# Complexity Assessment of Analog Security Primitives 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 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 (PRGNs), Physical Unclonable Functions (PUFs).**

## I. Introduction

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

PUF security is based on the disorder, unpredictability, and randomness of the manufacturing processes.

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, Universite 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, IMEP-LAHC, 38000, Grenoble, France

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

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, therefore multiple CRPs (CRPs library). Hence, the PUF should fulfil some characteristics [3], [4]:

- Physical uniqueness: As explained before, the PUF should act as a fingerprint due to fabrication variations.
- 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, ideally uncountable. In that case, it should not be possible for an adversary to have access to a large portion of the CRPs dataset in a reasonable amount of time. 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 both within each individual response and between responses. 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. Moreover, 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 [5]. For these PUFs, the NIST test suite [6], especially suited for the analysis of binary data, can

be used to evaluate some of the statistical properties of the responses [7]. However, PUFs can also operate in the analog domain, especially in some recent implementations of optical and photonic PUFs [3].

Nevertheless, there is an important question to address when considering 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 [8]- [11]. 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 [12], the optical output of the PUF is a speckle pattern recorded on a $320 \times 240$ 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 [13], 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 [8] 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.

Our goal is therefore to find a way of evaluating the physical quality of a PUF design by looking at the complexity of its responses in the analog domain, directly. In that case, the Shannon entropy [14] used for binary data cannot be applied. Since one of the basic requirements is the absence of repeating patterns within or between responses, 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 [15], [16], but also to obtain a score based on the autocorrelation function [17]. 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 and afterwards we will compare them with the disentropy. Finally, its performance will be evaluated on PRNGs and PUF responses to assess whether this metric is a good candidate or not to evaluate the quality of security primitives working in the analog domain.

## 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 $\tau$. In [16], [17] the authors claim that this function can be used to measure some randomness present in the signal. In case of a continuous signal $s(t)$ in $\mathbb{R}$, function of time $t$, the autocorrelation function $R(\tau)$ is given by:

$$R(\tau) = \int_{-\infty}^{\infty} s(t)s^*(t-\tau)dt \tag{1}$$

For discrete signals, the autocorrelation function is given by:

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

With $k$ the considered lag (delay), $s_t$ the discrete signal value at time $t$, $\bar{s}$ the mean value of $s_t$ and $\sigma_0^2$ 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 in [17] 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 [14]:

$$H = -\sum_{i=1}^{n} p_i \log(p_i) \tag{3}$$

Eq. 3 is expressed in terms of a discrete set of probabilities $\{p_i\}$. We can link the definition of entropy in Eq. 3 with the Boltzmann-Gibbs entropy of thermodynamics as:

$$S = -k\sum_{i=1}^{W} p_i \ln(p_i) \tag{4}$$

with $W \in \mathbb{N}$ the total number of possible microscopic configurations, $p_i$ their associated probabilities, and $k$ the Boltzmann constant. If the entropy is measured in units of $k$ per nat (natural unit of information) and $p_i = 1/W$, Eq. 4 becomes:

$$S = k \ln W \tag{5}$$

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 multifractals systems with $q \in \mathbb{R}$ as [18]:

$$S_q = k\frac{1-\sum_i p_i^q}{q-1} \tag{6}$$

Eq. 6 tends to the Boltzmann-Gibbs entropy in the limit of $q \to 1$. Then, in 1994 he proposed a new way of interpreting the experimental measurements as $q-$expectation values [19]. In this work he defines the generalised logarithmic function as:

$$\ln_q(x) = \frac{x^{(1-q)}-1}{1-q} \ \forall x \in \mathbb{R}^+ \text{ and } q \neq 1 \tag{7}$$

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

$$e_q^x = (1+(1-q)x)^{\frac{1}{1-q}} \ \forall x/(1+(1-q)x \geq 0) \text{ and } q \neq 1 \tag{8}$$

With this new logarithm definition (see Eq. 7), it is possible to express the Tsallis $q$-entropy $S_q$ as in Eq. 9.

