



HAL
open science

Complexity Assessment of Analog and Digital Security Primitives Signals Using the Disentropy of Autocorrelation

Paul Jimenez, Raphael Cardoso, Maurício Gomes de Queiroz, Mohab Abdalla, Cédric Marchand, Xavier Letartre, Fabio Pavanello

► **To cite this version:**

Paul Jimenez, Raphael Cardoso, Maurício Gomes de Queiroz, Mohab Abdalla, Cédric Marchand, et al.. Complexity Assessment of Analog and Digital Security Primitives Signals Using the Disentropy of Autocorrelation. 2024. hal-04740477

HAL Id: hal-04740477

<https://hal.science/hal-04740477v1>

Preprint submitted on 16 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Complexity Assessment of Analog and Digital Security Primitives Signals Using the Disentropy of Autocorrelation

Paul Jimenez, Raphael Cardoso, Maurício Gomes de Queiroz, Mohab Abdalla, Cédric Marchand, Xavier Letartre, and Fabio Pavanello

Abstract—The study of regularity in signals can be of great importance, typically in medicine to analyse electrocardiogram (ECG) or electromyography (EMG) signals, but also in climate studies, finance or security. In this work we focus on security primitives such as Physical Unclonable Functions (PUFs) or Pseudo-Random Number Generators (PRNGs). Such primitives must have a high level of complexity or entropy in their responses to guarantee enough security for their applications. There are several ways of assessing the complexity of their responses, especially in the binary domain. With the development of analog PUFs such as optical (photonic) PUFs, it would be useful to be able to assess their complexity in the analog domain when designing them, for example, before converting analog signals into binary. In this numerical study, we decided to explore the potential of the disentropy of autocorrelation as a measure of complexity for security primitives as PUFs, TRNGs or PRNGs with analog output or responses. We compare this metric to others used to assess regularities in analog signals such as Approximate Entropy (ApEn) and Fuzzy Entropy (FuzEn). We show that the disentropy of autocorrelation is able to differentiate between well-known PRNGs and non-optimised or bad PRNGs in the analog and binary domain with a better contrast than ApEn and FuzEn. Next, we show that the disentropy of autocorrelation is able to detect small patterns injected in PUFs responses and then we applied it to photonic PUFs simulations.

Index Terms—Complexity, disentropy, Pseudo Random Number Generators (PRNGs), Physical Unclonable Functions (PUFs).

I. INTRODUCTION

PHYSCAL Unclonable Functions (PUFs) represent a class of physical security primitives. They play the role of a physical key protecting an object that may be digital/analog data or electronic (and photonic) hardware. For some applications, they might be perceived as the electronic or photonic analogy of the biometric characteristics of a human being,

This work was supported by the French Agence Nationale de la Recherche under project number ANR-20-CE39-0004 - PHASEPUF project. F.P., X.L., and C.M. acknowledge funding from the European Union’s Horizon Europe research and innovation program under grant agreement No. 101070238. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

P. Jimenez, R. Cardoso, M. Gomes de Queiroz, M. Abdalla, C. Marchand and X. Letartre are with Ecole Centrale de Lyon, INSA Lyon, CNRS, Université Claude Bernard Lyon 1, CPE Lyon, INL, UMR5270, 69130 Ecully, France

F. Pavanello is with Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, Grenoble INP, CROMA, 38000, Grenoble, France

M. Abdalla is also affiliated with School of Engineering, RMIT University, Melbourne, VIC 3000, Australia

such as fingerprints [1], and provide interesting solutions in the context of the Internet of Things (IoT) [2].

PUF security is based on fabrication variations, i.e, variations from the designed dimensions of the structure due to the randomness of the manufacturing process.

The sensitivity to this randomness is what confers on the PUF its properties of uniqueness and non-clonability [3].

A PUF links an input information e.g., a binary string, called a “challenge” to a “response” in an ideally deterministic way forming a so-called challenge-response pair (CRP), that is not possible to be predicted in advance. Most of the time PUFs operate with multiple challenges, to build one or more responses from the set of all possible CRPs (CRPs library). Hence, the PUF should fulfil some characteristics [3], [4]:

- **Physical uniqueness:** As explained before, the PUF should act as a fingerprint and its responses should be mostly determined by the fabrication variations. Hence, for a same challenges different PUF instances should have different responses.
- **Physical unclonability:** Even with good equipment, an adversary should not be able to reproduce the PUF thanks to the unpredictability of fabrication variations.
- **Digital unpredictability:** It should be impossible to model the behavior of the PUF numerically (using machine learning attacks for example).
- **Reliability:** The PUF should be stable, i.e, for the same PUF instance, the same challenge should give the same response (within a certain error tolerance). This stability should hold over time, even in slightly different experimental conditions.

Depending on the application, some conditions have to be established on the challenge library size and on the quality of the responses. Especially in the case of a strong PUF, mostly suitable for authentication protocols, the CRP library should be very large to ensure that it would not be possible for an adversary to have access to a large portion of the CRP library in a reasonable amount of time [5]. Furthermore, the responses should be complex enough so that responses potentially harvested by an attacker cannot be used to train an algorithm able to predict the rest of the CRPs and to model the PUF behavior.

Consequently, the responses of a PUF must not have patterns that are repeated over all instances, within each individual response of the same instance and between the responses of the

same instance (if the responses are made of multiple bits). As a result, we should not detect patterns in the responses taken individually, nor in the concatenation of responses coming from the same PUF instance. Moreover, no patterns should be observed from the concatenation of responses of different PUF instances for a same challenge [6]. This argument applies also to cryptographic primitives for random number generators (RNGs) in the analog domain which require the absence of repeating patterns.

There is an entire zoo of PUFs, most of which are electronic ones and operate in the binary domain as the Arbiter, SRAM, or Butterfly PUFs [7]. Most of these electronic PUFs have responses of one bit for one given challenge. A multi-bits sequence is therefore built by concatenating the one-bit responses of the PUF for different challenges. In that case there is no single unambiguous way to assemble its responses into a sequence. For these PUFs, the NIST test suite [8], especially suited for the analysis of binary data, can be used to evaluate some of the statistical properties of the responses [9]. However, PUFs can also operate in the analog domain, especially in some recent implementations of optical and photonic PUFs [3]. For some of these PUFs one challenge give a ordered response of multiple bits [10]–[14].

Nevertheless, there is an important question to address when considering these PUFs that operate in the analog domain. The impact of fabrication variations affect some physical parameters, but PUFs output signals may present the need for a conversion from the analog to the binary domain to obtain responses that can be further used at a system level [10]–[14]. The quality of such responses will be impacted by the chosen conversion scheme: a different scheme will lead to different responses and probably different responses quality. Here by responses quality we refer to the responses complexity and entropy. Besides, post-processing may be needed to improve the responses quality. For example, in the first optical PUF by Pappu and co-workers in 2002 [13], the optical output of the PUF is a speckle pattern recorded on a 320×240 pixel camera that passes through a threshold filter to obtain an output in the binary domain, and finally through a Gabor transform to obtain a 2400-bit key. As discussed in [14], the entropy of the output response strongly depends on this transformation. A Gabor transformed image has regularities (zebra-stripes) leading to important patterns in the responses and hence to low entropy. Therefore, the Gabor transform may remove parts of the complexity contained in the optical speckle pattern.

This discussion can be also applied to silicon-photonic PUFs. For example, in [10] the authors convert their analog signal in the binary domain using a suite of Hadamard matrices. They also apply different transformations on the obtained binary signals to improve their quality (equalization, grey code, and XOR).

In those cases, we observe that the final entropic quality of a PUF is in fact the result of a combination of several steps:

- 1) The complexity of the physical device, its sensitivity to fabrication variations and its randomness properties.
- 2) The quality of the analog to binary signal conversion i.e., if the conversion retains the entropy contained in the analog signal.

