, \ell$, it is clear that $(X_j^i)_{j\geq 0}^{i=1,\ldots,\ell}$ is a Markov chain. We can compute:

$$\mathbb{P}(X_n^{\zeta_\ell} = x_n | X_{n-1}^{\zeta_\ell} = x_{n-1}, \ldots X_{n-k}^{\zeta_\ell} = x_{n-k}) \tag{20}$$

as the sum:

$$\sum \mathbb{P}(((X_n^i) = (s_n^i))^{i=1,\ldots,\ell} | ((X_{n-1}^i) = (s_{n-1}^i))^{i=1,\ldots,\ell}, \ldots, ((X_{n-k}^i) = (s_{n-k}^i))^{i=1,\ldots,\ell}), \tag{21}$$

which can be taken over all the $(s_\nu^i)_{\nu=n-k,\ldots,n}^{i=1,\ldots,\ell} \in (S_m^\ell)^{k+1}$ such that for $\nu = n-k, \ldots, n$, $\zeta_\ell^m(s_\nu^1, \ldots, s_\nu^\ell) = x_\nu$. Using the fact that $(X_j^i)_{j\geq 0}$ is a Markov chain, we observe that Eq. (21) is equal to:

$$\sum \mathbb{P}(((X_n^i) = (s_n^i))^{i=1,\ldots,\ell} | ((X_{n-1}^i) = (s_{n-1}^i))^{i=1,\ldots,\ell}), \tag{22}$$

the sum being taken over all the $(s_\nu^i)_{\nu=n-1,n}^{i=1,\ldots,\ell} \in (S_m^\ell)^2$ such that $\zeta_\ell^m(s_\nu^1, \ldots, s_\nu^\ell) = x_\nu$. But expression (22) is non other than

$$\mathbb{P}(X_n^{\zeta_\ell} = x_n | X_{n-1}^{\zeta_\ell} = x_{n-1}). \tag{23}$$

So that we have proved that

$$\mathbb{P}(X_n^{\zeta_\ell} = x_n | X_{n-1}^{\zeta_\ell} = x_{n-1}, \ldots X_{n-k}^{\zeta_\ell} = x_{n-k}) = \mathbb{P}(X_n^{\zeta_\ell} = x_n | X_{n-1}^{\zeta_\ell} = x_{n-1}). \tag{24}$$

Because $(X_j^i)_{j\geq 0}$ is a Markov chain, we know that $\mathbb{P}(((X_n^i) = (s_n^i))^{i=1,\ldots,\ell} | ((X_{n-1}^i) = (s_{n-1}^i))^{i=1,\ldots,\ell})$ does not depend on n. Thus, as expressions (22) and (23) are identical,

we confirm that $\mathbb{P}(X_n^{\zeta_\ell} = x_n | X_{n-1}^{\zeta_\ell} = x_{n-1})$ does not depend on n. Moreover it is clear that if $x_n \notin \mathrm{succ}(x_{n-1})$ then $\mathbb{P}(X_n^{\zeta_\ell} = x_n | X_{n-1}^{\zeta_\ell} = x_{n-1}) = 0$.

Next, we prove the second claim of Lemma 3. It is clear that Eq. (19) is true for $k = m$, because using condition (9) we obtain:

$$\begin{aligned} \hat{X}^{\zeta_\ell}(m) &= \lim_{j\to\infty} \zeta_\ell^m(X_j^1, \ldots, X_j^\ell) \\ &= \zeta_\ell^m(\lim_{j\to\infty} X_j^1, \ldots, \lim_{j\to\infty} X_j^\ell) = \zeta_\ell^m(\hat{X}^1(m), \ldots, \hat{X}^\ell(m)) \end{aligned} \tag{25}$$

Moreover, by definition, we have for $k > m$, $b_1 \ldots b_k \in S_k$

$$\begin{aligned} &\mathbb{P}(X_n^{\zeta_\ell} = b_{k-m+1} \ldots b_k | X_{n-1}^{\zeta_\ell} = b_{k-m} \ldots b_{k-1}) \\ &\quad = \mathbb{P}(\zeta_\ell^m(X_n^1, \ldots, X_n^\ell) = b_{k-m+1} \ldots b_k | \zeta_\ell^m(X_{n-1}^1, \ldots, X_{n-1}^\ell) = b_{k-m} \ldots b_{k-1}) \end{aligned} \tag{26}$$

so by an inductive application of (5), we obtain Eq. (19). □

Remark 6. *From the point of view of the two security models, each of the $(X_j^i)_{j\geq 0}$ determines the probability of guessing the next bit at the output of EO-TRNG$_i$ from knowledge of the m preceding bits. At first sight, one could think that $(\zeta_\ell^m(X_j^1, \ldots, X_j^\ell))_{j\geq 0}$ models the security threat when the attacker knows each of the output bits of the EO-TRNG$_i$ for $i = 1, \ldots, \ell$ and tries to guess the next output bit of the entropy conditioner. However, this not the case: $(\zeta_\ell^m(X_j^1, \ldots, X_j^\ell))_{j\geq 0}$ models the probability of guessing the next bit at the output of ζ_ℓ from knowledge of the m preceding bits at the output of ζ_ℓ, which is precisely our security model of the MO-TRNG.*

The preceding Lemma says that $(\zeta_\ell^m(X_j^1, \ldots X_j^\ell))_{i\geq 0}$ is a Markov chain with memory m. The second part of the Lemma makes it possible to compute the encoding function (see Definition 4) of this Markov chain.

