Development of the Unified Security Requirements of PUFs During the Standardization Process

Nicolas Bruneau, Jean-Luc Danger, Adrien Facon, Sylvain Guilley, Soshi Hamaguchi, Yohei Hori, Yousung Kang, Alexander Schaub

Development of the unified security requirements of PUFs during the standardization process

Nicolas Bruneau[1], Jean-Luc Danger[2], Adrien Facon[1,3], Sylvain Guilley[1,2,3], Soshi Hamaguchi[4], Yohei Hori[5], Yousung Kang[6], and Alexander Schaub[2]

[1] Secure-IC S.A.S., 15 Rue Claude Chappe, Bât. B, 35 510 Cesson-Sévigné, France
[2] LTCI, Télécom ParisTech, Université Paris-Saclay, 75 013 Paris, France
[3] École Normale Supérieure, France
[4] Cosmos Corporation, Japan
[5] National Institute of Advanced Industrial Science and Technology (AIST), Japan
[6] ETRI, Korea

Abstract. This paper accounts for some scientific aspects related to the international standardization process about physically unclonable functions (PUFs), through the drafting of ISO/IEC 20897 project. The primary motivation for this standard project is to structure and expand the market of PUFs, as solutions for non-tamperable electronic chips identifiers.
While drafting the documents and discussing with international experts, the topic of PUF also gained much maturity. This article accounts how scientific structuration of the PUF as a field of embedded systems security has been emerging as a byproduct. First, the standardization has allowed to merge two redundant security requirements (namely *diffuseness* and *unpredictability*) into one (namely *randomness*), which in addition better suits all kinds of PUFs. As another contribution, the standardization process made it possible to match unambiguous and consistent tests with the security requirements. Furthermore, the process revealed that tests can be seen as *estimators* from their theoretic expressions, the so-called *stochastic models*.

1 Introduction

Security is an enabler for the expansion of our digital society. The trend is all the more important with the settling of Internet of Things (IoT). Communicating parties become not only humans but also machines, and without surprise, there are today more machines than humans on our planet. Typically, each human uses many machines to service him personally. Besides, some mutualized infrastructures also leverage on IoT or Machine-To-Machine (M2M) objects.

In this context, it is crucial to attest of the identity of parties. Indeed, communicating with unidentified IoT objects can lead to malware infection and can of course be subverted by attackers to build attacks. Most simple attacks consist in masquerade, that is malevolent situation whereby one device pretends it is another one. These vulnerabilities are the premise of building botnets.

Beyond these attacks, which target individual devices, the issue propagates to service providers. With the possibility of such device impersonation, the cloud operators happen to exchange and receive information from unreliable sources. Consequently, the information manipulated by the cloud is thus unreliable too. However, it is known that big data market can only thrive provided the collected data can be relied on. This is captured by the attribute "veracity" of big data characterization in 5Vs (Volume, Velocity, Variety, Veracity, and Value).

Therefore, trustworthy identification is paramount. Besides, it is also important to safeguard other immaterial assets, such as data confidentiality (for privacy concerns) and program authentication (in the case of software updates). All these properties are part and parcel of the scope of *security* requirements.

Security technologies. Security is thus an important aspect of digital communications. It must thus be addressed rigorously. Building security from a technical point of view requires the collaboration of various technologies, including:

- cryptographic algorithms, which implement building blocks for confidentiality (through encryption), integrity (through hash functions) and authentication (through message authentication codes and/or digital signatures),
- sensitive security parameters[1] management, which consists in their generating, in ensuring their protection while they are in memory (in use or backed-up), and in warranting their erasure when they shall be disposed of,
- proving the correctness of the implementation, since bugs are known to be devastating in their exploitation [13], included but not limited by cyber-attacks,
- checking for the absence of side-channels in cryptographic implementations, which can be revealed by timing biases in micro-architectural behaviors, or by effects related to reliability issues in Double Data Rate (DDR) modules [15,17] or in FLASH memories [3] (collectively being known as *RowHammer* attacks).

The complexity of implementing such security technologies has led to many attacks, which leverage on inconsistent interaction between primitives. Typical examples are authenticated encryption using AES-CBC with AES-based CMAC, which natively does not provide authentication, or compression oracles, whereby the combination of compression and use of MAC-then-encrypt paradigm makes it possible to extract plaintext bytes just by analyzing the size of the encrypted data.

[1] Sensitive security parameters (SSPs) consist either in critical security parameters (CSPs) or public security parameters (PSPs). CSPs shall be kept secret. Examples are nonces, long-term and ephemeral keys. PSPs are public, but shall not be chosen. Examples are initialization vectors.

Specificities of standardization in security. Because of this complexity and in order to prevent the thrive of so numerous attacks, security mechanisms have been standardized. Standardization ensures that the design and the choice of cryptographic primitives are done without errors (trivially) leading to attacks.

