Testing and reliability enhancement of security primitives: Methodology and experimental validation

Md Toufiq Hasan Anik, Jean-Luc Danger, Omar Diankha, Mohammad Ebrahimabadi, Christoph Frisch, Sylvain Guilley, Naghmeh Karimi, Michael Pehl, Sofiane Takarabt

Testing and Reliability Enhancement of Security Primitives: Methodology and Experimental Validation

Md Toufiq Hasan Anik[d], Jean-Luc Danger[a], Omar Diankha[e], Mohammad Ebrahimabadi[d], Christoph Frisch[c], Sylvain Guilley[b], Naghmeh Karimi[d], Michael Pehl[c] and Sofiane Takarabt[b]

[a]*LTCI, Télécom Paris, Institut polytechnique de Paris, France*
[b]*Secure-IC, France*
[c]*Technical University of Munich, Department of Electrical and Computer Engineering, Germany*
[d]*University of Maryland Baltimore County, United States*
[e]*Paris 8 University, France*

ARTICLE INFO

Keywords:
Test
Physically Unclonable Function (PUF)
High order Alphabet (HoA)
True Random Number Generation (TRNG)
side-channel analysis (SCA)
fault injection attack (FIA)
Digital Sensor (DS)
Automatic Test Pattern Generation (ATPG)

Abstract

The test of security primitives is particularly strategic as any bias coming from the implementation or environment can wreak havoc on the security it is intended to provide. This paper presents how some security properties are tested on hardware security primitives including TRNG, PUF, and cryptographic modules. Moreover, we discuss how the sensors embedded to protect cryptographic modules against fault injection attacks should be calibrated over time to fulfill the requirement it was designed for. The testing we discuss in this paper is different from the conventional testing where we consider a fault model and generate test patterns via an ATPG to detect such faults. The test of TRNG and PUF to ensure a high level of security is mainly about the entropy assessment, which requires specific statistical tests. The security against SCA of cryptographic primitives, like the substitution box in symmetric cryptography, is generally ensured by masking. However, the hardware implementation of masking can be damaged by glitches, which create leakages on sensitive variables. Accordingly, a test method is to search for nets of the cryptographic netlist, which are vulnerable to glitches. Finally, the DS is an efficient primitive to detect disturbances and raise alarms in the case of FIA. The dimensioning of this primitive requires a precise test to take into account the environmental variations including aging.

This paper extends on a conference paper presented at DFTS '21 by the same co-authors, where the test methodology for three critical security primitives is presented. In addition, in this paper, we add experimental validation to show how such testing methodology is applied in practice.

1. Introduction

Functional testing has become a mandatory requirement for circuits to be admitted in downstream supply chain. Involved techniques are JTAG for boundary scan, and inner logic validation, BIST for memories, etc. Although these methods are well-known and have been deployed for a long time in digital circuits, their suitability for security functions appears to be insufficient. Indeed, those techniques only assess the correct functional behavior, but fail to test security functionalities (which are often non-functional).

Typically, regarding security applications, it is expected that some domain-specific tests are carried out. A secure chip typically embeds key generation logic (such as a PUF and/or a TRNG), cryptographic algorithms, and embedded sensors to monitor the operating conditions and/or prevent fault injection attacks. Obviously, PUFs used as *master keys* shall be reliable[1]. In addition, as any cryptographic key, they must be unpredictable; hence their randomness shall be ensured. Indeed, cryptographic algorithms are designed and proven secure assuming keys of maximal entropy (see for instance the recommendation from [39, § B.3.14.8]). Besides, cryptographic key management shall be secure against side-channel attacks, such as those exploiting masking countermeasures. Such protections are nowadays well known, in general. But without special care, they are vulnerable to glitches, which shall be managed responsibly. Eventually, digital sensors, which are standard-cell based structures shall be calibrated in terms of aging, so that they remain as efficient as possible across device utilization stages.