Proposition 4. *Let $(X^i)_{j\geq 0} =$ for $i = 1, \ldots, \ell$ be Markov chains with memory m over alphabet* **bits**. *We denote by f_{X^i} their respective encoding function (see Definition 4).*

Let $f_{X^{\zeta_\ell}}$ be the encoding function for $\zeta_\ell(X_j^1, \ldots, X_j^\ell)$. For all $s \in S_{m+1}$, we have:

$$f_{X^{\zeta_\ell}}(s) = \sum_{s'=(s^1,\ldots,s^\ell)\in S_{m+1}^\ell, \zeta_\ell^{m+1}(s')=s} \prod_{i=1}^{\ell} f_{X^i}(s^i). \tag{27}$$

Proof. This is an immediate consequence of the fact that for all $s \in S_{m+1}$, $f_{X^{\zeta_\ell}}(s) = \mathbb{P}(\hat{X}^{\zeta_\ell}(m+1) = s)$ by definition and the preceding Lemma. □

We can use the expression from Proposition 4 to recover $f_{X^{\zeta_\ell}}$ from the knowledge of f_{X^i}. However, the algorithm obtained in this way is not practical, because the sums in Eq. (27) are calculated over the big sets S_{m+1}^ℓ and S_m^ℓ so that the resulting run time complexity is in $O(|S_{m+1}|^\ell)$.

Fortunately, two optimizations are possible in most real world cases. Recall that $\mathbf{bits} = \{0, 1\}$. In this case, $\zeta_\ell : \mathbf{bits}^\ell \to \mathbf{bits}$ is a Boolean function that can be represented as an algebraic normal form. The algebraic normal form of ζ_ℓ is just a polynomial

$P \in \mathbb{F}_2[x_1, \ldots, x_\ell]$, the evaluation of which enables recovery of ζ_ℓ. It can be shown that the degree of P is bounded by ℓ. There are $\binom{2\ell}{\ell}$ monomials of degree ℓ in $\mathbb{F}_2[x_1, \ldots, x_\ell]$. So P can be evaluated by performing $\binom{2\ell}{\ell}(\ell+1)$ operations each of which is of the form $\zeta_2 : \mathbf{bits}^2 \to \mathbf{bits}$. Algorithm 5 deduced from Proposition 4, makes it possible to compute $\zeta_2^m(X_j^1, X_j^2)$ for an arbitrary $\zeta_2 : \mathbf{bits}^2 \to \mathbf{bits}$ in time $O(|S_{m+1}|^2)$. We deduce that $\zeta_\ell^m(X_j^1, \ldots, X_j^\ell)$, where ζ_ℓ^m is a general Boolean function, can be evaluated in $O(\binom{2\ell}{\ell}(\ell+1)|S_{m+1}|^2)$.

We can further improve the computation run time of $\zeta_\ell^m(X_j^1, \ldots, X_j^\ell)$, if we assume that $\zeta_\ell : \mathbf{bits}^\ell \to \mathbf{bits}$ is given as the composition of ℓ times the same associative composition law $\zeta_\ell^0 : \mathbf{bits}^2 \to \mathbf{bits}$ to get a running time of $O(\ell|S_{m+1}|^2)$. If, in addition, the Markov chains $X_j^1, \ldots, X_j^\ell$ represent ℓ copies of the same Markov chain we can use a square and multiply algorithm to compute $\zeta_\ell^m(X_j^1, \ldots, X_j^\ell)$ in time $O(\log(\ell)|S_{m+1}|^2)$. This last condition is fulfilled in the classical design where the MO-TRNG is obtained as the exclusive or function of outputs of ℓ EO-TRNGs that have the same specification.

Algorithm 5: Algorithm for computing $\zeta_2^m(X_j^1, X_j^2)$.

input :

- f_{X^i} $i = 1, 2$ the encoding functions of two Markov chains $(X_j^i)_{j\geq 0}$ with memory m associated to EO-TRNG$_i$;
- $\zeta_2^m : S_m \times S_m \to S_m$ obtained as the bitwise operation of $\zeta_2 : \mathbf{bits} \times \mathbf{bits} \to \mathbf{bits}$.

output: The encoding function $f_{X^{\zeta_2}}$ of the Markov chain $\zeta_2^m(X_1, X_2)$.

```
1 ∀i ∈ S_{m+1}, X_3[i] ← 0;
2 for x, y ∈ S_{m+1} × S_{m+1} do
3 |   i ← ζ_2^{m+1}(x, y);
4 |   X_3[i] ← X_3[i] + f_{X̂1}[x].f_{X̂2}[y] ;
5 end
6 return f_{X^{ζ2}} = X_3; /* Comment: We represent f_{X^{ζ2}} as a table of
                floats indexed by S_{m+1}                                  */
```

The second optimization is possible if for all $k \geq m$, S_k together with the composition law $\zeta_2^k : S_k^2 \to S_k$ is a group. In this case, we use the convenient notation $+^{\zeta_\ell} : S_k^2 \to S_k$ for $\zeta_2^k : S_k^2 \to S_k$ and denote by $-^{\zeta_2}$ the inverse operation. For $\ell = 2$, we can rewrite Eq. (27) as:

$$f_{X^{\zeta_2}}(s) = \sum_{s' \in S_{m+1}} f_{X^1}(s -^{\zeta_2} s') f_{X^2}(s'), \tag{28}$$

Note that Eq. (28) is a convolution product of functions $S_k \to \mathbb{R}$. Recall that if $(G, +^G)$ is a finite commutative group, $f, g : G \to \mathbb{R}$ are functions, $(f * g)(x) = \sum_{t\in G} f(t)g(x -^G t)$ is the convolution product of f and g that can easily be computed using a Fourier transform.