But standardization goes beyond simplification of choice regarding security algorithms. It also simplifies the understanding of the whole solution, thereby provided a more clear vision of the design rationale. This is important because the adoption of security can be considered as trust delegation into a security standard. Therefore, standardized solutions give more confidence in the fact that no security issue will happen. As a matter of facts (recall Snowden affair), one of the risks is about backdoors in security protocols. It is indeed cynical to invest in a security solution which actually helps attackers hack your devices.

Another aspect related to security is that the Devil lays in the details. In particular, unintentional vulnerabilities might be injected while implementing cryptographic solutions. Typically, side-channels (i.e., non-functional albeit adversarially observable behaviors) might leak information about the secrets. This includes timing, cache hit/miss patterns, power consumption, electromagnetic field radiation, etc.

Scope of the paper. In this article, we survey the role of standardization for Physically Unclonable Functions (PUFs) as a key generation technology. Such technology is a cornerstone, since cryptography bases itself on the principle that all the design can be made public, whilst the only secrecy lays in the key, according to a doctrine which is known as the Auguste Kerckhoffs law.

In particular, we detail the service which is expected from the PUFs, derive security requirements, and expose ways to test and evaluate them. These aspects are part and parcel of the mandatory normative features of project ISO/IEC 20897.

Outline and contributions. The rest of the paper is organized as follows. First of all, we provide the reader with a high-level description of PUFs. This topic is addressed in Sec. 2. In this section, we stress the progress which has been made during standardization process concerning the definition of independent features of PUFs of seemingly different nature. Our main contribution lays in Sec. 3. We here account for a remarkable progress in the modelization of PUFs, namely the duality between challenges and responses. Based on this achievement, we enunciate clearly in this Sec. 3 the unified security requirements for PUFs. Eventually, we explain in Sec. 4 how to test and/or evaluate whether or not security requirements are met, and how, in a given device. Conclusions are in Sec. 5.

2 Description of a PUF

A PUF is a function implemented in a device that is produced or configured with the security objective that random fluctuations in the production process lead to different behaviours and are hard to reproduce physically.

This means that a PUF generates data which do not result from a read access in a Non Volatile Memory (NVM). Indeed, no data are stored in the PUF during its whole life cycle.

2.1 Towards a standard related to PUFs

The international standards of PUFs are discussed in ISO/IEC 20897-1 and 20897-2 [12]: the former deals with the security requirements and the latter deals with the test and evaluation methods of PUFs. The documents of the standards are edited within ISO/IEC JTC 1/SC 27/WG 3. Notice that the contents of this paper concerns only the authors, and is not endorsed by ISO/IEC.

There have been many PUF variants reported so far [2,1], and excerpt of technologies taken from a selection of published academic papers is reported below:

- Optical PUF [20],
- Coating PUF [28],
- SRAM PUF [9],
- Glitch PUF [27],
- Arbiter PUF [19,6],
- Loop PUF [4],
- Memory contention PUF [8],
- Oxide rupture PUF [29],
- Transistor voltage threshold [26].

All those very different structures aim at providing a similar service to the final users. Namely, they all implement a function which is hard if not impossible to copy. All those objects are covered by the scope of ISO 20897-1/-2 drafts.

2.2 Parameters of PUF

A PUF is, by definition, a function. The input parameters are termed challenges, and output parameters are termed responses.

In this paper we focus on so-called silicon PUFs. Their challenges and responses are digital, i.e., consist in bitstrings. Precisely, PUFs obtain digital(ized) responses by amplifying analog signals from physical properties of tiny imperfections of the device implementation. This property allows PUFs to be unclonable.

The goal of standardization is to provide portable metrics along with tests and evaluation methods which are comparable whilst they embrace a wide array of very different PUF technologies. The effort to be carried out is thus to abstract away all peculiarities inherent to various PUF technologies. The method to do so is to think about the services expected by a PUF in general, without referring to any implementation specific choice.

It is customary in the literature to classify PUFs as either *weak* or *strong*. A weak PUF has no input, but usually a (unique) response of large size. A strong PUF has many inputs, but usually short responses in size. As the standardization process has revealed, this classification is a bit caricatural. A softer classification will be introduced in Sec. 3.1.

We provide here-after some useful notations.

The responses from multiple PUFs are arranged into a cube as Fig. 1 shows. The repetitive calls to a PUF are illustrated in Fig. 2. The single small cube describes a 1-bit response from a PUF. The three axes of the cube and the time are described hereafter, as directions:

- **direction B**: "#bit" shows the bit length of the response obtained from a single challenge. In a 1-bit response PUF, e.g., arbiter PUF, the dimension B collapses.
- **direction C**: "#challenge" shows the number of different challenges given to a PUF. In a no-challenge PUF (or, more rigorously, a one-challenge PUF, see discussion in Sec. 3.1), e.g., SRAM PUF, the dimension C collapses.
- **direction D**: "#PUF" shows the number of different PUF devices under test.
- **direction T**: "#query" shows the number of query iterations under the fixed PUF device and challenge.

2.3 Use-cases of PUFs

By definition, a PUF is a "non-stored sensitive security parameter". Therefore, PUFs can be used to fulfill different needs where non-tamperability is a strong requirement.