- 3) The post-processing step to increase the complexity or stability of the PUF.

Hence, while designing a PUF, it would be ideal for a researcher to be able to work independently on each of these steps. However, by using common benchmarks like the NIST test suite, it is impossible to evaluate the PUF physical quality since it requires a binary conversion.

This is also a problem when using TRNGs that operate in the analog domain, such as photonic TRNGs [15], [16], and it is important to extract as much information as possible about the analog signal from the TRNG if we want to analyze the impact of the binary conversion for instance.

Our goal is therefore to find a way of evaluating the physical quality of a PUF design or a TRNG by looking at the complexity of their outputs in the analog domain directly. In that case, the Shannon entropy [17] used for binary data cannot be applied. Since one of the basic requirements is the absence of repeating patterns within or between responses/outputs, metrics based on the autocorrelation function may be ideal candidates. Therefore, in this paper, we propose to evaluate the performance of the disentropy, a metric developed by R. V. Ramos for quantum applications [18], [19], but also to obtain a score based on the autocorrelation function [19], [20]. We will first describe the disentropy of the autocorrelation mathematically, then we will introduce other metrics that are used to evaluate the complexity of analog signals like the Approximate Entropy also used in cryptography and security. Finally, we will compare and evaluate the performance of these metrics on PRNGs and PUFs responses (analog but also binary) to assess whether the disentropy of autocorrelation is a good candidate or not to evaluate the quality of security primitives. For testing the metrics, rather than TRNG, we decided to test the metrics on common deterministic PRNGs whose random properties are easy to modify.

II. THE DISENTROPY OF THE AUTOCORRELATION FUNCTION

A. The autocorrelation function

The autocorrelation function measures the similarity between a function or a signal and a delayed version of itself with delay τ . Castro et al. [19] and Ramos [20] claim that this function can be used to measure some randomness present in the signal.

For discrete signals, the autocorrelation function is given by:

$$r_k = \frac{c_k}{\sigma_0^2} = \frac{\frac{1}{N} \sum_{t=1}^{N-k} (s_t - \bar{s})(s_{t+k} - \bar{s})}{\sigma_0^2} \quad (1)$$

With k the considered lag (delay), s_t the discrete signal value at time t , \bar{s} the mean value of s_t and σ_0^2 its sample variance. It is important to note that the autocorrelation function can be either positive or negative depending on whether two events are correlated or anti-correlated.

As explained by Ramos in [20] the goal is to obtain a score from the function, hence, we need to map the function to a scalar. An intuitive way to obtain it would be to use the Shannon entropy H defined as [17]:

$$H = - \sum_{i=1}^n p_i \log(p_i) \quad (2)$$

Eq. 2 is expressed in terms of a discrete set of probabilities $\{p_i\}$.

However, since the autocorrelation function can have negative values, it cannot be associated to a probability distribution. Moreover, the logarithm would not be defined on negative values so the classical definition of entropy as Shannon entropy cannot be used to map the autocorrelation function to a scalar.

B. The construction of disentropy

In 1988 Tsallis generalised the definition of entropy for multifractal systems [21] with $q \in \mathbb{R}$ as:

$$S_q = k \frac{1 - \sum_i p_i^q}{q - 1} \quad (3)$$

Eq. 3 tends to the Boltzmann-Gibbs entropy in the limit of $q \rightarrow 1$. Then, in 1994 he proposed a new way of interpreting the experimental measurements as q -expectation values [22]. In this work he defines the generalised logarithmic function as:

$$\ln_q(x) = \frac{x^{(1-q)} - 1}{1 - q} \quad \forall x \in \mathbb{R}^+ \text{ and } q \neq 1 \quad (4)$$

This logarithm tends to the natural logarithm when $q \rightarrow 1$ and a generalised exponential function can be attributed to it:

$$e_q^x = (1 + (1 - q)x)^{\frac{1}{1-q}} \quad \forall x / (1 + (1 - q)x \geq 0) \text{ and } q \neq 1 \quad (5)$$

With this new logarithm definition (see Eq. 4), it is possible to express the Tsallis q -entropy S_q as in Eq. 6.

$$S_q = -k \sum_i p_i^q \ln_q(p_i) \quad (6)$$

Then, da Silva and Ramos introduced the Lambert function $W(z)$ [20], [23] to further extend the meaning of this definition of entropy. $W(z)$ is obtained by solving:

$$W(z)e^{W(z)} = z \quad (7)$$

This equation has an infinite number of solutions, but only two branches give real values for $z \in \mathbb{R}$. By taking the logarithm of Eq. 7, and defining $z = p_i$ it is possible to obtain a definition of entropy as:

$$S = - \sum_i p_i \ln(W(p_i)) - \sum_i p_i W(p_i) \quad (8)$$

The term $\sum_i p_i W(p_i)$ is called the disentropy. It is minimal when the entropy is maximal and vice-versa. It is important to note here that the disentropy does not contain a logarithm in its expression.

Next, this entropy is generalised using the Tsallis generalised exponential and Lambert-Tsallis W_q function solution of:

$$W_q(z)e_q^{W_q(z)} = z \quad (9)$$

By taking the Tsallis q -logarithm of Eq. 9:

$$\ln_q(z) = W_q(z) + ((1 - q)W_q(z) + 1) \ln_q(W_q(z)) \quad (10)$$

At last, a probability p_i is inserted in Eq. 9 with $z = p_i$ and by using the Tsallis q -entropy of Eq. 6 neglecting k , one obtains [19], [20]:

$$S_q = - \sum_i p_i^q W_q(p_i) - \sum_i p_i^q \ln_q[W_q(p_i)] - (1 - q) \sum_i p_i^q W_q(p_i) \ln_q[W_q(p_i)] \quad (11)$$

where the term:

$$D_q = \sum_i p_i^q W_q(p_i) \quad (12)$$

is the Tsallis q -disentropy. Recall that the autocorrelation function can take values ranging from -1 to 1 . Therefore, a function $W_q(r_k)$ defined on this interval shall be found. The $W_2(z)$ function below is defined on the interval of values taken by the autocorrelation function:

$$W_2(z) = \frac{z}{z + 1}, \quad z > -1 \quad (13)$$

So, by replacing $z = p_i$ in Eq. 12 one obtains a value for the disentropy in the continuous and discrete probability distributions:

$$D_2 = \sum_i \frac{p_i^3}{p_i + 1} \quad (14)$$

Ramos has shown that this metric can be defined for functions that are not probability distributions. For example in the case of the autocorrelation function in [19], [20] and for the Wigner function in quantum mechanics in [24]. Therefore, in the case of autocorrelation function, they obtain the disentropy given by Eq. (15).

$$D_2 = \sum_{k=0}^{N-1} \frac{r_k^3}{r_k + 1} \quad (15)$$

With this metric it is possible to have a defined scalar value for the complexity of a signal using the autocorrelation function. A large positive or negative disentropy shows the presence of correlations (or anti-correlations) within the signal. Its ideal value is $D_2 = 0.5$ obtained if $r_0 = 1$ and $r_k = 0 \quad \forall k \neq 0$. Note that the disentropy is not defined for $r_k = -1$ i.e, it is not defined in the case of perfect anti-correlation. In that case, the metric diverges.

III. APPROXIMATE ENTROPY AND FUZZY ENTROPY

In our study of complexity measures for analog signals based on the disentropy, we need to compare its performances with those of other metrics. Below, we will discuss and compare two of the most used metrics for assessing the complexity of signals.