Namely, let $\hat{G}$ be the dual group of G that is the group of characters $\chi : G \to \mathbb{C}^*$. If

$f : G \to \mathbb{R}$ is a function, we denote by $\hat{f} : \hat{G} \to \mathbb{R}$ the function such that:

$$\hat{f}(\chi) = \frac{1}{|G|} \sum_{x \in G} f(x)\chi(-x). \tag{29}$$

Using the fact that $\sum_{\chi \in \hat{G}} \chi(t)$ is 0 if $t = 0$ or $|G|$ if $t \neq 0$, we deduce that for all $x \in G$ $f(x) = \sum_{\chi \in \hat{G}} \hat{f}(\chi)\chi(x)$. This shows that $\hat{f}$ makes it possible to recover f. In our context, $\hat{f}$ is the Fourier transform of f. Using the fact that $\hat{\hat{G}}$ is canonically isomorphic to G, one can show that the map $f \mapsto \hat{f}$ is an involution i.e. $\hat{\hat{f}} = f$. Using the fact that $\sum_{t \in G} \chi(t)$ is 0 if $\chi \neq 0$ and $|G|$ if $\chi = 0$, we easily get the well known proposition:

Proposition 5. *Let $f, g : G \to \mathbb{R}$ be functions, we have for all $\chi \in \hat{G}$:*

$$\widehat{f * g}(\chi) = \hat{f}(\chi)\hat{g}(\chi). \tag{30}$$

This proposition gives us a way to rapidly compute convolution product of $f, g : G \to \mathbb{R}$, provided that one can easily compute the Fourier transform in G.

Example 1. *In the case that $\zeta_2 : \mathbf{bits}^2 \to \mathbf{bits}$ is the xor map $(b_1, b_2) \mapsto b_1 \oplus b_2$ then (S_m, ζ_2^m) is a group. The associated Fourier transform is called the Walsh transform and a quasi-optimal algorithm is known to compute it [8]. In this case we can use Algorithm 6 to compute $\zeta_2^m(X_j^1, X_j^1)$ in time of order $O(|S_{m+1}|)$.*

By combining the two optimizations, if the MO-TRNG is obtained by XOR-ing the outputs of ℓ EO-TRNGs with the same specification, one obtains an algorithm to compute the entropy rate of the TRNG with the running time complexity $O(\log(\ell)|S_{m+1}|)$.

6 Validation of the model by simulations and its comparison with state-of-the-art MO-TRNG models

Using the proposed algorithms, we simulated several MO-TRNG configurations featuring different numbers of rings in order to compare our model with other state-of-the-art MO-TRNG models and to assess the impact of design choices on the entropy rate. All the simulated MO-TRNGs had the same architecture as that presented in Fig. 2 where, for the sake of simplicity, the oscillators $O_1, \ldots, O_\ell$ producing the sampled signals had the same number of inverters. The entropy conditioner was implemented as an exclusive or (XOR) function.

The set of algorithms presented in this paper was implemented in Python available under GPL License. The whole set is freely accessible via GitHub [16].

We denote by EO-TRNG$_i$ the EO-TRNG made of the oscillator O$_i$ sampled at rising edges of the output signal of O$_0$. For our computations, we assumed that EO-TRNG$_i$ for $i = 1, \ldots, \ell$ has the following fixed parameters (see Definition 1): $\alpha_i = 0.5$ for the duty cycle and $\mu_i = 1$ for the drift of the Wiener process. We computed the entropy rate of the resulting TRNG with the quality factor varying in the interval $[0.001; 0.1]$. Recall that the quality factor (see [3, Section 2.4]) is given by $\sigma^2 \Delta t$ where σ^2 is the volatility of the Wiener process and Δt is the time interval between two subsequent samplings.

Algorithm 6: Algorithm for computing $\zeta_2^m(X_j^1, X_j^2)$ using Fourier transform.

input :

- f_{X^i} $i = 1, 2$ the encoding functions of two Markov chains $(X_j^i)_{j\geq 0}$ with memory m associated with EO-TRNG$_i$;
- $\zeta_2^m : S_{m+1} \times S_{m+1} \to S_{m+1}$ a composition law such that:
  - S_{m+1} together with the composition law $\zeta_2^m : S_{m+1} \times S_{m+1} \to S_{m+1}$ is a group;
  - an efficient algorithm exists to compute the Fourier transform of $f_{X^i} : S_{m+1} \to \mathbb{R}$ for $i = 1, 2$.

output: The encoding function $f_{X^{\zeta_2}}$ of the Markov chain $\zeta_2^m(X_1, X_2)$.