$$S_q = -k\sum_i p_i^q \ln_q(p_i) \tag{9}$$

Then, the authors introduced the Lambert function $W(z)$ [17], [20] to further extend the meaning of this definition of entropy. $W(z)$ is obtained by solving:

$$W(z)e^{W(z)} = z \tag{10}$$

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. 3, neglecting $k$ and defining $z = p_i$ it is possible to obtain a definition of the Boltzmann-Gibbs entropy:

$$S = -\sum_i p_i \ln(W(p_i)) - \sum_i p_i W(p_i) \tag{11}$$

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:

$$W_q(z)e_q^{W_q(z)} = z \tag{12}$$

By taking the Tsallis $q$-logarithm of Eq. 12:

$$\ln_q(z) = W_q(z) + ((1-q)W_q(z)+1)\ln_q(W_q(z)) \tag{13}$$

At last, a probability $p_i$ is inserted in Eq. 12 with $z = p_i$ and by using the Tsallis $q$-entropy of Eq. 9 neglecting $k$, one obtains [16], [17]:

$$\begin{aligned} S_q = &-\sum_i p_i^q W_q(p_i) - \sum_i p_i^q \ln_q[W_q(p_i)] \\ &-(1-q)\sum_i p_i^q W_q(p_i)\ln_q[W_q(p_i)] \end{aligned} \tag{14}$$

where the term:

$$D_q = \sum_i p_i^q W_q(p_i) \tag{15}$$

is the Tsallis q-disentropy. The authors define it also in the continuous case as:

$$D_q = \int_{-\infty}^{\infty} p^q(x) W_q(p(x))dx \tag{16}$$

Recall that the autocorrelation function can take values ranging from $-1$ to 1. Therefore, a function $W_q(R(\tau))$ 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:

$$W_2(z) = \frac{z}{z+1}, \ z > -1 \tag{17}$$

So, by replacing $z = p(x)$ in Eq. 15 and Eq. 16 one obtains a value for the disentropy in the continuous and discrete probability distributions:

$$D_2 = \int_{-\infty}^{\infty} \frac{p^3(x)}{p(x)+1}dx \tag{18}$$

$$D_2 = \sum_i \frac{p_i^3}{p_i+1} \tag{19}$$

The authors have shown that this metric can be used for functions that are not probability distributions. For example in the case of the autocorrelation function in [16], [17] and for the Wigner function in quantum mechanics in [21]. Therefore, in the case of autocorrelation function, they obtain the disentropies given by Eq. (20) for continuous signals and Eq. (21) for discrete ones.

$$D_2 = \int_{-\infty}^{\infty} \frac{R(\tau)^3}{R(\tau)+1}d\tau \tag{20}$$

$$D_2 = \sum_{k=1}^{N} \frac{r_k^3}{r_k+1} \tag{21}$$

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 $R(\tau) = \delta(\tau)$. Note that the disentropy is not defined for $R(\tau) = -1$ and $r_k = -1$ i.e, it is not defined in the case of perfect anti-correlation. In that case, the metric diverges.

## III. Other measures of complexity

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-rando generators generating binary signals [6] is commonly used in cryptography and security to evaluate the quality of security primitives [22], [23]. In this test suite the Approximate Entropy (ApEn) is used to measure the complexity and detect the presence of repeating patterns in

a binary signals. The ApEn has actually been developed by Steven M. Pincus in 1991 [24] for any kind of vector in $\mathbb{R}^N$. This metric is currently used in medicine for ECG and EMG signals [25]–[27], but also in other domains such as climate studies [28] or finance [29]. The full algorithm of ApEn can be found in [24], [30] 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. If a signal is completely repeatable, it remains repeatable by taking $m$ or $m+1$ 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 [27].

**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:
$\mathbf{x}(i) = [s(i), s(i+1), ..., s(i+m-1)] \ \forall i \in [1, n]$
Then compute:

$$C_i^m(r) = \frac{\#j \text{ such that } d[\mathbf{x}(i), \mathbf{x}(j) \leq r]}{n} \text{ with } j \in [1, n]$$

With $d$ a metric comparing two vectors:

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

Next, compute $\phi^m$:

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

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 the difference is smaller than $r$ the patterns are considered similar, therefore $r$ is usually a function of the standard deviation of the signal.

### 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 [27], [30], [31]. Therefore, other metrics emerged, for example the Sample Entropy (SampEn) developed by Richman and Moorman in 2000 [31] 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 $N$ [27], [32]. 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 [27], [32]. The Fuzzy Entropy (FuzEn) has been introduced [27] to overcome these problems by using the fuzzy sets theory developed by Zadeh in 1965 [33]. 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 $A$" [33]
In [27] the authors 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 [32]. 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.

## IV. Some PRNGs

### A. Linear congruential generator (LCG)

This generator produces a sequence of pseudo-randomised numbers based on linear recursions given by [34], [35]:.

$$x_{k+1} = ax_k + c \bmod M \tag{22}$$

$x_k$ being the sequence with $k \in \mathbb{N}$, $M > 0$ is the modulo, $x_0 \in [0, M[$ the seed, $a \in [0, M[$ the multiplier and $c \in [0, M[$ the increment, such that $x_0$, $a$, $c \in \mathbb{Z}_M$ [35]. 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 [36] and the linear congruential generator in 1958 by W. E. Thomson and A. Rotenberg [37].