A first use-case consists in using the value of the PUF as a private data, which can play the role of some secret key, known by nobody (neither by the designer nor by the tester nor by the firmware developer, etc.). Therefore, such solution solves the problem of key management, as each device handles its own key.

A second use-case consists in device unfalsifiable identification. The PUF is used as a public identifier, which shall therefore be non-tamperable.

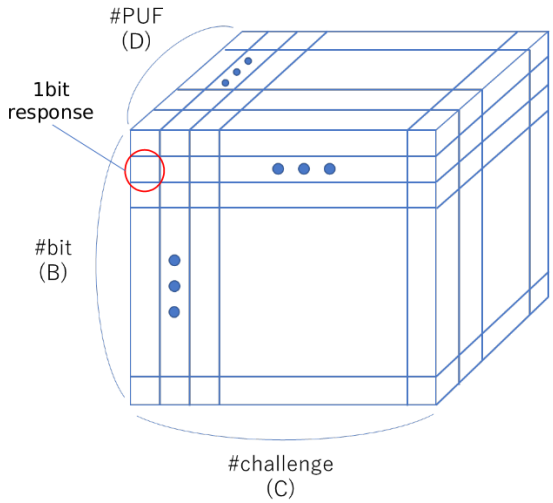


Fig. 1. The three dimensions involved in the PUFs entropy metrics

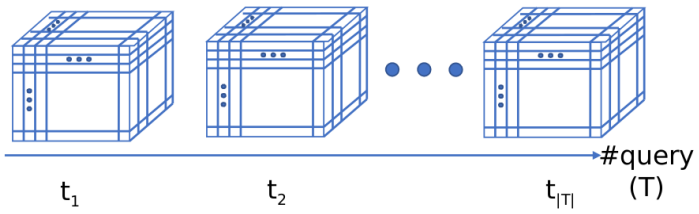


Fig. 2. Illustration of the PUF repetitive usage, involved in the PUF steadiness metric

A third use-case consists in device authentication. This application is demanding more security than the plain identification, as the protocol shall be protected against replay attacks. Therefore, in this use-case, a subset of PUF's challenges and responses is saved and this whitelist enables future interactive attestation that the device is genuine. The scenario is referred to as challenge-response protocol (CRP).

Yet a forth use-case consists simply in generating randomness: this can consist in the replacement of an unavailable true random number generator, or for a source of random numbers, which can be reseeded (as if `srand(0)` is called before a sequence of several `rand()` in a software implementation). For this purpose, a list of selected challenges is queried, and the associated responses are taken as random source unique to the device under consideration.

2.4 PUF life-cycle

The usual steps involved in the life-cycle of a PUF are depicted in Fig. 3 for the exemplar use-cases presented in Sec. 2.3. The design must first be realized, and then it is sent to the silicon foundry for fabrication. Most PUFs are becoming unique as soon as they are fabricated. However, some PUFs are prepared by the foundry, but must be "revealed" later on (like a photographic picture after exposition). For example, the oxide rupture PUF [29] is not operational right after fabrication, but becomes operational subsequent to some locking process. Besides, some PUFs are not steady by nature, and for them to be of practical use, their overall steadiness must be improved by removing those entropy sources which are the less steady. This step consists in so-called helper data extraction. At this stage, the PUF is ready to be used, except for the authentication protocol. Indeed, it requires first to register all challenge-response pairs which will be used later on to attest of the PUF identity. All those steps are carried out in a safe environment, typically the PUF designers factory, hence no risk of attacks.

From this stage on, the PUFs are deployed in their *operational environment*: responses are gathered, to either build a key, an identifier (ID), a response to compare to formerly registered challenge-response pairs in the case of authentication protocols, or responses for static entropy gathering. Notice that most of the PUFs require a response correction step, which attempts to remove errors in the responses when they are not reliable enough.

Notice that Fig. 3 presents optional steps (which can be removed for some instances of PUF if they do not jeopardize security requirements) between braces.

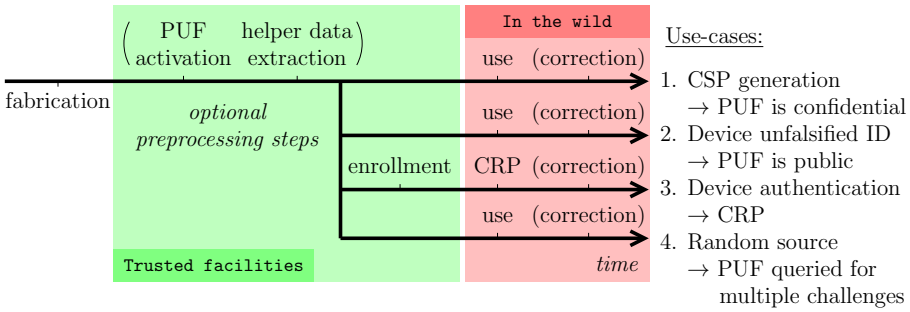


Fig. 3. Life cycle of a PUF in four use-cases

3 Security requirements

One of the virtues of the standardization process has been to clarify and to unite the vision of PUFs, as diverse as they are. In particular, we account in Sec. 3.1 for the symmetrical role of challenges and responses, and then introduce in Sec. 3.2 a consistent list of security requirements which apply equally to any PUF avatars.