A. The Approximate Entropy

As mentioned in the introduction, the NIST test suite for random and pseudo-random generators generating binary signals [8] is commonly used in cryptography and security to evaluate the quality of security primitives [25], [26]. In this test suite

the Approximate Entropy (ApEn) is used to measure the complexity and detect the presence of repeating patterns in binary signals. The ApEn has actually been developed by Steven M. Pincus in 1991 [27] for any kind of vector in \mathbb{R}^N . This metric is currently used in medicine [28], [29], but also in other domains such as climate studies [30] or finance [31]. The full algorithm of ApEn can be found in [27], [32] and summarized in Algorithm 1. This algorithm has three parameters: $m \in \mathbb{N}^*$ called the embedding dimension, $r \in \mathbb{R}^+$ called the noise filter or scaling parameter, and N the number of samples. The ApEn takes patterns of m points in the signal, then identifies other patterns that are similar across the signal, and determines which of these patterns remains similar for the following $m + 1$ points. In more mathematical terms, ApEn is based on the conditional probability that a signal that repeated itself for m points will repeat itself for $m + 1$ points [33].

Algorithm 1 The Approximate Entropy algorithm.

Let $s \in \mathbb{R}^N$ be a time series of length N , and $n = N - m + 1$. Define $\mathbf{x} \in \mathbb{R}^m$ as:

$$\mathbf{x}(i) = [s(i), s(i + 1), \dots, s(i + m - 1)] \quad \forall i \in \mathbb{N}^* \text{ and } i \leq n$$

Then compute:

$$C_i^m(r) = \frac{\text{Number of values } j \text{ such that } d[\mathbf{x}(i), \mathbf{x}(j)] \leq r}{n}$$

With d a metric comparing two vectors:

$$d[\mathbf{x}(i), \mathbf{x}(j)] = \max_{k=1, \dots, m} (|s(i + k - 1) - s(j + k - 1)|)$$

Next, compute ϕ^m :

$$\phi^m(r) = \frac{1}{n} \sum_{i=1}^n \log(C_i^m(r))$$

The ApEn is then defined by:

$$\text{ApEn}(m, r, N) = \phi^m(r) - \phi^{m+1}(r)$$

In the computation of $C_i^m(r)$, we see that for similarity the algorithm compares blocks within the resolution r based on the Heaviside function: if $d[\mathbf{x}(i), \mathbf{x}(j)] \leq r$, then the patterns are considered similar. Note that r is usually a function of the standard deviation of s [27], [32].

B. The Fuzzy Entropy

The ApEn can be biased and may indicate more similarities than contained in the series. It can also be inconsistent and sensitive to a change in r , and depends on the length N of the series [32]–[34]. Therefore, other metrics emerged, for example the Sample Entropy (SampEn) developed by Richman and Moorman in 2000 [34] fixes some of ApEn problems. In fact, they have shown that SampEn does not depend on the series length if N is big enough, and is less biased. This makes SampEn interesting, but its results can still be untrustworthy for small N [33], [35]. This problem is mainly linked to the fact that both SampEn and ApEn use a Heaviside function as a two state classifier for the blocks similarity. In reality, this frontier is blurry and it is not easy to determine whether a

pattern belongs to one class or to the other [33], [35]. The Fuzzy Entropy (FuzEn) has been introduced [33] to overcome these problems by using the fuzzy sets theory developed by Zadeh in 1965 [36]. In this paper, Zadeh introduced the idea of fuzzy sets as a “class with a continuum of grades of membership” with a “membership function” $f_A(x)$ associating every object x of a space X to a real number in $[0, 1]$. $f_A(x)$ represents this “grade of membership” of x in A . The closer $f_A(x)$ is to 1, “the higher the grade of membership of x in A ” [36].

In FuzEn algorithm, this membership function f_A is therefore used to replace the Heaviside function in ApEn algorithm.

In [33] Chen et al. use a family of exponential functions as membership functions. However, other membership functions can be used such as triangular, Z-shaped, constant-Gaussian as presented in [35]. They are all functions of r and of a defined distance metric comparing two vectors, as in Algorithm 1.

In our study, we will compare the results obtained with the disentropy of the autocorrelation to the results obtained with ApEn because of its use in cryptography for binary signals and FuzEn to dispose of ApEn defaults. These metrics will be first tested on different PRNGs of good and poor quality. All metrics will be used to find regularities and patterns in their outputs; ApEn and FuzEn by looking for repeating patterns of size m and $m + 1$ along the signal and the disentropy of autocorrelation by comparing the signal with a delayed version of itself using autocorrelation.

If a signal does exhibit clear repeating patterns ApEn and FuzEn would have a value close to 0, on the other side this signal would obtain a high $|D_2|$ value.

IV. SOME PRNGS

A. Linear congruential generator (LCG)

This generator produces a sequence of pseudo-randomised numbers based on linear recursions given by [37], [38]:

$$x_{k+1} = ax_k + c \text{ mod } M \quad (16)$$

x_k being the sequence with $k \in \mathbb{N}$, $M \in \mathbb{N}^*$ is the modulo, x_0 is the seed, a is the multiplier and c the increment, such that $x_0, a, c \in \mathbb{Z}_M$ [38]. This generator is a common and old method to make a PRNG, the linear method with $c = 0$ has been developed in 1951 by D.H. Lehmer [39].

In this study, we use Lehmer generators and linear congruential generators with parameters shown in Table I. To obtain optimised PRNGs of good quality with a period of M , the parameters M , a , c and x_0 must meet certain conditions [37], [38]. Among the requirements, in the case where M is a multiple of 4, $a - 1$ should also be multiple of 4; further, when $c = 0$, the seed x_0 should be a co-prime of M . Some parameters in Table I have been chosen to match the parameters of commonly used random functions as the C++11 `minstd_rand` function or the GNU C Library `rand` function [38]. Since x_0 should be co-prime of M when $c = 0$, we chose $x_0 = 1$.

Using Eq. 16 to generate PRNGs allows us to create non-optimised ones or PRNGs exhibiting periodic patterns by

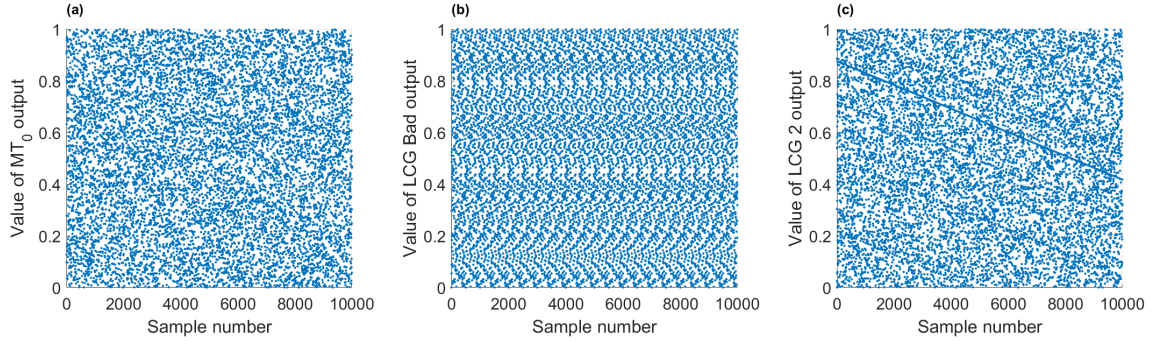


Fig. 1. Examples of normalised PRNGs output for 10000 samples (a) MT_0 (b) LCG Bad (c) LCG 2. Before normalisation, MT_0 output have been generated using the MATLAB™ `rand` function, LCG Bad and LCG 2 with Eq. 16 equation and parameters in Tab. I.

choosing parameters a , M , and c that do not meet the conditions required to create an optimized PRNG. These PRNGs presented in Table I (LCG Bad, LCG 1, 2, 3, 4) will therefore be compared to the optimised ones using the metrics presented earlier. The parameters of LCG 1, 2, 3 and 4 have been chosen in order to observe patterns in their output visually, while still exhibiting randomness as represented on Fig. 1c. LCG 1, 2 and 3 have the same a and M but different c while LCG 3 and 4 only have a different M .