In this study, we use Lehmer generators and linear congruential generators with parameters shown in Table I. Lehmer suggested to have $M$ as a Mersenne-prime number such that $M = 2^p - 1$ with $p$ a prime number. Note also that the seed $x_0$ should be a co-prime of $M$, henceforth we chose $x_0 = 1$. To obtain PRNGs of good quality, 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 [35].

This way of generating PRNGs allows us to create non-optimised PRNGs or bad PRNGs by changing the parameters $a$, $M$, and $c$. These PRNGs presented in Table I (LCG Bad, LCG 1, 2, 3, 4) will therefore be compared to the optimised ones using the metric 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.

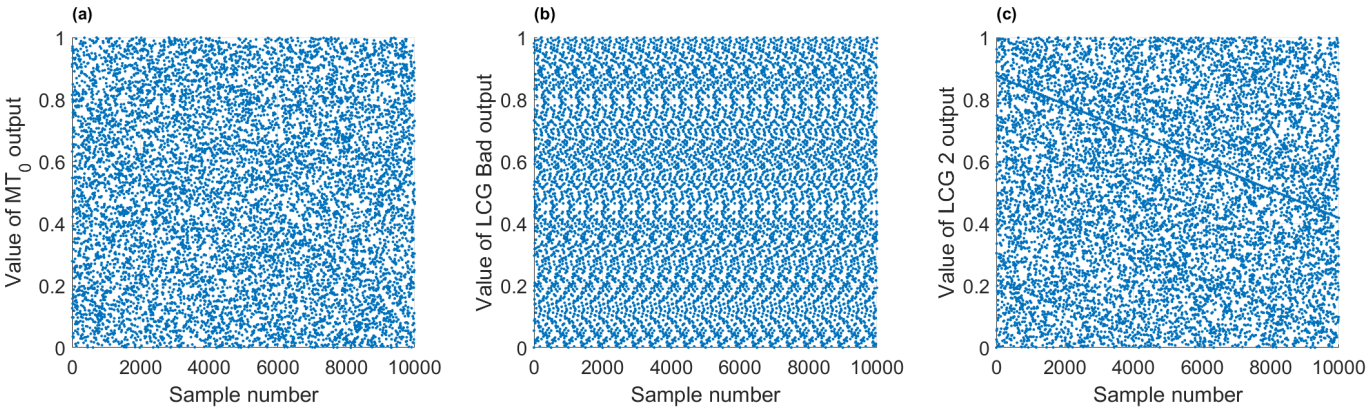


Fig. 1. Examples of normalised PRNGs output for 10000 samples (a) $MT_0$ (b) Bad LCG (c) LCG 2.

The parameters of LCG Bad have been chosen to create a bad PRNG with poor complexity 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 [34] | $2^{31}-1$ | 16807 | 0 |
| C++11 `minstd_rand` | $2^{31}-1$ | 48271 | 0 |
| GNU C Library `rand` | $2^{31}$ | 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 [38] is a common tool to generate sequences of random numbers. For example, it is used in python `random` module [39] and MATLAB™ `rand` function [40]. Furthermore, the MT algorithm can be used as a base for cryptographic cyphers, for example CryptMT [41], or image watermarking techniques [42]. MT uses the twisted generalized feedback back shift register (TGFSR) algorithm developed in [43]. The MT19937 algorithm described in [38] 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. Analog signals

For the ApEn metric, we utilised the MATLAB™ `approximateEntropy` function [44]. It is recommended to use the ApEn algorithm with $m = 2$ or $m = 3$ as well as $r$ between $0.1\sigma_0$ and $0.2\sigma_0$ [24], [30]. 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 [32] 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 and $r = 0.1253\sigma_0$ 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 [45] for each lag (delays) from lag $= 0$ (no delay) to lag $=$ length$(s) - 1$.
Next, the disentropy of the autocorrelation function ($D_2$) is obtained using Eq. 21 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 [17]. 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, a convergence study has been conducted with $MT_0$ and $m = 2$. Results for ApEn and FuzEn 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 the standard deviation to an order of magnitude of $10^{-2}$. On the other hand, the disentropy tends to 0 after few oscillations; for 1000 samples its standard deviation is $\sim 10^{-3}$ in order of magnitude.

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 a small standard deviation.