This paper addresses all these issues in a pedagogical manner. Three sections are devoted to security-specific tests that shall be carried out in addition to the usual functional tests. Given the fast spread of security features in chips, these tests shall not only be considered as *nice* features, but very soon, as *mandatory* features. Indeed, silicon security pieces of "intellectual properties" (IP) are required to fulfill the requirements of some safety or cyber-security certification schemes. Table 1 lists some such schemes and indicates which IP allows getting compliance. The safety schemes are:

- **IEC 61508**, entitled "Functional Safety of Electrical/ Electronic/Programmable Electronic Safety-related Systems", published by the International Electrotechnical Commission (IEC) and
- **ISO 26262**, entitled "Road vehicles – Functional safety", is published by International Organization for Standardization (ISO) sub-committee ISO/TC 22/SC 32, and is an adaptation for the automotive market.

ORCID(s):

[1] International standard ISO/IEC 20897-2 [2] gives a comprehensive list of requirements regarding PUFs, which can be found in [10, §3.2].

The security schemes are multiple. We list them hereafter:

- The "Security Evaluation Standard for IoT Platforms" (**SESIP**) is a Global Platform standard specifying lightweight requirements dedicated to the Internet of Things (IoT) market. The SESIP can be seen as a simplified profile from Common Criteria.
- The standard **ISO/IEC 15408**, also known as "**Common Criteria**" (CC), is an open, rigorous, scheme aiming at providing a given level of assurance of the platform. Part 2 of the CC defines a list of Security Functional Requirements, which are referred to as `<CLASS>_<FAMILY>`.
- Regarding key management, the USA National Institute of Standards and Technology (NIST) defines the **FIPS 140-3** standard for hardware security modules (HSM). This Federal Information Processing Standard (FIPS) 140-3 lists security requirements. The 3rd revision (hence the "-3" suffix) is actually the same as the international standard **ISO/IEC 19790:2012**.
- The Office of State Commercial Cryptography Administration (**OSCCA**) is a Chinese administration organized in a similar manner as NIST. Its resulting norm is referred to as **GM/T 0008:2012**.
- The European Telecommunications Standards Institute (ETSI) is endowed with the capability to define European Norms (abridged EN). In the IoT market, the ETSI has published the "Cyber Security for Consumer Internet of Things: Baseline Requirements" as **ETSI EN 303 645** V2.1.1.
- Eventually, the European project "E-safety Vehicle Intrusion proTected Applications" (or **EVITA**, pertaining to the Seventh Framework Program) defines *de facto* the normative requirements for automotive HSMs.

It can be seen that only sensors matter in terms of safety. The reason is that *weak* keys and *leaky cryptography* do not hinder systems' safety at all. Regarding cybersecurity, more requirements come into play. Indeed, security requires safety, but calls for more caution, namely:

- *secrecy of keys* upon generation (when spawned by a PUF or a TRNG) and upon use (in cryptographic algorithms exposed to side-channel attacks);
- *protection against perturbations*, which can lead to cryptanalysis [40]. The very same sensors as leveraged for achieving safety goals can be reused verbatim in this respect.

In Tab. 1, the EFP acronym in FIPS standard line stands for "Environmental Failure Protection" (see §4), whereas for CC, one has `FCS_CKM` = "Cryptographic key generation", `FPT_PHP` = "Passive detection of physical attack", and `FRU_FLT` = "Fault tolerance". Notice that there is currently no mature test suite for PUF.

Certif. scheme	PUF / TRNG	Masking	Sensors
Safety (IEC 61508, ISO 26262)	N/A	N/A	A safety mechanism whose "*diagnostic coverage*" shall be measured by tests
SESIP v1.1	N/A	Level 3 onward	Level 3 onward
Common Criteria	`FCS_CKM`	`FPT_PHP`	`FRU_FLT`
NIST FIPS 140-3	Sensitive Security Parameter Management	Module is designed to mitigate against non-invasive attacks specified in Annex F.	Tamper detection and response envelope. EFP. Fault injection mitigation.
OSCCA, GM/T 0008	Same as above	Same as above	Same as above
ETSI EN 303 645	No universal default passwords	N/A	N/A
EVITA	Yes (D3.2)	N/A	N/A

Table 1
Mapping between certification schemes and applicable IPs to meet their safety/security requirements.