3.1 Duality between challenges and responses

It has been highlighted earlier in Sec. 2.2 that the PUFs are varied in terms of number of challenges and responses. However, there is a way to unify this notion, by analyzing the tradeoff between number of challenges and of responses.

Actually, we would like to introduce a transformation from a PUF with a given number of challenge and response bits into a PUF with other such parameters. In this respect, we highlight a broader interpretation which allows to derive a **duality** notion between response number of bits and challenge number of bits. It is indeed possible to trade some response bits for some challenge bits. Let us introduce here-after the transformation.

Sub-selection to trade large response bitwidth for short challenge bitwidth. Let us model an SRAM by a module with no input (or equivalently, an input of 0 bitwidth), and an output word on 2^n bits; notice that memories of maximum capacity always contain a number of bits which is a power of 2, since the n-bit address is fully decoded. Thus, the address bitwidth is zero[2] and the data read out has a bitwidth of 2^n bits. The SRAM can also be seen as two memories of 2^{n-1} bits each, or four memories memories of 2^{n-2} bits each, etc. Those tradeoffs are illustrated for $n=8$ in Fig. 4. The table 1 explains how such tradeoffs work in general.

Actually, most SRAM modules are already partitioned in banks, hence the contents of Fig. 4 is fairly natural. For instance, for small sizes, it is not uncommon to have so-called *cuts of RAM* of 32 bytes, that is 5-bit addresses and data out on 8 bits. This is the equivalent of a memory with no address and size $8\times 2^5 = 2^8$, i.e., with 256 data (single bit) outputted.

A similar situation is encountered for Look-up-Tables (LUTs) or Block RAMs (BRAMs) in FPGAs, which can have multiple form factors. See the example depicted in Fig. 5 of a 4-LUT (memory of 4 input bits and 1 output bit) which is equivalent to two 3-LUTs.

[2] To be accurate, there is thus $2^0 = 1$ input, which is consequently constant.

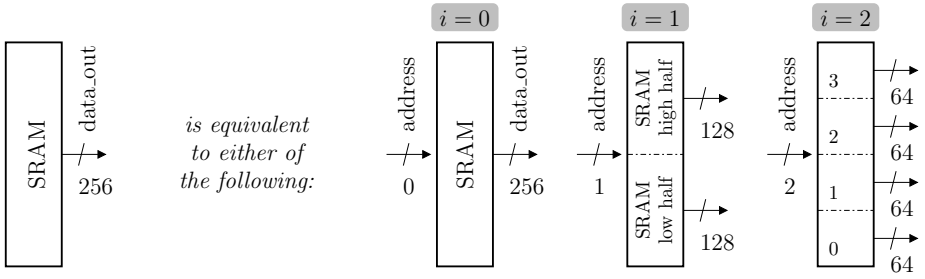


Fig. 4. Transformation of 0-bit address (challenge) with 2^n-bit unique output data (response) into equivalent i-bit challenges with (2^{n-i})-bit responses, for $i \in \{0,1,2\}$ and for $n=8$

Table 1. Tradeoff between challenges (addresses) and response number of bits per input for an SRAM-PUF

Number of inputs	Number of outputs
$1=2^0$	2^n
$2=2^1$	2^{n-1}
$4=2^2$	2^{n-2}
$\vdots$	$\vdots$
2^n	$1=2^0$

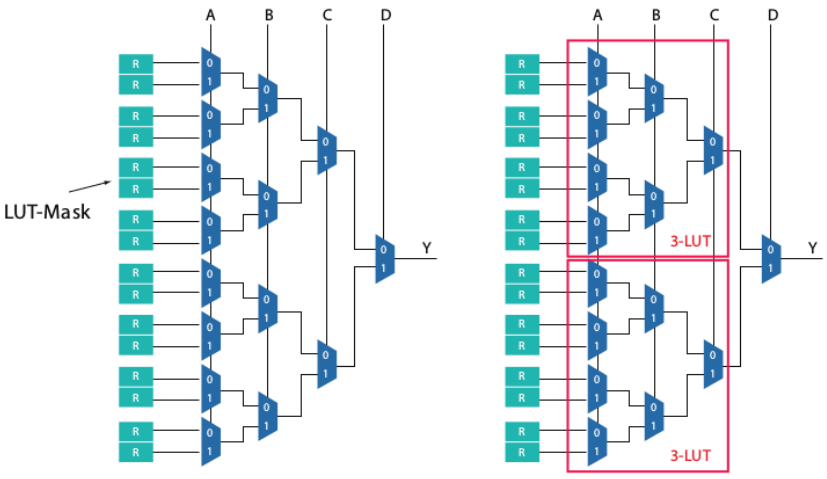
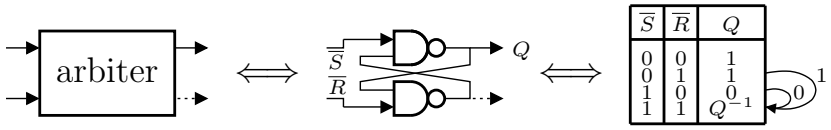