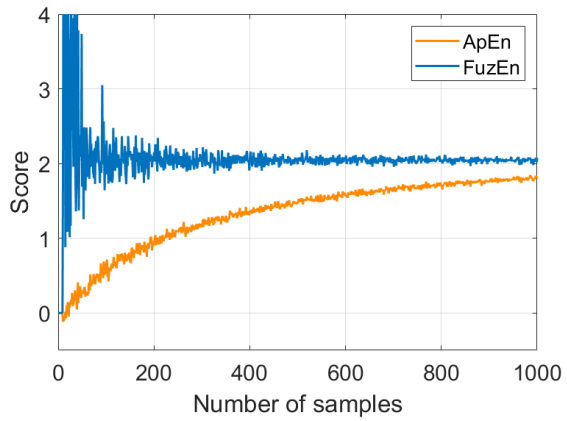


Fig. 2. Convergence study of ApEn and FuzEn for $MT_0$ and $m = 2$.

After 10000 samples, the order of magnitude of the variations for ApEn and FuzEn was $10^{-3}$, and $10^{-4}$ for 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 that the output of $MT_0$ does not exhibit observable patterns. On the other hand, LCG Bad has clear repeating patters and should have a very low score for all the metrics. LCG 2 does not necessarily have repeating patterns. However, straight lines translated on the y-axis can be observed for LCG 2 in its output.

Then, we decided to test different PRNGs for the different metrics and compared 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

| $\mathcal{D}_0$ | | $\text{ApEn}_0$ | $\text{FuzEn}_0$ |
|---|---|---|---|
| $4.80 \cdot 10^{-5}$ | $m = 2$ | 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 B.

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. However, one observes that LCG 2 obtains good results for the disentropy and the best score 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). Besides, it can be also seen that $m = 3$ obtains better contrast than $m = 2$ for ApEn and FuzEn.

By looking at Eq. (20) and Eq. (21), 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 $\tau_p$ 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 $j \in \mathbb{N}^*$.

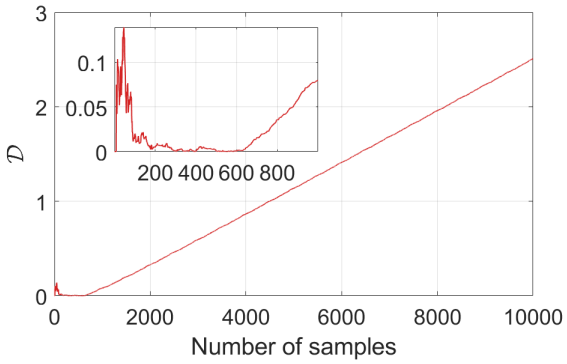


Fig. 4. Evolution of LCG Bad disentropy with $N$.

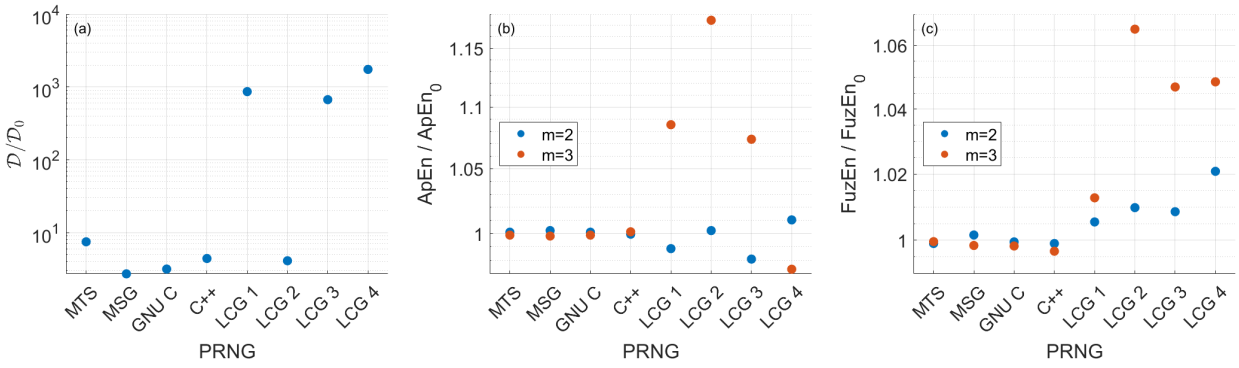


Fig. 3. Score ratio of PRNGs obtained from (a) $\mathcal{D} = |D_2 - 0.5|$, (b) ApEn, and (c) FuzEn compared to $MT_0$ scores for analog signals of Table. II.

### B. Binary signals

Since Eq. (1) and Eq. (2) are 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 [6] 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 $\text{MT}_0$ scores shown in Tab. III.

TABLE III
$\text{MT}_0$ SCORES FOR BINARY SIGNALS.