$\widehat{f_{X^i}} \leftarrow FT(f_{X^i})$ for $i = 1, 2$;
for $\chi \in \hat{S}_{m+1}$ **do**
  $\widehat{f_{X^{\zeta_2}}}(\chi) \leftarrow \widehat{f_{X^1}}(\chi).\widehat{f_{X^2}}(\chi)$;
end
return $f_{X^{\zeta_2}} = FT^{-1}(\widehat{f_{X^{\zeta_2}}})$; `/* Comment: We represent` $f_{X^{\zeta_2}}$ `as a table of floats indexed by` S_{m+1} `*/`

First, we compared our model with the EO-TRNG model of Baudet *et al.* provided in [3]. In the proof of Corollary 1 presented in [3], one can find the following closed approximate expression for the entropy rate at the output of an EO-TRNG depending on its quality factor:

$$\mathcal{E}(Q) = 1 - \frac{16}{\pi^2 \ln(2)} e^{-4\pi^2 Q} + O(e^{-6\pi^2 Q}). \tag{31}$$

The results obtained using this expression can be compared with the results given by our model using Fig. 5. Note that Eq. (31) is not accurate for small quality factors, since the error term becomes too big (note in particular the negative entropy rate for the quality factor sizes up to 0.02). When the quality factor increases further, the two models converge toward the same value of entropy and always remain within the margin of error. Note that the model of Baudet *et al.* proposed in [3] always underestimates entropy compared to the new model, the latter being more precise. We also checked that the results of our model regarding the computed bias were always within the error margin appearing at the end of Eq. (31).

One could expect that, as for a cryptographic application, we would need to produce bits with high entropy rate and consequently, we would use a TRNG with a big quality factor where the discrepancy between the two models is small. Table 2 shows that this assumption is not correct: in the case of an MO-TRNG obtained by XOR-ing the outputs of $\ell = 64$ EO-TRNGs, which can be a realistic number of rings in practice, to achieve the MO-TRNG entropy rate of 0.997 required by recommendations AIS 31 [14], the required quality factor of individual EO-TRNGs would be 0.011. Note that this value is in the

ℓ	2	4	8	16	32	64
Quality factor	0.082	0.047	0.029	0.020	0.015	0.011

Table 2: Quality factor required to achieve an entropy rate of 0.997 at the output of the MO-TRNG obtained by XOR-ing outputs of ℓ EO-TRNGs.

area where results of the two models diverge significantly.

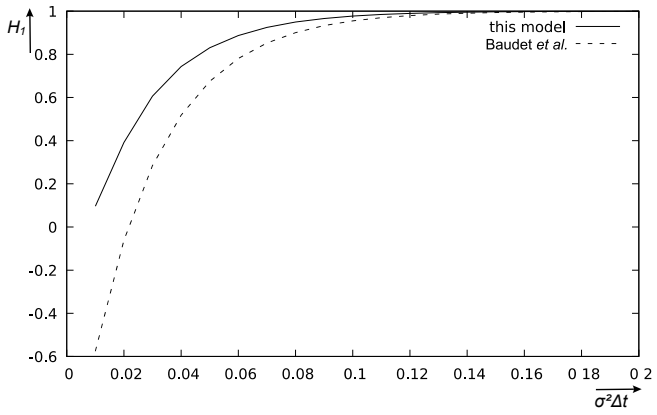

Figure 5: Entropy rate at the output of an EO-TRNG estimated using the model of Baudet *et al.* from [3] and our model as a function of the quality factor (the vertical axis represents Shannon entropy and the horizontal axis represents quality factor).

The previous results concerning the EO-TRNG confirm that our approach based on Markov chains is more precise than that presented in [3]. The MO-TRNG model presented in [26] extends the model from [3] to the use of multiple oscillators as sources of randomness. Consequently, both models have the same characteristics regarding the impact of the quality factor on the entropy.

In Section 5, we showed that our method makes it possible to evaluate a lower entropy bound following Security model A from Definition 7, on which the above cited models [3, 26] (implicitly) rely, but, the new method also makes it possible to compute entropy with the more realistic Security model B.

Next, we assessed the impact of the choice of the security model on the proved lower bound of the entropy rate at the output of the MO-TRNG for $\ell = 1, 2$ and 4. The results are presented in Fig. 6. On the left panel in Fig. 6, we present the entropy rate computed with Security model A, and on the right one, the rate was computed with Security model B.

We can see that the entropy rate of the latter is significantly bigger than that of the former. Also, as expected, in both cases, the entropy conditioner, which combines the

outputs of several EO-TRNGs by an exclusive or function, significantly increases the entropy rate.

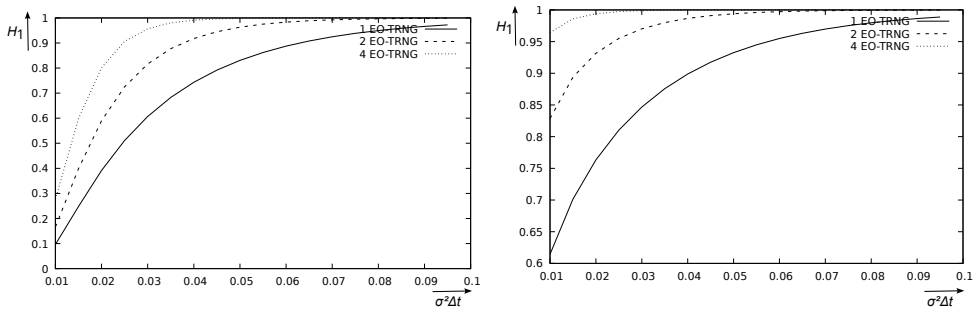

Figure 6: Shannon entropy rate as a function of quality factor in an MO-TRNG composed of 1, 2 and 4 EO-TRNGs computed according to Security model A (left) and Security model B (right).