The parameters of LCG Bad have been chosen to create a bad PRNG exhibiting periodicity as illustrated on Fig. 1b. In this study, the output of all LCGs will be normalised between 0 and 1.

TABLE I
PARAMETERS OF LEHMER AND LINEAR CONGRUENTIAL GENERATORS.

PRNG	M	a	c
Minimal standard generator [37]	$2^{31} - 1$	16807	0
C++11 <code>minstd_rand</code>	$2^{31} - 1$	48271	0
GNU C Library <code>rand</code>	2^{31}	1103515245	12345
LCG Bad	5000	17	256
LCG 1	2^{20}	1487	25436
LCG 2	2^{20}	1487	25236
LCG 3	2^{20}	1487	25336
LCG 4	2^{19}	1487	25336

B. Mersenne-Twister PRNG

The Mersenne-Twister (MT) algorithm [40] is a common tool to generate sequences of random numbers. For example, it is used in python `random` module [41] and MATLAB™ `rand` function [42]. Furthermore, the MT algorithm can be used as a base for cryptographic cyphers, for example CryptMT [43], or image watermarking techniques [44]. MT uses the twisted generalized feedback back shift register (TGFSR) algorithm developed in [45]. The MT19937 algorithm described in [40] has a remarkably large prime period $M = 2^{19937} - 1$ making it a good PRNG.

In this work, the MT19937 algorithm in MATLAB™ `rand` function will be used with seed 0 and a randomly picked

seed i.e., $S = 1773456103$, two PRNGs which will be called MT_0 and MT_S , respectively. Now that we have defined which PRNGs are going to be used in this study, we can apply to them the various metrics discussed so far.

V. PRNGS RESULTS

A. Tests on analog signals

For the ApEn metric, we utilised the MATLAB™ `approximateEntropy` function [46]. It is recommended to use the ApEn algorithm with $m = 2$ or $m = 3$ as well as r between $0.1\sigma_0$ and $0.2\sigma_0$ [27], [32]. Therefore, we took $m = 2$ and $m = 3$, and the `approximateEntropy` function uses $r = 0.2\sigma_0$.

For the FuzEn metric, we used the algorithm provided in [35] with a Gaussian membership function as recommended for long signals for a faster computation time. We also took $m = 2$ and $m = 3$ as well as $r = 0.1253\sigma_0$ as recommended in the algorithm.

A sweep in m from $m = 2$ to $m = 8$ has been performed for ApEn and FuzEn, results are shown in Appendix A and confirm the choice of $m = 2$ and $m = 3$.

To compute the autocorrelation function we used the MATLAB™ `autocorr` function [47] for each lag (delays) from lag = 0 (no delay) to lag = $\text{length}(s) - 1$.

Next, the disentropy of the autocorrelation function (D_2) is obtained using Eq. 15 by summing for each lag value.

Recall here that D_2 can be positive or negative with an ideal value at $D_2 = 0.5$ in the absence of patterns in the signal [20]. Therefore, we decided to focus on $\mathcal{D} = |D_2 - 0.5|$ to better appreciate the variations around 0.

To know how many samples are needed to perform the analysis, we analysed how ApEn and FuzEn with $m = 2$ evolve with the number of samples. Results obtained by ApEn and FuzEn applied on MT_0 are shown in Fig. 2. With the parameters defined previously, ApEn needs more than 1000 samples to converge. The dependency of ApEn on the number of samples was expected as discussed in Section III-A. FuzEn converges faster than ApEn, but still needs at least 1000 samples to reduce its oscillations to $\sigma_{FuzEn} \sim 10^{-2}$ in order of magnitude. On the other hand, the disentropy tends to 0 after few oscillations; for 1000 samples its oscillations are in the order of $\sigma_{\mathcal{D}} \sim 10^{-3}$. It is not represented on Fig. 2 for reasons of visibility.

In the next studies, we chose to take a number of samples $N = 10000$ to make sure that both ApEn and FuzEn have enough samples to converge and have small oscillations.

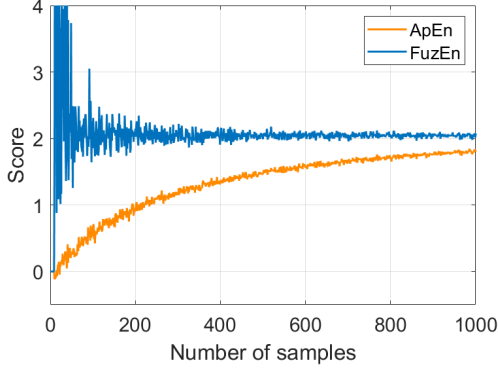


Fig. 2. Convergence study of ApEn and FuzEn for MT_0 and $m = 2$.

After 10000 samples, the oscillations for ApEn and FuzEn are in the order of $\sigma_{FuzEn} \sim \sigma_{ApEn} \sim 10^{-3}$, and $\sigma_{\mathcal{D}} \sim 10^{-4}$ for the the disentropy. Hence, we generated outputs of 10000 samples for each PRNG. For example, the output of MT_0 , LCG Bad, and LCG 2 are presented on Fig. 1.

We observe on Fig. 1a that the output of MT_0 does not exhibit observable patterns. On the other hand, by looking at Fig. 1b we see that LCG Bad has clear repeating patterns and should have bad results for all the metrics, i.e., ApEn and FuzEn close to zero and a high disentropy. As seen on Fig. 1c, LCG 2 does not necessarily have repeating patterns, but straight lines can be observed in its output.

Then, we decided to apply the different metrics on the PRNGs and compare their results to the scores obtained by MT_0 (for each respective metric) presented in Tab. II.

TABLE II
 MT_0 SCORES FOR ANALOG SIGNALS

\mathcal{D}_0		ApEn ₀	FuzEn ₀
$4.80 \cdot 10^{-5}$	$m = 2$	2.156	2.040
	$m = 3$	1.848	1.926

Results are shown on Fig. 3. All metrics were clearly able to distinguish the repeating patterns of LCG Bad, but in order

to have a proper visibility of the other PRNGs results, it was decided to remove its scores from Fig. 3. However, results from LCG Bad are presented in the Appendix A.

As expected, we observe that the different good or well-known PRNGs (MT_S , MSG, GNU C, C++) obtain good scores for all metrics, close to those of MT_0 with $D \sim 10^{-4}$. However, only the disentropy is capable of clearly distinguishing between these known PRNGs and LCG 1, 3 and 4. FuzEn even perceives them as better PRNGs. It can be also seen that $m = 3$ obtains better contrast than $m = 2$ for ApEn and FuzEn.

However, one observes that LCG 2 obtains a good score for the disentropy and the best one for ApEn and FuzEn with $m = 3$. This means that \mathcal{D} , ApEn and FuzEn are not capable of perceiving the lines observed for LCG 2 in Fig. 1(c). These lines are actually dotted lines with a certain periodicity; this period p_{line} has an impact on the metrics scores. A test has been conducted with MT_0 output in which we included a diagonal line of a certain periodicity. With $m = 3$, ApEn and FuzEn are not able to detect the line for $p_{line} > 3$. With $m = 2$ they don't detect it for $p_{line} > 2$, we also tested with $m = 4$ and concluded that ApEn and FuzEn would not detect the line for $p_{line} > m$. On the other side, the disentropy is not capable of detecting the line for $p_{line} > 40$ while the line with the smallest p_{line} in LCG2 has a period of 64 samples.