Contributions. As mentioned earlier, in this paper, we DO NOT apply conventional testing for which we need to deploy an ATPG to generate patterns for the targeted fault model and apply these patterns to ensure that the circuit output does not deviate from the expected output. Rather we discuss the requirements that each of our targeted security primitives (PUFs, TRNGs, Cryptographic Modules, and Digital Sensors) should meet to ensure security and/or safety. In sum, in this paper, we aim at providing a full overview of tests related to security chips (this test is a non-functional test needed for each of the targeted primitives to fulfill the application they were designed for). Such information is usually only available in specialized publications, or even worse, is not publicly discussed. Based on our results in [6], we detail the nature of the tests for three classes of security functions, namely:

1. Functions managing the need for randomness, namely through TRNGs (dynamic randomness) and PUFs (static randomness);
2. Leakage Analysis in cryptographic algorithms, whose resistance to side-channel attacks shall be extensively proven;

3. Resistance of sensors against perturbation, via considering aging into account.

We provide examples of experimental validation to show how these tests apply in practice. Namely, we discuss differences in testing binary and higher-order alphabet PUFs and illustrate our findings based on measurement results taken from 180 FPGA boards. Also, we show that side-channel attacks can extract secret keys if the netlist-level testing reveals a vulnerability owing to spurious glitches whose activity discloses clear (unmasked) information. Eventually, regarding the sensor, we show on an FPGA platform how to calibrate a digital sensor, and explain the impact of intra- and inter-die variability.

Scope of the paper. The paper covers a broad scope, as it aims at addressing relevant aspects of security testing. Still, in addition to the panorama, we offer some deep-dive into the addressed topics by providing some novel contributions. Namely:

- Regarding PUFs, we introduce and discuss the innovate concept of *high-order alphabets*;
- Regarding side-channel protection of cryptographic functions, we illustrate how to spot *glitches* from a theoretical manner (in the netlist) and we confirm that it leads to real exploitation (in practice);
- Regarding digital sensors, we show how to cope with *aging*, and in addition we provide *actual FPGA* results.

Outline. The rest of the paper is structured as follows. The question of entropy testing is tackled in Sec. 2. Also, in this section, an example of high-alphabet PUF is introduced and analyzed. Testing for harmfulness of glitches in side-channel protections is the topic of Sec. 3. The validation of digital sensors across aging is discussed in Sec. 4. Eventually, Sec. 5 concludes the paper. A technical proof is relegated in Appendix. A.

An orientation regarding the different security-related aspects covered throughout the paper is depicted in Figure 1.

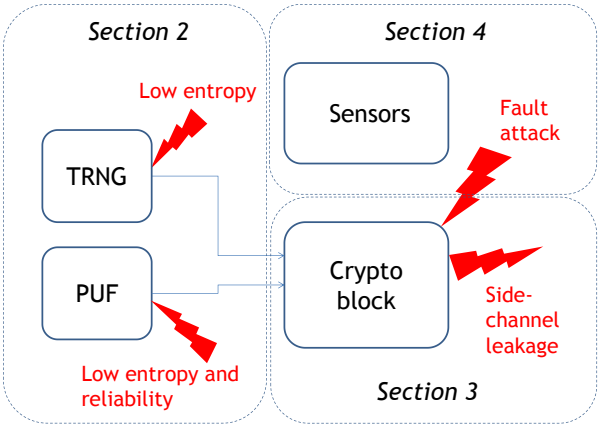


Figure 1: Security IPs and threats targeting them.