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

First, one observes that $\text{ApEn}_0$ and $\text{FuzEn}_0$ 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 $\log(2)$ [30].

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

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 [6]. Therefore, the ApEn algorithm from [46] has been used.

This algorithm obtains results for $m = 2, 3$ and 8 similar to the MATLAB™ `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 an analog PUF to be converted in binary. As a reference, 100 different responses with the MATLAB™ `rand` function were taken, then we concatenated them to obtain a series $C_0$ of size $N = 12800$. As no patterns are present in the responses, the concatenation $C_0$ should obtain a good score. 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 ∈ [2, n] do
    if s(k − 1) ≥ 0.9 then
        s(k) ← 0.1
    else if s(k − 1) ≤ 0.1 then
        s(k) ← 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 as if it was 100 different PUF instances and obtained the concatenations $C_i$ with $i \in [1, 100]$.
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 samples. Therefore, ApEn and FuzEn might perform better with $m + 1 = 2$ considering Algorithm 1.

TABLE IV
AVERAGE METRIC SCORE RATIO OF $C_i$ FOR THE REFERENCE AND TEST 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 $\sim 10^{-1}$ as 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 $62\%$.

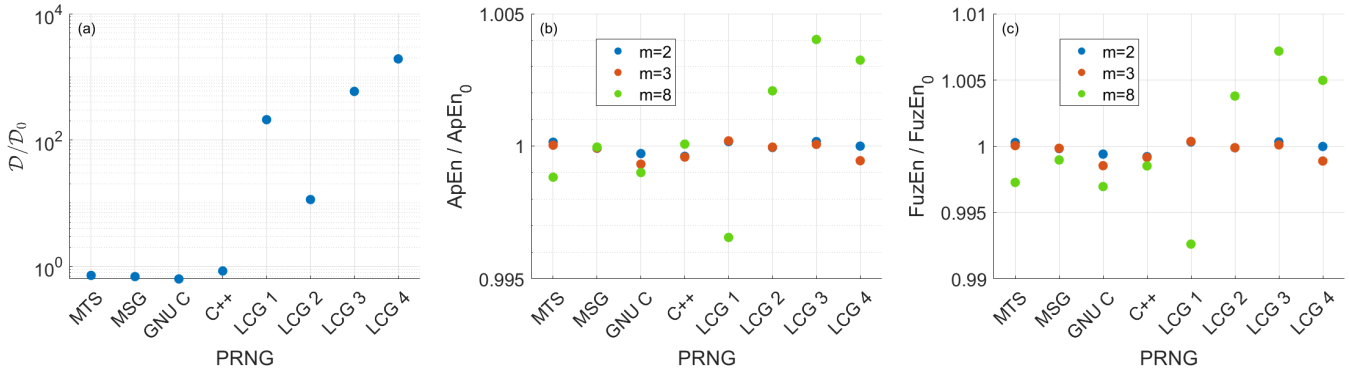


Fig. 5. Score ratio of PRNGs obtained from (a) $\mathcal{D} = |D_2 - 0.5|$ (b) ApEn (c) FuzEn compared to $\text{MT}_0$ scores for binary signals of Table. III.

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. 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 give $0\%$ for differences lower than $1\%$ and smaller than the standard deviation of signals.

TABLE V
AVERAGE SCORE RATIO BETWEEN REFERENCE CONCATENATIONS AND TEST CONCATENATION OF RESPONSES WITH THE [0.2, 0.1] PATTERN.

| 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 a photonic PUFs previously published [9]. Different PUF architectures were reported and here we focus our discussion on the architectures PUF 1 and PUF 3 because of their different NIST ApEn [46] scores. In this paper, the binary challenges are created using the MT algorithm and 150 instances of each architecture are simulated with fabrication variations. For each instance 100 responses of 128 bits are generated and 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 [46].

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 challenges | 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 challenges to generate unique instances because of its fabrication variations and architecture. It also degrades the quality of the challenges generated using the MT algorithm, while PUF 3 is not complex enough to have an impact on the challenges.

Given that $N^{\%}_{ApEn}$(PUF 1) $< N^{\%}_{ApEn}$(PUF 3) in the binary domain, one expects $\mathcal{D}$(PUF 1) $> \mathcal{D}$(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 $\mathcal{D}$(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 responses and show that the low $N^{\%}_{ApEn}$(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 responses of PUF 3 obtained a score very close to $\log(2)$ for both metrics. This means that ApEn and FuzEn are seeing the responses of PUF 3 as signals close to binary data.