7 Modeling and evaluation of the TRNG implemented in hardware and discussion

Next, we evaluated the relevance of our entropy estimation and entropy management methods for MO-TRNGs based on ring oscillators, which were implemented in hardware. We used a modular hardware dedicated to TRNG and PUF testing, which featured the Intel 5CEBA4F17C8N device from the Cyclone V FPGA family [15]. We implemented three versions of the MO-TRNG following the circuit diagram depicted in Fig. 2 where:

- the ring oscillator O_0 was always composed of 29 delay elements (one NAND gate and 28 buffers) and it generated the sampling (reference) clock signal of 69 MHz;
- in the three versions, the oscillators O_i were made of respectively 39, 49, and 68 delay elements and produced sampled clock signals with mean frequencies of, respectively, 51, 40, and 28.9 MHz. Note that to evaluate the impact of the ring frequencies on the entropy, we set the number of delay elements to get approximately the same difference in mean frequencies between the three versions of the rings, i.e. about 11 MHz.

Remark 7. *We recall that all ring oscillators are considered to be independent and unmanipulable sources of jittered clock signals in our MO-TRNG stochastic model. The independence of phases of generated clock signals can be ensured by a proper hardware design (e.g. by placing the rings far apart, avoiding parallel routing, placing a buffer at the ring output). The required independence can be easily verified as it was done in [18, Page 4] using a suitable oscilloscope (e.g. LeCroy WaveRunner 640Zi). Indeed, the authors show that the cumulative distribution function of the phase shift between two independent*

	Duty cycle	$1/2\mathbb{E}(b_i \neq b_{i+1})$	$T_0/T_x \mod 1$	$\sigma^2_{T_0}$	Entropy rate
EO-TRNG$_{39}$	0.522	0.285	0.260	$7.5 \cdot 10^{-6}$	0.637
EO-TRNG$_{49}$	0.503	0.424	0.420	$4 \cdot 10^{-6}$	0.561
EO-TRNG$_{68}$	0.508	0.407	0.418	$2.5 \cdot 10^{-6}$	0.468

Table 3: Parameters of EO-TRNG$_\nu$ for a number of delay elements $\nu = 39, 49$, and 68, computed using the algorithms presented in [10] together with the Shannon entropy rate according to model B for accumulation time $D = 2000$.

signals must be uniformly distributed, which corresponds to the standard deviation of the phase shift of 104 degrees. This standard deviation can be easily measured and verified by the chosen oscilloscope [18, Figure 24].

It is also well known that ring oscillators are vulnerable to frequency attacks [17, 4] during which the rings can lock to the frequency of the external signal. Fortunately, this kind of attacks can be easily detected by measuring online the jitter variance, as it was proposed in [10]. This possibility is important to ensure that the Security model B is valid. The same measures, which can constitute a basis for dedicated statistical tests, would be able to detect entropy variations caused by aging, process variation, self-heating up, etc.

To obtain comparable results, we fixed the placement and routing of all oscillators (i.e. of that generating the reference clock, but also of those generating sampled clocks) for all MO-TRNG versions. We placed the rings far apart to avoid their mutual dependence and confirmed their independent behavior by oscilloscope. We denote by EO-TRNG$_\nu$ the EO-TRNG made of ν delay elements (inverters and buffers) and by MO-TRNG$_{(\ell,\nu)}$ the MO-TRNG made of ℓ EO-TRNG$_\nu$.

First, we set the division factor D to 1 and generated bit streams $(b_i)_{i=1,\dots,\kappa}$ of $\kappa = 10^6$ EO-TRNG$_\nu$ output bits. From the bit streams obtained, using algorithms published in [10], we computed:

- the duty cycle of the sampled oscillator as $\mathbb{E}(b_i) = (\sum b_i)/\kappa$;
- the ratio $T_0/T_x \mod 1$ (more precisely $\min(T_0/T_x \mod 1, 1 - T_0/T_x \mod 1)$), where we denote by T_0 (resp. T_x) the mean period of the sampling (resp. the sampled) oscillator in the EO-TRNG$_x$ as $1/2\mathbb{E}(b_i \neq b_{i+1})$ according to [10, Fact 2];
- the volatility $\sigma^2_{T_0}$ of the associated Wiener process σ^2_0 according to Eq. (2) using [10, Fact 1].

The results are presented in Table 3. Recall that the method in [10] outputs $\sigma^2_{T_0}$, which is the variance of the jitter accumulated during T_0 when T_1 has been scaled to 1. It means that if D is the value of the divider from Fig. 1 then the quality factor of the EO-TRNG is equal to $D\sigma^2_{T_0}$.

Note that in Table 3, the value of $\sigma^2_{T_0}$ increases with the clock frequency (i.e. in our case, it decreases with number of delay elements included in the ring). But in the previous paragraph, we have seen that the quality factor of the EO-TRNG depends

linearly on $\sigma^2_{T_0}$. This means that an EO-TRNG is more efficient (from the point of view of the entropy rate at its output), if the sampled signal has a higher frequency. This quite natural conclusion is confirmed by our model, as can be seen in the last column of Table 3, which gives the entropy rate of the EO-TRNG$_\nu$ for $\nu = 39, 49$, and 68 according to Security model B and the frequency divisor D set to 2000.

We underline the fact that the sampling oscillator had the same frequency in all implemented EO-TRNG$_\nu$ versions and consequently, the output bit rate remained the same, i.e. it was independent of the number of delay elements of the sampled rings. At first glance, it may consequently seem that in order to increase the entropy rate, it is preferable to select the highest frequency of the sampled clock signals possible.