2. Evaluation of PUFs and TRNGs

Two kinds of randomness are necessary for secure devices: A secret key requires randomness, which is stable over time. However, security also relies on randomness which is fresh whenever sampled, e.g., for nonces. These different kinds of randomness for the overall cryptosystem are provided by security primitives such as PUFs and TRNGs. PUFs extract static randomness from manufacturing variations. Dynamic randomness, e.g., for nonces, stems from TRNGs whose randomness consist in the digitization of noise. Nonetheless, in either case an attacker should not be able to predict the random data. Therefore, the entropy of PUFs and TRNGs is crucial for the security of a system, as it has been shown in Fig. 1. To make this entropy usable as a key, reliability – i.e., the property that always the same realization of a random number is derived – is an important property for PUFs, too. However, evaluating the reliability of PUFs is a separate issue that needs to be addressed and out-of-scope in this work. An exemplary approach can be found in [41]. Other PUF metrics, like Uniformity, Uniqueness, Bit-Alias [19, 27] are generally used in literature for testing PUFs regarding the quality of their randomness and standardized test suits are used to test TRNGs. However, such security-specific statistical tests ensuring a sufficient quality of the randomness, and ultimately the correct functionality of PUFs and TRNGs are hard to realize without infusing a backdoor, since access to security relevant internal data must be provided at least temporarily for testing. For TRNGs, test suites partly consist out of complex tests and are thus not suitable for testing on a device. They require offline testing. For PUFs, some metrics additionally require data from multiple devices which also necessitates offline tests.

2.1. Fundamental Differences in Statistical Properties

At first glance, PUFs and TRNGs both produce randomness. However, applying the same tests to them is problematic because their randomness has several differences:

Source of Randomness: The most important difference between PUFs and TRNGs is their basis for randomness. A PUF extracts randomness from the variations in the manufacturing process which should stay constant. For a TRNG, the randomness stems from noise. This disparity is the foundation for more differences.

Amount of Data: Statistical tests require large amounts of data to ensure a sufficient significance of the results. Disregarding latency, TRNGs can output arbitrarily much random data because the noise keeps changing over time. In contrast, PUFs can provide only a limited amount of randomness since process variations are fixed after the production of the chip. Hereby, the PUF primitive determines the amount of

data that can be extracted. Normally, PUFs as key storage output one bit (e.g., SRAM PUF [17]) or only few bits (e.g. Loop PUF [12]) per PUF instance, and a chip possesses a limited number of PUF instances[2]. As a result, many devices are necessary to enable testing with enough data.

Dimension of data: Whereas the bit-stream of a TRNG is one-dimensional and from a single source, testing of PUFs has to be along multiple dimensions: (i) For many PUFs, a combination of multiple PUF instances on a device results in a key of the desired length. Consequently, there are two dimensions to test – the unpredictability of a single PUF instance at a specific position evaluated over several devices and the relation of the PUF responses on a single device. (ii) Oftentimes, PUF instances on a single device are structured in a two-dimensional array. The findings in [31] show that the way of concatenating the data row- or column-wise to get a stream of responses has an impact on the test result. (iii) Some PUFs are configurable by a challenge; in this case, the choice of the challenges adds another dimension. (iv) In addition, so called higher-order alphabet (HoA) PUFs output multiple symbols per challenge or position.

One approach to tackle the different dimensions in (i), (ii), and (iii) is to understand the PUF not as one source outputting multiple bits, but as a multi-bit source or as multiple one-bit sources. Beyond, HoA PUFs need even more advanced techniques for testing as discussed below.

Impact of Noise: PUFs extract a constant randomness from noisy data, whereas TRNGs generate fresh randomness from noisy data. There are two aspects linked to that: (i) Testing a PUF means dealing with samples from a joint distribution of manufacturing variations and noise[3]. (ii) The entropy of a PUF is restricted by the limited variations in the manufacturing process and is effectively lowered by noise which often imposes remediation means such as error correction[4].

2.2. Properties to be Tested

Because of the different nature of TRNGs and PUFs, the tests themselves have to evaluate distinct properties. TRNG tests can be divided into two categories: Either, the test compares the TRNG output sequences to sequences of independent and identically distributed (iid) numbers, i.e. ideal randomness. If they cannot be distinguished, the TRNG is considered to be of a high quality. Or, a test estimates the entropy of a TRNG output.