As shown in Fig. 6, the normalised responses 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 binary challenges. Therefore, ApEn(PUF 1) $>$ ApEn(PUF 3) and FuzEn(PUF 1) $>$ FuzEn(PUF 3) even if PUF 1 has patterns and $N^{\%}_{ApEn}$(PUF 1) $<$ $N^{\%}_{ApEn}$(PUF 3).

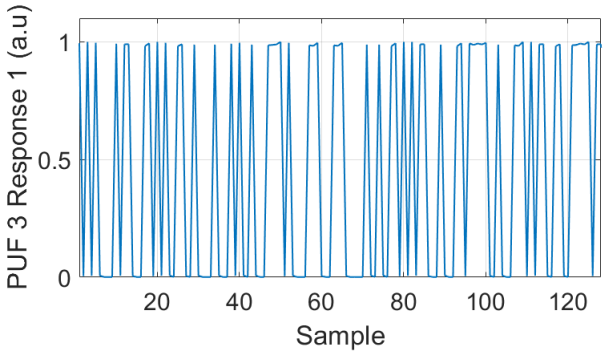


Fig. 6. First analog response of PUF 3 first instance (normalised).

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 [30], 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 is 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 responses of PUFs. However, 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.

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 with a small amount of 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 making it more interesting to use for signals that have to be converted into binary data.

## Appendix A
## Influence of $m$ on ApEn and FuzEn scores for analog signals

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 and Fig. 8, 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 $MT_0$.

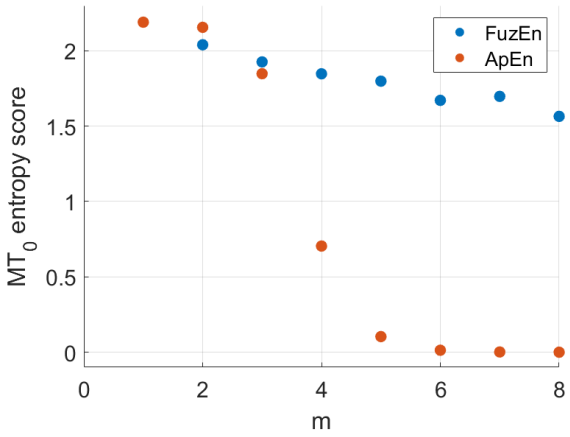


Fig. 7. $MT_0$ ApEn and FuzEn score evolution with $m$ for analog signals.

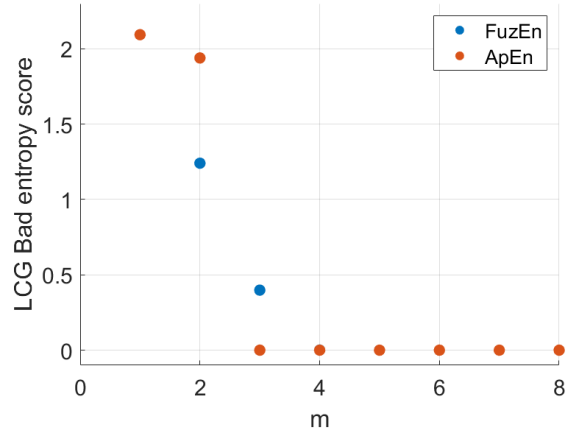


Fig. 8. LCG Bad ApEn and FuzEn score evolution with $m$ for analog signals.

## APPENDIX B
## LCG BAD SCORES

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] R. Maes, *Physically Unclonable Functions:Constructions, Properties and Applications*.

[2] A. Babaei and G. Schiele, *Physical Unclonable Functions in the Internet of Things: State of the Art and Open Challenges*, Sensors, vol. 19, no. 14, p. 3208, Jul. 2019, doi: 10.3390/s19143208.

[3] F. Pavanello, I. O'Connor, U. Ruhrmair, A. C. Foster, and D. Syvridis, *Recent Advances in Photonic Physical Unclonable Functions*, in 2021 IEEE European Test Symposium (ETS), Bruges, Belgium: IEEE, May 2021, pp. 1–10. doi: 10.1109/ETS50041.2021.9465434.

[4] A. Maiti, V. Gunreddy, and P. Schaumont, *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*, in Embedded Systems Design with FPGAs, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds., New York, NY: Springer New York, 2013, pp. 245–267. doi: 10.1007/978-1-4614-1362-2_11.

[5] M. Al-Haidary and Q. Nasir, *Physically Unclonable Functions (PUFs): A Systematic Literature Review*, in 2019 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates: IEEE, Mar. 2019, pp. 1–6. doi: 10.1109/ICASET.2019.8714431.

[6] A. Rukhin et al., *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*.

[7] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, *Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF*, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 37, no. 1, pp. 97–109, Jan. 2018, doi: 10.1109/TCAD.2017.2702607.