Fig. 5. Architecture of a 4-LUT, equivalent to two 3-LUTs (diagram courtesy of Altera [5])

Repeating to trade numerous challenges for short responses. Let us model an arbiter-PUF [19] by a module with an input of n bits and a single-bit response, which can be seen as a word on 1 ($=2^0$) bit. The challenge is seen as an integer c comprised between 0 and 2^n-1 and the response r is seen as an integer comprised between 0 and 1. Challenges drive conditional switches ($\overset{1}{\underset{0}{f}}$), and responses are produced by an arbiter, as fair as possible, e.g., using an RS-latch:



The figure 6 for $i \in \{0,1,2\}$ shows how to turn this mono-bit response PUF into an equivalent PUF with challenges on $(n-i)$ bits and responses on 2^i bits. Notice that this figure represents a *sequential protocol*, where the same n-bit arbiter-PUF is called 2^i times in a row. The resulting quantitative tradeoff is given in Tab. 2.

Table 2. Tradeoff between challenges (addresses) and response number of bits per input for an arbiter-PUF

Number of inputs	**Number of outputs**
2^n	$1=2^0$
2^{n-1}	$2=2^1$
2^{n-2}	$4=2^2$
$\vdots$	$\vdots$
$1=2^0$	2^n

Summary of the duality between response number of bits and the number of challenges. For both SRAM-PUF and arbiter-PUF, the trade-off between the number of bits in the challenge and in the response are linked with the following **conservation rule**:

$$\boxed{\#\text{challenge} \times \#\text{response} = \text{volume} = 2^n.}$$

The constant 2^n is an intrinsic property of the PUF. It stays unchanged as its "aspect ratio" is changing in terms of number of bits of inputs (challenges) traded for number of bits of output (responses). This highlights that the constant 2^n can be referred to as the **volume** (or the **surface**) of the PUF.

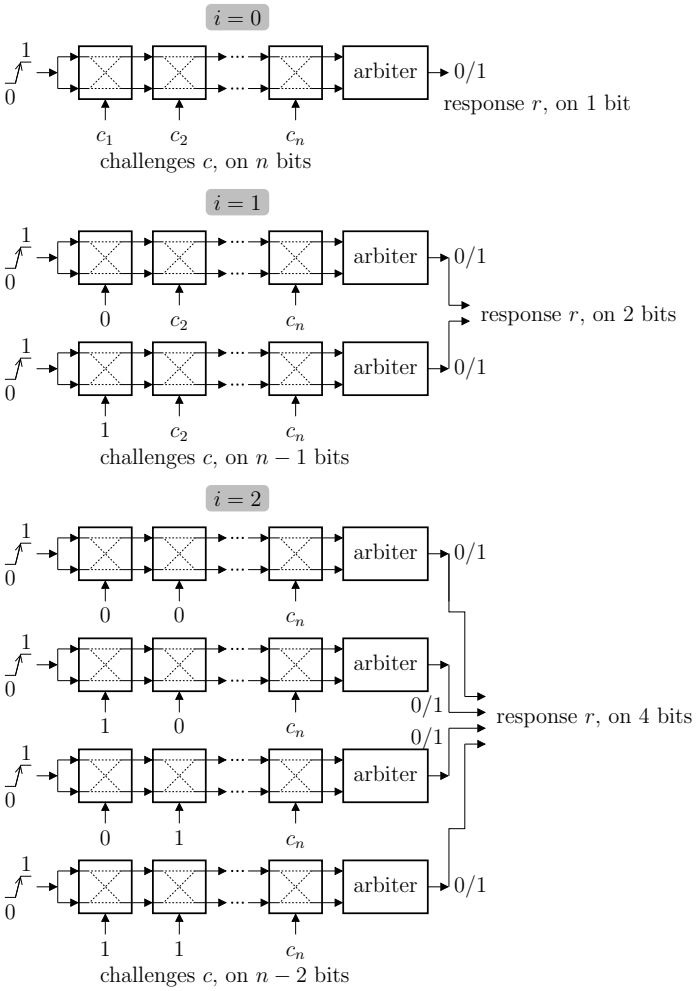


Fig. 6. Transformation of n-bit challenge with 1-bit response into equivalent $(n-i)$-bit challenges with 2^i-bit responses, for $i \in \{0,1,2\}$

The same equation can be seen additively, as:

$$\log_2(\#\text{challenge}) + \log_2(\#\text{response}) = \#\text{challenge\_bits} + \#\text{response\_bits} = n.$$

The figure 7 summarizes this law, which takes into account behaviors of Tab. 1 and 2.