However, other constrains also have to be considered when taking decisions regarding the frequency of the sampled clocks:

- since the slopes of the rising edges and falling edges of the clock signal are usually very different and do not depend on the clock frequency, with increasing frequencies, the duty cycle moves further from its ideal value (0.5) – this increases the bias of the output bit values and decreases the entropy rate;
- rings can interlock more easily at high frequencies [18, Figures 18 and 19] and decrease the output entropy rate, it is therefore preferable (especially in multiple ring oscillator based TRNGs) to reduce the frequency of generated clocks and to spread their frequencies, while also paying attention to their harmonics.

The aim of entropy management in MO-TRNG design is to achieve a prescribed entropy rate, e.g. 0.997 according to the current version of recommendations AIS 31 (version 2.0) [14] or even 0.9998, as required by the new draft of AIS 31 (version 2.35) [19], by choosing the number of EO-TRNGs and the value of the frequency divider. Of course there is a trade-off between the entropy rate, the bitrate and number of gates in the TRNG, i.e. its cost. In practice, two entropy management strategies can be applied:

- the number and the size of oscillators (i.e. the cost) is fixed and the designer needs to determine the accumulation time (i.e. the bit rate) required to reach the required entropy rate,
- the required bit rate is fixed and the designer needs to determine the size (the frequency) and the number of oscillators, required to reach the targeted entropy rate.

Applying the first strategy, we computed the value of D to achieve a Shannon entropy rate of 0.997 according to Security models A and B for the MO-TRNG$_{(2,\nu)}$ with $\nu = 39, 49$, and 68. The results are given in Table 4. Applying the second strategy, we computed the number of oscillators ℓ, so that the Shannon entropy rate at the output of the MO-TRNG$_{(\ell,\nu)}$ for $\nu = 39, 49$, and 68 was 0.997, while the divider value D was fixed to 2000. The results are shown in Table 5.

The results presented in Tables 4 and 5, allows us to conclude that while both security models (A and B) make it possible to compute a lower bound of the entropy rate at the MO-TRNG output, being overly pessimistic, Security model A has a huge impact in terms of cost and performance. Specifically, we can see that to achieve the same proven entropy rate:

	Value of D for model A	Value of D for model B
MO-TRNG$_{(2,39)}$	10 483	7 898
MO-TRNG$_{(2,49)}$	20 560	14 809
MO-TRNG$_{(2,68)}$	32 896	23 694

Table 4: Setting the division factor D to achieve an entropy rate of 0.997 for MO-TRNG$_{(2,\nu)}$ and $\nu = 39, 49$, and 68, according to Security models A and B

	Value of ℓ for model A	Value of ℓ for model B
MO-TRNG$_{(\ell,39)}$	33	6
MO-TRNG$_{(\ell,49)}$	273	9
MO-TRNG$_{(\ell,68)}$	3562	12

Table 5: Setting the number of oscillators ℓ to achieve entropy rate of 0.997 for MO-TRNG$_{(\ell,\nu)}$ with $\nu = 39, 49, 68$ and $D = 2000$, according to Security models A and B.

- while using the same area, the output bit rate computed using Security model A is only about 70% of what we can achieve using Security model B (see Table 4);
- to reach the same output bit rate with model A as with model B, we have to multiply the number of rings and consequently the generator cost by up to 300 (see Table 5).

In the last phase of our experiments, we compared the entropy estimations based on Security model B and the two entropy management strategies (fixed cost versus fixed bit rate) presented in Tables 4 and 5 with results of five statistical tests following Test procedure B of AIS 31, version 2.0 [14]: test T6 a) and b), test T7 a) and b) and T8. Note that for high quality results, the test T8 estimates Shannon entropy rate per output byte.

For each MO-TRNG configuration tested, we generated one million bytes of random data: by varying division factor D from 2000 to 25000 for the first entropy management strategy and by modifying the number of rings ℓ from 1 to 13, when the second strategy was evaluated. The results of tests are presented in Tables 6 to 8 for the first entropy management strategy and in Table 9 for the second one.

In addition to showing the success of the tests and entropy estimated by test T8, we also specify, for how many of the five tests of the AIS 31 Test procedure B the generated data failed. Here, it is interesting to note that the data always passed test T6 a).

The dark gray cells in Tables 6 to 8 correspond to values of division factor D from the right column of Table 4, while the cells highlighted in Table 9 correspond to the number of oscillators ℓ from the right column of Table 5. As could be expected, our entropy estimations are always more stringent than those given by test T8 in both entropy management strategies, mainly because the statistical tests can not distinguish the contribution of unpredictable and unmanipulable thermal noises from the contribution of auto-correlated low frequency noises such as flicker noise and the contribution of (ma-

	Results of AIS31, procedure B/Shannon entropy from T8			
D	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
2000	Failed 4x/-	Failed 3x/0.576	Failed 3x/0.994	Failed 2x/0.999
3000	Failed 4x/0.540	Failed 3x/0.969	Passed/0.999	Passed/1.000
4000	Failed 3x/0.977	Failed 2x/0.998	Passed/1.000	Passed/1.000
5000	Failed 3x/0.992	Failed 1x/0.999	Passed/1.000	Passed/1.000
6000	Failed 1x/0.998	Passed/1.000	Passed/1.000	Passed/1.000
7000	Failed 1x/1.000	Passed/1.000	Passed/1.000	Passed/1.000
8000	Passed/1.000	Passed/1.000	Passed/1.000	Passed/1.000
10000	Passed/1.000	Passed/1.000	Passed/1.000	Passed/1.000

Table 6: Comparison of our 0.997 Shannon entropy estimation (in gray) with the results of statistical tests AIS31, Test procedure B and its Shannon entropy estimation according to test T8 for different accumulation times D and number of rings ℓ with mean output frequency of 51.0 MHz (39 elements per ring).