[8] B. T. Bosworth et al., *Unclonable photonic keys hardened against machine learning attacks*, APL Photonics, vol. 5, no. 1, p. 010803, Jan. 2020, doi: 10.1063/1.5100178.

[9] P. Jimenez et al., *Photonic Physical Unclonable Function Based on Symmetric Microring Resonator Arrays*, Oct. 2023, doi: 10.5281/ZENODO.8436663.

[10] D. Dermanis, A. Bogris, P. Rizomiliotis, and C. Mesaritakis, *Photonic Physical Unclonable Function Based on Integrated Neuromorphic Devices*, J. Lightwave Technol., vol. 40, no. 22, pp. 7333–7341, Nov. 2022, doi: 10.1109/JLT.2022.3200307.

[11] F. B. Tarik, A. Famili, Y. Lao, and J. D. Ryckman, *Scalable and CMOS compatible silicon photonic physical unclonable functions for supply chain assurance*, Sci Rep, vol. 12, no. 1, p. 15653, Sep. 2022, doi: 10.1038/s41598-022-19796-z.

[12] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, *Physical One-Way Functions*, Science, vol. 297, no. 5589, pp. 2026–2030, Sep. 2002, doi: 10.1126/science.1074376.

[13] U. Rührmair, C. Hilgers, S. Urban, A. Weiershäuser, E. Dinter, B. Forster, and C. Jirauschek (2013). Optical pufs reloaded. Cryptology ePrint Archive.

[14] C. E. Shannon, *A mathematical theory of communication*, in The Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, July 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[15] R. V. Ramos, *Quantum and Classical Information Theory with Disentropy*.

[16] G. S. Castro and R. V. Ramos, *Enhancing eavesdropping detection in quantum key distribution using disentropy measure of randomness*, Quantum Inf Process, vol. 21, no. 2, p. 79, Feb. 2022, doi: 10.1007/s11128-022-03422-y.

[17] R. V. Ramos, *Estimation of the Randomness of Continuous and Discrete Signals Using the Disentropy of the Autocorrelation*, SN COMPUT. SCI., vol. 2, no. 4, p. 254, Jul. 2021, doi: 10.1007/s42979-021-00666-w.

[18] C. Tsallis, *Possible generalization of Boltzmann-Gibbs statistics*, J Stat Phys, vol. 52, no. 1–2, pp. 479–487, Jul. 1988, doi: 10.1007/BF01016429.

[19] Tsallis, C. (1994). *What are the numbers that experiments provide.* Quimica Nova, 17(6), 468-471.

[20] G. B. Da Silva and R. V. Ramos, *The Lambert–Tsallis Wq function*, Physica A: Statistical Mechanics and its Applications, vol. 525, pp. 164–170, Jul. 2019, doi: 10.1016/j.physa.2019.03.046.

[21] R. V. Ramos, *Disentropy of the Wigner function*, J. Opt. Soc. Am. B, vol. 36, no. 8, p. 2244, Aug. 2019, doi: 10.1364/JOSAB.36.002244.

[22] M. Garcia-Bosque, A. Perez-Resa, C. Sanchez-Azqueta, C. Aldea, and S. Celma, *Chaos-Based Bitwise Dynamical Pseudorandom Number Generator On FPGA*, IEEE Trans. Instrum. Meas., vol. 68, no. 1, pp. 291–293, Jan. 2019, doi: 10.1109/TIM.2018.2877859.

[23] A. Cherkaoui, L. Bossuet, and C. Marchand, *Design, Evaluation, and Optimization of Physical Unclonable Functions Based on Transient Effect Ring Oscillators*, IEEE Trans.Inform.Forensic Secur., vol. 11, no. 6, pp. 1291–1305, Jun. 2016, doi: 10.1109/TIFS.2016.2524666.

[24] S. M. Pincus, *Approximate entropy as a measure of system complexity.*, Proc. Natl. Acad. Sci. U.S.A., vol. 88, no. 6, pp. 2297–2301, Mar. 1991, doi: 10.1073/pnas.88.6.2297.

[25] A. Holzinger et al., *On Applying Approximate Entropy to ECG Signals for Knowledge Discovery on the Example of Big Sensor Data*, in Active Media Technology, vol. 7669, R. Huang, A. A. Ghorbani, G. Pasi, T. Yamaguchi, N. Y. Yen, and B. Jin, Eds., in Lecture Notes in Computer Science, vol. 7669. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 646–657. doi: 10.1007/978-3-642-35236-2_64.