By looking at Eq. (15), it is possible to deduce that the score obtained by the disentropy will depend highly on the number of samples if the pattern repeats itself periodically. Indeed, if one defines T as the period of the pattern, and τ_p a point in the periodic pattern, then $r_{\tau_p} = r_{\tau_p+T} > 0$. Hence, the contribution of the periodic pattern will be added each time it is observed. This behavior has been verified with LCG Bad that repeats itself every 500 samples. Fig. 4 shows that the disentropy score is small for a small number of samples i.e., below 500. However, above 500 samples the metric begins to analyze redundancies and the score $\mathcal{D} = |D_2 - 0.5|$ grows linearly with the number of samples due to the increase of autocorrelation functions r_{500j} with $j \in \mathbb{N}^*$.

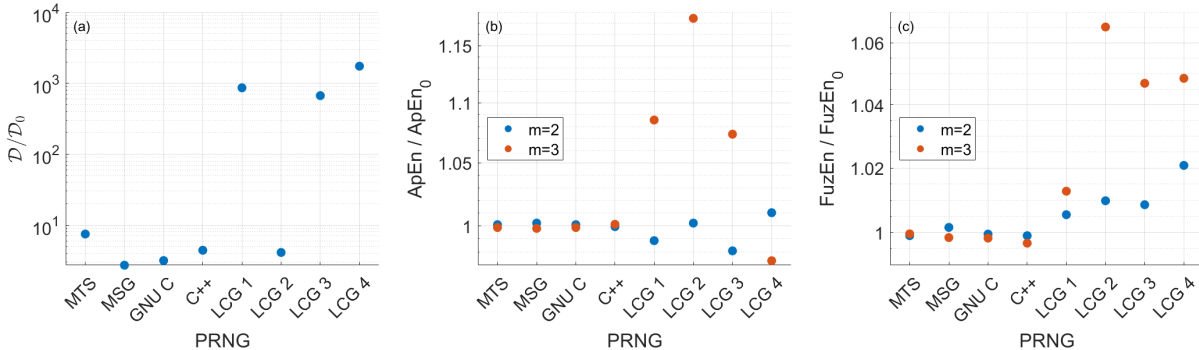


Fig. 3. Score ratio of PRNGs obtained from (a) $\mathcal{D} = |D_2 - 0.5|$, (b) ApEn, and (c) FuzEn compared to MT_0 scores for analog signals of Table. II.

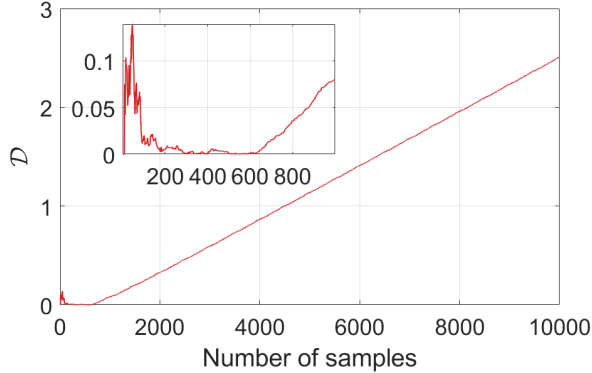


Fig. 4. Evolution of LCG Bad disentropy with N .

B. Tests on binary signals

Since Eq. (1) is also defined for binary signals we want to test how the disentropy behaves in that case.

By placing a comparator at 0.5 on the PRNGs output of section V-A we converted these analog signals into binary ones. For binary signals the NIST test suite manual [8] recommend to have $m < \lfloor \log_2(N) \rfloor - 5$. With $N = 10000$ this condition becomes $m < 8$, hence we decided to compute the entropies with $m \in [2, 8]$ with the last point $m = 8$ included. We decided to do the same analysis as in Fig. 3 by comparing the PRNG scores to the MT_0 scores shown in Tab. III.

TABLE III
 MT_0 SCORES FOR BINARY SIGNALS.

\mathcal{D}_0		ApEn ₀	FuzEn ₀
$2.63 \cdot 10^{-4}$	$m = 2$	0.6930	0.6932
	$m = 3$	0.6929	0.6935
	$m = 8$	0.6798	0.6931

First, one observes that ApEn₀ and FuzEn₀ scores decrease from values close to 2 in Tab. II to values close to $\log(2) \approx 0.6931$. In fact, the highest ApEn value for binary signals is known to be equal to $\log(2)$ [32].

As in the analog case, all metrics were able to discriminate LCG Bad; its scores are presented in the Appendix A.

Results for the other PRNGs are presented in Fig. 5 where we observe that only the disentropy is capable to distinguish between the well-known PRNGs and LCG 1, 2, 3 and 4. In Fig. 5(a) and Fig. 5(b) we decided to represent $m = 2$ and $m = 3$ because they were the inner dimensions used in Section V-A and $m = 8$ because it exhibited the highest contrast between the PRNGs. The well-known PRNGs still obtain results close to $\mathcal{D}_0 \sim 10^{-4}$. On the other hand, ApEn and FuzEn are only capable to discriminate LCG Bad and LCG 1. By increasing m further, both are more and more discriminated, but all the other PRNGs obtain scores around $\log(2)$.

It is possible to obtain a p -value from ApEn in the NIST test suite to discriminate between random and non-random binary series; if $p > 0.01$ the series is considered random [8]. Therefore, the ApEn algorithm from [48] has been used.

This algorithm obtains results for $m = 2, 3$ and 8 similar to the MATLABTM

approximateEntropy function. In terms of p -values, only LCG Bad is considered as non-random except for $m = 8$ where LCG 1 obtains $p = 0.0096$ and is therefore near the limit.

Here, the disentropy of autocorrelation outperformed ApEn and FuzEn and, this time, it is capable of discriminating LCG 2 from the other good PRNGs with one order of magnitude difference.

VI. PUFs

A. Deterministic patterns

In this section we initially consider series of analog signals that are not experimental data generated by PUFs.

First, we generate the responses with the MT algorithm, then we insert patterns inside and, finally, we test them.

We generated the responses R_i of $n = 128$ samples as if they were responses coming from the same analog PUF instance or responses coming from different PUF instances to be converted in binary. As a reference, 100 different responses with the MATLABTM rand function were taken, then we concatenated them to obtain a series C_0 of size $N = 12800$. From an application point of view analyzing the concatenation of responses of different PUF instances to the same challenge could be used to detect correlations between PUF instances for example as explained in [6]. Detecting patterns between instances or between responses of the same instances can also give information about the PUF to an attacker. The reader may wonder how to order the concatenation of PUFs responses to extract as much information as possible, this would not be addressed in this paper but any patterns found in any responses ordering might give information about the PUF to an attacker in case they have access to PUF instances.

As no patterns are present in the responses, the concatenation C_0 should obtain good scores. Besides, we generated 128-sample-long responses with rand, but this time we added the conditions presented in Algorithm 2.

Algorithm 2 Deterministic dynamics algorithm.

```

for  $k \in [2, n]$  do
  if  $s(k-1) \geq 0.9$  then
     $s(k) \leftarrow 0.1$ 
  else if  $s(k-1) \leq 0.1$  then
     $s(k) \leftarrow 0.9$ 
  end if
end for

```

By doing so, we insert two patterns in the responses of the PUF with a probability of $p = 0.1$ each. This simulates a PUF with a certain deterministic dynamics. These responses will be the test responses and will be compared to the reference ones for each metrics. This study has been performed 100 times to emulate 100 different PUF instances. For each instance we obtained the concatenation of their responses C_i with $i \in \mathbb{N}^*$ and $i \leq 100$.