	Results of AIS31, procedure B/Shannon entropy from T8			
D	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
2000	Failed 4x/-	Failed 4x/-	Failed 4x/0.991	Failed 2x/0.978
3000	Failed 4x/0.418	Failed 3x/0.455	Failed 1x/0.997	Passed/0.999
4000	Failed 4x/0.334	Failed 3x/0.855	Failed 2x/0.999	Passed/1.000
5000	Failed 3x/0.958	Failed 1x/0.922	Failed 1x/0.999	Passed/1.000
6000	Failed 3x/0.983	Failed 2x/0.998	Passed/1.000	Passed/1.000
7000	Failed 3x/0.984	Failed 1x/0.999	Passed/1.000	Passed/1.000
8000	Failed 3x/0.997	Failed 1x/0.999	Passed/1.000	Passed/1.000
10000	Failed 1x/0.999	Passed/0.999	Passed/1.000	Passed/1.000
15000	Passed/1.000	Passed/1.000	Passed/1.000	Passed/1.000
20000	Passed/1.000	Passed/1.000	Passed/1.000	Passed/1.000

Table 7: Comparison of our 0.997 Shannon entropy estimation (in gray) with the results of statistical tests AIS31, Test procedure B and its Shannon entropy estimation according to test T8 for different accumulation times D and number of rings ℓ with mean output frequency of 40.0 MHz (49 elements per ring).

nipulable) global noises. This fact further confirms the validity and usefulness of our approach.

8 Conclusion and perspectives

In this paper, we proposed a complete set of algorithms aimed to compute the entropy rate at the output of oscillator based TRNGs using differences in phases of generated clocks as a source of randomness. The algorithms were implemented in Python and

	Results of AIS31, procedure B/Shannon entropy from T8			
D	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
2000	Failed 4x/-	Failed 4x/0.809	Failed 4x/0.991	Failed 4x/0.996
3000	Failed 4x/-	Failed 4x/0.566	Failed 4x/0.928	Failed 4x/0.982
4000	Failed 4x/-	Failed 4x/0.752	Failed 3x/0.733	Failed 1x/0.999
5000	Failed 4x/0.610	Failed 3x/0.521	Failed 3x/0.985	Failed 1x/0.999
6000	Failed 4x/0.805	Failed 4x/0.810	Failed 2x/0.999	Passed/1.000
7000	Failed 3x/0.711	Failed 3x/0.667	Passed/1.000	Passed/1.000
8000	Failed 2x/0.978	Failed 2x/0.994	Passed/1.000	Passed/1.000
10000	Failed 2x/0.992	Failed 3x/0.976	Passed/1.000	Passed/1.000
15000	Failed 1x/0.999	Passed/0.999	Passed/1.000	Passed/1.000
20000	Passed/1.000	Passed/1.000	Passed/1.000	Passed/1.000
25000	Passed/1.000	Passed/1.000	Passed/1.000	Passed/1.000

Table 8: Comparison of our 0.997 Shannon entropy estimation (in gray) with the results of statistical tests AIS31, Test procedure B and its Shannon entropy estimation according to test T8 for different accumulation times D and number of rings ℓ with mean output frequency of 28.9 MHz (68 elements per ring).

	Results of AIS 31, procedure B/Shannon entropy from T8		
ℓ	$\nu = 39$	$\nu = 49$	$\nu = 68$
1	Failed 4x/-	Failed 4x/-	Failed 4x/-
2	Failed 4x/0.746	Failed 4x/0.720	Failed 4x/0.467
3	Failed 3x/0.923	Failed 4x/0.973	Failed 3x/0.865
4	Passed/0.999	Failed 1x/0.998	Failed 4x/0.921
5	Passed/1.000	Passed/1.000	Failed 1x/0.999
6	Passed/1.000	Passed/1.000	Passed/1.000
7	Passed/1.000	Passed/1.000	Passed/1.000
8	Passed/1.000	Passed/1.000	Passed/1.000
9	Passed/1.000	Passed/1.000	Passed/1.000
10	Passed/1.000	Passed/1.000	Passed/1.000
11	Passed/1.000	Passed/1.000	Passed/1.000
12	Passed/1.000	Passed/1.000	Passed/1.000
13	Passed/1.000	Passed/1.000	Passed/1.000

Table 9: Comparison of our 0.997 Shannon entropy estimation (in gray) with the results of statistical tests AIS31, Test procedure B and its entropy estimation according to test T8 for different number of rings ℓ with 39, 49 and 68 ring elements for accumulation time $D = 2000$.

are publicly available. They should play an essential role in the design and evaluation of oscillator-based TRNGs, where they can be used to assess the level of security of the TRNG aimed at cryptographic applications. They can also be used as an entropy

management tool for the evaluation of different options in the design space that represent a compromise between the cost and the best output bit rate while achieving the required level of security.