[26] R. K. Udhayakumar, C. Karmakar, P. Li, and M. 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), Orlando, FL, USA: IEEE, Aug. 2016, pp. 6230–6233. doi: 10.1109/EMBC.2016.7592152.

[27] W. Chen, Z. Wang, H. Xie, and W. Yu, *Characterization of Surface EMG Signal Based on Fuzzy Entropy*, IEEE Trans. Neural Syst. Rehabil. Eng., vol. 15, no. 2, pp. 266–272, Jun. 2007, doi: 10.1109/TNSRE.2007.897025.

[28] A. Delgado-Bonal, A. Marshak, Y. Yang, and D. Holdaway, Analyzing changes in the complexity of climate in the last four decades using MERRA-2 radiation data, Sci Rep, vol. 10, no. 1, p. 922, Jan. 2020, doi: 10.1038/s41598-020-57917-8.

[29] A. Delgado-Bonal, *Quantifying the randomness of the stock markets*, Sci Rep, vol. 9, no. 1, p. 12761, Sep. 2019, doi: 10.1038/s41598-019-49320-9.

[30] A. Delgado-Bonal and A. Marshak, *Approximate Entropy and Sample Entropy: A Comprehensive Tutorial*, Entropy, vol. 21, no. 6, p. 541, May 2019, doi: 10.3390/e21060541.

[31] J. S. Richman and J. R. Moorman, *Physiological time-series analysis using approximate entropy and sample entropy*, American Journal of Physiology-Heart and Circulatory Physiology, vol. 278, no. 6, pp. H2039–H2049, Jun. 2000, doi: 10.1152/ajpheart.2000.278.6.H2039.

[32] H. Azami, P. Li, S. E. Arnold, J. Escudero, and A. Humeau-Heurtier, *Fuzzy Entropy Metrics for the Analysis of Biomedical Signals: Assessment and Comparison*, IEEE Access, vol. 7, pp. 104833–104847, 2019, doi: 10.1109/ACCESS.2019.2930625.

[33] L. A. Zadeh, *Fuzzy sets*, Information and Control, vol. 8, no. 3, pp. 338–353, Jun. 1965, doi: 10.1016/S0019-9958(65)90241-X.

[34] S. K. Park and K. W. Miller, *RANDOM NUMBER GENERATORS: GOOD ONES ARE HARD TO FIND*, vol. 31, no. 10, 1988.
[35] K. Bhattacharjee and S. Das, *A search for good pseudo-random number generators: Survey and empirical studies*, Computer Science Review, vol. 45, p. 100471, Aug. 2022, doi: 10.1016/j.cosrev.2022.100471.
[36] D. H. Lehmer, *Mathematical models in large-scale computing units*, Ann. Comput. Lab.(Harvard University), vol. 26, pp. 141–146, 1951.
[37] W. E. Thomson, *A modified congruence method of generating pseudo-random numbers*, The Computer Journal, vol. 1, no. 2, pp. 83–83, 1958.
[38] M. Matsumoto and T. Nishimura, *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Trans. Model. Comput. Simul., vol. 8, no. 1, pp. 3–30, Jan. 1998, doi: 10.1145/272991.272995.
[39] *random — Generate pseudo-random numbers*, Python documentation. Accessed: Jan. 05, 2024. [Online]. Available: https://docs.python.org/3/library/random.html
[40] *Control random number generator - MATLAB rng - MathWorks France*. Accessed: Jan. 05, 2024. [Online]. Available: https://fr.mathworks.com/help/matlab/ref/rng.html
[41] M. Matsumoto, T. Nishimura, M. Hagita, and M. Saito, *CRYPTOGRAPHIC MERSENNE TWISTER AND FUBUKI STREAM/BLOCK CIPHER*.
[42] K. L. Prasad, T. Ch. M. 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), Bhimavaram, India: IEEE, Feb. 2016, pp. 517–523. doi: 10.1109/IACC.2016.102.
[43] M. Matsumoto and Y. Kurita, *Twisted GFSR generators*, ACM Trans. Model. Comput. Simul., vol. 2, no. 3, pp. 179–194, Jul. 1992, doi: 10.1145/146382.146383.
[44] *Measure of regularity of nonlinear time series - MATLAB approximateEntropy - MathWorks France*. Accessed: Jan. 05, 2024. [Online]. Available: https://fr.mathworks.com/help/predmaint/ref/approximateentropy.html
[45] *Sample autocorrelation - MATLAB autocorr - MathWorks France*. Accessed: Jan. 05, 2024. [Online]. Available: https://fr.mathworks.com/help/econ/autocorr.html
[46] GitHub, NIST Randomness Testsuit, https://github.com/stevenang/randomness_testsuite