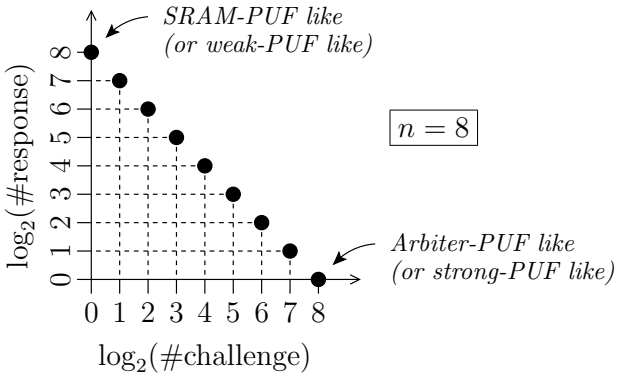


Fig. 7. Illustration of the duality between number of bits in challenges and in responses

3.2 List of metrics (security requirements)

The definition and the explanation of security requirements for PUFs are given here-after:

- **Steadiness**[3]: it is a measurement of stability of PUF responses in time. This metric can be seen as a *safety* requirement. However, PUFs with unsteady responses could be prone to prediction attacks (if the response is very biased) or to related keys attacks.
- **Randomness**: it assesses how unpredictable are PUF responses when considering a collection of response bits under all the possible challenges. The obtained *intra-PUF* bitstring shall be, ideally, unpredictable. Such security requirement attests of the PUF unclonability.
- **Uniqueness**: it estimates how different are any two pairs of different PUFs. This *inter-PUF* metric is required to quantify in which respect the fab is unable to generate clones of PUFs.
- **Unpredictability**: it estimates how hard it is to predict the responses of an $(n+1)^{\text{th}}$ PUF knowing all previous n instances. This metric relates to randomness, but is more pragmatic as it involves machine learning or *ad hoc* tests.
- **Unclonability**: this metric makes sure no easy exploitable bias exist in the PUF architecture, by design. The goal of this security requirement is to validate for the absence of trap or backdoor in the PUF rationale.

[3] Notice that *steadiness* is a word reserved for stability of a given PUF response corresponding to a fixed challenge. The synonymous terms *reliability*, *reproducibility* and *stability* are not preferred. In particular, "reliability" is discarded as it would make some confusion regarding the metric related to the yield in the CMOS manufacturing processes.

Impact of input and output bits duality on the PUF metrics. As shown in Sec. 3.1, the same metric can be used to measure the *randomness* with respect to challenges (aka *diffuseness*) and with respect to response bits (aka *unpredictability*). Therefore, trading bits of challenge for bits of response, or vice-versa, does not affect the properties of the PUF. We say that the transformation of PUF form-factor change is iso-metric, i.e., it does not modify its randomness metric. Thus, the transformation is safe, and can be applied at the convenience of the user without compromising the randomness of the PUF. We say that randomness consists in computing entropy "in the C-B plane".

4 Tests and evaluations

The security requirements defined in Sec. 3 have some value only provided one has a means to attest that a device indeed implements them. There are two methods to do so: *test* and *evaluation*.

In testing philosophy, an automatic procedure is launched to check each and any security requirement. This allows for fast and reproducing checking. However, subtle issues (e.g., corner cases, weak vulnerabilities, problems not covered by the test suite, etc.) could inadvertently pass successfully through the test.

This explains why testing is complemented by the evaluation philosophy. Evaluation is conducted by an expert, who attempts to think out-of-the-box in a view to derive attacks. This expert defines a couple of attack paths, performs a quotation (i.e., scores which reflect the cost of the attacks) for them, selects and realizes the attack of lowest quotation.

4.1 Well established tests

The tests base themselves on a *metric*. For the tests to be consistent even in heterogeneous conditions, the metric must be generic. The idea is that the metric must suit to a variety of PUFs.

Entropy is a metric which can compare data of various nature. The output of discussions at standardization committee meetings is to use this very same metric for different security requirements. The method is termed "**multiple data, one same metric**". This enables consistency within metrics, and simplifies the test of security requirements. Notice that the entropy for the steadiness is simply $H_2(\mathrm{BER}) = -\mathrm{BER}\log_2\mathrm{BER} - (1-\mathrm{BER})\log_2(1-\mathrm{BER})$, where BER is the bit error rate. The BER is laying between 0 and 1, and its value can be interpreted as follows:

- when $0 \leqslant \mathrm{BER} < 1/2$, the bit is most of the time correct;

- when $1/2 < \mathrm{BER} \leqslant 1$, the bit is most of the time incorrect – it behaves the same as if the bit is actually inverted, with BER corresponding to the previous case ($0 \leqslant \mathrm{BER} < 1/2$);
- when $\mathrm{BER} = 1/2$, the PUF is actually a perfect TRNG (True Random Number Generator).

Therefore, in practice, it is assumed that $0 \leqslant \mathrm{BER} < 1/2$. As the function $p \mapsto H_2(p)$ is strictly increasing on interval $[0,1/2[$, it is equivalent to minimize the entropy $H_2(\mathrm{BER})$ and to minimize the BER itself.