For each C_i we compute the metrics for $m = 1, 2$ and 3. We decided to use $m = 1$ because the patterns implied two

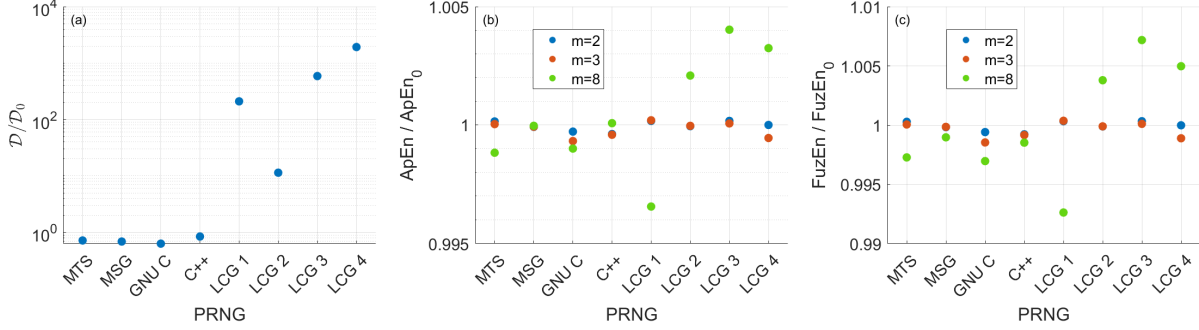


Fig. 5. Score ratio of PRNGs obtained from (a) $\mathcal{D} = |D_2 - 0.5|$ (b) ApEn (c) FuzEn compared to MT_0 scores for binary signals of Table. III.

samples. Therefore, ApEn and FuzEn might perform better with $m + 1 = 2$ considering Algorithm 1.

TABLE IV
AVERAGE METRIC SCORE RELATIVE DIFFERENCE OF C_i BETWEEN THE TEST AND REFERENCE RESPONSES.

\mathcal{D}		ApEn	FuzEn
+27742%	$m = 1$	-15%	-26%
	$m = 2$	-20%	-16%
	$m = 3$	-12%	-19%

In Table IV, we observe that all the metrics were able to differentiate between the reference and the test responses. However, for the test responses, the \mathcal{D} scores increased by $\sim 10^2$, while ApEn and FuzEn only decreased by $\sim 10^{-1}$ in order of magnitude.

It is possible to analyse the responses individually by removing the concatenation step. In that case, the metrics are applied on series of 128 samples making it more difficult to differentiate between the reference responses and the test responses. The study has been performed with 500 responses to have enough data for the mean values of the metrics to converge. Between the reference and the test signals, the mean value of \mathcal{D} over all responses increased by 331%. FuzEn($m = 1$) and FuzEn($m = 3$) decreased by 25% and 20%, respectively and ApEn($m = 1$) decreased by 13%, while ApEn($m = 3$) increased by 62%.

Note that ApEn($m = 3$) gives a wrong result in that case, in fact its mean value should decrease as ApEn($m = 1$) and FuzEn. This error is probably due to shortness of the responses that hinders ApEn to give meaningful results as the pattern is too short to be observed with $m = 3$.

Now we consider 100 instances of a PUF that has a defect where some samples will always be equal. For example, one can set the value of the first sample, forcing it to be equal to 0.5. In that case, no metrics are able to differentiate between the reference and the test concatenations. If we force the first two response samples to be equal to 0.2 and 0.1 respectively, the [0.2, 0.1] pattern will repeat itself every 128 bits in the concatenation. The tests are performed with different number of responses N_{resp} . Intuitively, by increasing the number of responses, we increase the length of the concatenation and number of repeating patterns inside. Results are presented

in Tab. V. We decided to put 0% if the relative difference between the reference and test responses is lower than 1% and smaller than the standard deviation of the metrics over the 100 instances.

TABLE V
AVERAGE METRIC SCORE RELATIVE DIFFERENCE OF C_i BETWEEN THE TEST RESPONSES WITH THE [0.2, 0.1] PATTERN AND REFERENCE RESPONSES.

Metrics	$N_{resp} = 100$	$N_{resp} = 200$	$N_{resp} = 500$
\mathcal{D}	0%	+719%	+4930%
ApEn($m = 1$)	0%	-0.3%	-0.3%
FuzEn($m = 1$)	-0.5%	-0.5%	-0.5%
ApEn($m = 2$)	0%	-0.3%	-0.3%
FuzEn($m = 2$)	-0.4%	-0.3%	-0.3%
ApEn($m = 3$)	0%	-0.2%	-0.2%
FuzEn($m = 3$)	0%	-0.3%	-0.3%

For $N_{resp} = 100$, we observe that only FuzEn is capable of detecting the pattern with a very small contrast of 0.5% for $m = 1$ and 0.4% for $m = 2$. However, by increasing the number of responses, and therefore by increasing the number of patterns present in the signal, the disentropy contrast increases drastically, while the FuzEn and ApEn contrast remains very small for all m .

B. Test on PUFs responses

Here, we use the simulated outputs of photonic PUFs previously published [11]. In this paper different photonic PUF architectures based on symmetric ring resonator arrays have been simulated and their responses have been tested using NIST tests (including ApEn) for two different binary conversions. We will focus our discussion here on PUF 1 and PUF 3 because of their different NIST ApEn [48] scores. PUF 3 is very simple and composed of one ring per array while PUF 1 is more complex and composed of 10 rings per array.

In this paper, the binary challenges are created using the MT algorithm and encoded in light. 150 instances of each architecture are simulated with fabrication variations. For each instance 100 responses of 128 bits are generated by sampling and converting the analog outputs into binary. At the end, these responses have been concatenated for the NIST tests

evaluation. Their results are compared in Table VI with $N_{ApEn}^{\%}$ the percentage of instances passing the ApEn test [48].

TABLE VI
RESULTS SUMMARY OF PUF 1 AND PUF 3 ARCHITECTURES.

	PUF 1	PUF 3
$N_{ApEn}^{\%}$	1-7%	46%
Uniqueness	High	Low
Impact on challenge-encoded inputs	High	Low

For two different binary conversion schemes, PUF 1 obtains low $N_{ApEn}^{\%}$ compared to PUF 3 despite the bias in PUF 3 responses due to the conversion to a binary format. PUF 1 can operate effectively on the inputs encoded with the challenges to generate unique instances because of its fabrication variations and architecture. It also degrades the quality of the inputs, while PUF 3 is not complex enough to have an impact on the inputs.

Given that $N_{ApEn}^{\%}(\text{PUF 1}) < N_{ApEn}^{\%}(\text{PUF 3})$ in the binary domain, one expects $\mathcal{D}(\text{PUF 1}) > \mathcal{D}(\text{PUF 3})$ in the analog domain as well. For the 150 instances we indeed obtain $\mathcal{D}(\text{PUF 1}) \sim 10^{-2}$ and $\mathcal{D}(\text{PUF 3}) \sim 10^{-4}$.

Moreover, with a number of samples in the same order of magnitude $N = 100 \times 128 \sim 10^4$ we expected $\mathcal{D}(\text{PUF 3})$ to be close to $\mathcal{D}(\text{MT}_0) \sim 10^{-4}$ of Fig. 3(a).

Also, as shown in Fig. 3(a), the score $\mathcal{D}(\text{PUF 1}) \sim 10^{-2}$ has the same order of magnitude as \mathcal{D} of LCG 1, 3 and 4 exhibiting patterns. This indicates the presence of patterns in the analog PUF 1 outputs and indicates that the low $N_{ApEn}^{\%}(\text{PUF 1})$ is not entirely caused by a bad binary conversion scheme or the bias in the responses.

On the ApEn and FuzEn side, we discovered that the analog outputs of PUF 3 obtained a score very close to $\log(2)$ for both metrics. This means that ApEn and FuzEn are seeing the outputs of PUF 3 as signals close to binary data.