In the course of our study, we introduced two security models that are relevant when the entropy rate at the output of an MO-TRNG needs to be estimated. While Security model A has been implicitly used in previous works [3, 26], especially because it greatly simplifies computations, we have shown that it leads to an overly pessimistic underestimation of the entropy rate at the TRNG output, which can impact the cost and performance of the TRNG when one needs to reach a target entropy rate. Consequently, Security model B, which is even more realistic than model A, should be preferred when the entropy rate needs to be estimated. The only disadvantage of the Security model B compared to model A is that it involves more computations, caused essentially by the use of Markov chains, but, as we have shown in this paper, Security model B is still amenable to computations for real word MO-TRNG designs.

An interesting question for future research would be whether some function or a subset of functions could behave better with respect to the entropy rate per bit in the class of entropy conditioner functions. One possibility in this direction would be to generalize the analysis made by Dichtl in [7]. Since in our model, we are dealing with a general class of logic functions, this problem should not represent an insurmountable obstacle. The new TRNG model based on Markov chains should make evaluation of the efficiency of the new entropy conditioner even easier.

References

[1] R. B. Ash. *Information Theory*. Dover Publications, Inc., New York, 1990. Corrected reprint of the 1965 original.

[2] B. Barak and S. Halevi. An architecture for robust pseudo-random generation with applications to /dev/random. In *Proc. of the 12th ACM Conference on Computer and Communication Security (CCS'05)*, pages 203–212, 2005.

[3] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux. On the Security of Oscillator-based Random Number Generators. *Journal of Cryptology*, 24:398–425, 2011.

[4] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer. Electromagnetic analysis on ring oscillator-based true random number generators. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1954–1957, 2013.

[5] F. Bernard, V. Fischer, and B. Valtchanov. Mathematical Model of Physical RNGs Based on Coherent Sampling. *Tatra Mountains Mathematical Publications*, 45(1):1–14, 2010.

[6] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert. A Very High Speed True Random Number Generator with Entropy Assessment. In G. Bertoni and J. S. Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *LNCS*, pages 179–196. Springer, 2013.

[7] M. Dichtl. Bad and Good Ways of Post-processing Biased Physical Random Numbers. In *Fast Software Encryption - FSE 2007*, volume 4593 of *LNCS*, pages 137–152. Springer Verlag, 2007.

[8] B. Fino and V. Algazi. Unified Matrix Treatment of the Fast Walsh-Hadamard Transform. *IEEE Transactions on Computers*, 100(11):1142–1146, 1976.

[9] V. Fischer, F. Bernard, and N. Bochard. Modern random number generator design – Case study on a secured PLL-based TRNG. *it - Information Technology*, 61(1):3–13, feb 2019.

[10] V. Fischer and D. Lubicz. Embedded Evaluation of Randomness in Oscillator Based Elementary TRNG. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *LNCS*, pages 527–543. Springer, 2014.

[11] T. Guneysu. True Random Number Generation in Block Memories of Reconfigurable Devices. In *International Conference on Field-Programmable Technology – FPT 2010*, pages 200–207. IEEE Press, 2011.

[12] P. Haddad, F. Berdnard, V. Fischer, and Y. Teglia. On the Assumption of Mutual Independence of Jitter Realizations in P-TRNG Stochastic Models. In *Design, Automation and Test in Europe – DATE 2014, Dresden, Germany*. IEEE, 2014.

[13] W. Killmann and W. Schindler. A Design for a Physical RNG with Robust Entropy Estimators. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *LNCS*, pages 146–163. Springer, 2008.

[14] W. Killmann and W. Schindler. A proposal for: Functionality classes for random number generators, version 2.0. [online] Available at: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf, 2011. Accessed: 2023-10-31.

[15] M. Laban, M. Drutarovsky, V. Fischer, and M. Varchola. Platform for testing and evaluation of PUF and TRNG implementations in FPGAs. In *TRUDEVICE – 6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016), Barcelona, Spain*, November 2016.

[16] D. Lubicz. TRNG Test Suite. [online] Available at: https://github.com/dlubicz/MOTRNG, 2022. Accessed: 2022-01-23.

[17] A. T. Markettos and S. W. Moore. The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *LNCS*, pages 317–331. Springer Berlin Heidelberg, 2009.

[18] U. Mureddu, N. Bochard, L. Bossuet, and V. Fischer. Experimental study of locking phenomena on oscillating rings implemented in logic devices. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(7):2560–2571, 2019.

[19] M. Peter and W. Schindler. A proposal for Functionality classes for random number generators, version 2.35 - DRAFT. [online] Available at: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e.pdf, 2022. Accessed: 2023-10-31.

[20] W. Schindler and W. Killmann. Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications. In B. S. Kaliski, C. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *LNCS*, pages 431–449. Springer, 2003.

[21] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[22] B. Sunar, W. J. Martin, and D. R. Stinson. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE Transactions on Computers*, 56(1):109–119, 2007.

[23] G. Taylor and G. Cox. Behind Intel's New Random-Number Generator. [online] Available at: https://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator, 2011. Accessed: 2023-10-31.

[24] M. Varchola and M. Drutarovsky. New High Entropy Element for FPGA Based True Random Number Generators. In S. Mangard and F.X. Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *LNCS*, pages 351–365. Springer, 2010.

[25] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy. Fast Digital TRNG Based on Metastable Ring Oscillator. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *LNCS*, pages 164–180. Springer, 2008.

[26] X. Wu, Y. Ma, J. Yang, T. Chen, and J. Lin. On the Security of TRNGs Based on Multiple Ring Oscillators. In S. Chen, K. Choo, X. Fu, W. Lou, and A. Mohaisen, editors, *Security and Privacy in Communication Networks - SecureComm 2019*, volume 305 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer, 2019.