Remark 1. Notice that metrics different from entropy might be misleading. For instance, measuring the randomness by average Hamming distances can hide some weaknesses in terms of security level. For instance, as a motivating example, let us consider the following 8-bit responses to three challenges:

- R1: $(10101100)_2$
- R2: $(01101100)_2$
- R3: $(10100011)_2$

Those responses are balanced (bitstrings with as many 0s as 1s). But their Hamming distances (d_H) are not balanced, since:

- $d_H(\mathrm{R1}, \mathrm{R2}) = 2$,
- $d_H(\mathrm{R2}, \mathrm{R3}) = 6$,
- $d_H(\mathrm{R3}, \mathrm{R1}) = 4$.

However, the average Hamming distance is $(2+6+4)/3 = 4$, which might give a false confidence that responses to different challenges are "independent" (random).

Only one security requirement cannot be tested, namely *physical unclonability*.

4.2 Well established evaluations

Evaluation is required for those security requirements which cannot be tested, because they cannot be decided based on measured data. This happens for requirements which are "*negative*": this means that the security requires *not to* verify a property.

This holds for instance for "unclonability". One aspect of unclonability could be termed "supervised" unclonability: the attacker manages to predict responses from unseen challenges, after having seen enough responses from known challenges. This metric, termed *mathematical unclonability*, can be estimated as the entropy in the C-B space, hence can be turned into a test.

However, *physical unclonability*, consists in evaluating the difficulty of fabricating a PUF that has the same CRPs as a specific PUF. This task can only be achieved by a thorough qualitative analysis of the PUF design.

4.3 Further tests and evaluations

The tests described in Sec. 4.1 and the evaluations described in Sec. 4.2 are natural. Still, some more "specialized" (if not bespoke) methods can be leveraged for challenging more drastically the security requirements.

Regarding tests, entropy can also be *characterized* (instead of *estimated*) by some testsuites (e.g., DieHard [16], NIST FIPS 140-2 [25], FIPS SP 800-90 [18], FIPS SP 800-22 [22], BSI AIS31 [14], ISO/IEC 18031 [10] and companion 20543 project [11], etc.). This has been suggested already by several papers, such as [21,7]. Still, it is important those tests adhere to the principle: **multiple data, one same testsuite**

Still regarding tests, machine learning (ML) of challenge and response pairs has been shown to be able to predict with good accuracy responses from unseen challenges [21]. Thus, the general-purpose entropy metric can be traded for crafted tools, e.g., using ML, or any taylored distinguisher. Still, for consistency reasons, this analysis shall adhere to the principle: **multiple data, one same distinguisher**. For the sake of clarification, the equivalent of *randomness* when trading entropy for ML tools is referred to as *unpredictability* (or alternatively: *mathematical unclonability*).

4.4 Conclusion on tests and evaluations

The current *statu quo* on tests and evaluations is recalled in Tab. 3.

The steadiness is evaluated by the entropy of BER, and shall be minimized. The corresponding test would be to check for the balanced, that is situations where $\mathrm{BER} \approx 1/2$. However, for PUFs, the ideal situation is when $\mathrm{BER} \approx 0$, therefore classical tests are ignored.

On the contrary, randomness and uniqueness can rely on tests since their entropy metric shall be maximized. Regarding ML test, we expect that:

- high entropy goes hand in hand with passing of all tests and failure of ML
- low entropy goes hand in hand with failing of all tests and success of ML
- intermediate situations might exist, such as machine learning to succeed, but still high entropy; though, in this case, one expects the entropy to be still less than expected (e.g., 127 instead of 128 for a randomness on 128 bits).

Each metric is thus related to entropy. The entropy can be *estimated*, albeit with limited precision, since a PUF can only be queried a finite number of times. Therefore, it is reassuring to employ for PUF metrics *estimations* of *abstract random metrics*, through *stochastic models* (see e.g., [23] for steadiness and [24] for entropy metrics).

Table 3. Current *statu quo* on relevant tests and evaluations for PUFs

Metric	**Mathematical tests** (with data)			**Rationale evaluation** (without data)
	Entropy	**Tests**	**ML**	
Steadiness	Entropy in T dimension	NO	NO	NO
Randomness	Entropy in B-C plane	YES	NO	NO
Uniqueness	Entropy in D dimension	YES	NO	NO
Unpredictability[†]	NO	NO	YES	NO
Unclonability[‡]	NO	NO	NO	YES

[†] **Unpredictability** is also called **mathematical unclonability**
[‡] **Unclonability** is also called **physical unclonability**

5 Conclusions

The standardization process on PUFs has allowed to clarify security requirements. For instance, *diffuseness* (variability in the dimension of the challenges) and *unpredictability* (variability in the dimension of the response bits) are related requirements, which both concern the *randomness* of one instance of the PUF at one instant of time. Besides, diffuseness is not suitable for PUFs without challenges, where the unpredictability is irrelevant for those PUFs with only one bit of response. Our analysis has showed that a notion of randomness can encompass both diffuseness and unpredictability, thereby solving the case. Besides, we showed how randomness can be addressed both by a metric and machine learning tools.

Acknowledgments

This work was partly supported by both Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00399, Study on secure key hiding technology for IoT devices [KeyHAS Project]) and the project commissioned by the Japanese New Energy and Industrial Technology Development Organization (NEDO).