As shown in Fig. 6, the normalised analog outputs of PUF 3 are closely oscillating between two levels: 0 and 1 due to the low impact that the PUF 3 architecture has on the input light encoded with the binary challenges. Therefore, $\text{ApEn}(\text{PUF 1}) > \text{ApEn}(\text{PUF 3})$ and $\text{FuzEn}(\text{PUF 1}) > \text{FuzEn}(\text{PUF 3})$ even if PUF 1 has patterns and $N_{ApEn}^{\%}(\text{PUF 1}) < N_{ApEn}^{\%}(\text{PUF 3})$.

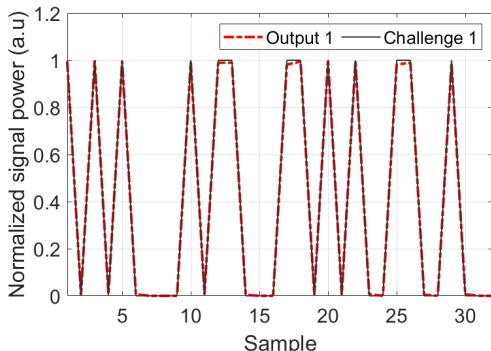


Fig. 6. Normalised output 1 of PUF 3 first instance compared to challenge 1. The delay caused by the PUF have been compensated to align the challenge with the output on the figure.

Hence, one should be careful while using ApEn and FuzEn on analog signals if they oscillate between two levels. As we observed on Fig 3(a) and Fig. 5(a), the disentropy is not highly affected by the transformation from analog to binary signal (c.f the well-known PRNGs and LCG Bad results). As mentioned in [32], the ApEn algorithm gives a relative value allowing to compare signals using the same “alphabet”. Therefore, depending on the application, this characteristic of ApEn and FuzEn to be sensitive to the type of signals can make a comparison between analog signals complicated.

One could argue that a binary signal is less complex than an analog signal with a very large number of levels. However, for security applications where the signal has to be converted into binary data, the detection of patterns and correlations in the signal might be more important than the complexity in terms of levels and the instability of ApEn and FuzEn can be detrimental in this regard.

VII. CONCLUSION

In this study we were able to show that the disentropy of autocorrelation is an interesting metric to assess the complexity of security primitives outputs, both in the analog as well as in the binary domain. The disentropy is able to differentiate between well-known and optimized PRNGs and ones of lower quality with a greater contrast than ApEn and FuzEn. It can also be used to detect patterns in the analog outputs and binary responses of PUFs.

Furthermore, the disentropy does not need inputs whereas ApEn and FuzEn results depend highly on their input parameters such as the inner dimension m . In addition, ApEn and FuzEn depend on the type of the input signal s ; with the chosen m and r we observed that the ideal value for ApEn and FuzEn is close to 2 for analog signals and $\log(2)$ for binary ones or analog signals fluctuating between two levels. On the other hand, the results obtained from the disentropy do not suffer from this characteristic since it only detects the amount of correlation in signals.

However, we saw that none of the metrics were able to detect the lines in LCG 2 analog output. ApEn and FuzEn failed in detecting these lines for a period $p_{line} > m$ and the disentropy for $p_{line} > 40$. Also, the score given by the disentropy depends highly on the length of the signal. Therefore, we recommend using it with a comparison. We suggest to compare the score obtained by the signal under test alongside a signal of the same length generated with the MT_0 algorithm.

APPENDIX

To confirm the choice of $m = 2$ and $m = 3$ for ApEn and FuzEn we decided to conduct a study of the influence of m in the ApEn and FuzEn scores. As shown in Fig. 7, with $m = 1$ we cannot differentiate between MT_0 and LCG Bad. Using $m = 2$ we obtain a high score for MT_0 as expected, but a small contrast compared to LCG Bad. $m = 3$, seems to be the best candidate since it has the largest contrast between the two PRNGs while giving a high score for MT_0 . For $m > 3$, the contrast decreases and ApEn gets close to 0 for MT_0 .

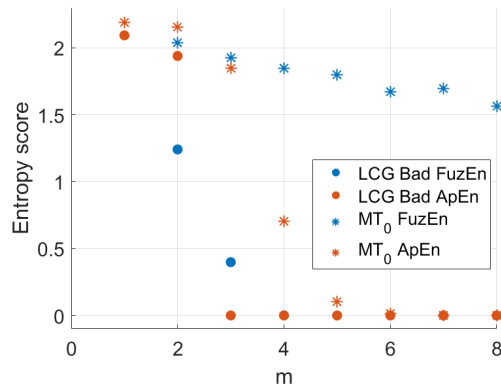


Fig. 7. LCG Bad and MT₀ entropy score evolution with m for analog signals.

The scores obtained by LCG Bad are hard to represent in Fig. 3 and Fig. 5 without losing visibility for the other PRNGs. Therefore, its results are presented in Tab. VII for analog signals and in Tab. VIII for binary signals.

TABLE VII
LCG BAD SCORES FOR ANALOG SIGNALS.

\mathcal{D}		ApEn	FuzEn
2.51	$m = 2$	1.939	1.241
	$m = 3$	0	0.398

TABLE VIII
LCG BAD SCORES FOR BINARY SIGNALS.

\mathcal{D}		ApEn	FuzEn
2.60	$m = 2$	0.692	0.690
	$m = 3$	0.685	0.677
	$m = 8$	0.367	0.381

ACKNOWLEDGMENTS

The authors would like to thank R. V. Ramos for the helpful and interesting discussion on disentropy and Stefano Giordano for useful feedback.