References

1. Halak Basel. Physically Unclonable Functions — From Basic Design Principles to Advanced Hardware Security Applications. DOI: 10.1007/978-3-319-76804-5.
2. Christoph Böhm and Maximilian Hofer. *Physical Unclonable Functions in Theory and Practice*. Springer Publishing Company, Incorporated, 2012.
3. Yu Cai, Saugata Ghose, Yixin Luo, Ken Mai, Onur Mutlu, and Erich F. Haratsch. Vulnerabilities in MLC NAND flash memory programming: Experimental analysis, exploits, and mitigation techniques. In *2017 IEEE International Symposium on High Performance Computer Architecture, HPCA 2017, Austin, TX, USA, February 4-8, 2017*, pages 49–60. IEEE Computer Society, 2017.
4. Zouha Cherif, Jean-Luc Danger, Sylvain Guilley, and Lilian Bossuet. An Easy-to-Design PUF based on a single oscillator: the Loop PUF. In *DSD*, September 5-8 2012. Çeşme, Izmir, Turkey; (Online PDF).
5. Altera Corporation. White paper: FPGA Architecture. July 2006, ver. 1.0, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01003.pdf, last accessed: April 19, 2018.
6. Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 148–160. ACM, 2002.
7. Sylvain Guilley and Youssel El Housni. Random numbers generation: tests and attacks. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018, Amsterdam, Netherlands, September 13, 2018*. IEEE Computer Society, 2018.
8. Tim Güneysu. Using Data Contention in Dual-ported Memories for Security Applications. *Signal Processing Systems*, 67(1):15–29, 2012.
9. Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Trans. Computers*, 58(9):1198–1210, september 2009.
10. ISO/IEC JTC 1/SC27/WG2. ISO/IEC 18031:2011 – Information technology – Security techniques – Random bit generation.
11. ISO/IEC JTC 1/SC27/WG3. ISO/IEC DIS 20543 – Information technology – Security techniques – Test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408.
12. ISO/IEC NP 20897. Information technology – Security techniques – Security requirements, test and evaluation methods for physically unclonable functions for generating nonstored security parameters. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=69403.
13. Marc Joye and Michael Tunstall, editors. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012. ISBN: 978-3-642-29655-0; DOI: 10.1007/978-3-642-29656-7.
14. Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes for random number generators, September 2011. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf?__blob=publicationFile.
15. Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*, pages 361–372. IEEE Computer Society, 2014.

16. George Marsaglia. The Marsaglia random number CDROM including the Diehard Battery of Tests of randomness. Web site at the Department of Statistics, Florida State University, Tallahassee, FL, USA., 1995.
17. Onur Mutlu. The RowHammer problem and other issues we may face as memory becomes denser. In David Atienza and Giorgio Di Natale, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 1116–1121. IEEE, 2017.
18. NIST. Recommendation for the entropy sources used for random bit generation, 2012. http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf.
19. Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, September 20 2002. DOI: 10.1126/science.1074376.
20. Ravikanth S. Pappu. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, March 2001.
21. Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 237–249. ACM, 2010.
22. Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications, april 2010. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf.
23. Alexander Schaub, Jean-Luc Danger, Sylvain Guilley, and Olivier Rioul. An Improved Analysis of Reliability and Entropy for Delay PUFs. In Martin Novotný, Nikos Konofaos, and Amund Skavhaug, editors, *21st Euromicro Conference on Digital System Design, DSD 2018, Prague, Czech Republic, August 29-31, 2018*, pages 553–560. IEEE Computer Society, 2018.
24. Alexander Schaub, Olivier Rioul, Joseph J. Boutros, Jean-Luc Danger, and Sylvain Guilley. Challenge Codes for Physically Unclonable Functions with Gaussian Delays: A Maximum Entropy Problem, July 22-27 2018. LAWCI, Latin American Week on Coding and Information, UNICAMP – Campinas, Brazil.
25. NIST FIPS (Federal Information Processing Standards). Security Requirements for Cryptographic Modules publication 140-2, May 25 2001. http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.
26. Ying Su, Jeremy Holleman, and Brian P. Otis. A Digital 1.6 pJ/bit Chip Identification Circuit Using Process Variations. *IEEE Journal of Solid-State Circuits*, 43(1):69–77, Jan 2008.
27. Daisuke Suzuki and Koichi Shimizu. The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 366–382. Springer, August 17-20 2010. Santa Barbara, CA, USA.
28. Pim Tuyls, Boris Skoric, and Tom Kevenaar. *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, December 2007. 1st Edition, ISBN 978-1-84628-983-5.
29. Meng-Yi Wu, Tsao-Hsin Yang, Lun-Chun Chen, Chi-Chang Lin, Hao-Chun Hu, Fang-Ying Su, Chih-Min Wang, James Po-Hao Huang, Hsin-Ming Chen, Chris Chun-Hung Lu, Evans Ching-Song Yang, and Rick Shih-Jye Shen. A PUF scheme using competing oxide rupture with bit error rate approaching zero. In *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*, pages 130–132. IEEE, 2018.