REFERENCES

- [1] Roel Maes. *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer, Berlin, Heidelberg, 2013.
- [2] Armin Babaei and Gregor Schiele. Physical Unclonable Functions in the Internet of Things: State of the Art and Open Challenges. *Sensors*, 19(14):3208, July 2019.
- [3] Fabio Pavanello, Ian O'Connor, Ulrich Rührmair, Amy C. Foster, and Dimitris Syvridis. Recent Advances in Photonic Physical Unclonable Functions. In *2021 IEEE European Test Symposium (ETS)*, pages 1–10, Bruges, Belgium, May 2021. IEEE.
- [4] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. In Peter Athanas, Dionisios Pnevmatikatos, and Nicolas Sklavos, editors, *Embedded Systems Design with FPGAs*, pages 245–267. Springer New York, New York, NY, 2013.
- [5] Ulrich Rührmair, Jan Sölter, and Frank Schnke. On the Foundations of Physical Unclonable Functions, 2009. Publication info: Published elsewhere. Unknown where it was published.
- [6] Mitsuru Shiozaki, Yohei Hori, and Takeshi Fujino. Entropy Estimation of Physically Unclonable Functions with Offset Error, 2020. Publication info: Preprint. MINOR revision.
- [7] Mohammed Al-Haidary and Qassim Nasir. Physically Unclonable Functions (PUFs): A Systematic Literature Review. In *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6, Dubai, United Arab Emirates, March 2019. IEEE.
- [8] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, N. Heckert, James Dray, San Vo, and Lawrence Bassham. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical Report NIST Special Publication (SP) 800-22 Rev. 1, National Institute of Standards and Technology, April 2010.
- [9] Cedric Marchand, Lilian Bossuet, Ugo Mureddu, Nathalie Bochar, Abdelkarim Cherkaoui, and Viktor Fischer. Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):97–109, January 2018.
- [10] Bryan T. Bosworth, Iskandar A. Atakhodjaev, Michael R. Kossey, Brian C. Grubel, Daniel S. Vresilovic, Jasper R. Stroud, Neil MacFarlane, Jesús Villalba, Najim Dehak, A. Brinton Cooper, Mark A. Foster, and Amy C. Foster. Unclonable photonic keys hardened against machine learning attacks. *APL Photonics*, 5(1):010803, January 2020.
- [11] Paul Jimenez, Raphael Cardoso, Maurício Gomes de Queiroz, Mohab Abdalla, Clément Zrounba, Ulrich Rührmair, Cédric Marchand, Xavier Letartre, and Fabio Pavanello. Photonic Physical Unclonable Function Based on Symmetric Microring Resonator Arrays. In *Frontiers in Optics + Laser Science 2023 (FiO, LS) (2023), paper JT4A.82*, page JT4A.82. Optica Publishing Group, October 2023.
- [12] Farhan Bin Tarik, Azadeh Famili, Yingjie Lao, and Judson D. Ryckman. Scalable and CMOS compatible silicon photonic physical unclonable functions for supply chain assurance. *Scientific Reports*, 12(1):15653, September 2022.
- [13] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, September 2002.
- [14] Ulrich Rührmair, Christian Hilgers, Sebastian Urban, Agnes Weiershäuser, Elias Dinter, Brigitte Forster, and Christian Jirauschek. Optical PUF reloaded. *Cryptology ePrint Archive*, 2013.
- [15] Apostolos Argyris, Stavros Deligiannidis, Evangelos Pikasis, Adonis Bogris, and Dimitris Syvridis. Implementation of 140 Gb/s true random bit generator based on a chaotic photonic integrated circuit. *Optics Express*, 18(18):18763–18768, August 2010. Publisher: Optica Publishing Group.
- [16] Kazusa Ugajin, Yuta Terashima, Kento Iwakawa, Atsushi Uchida, Takahisa Harayama, Kazuyuki Yoshimura, and Masanobu Inubushi. Real-time fast physical random number generator with a photonic integrated circuit. *Optics Express*, 25(6):6511–6523, March 2017. Publisher: Optica Publishing Group.
- [17] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948. Conference Name: The Bell System Technical Journal.
- [18] R. V. Ramos. Quantum and Classical Information Theory with Disentropy, January 2020. arXiv:1901.04331 [quant-ph].
- [19] G. S. Castro and R. V. Ramos. Enhancing eavesdropping detection in quantum key distribution using disentropy measure of randomness. *Quantum Information Processing*, 21(2):79, February 2022.
- [20] R. V. Ramos. Estimation of the Randomness of Continuous and Discrete Signals Using the Disentropy of the Autocorrelation. *SN Computer Science*, 2(4):254, July 2021.
- [21] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52(1-2):479–487, July 1988.
- [22] C. Tsallis. What are the Numbers that Experiments Provide. *Química Nova*, 1994.
- [23] G.B. Da Silva and R.V. Ramos. The Lambert-Tsallis W_q function. *Physica A: Statistical Mechanics and its Applications*, 525:164–170, July 2019.
- [24] R. V. Ramos. Disentropy of the Wigner function. *Journal of the Optical Society of America B*, 36(8):2244, August 2019.
- [25] Abdelkarim Cherkaoui, Lilian Bossuet, and Cedric Marchand. Design, Evaluation, and Optimization of Physical Unclonable Functions Based on Transient Effect Ring Oscillators. *IEEE Transactions on Information Forensics and Security*, 11(6):1291–1305, June 2016.
- [26] Miguel Garcia-Bosque, Adrián Pérez-Resca, Carlos Sánchez-Azqueta, Concepción Aldea, and Santiago Celma. Chaos-Based Bitwise Dynamical Pseudorandom Number Generator On FPGA. *IEEE Transactions on Instrumentation and Measurement*, 68(1):291–293, January 2019. Conference Name: IEEE Transactions on Instrumentation and Measurement.

- [27] S M Pincus. Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences*, 88(6):2297–2301, March 1991.
- [28] Andreas Holzinger, Christof Stocker, Manuel Bruschi, Andreas Auinger, Hugo Silva, Hugo Gamboa, and Ana Fred. On Applying Approximate Entropy to ECG Signals for Knowledge Discovery on the Example of Big Sensor Data. In Runhe Huang, Ali A. Ghorbani, Gabriella Pasi, Takahira Yamaguchi, Neil Y. Yen, and Beijing Jin, editors, *Active Media Technology*, pages 646–657, Berlin, Heidelberg, 2012. Springer.
- [29] Radhagayathri K. Udhayakumar, Chandan Karmakar, Peng Li, and Marimuthu Palaniswami. Effect of embedding dimension on complexity measures in identifying Arrhythmia. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6230–6233, Orlando, FL, USA, August 2016. IEEE.
- [30] Alfonso Delgado-Bonal, Alexander Marshak, Yuekui Yang, and Daniel Holdaway. Analyzing changes in the complexity of climate in the last four decades using MERRA-2 radiation data. *Scientific Reports*, 10(1):922, January 2020.
- [31] Alfonso Delgado-Bonal. Quantifying the randomness of the stock markets. *Scientific Reports*, 9(1):12761, September 2019.
- [32] Alfonso Delgado-Bonal and Alexander Marshak. Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. *Entropy*, 21(6):541, May 2019.
- [33] Weiting Chen, Zhizhong Wang, Hongbo Xie, and Wangxin Yu. Characterization of Surface EMG Signal Based on Fuzzy Entropy. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(2):266–272, June 2007.
- [34] Joshua S. Richman and J. Randall Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049, June 2000.
- [35] Hamed Azami, Peng Li, Steven E. Arnold, Javier Escudero, and Anne Humeau-Heurtier. Fuzzy Entropy Metrics for the Analysis of Biomedical Signals: Assessment and Comparison. *IEEE Access*, 7:104833–104847, 2019.
- [36] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.
- [37] S. K. Park and K. W. Miller. Random number generators: good ones are hard to find. *Commun. ACM*, 31(10):1192–1201, October 1988.
- [38] Kamalika Bhattacharjee and Sukanta Das. A search for good pseudo-random number generators: Survey and empirical studies. *Computer Science Review*, 45:100471, August 2022.
- [39] Derrick H. Lehmer. Mathematical models in large-scale computing units. *Ann. Comput. Lab.(Harvard University)*, 26:141–146, 1951.
- [40] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.
- [41] random — Generate pseudo-random numbers. Date Accessed: 03 September 2024, url: <https://docs.python.org/3/library/random.html>.
- [42] Control random number generator - MATLAB rng - MathWorks France. Version: MATLAB 2023b, url: <https://fr.mathworks.com/help/matlab/ref/rng.html>.
- [43] Makoto Matsumoto, Takuji Nishimura, Mariko Hagita, and Mutsuo Saito. CRYPTOGRAPHIC MERSENNE TWISTER AND FUBUKI STREAM/BLOCK CIPHER, 2005. Publication info: Published elsewhere. Unknown where it was published.
- [44] Koyi Lakshmi Prasad, T.Ch. Malleswara Rao, and V. Kannan. A Hybrid Semi-fragile Image Watermarking Technique Using SVD-BND Scheme for Tampering Detection with Dual Authentication. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 517–523, Bhimavaram, India, February 2016. IEEE.
- [45] Makoto Matsumoto and Yoshiharu Kurita. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation*, 2(3):179–194, July 1992.
- [46] Measure of regularity of nonlinear time series - MATLAB approximateEntropy - MathWorks France. Version: MATLAB 2023b, url: <https://fr.mathworks.com/help/predmaint/ref/approximateentropy.html>.
- [47] Sample autocorrelation - MATLAB autocorr - MathWorks France. Version: MATLAB 2023b, url: <https://fr.mathworks.com/help/econ/autocorr.html>.
- [48] Steven Kho Ang. [stevenang/randomness_testsuite](https://github.com/stevenang/randomness_testsuite), July 2024. original-date: 2017-12-31T15:55:33Z.