Similarly, a PUF test checks if a PUF response is random enough, i.e., unpredictable. Due to the multiple dimensions of a PUF, this unpredictability has to hold even if an attacker collects information across any of the dimensions. Consequently, a PUF test has to cover several aspects: PUF responses have to be free of bias or correlation effects, regardless of whether they are from different PUF instances on the same device, for different challenges, or over more devices for a fixed position.

In addition to statistically testing the unpredictability of a PUF, noise effects have to be analyzed as well. The noise should be low enough such that only a minimum amount of post-processing, e.g., in the form of error corrections, is needed.

Lastly, the entropy is important for PUFs as well as for TRNGs. Hereby once again, the entropy estimator has to incorporate the intricacies of PUFs, such as their multiple dimensions.

2.3. Test Methods for TRNGs

No test can prove randomness. Thus, the evaluation of a TRNG relies on statistical test suites with multiple tests. Whenever a test is passed, this strengthens the confidence in the overall output.

The NIST SP 800-22 standard [7] compares the TRNG output with iid bits. 15 tests evaluate the input by looking for patterns under the null hypothesis H_0 that the tested RNG's output sequence is random. A test rejects this null hypothesis if its p value is too low. The overall test suite finally interprets the individual tests and gives a concluding result.

The BSI AIS 31 standard [23] (as well as its current draft of an update [24]) defines a second standardized test suite. It has nine tests which in parts overlap with [7] and also analyze different criteria of a TRNG's output. Besides, it requires information about the structure of the TRNG to enhance the overall test result.

There are also non-standardized tests such as TESTU01 [25] with six test batteries which follow a similar concept as the standardized ones. Also, e.g., in the BSI AIS 31 a model for the source of entropy is required to substantiate the claim of true randomness.

Instead of comparing the observed randomness of the TRNG under test with an ideal one, estimating the entropy is also possible such as in the NIST SP 800-90B [35]. First, a user has to pick either the iid or the non-iid track. Then the estimators evaluate the min-entropy. The lowest value, finally is the output of the whole test.

2.4. Test Methods for PUFs with Binary Responses

Unlike for TRNGs, for PUFs there is only one standard at the moment: ISO/IEC 20897-2 [2] which evaluates randomness and reliability. Part of the test can be found in the standardized TRNG test suite, and part of them is based on PUF-specific publications, e.g., [19, 27]. It also applies the NIST SP 800-90B to estimate the entropy of a PUF.

Besides [2], research has proposed other methods. Overall, there is a large amount of various qualitative PUF tests. Visualizing their output can highlight issues of the underlying PUF, such as the Principal Component Analysis (PCA) in [45], which can show gradients of the process variations. In recent years, the general concept of testing has improved by introducing statistics in the form of confidence

[2]Please note that multi-challenge PUFs like Arbiter PUFs [26] or SUM PUFs [47] and their relatives are rarely used for key storage and are potentially weak against machine learning [36].

[3]Potentially, more effects can also be part of this distribution, e.g., temperature shifts, fluctuations in the supply voltage, or device parameters variation owing to its aging.

[4]Some PUFs, such as the Loop PUF [12], can achieve a required entropy / reliability by adapting the number of oscillations.

intervals or hypothesis testing (e.g., [46]). As an additional dimension, spatial information can be integrated [44], which highlights the difference to TRNG tests. Spatial information also enhances entropy estimation for PUFs [32]. However, in some scenarios, not only the entropy in the PUF is relevant, but also in the extracted key. The findings in [43] demonstrate how this key entropy can be evaluated.

The tests discussed so far mainly focus on PUFs with binary responses, even if few are extendable to PUFs with responses from a higher order alphabet. Therefore, next, we describe issues and solutions for testing such HoA PUFs.

2.5. Test Methods for HoA PUFs

Recently, higher-order alphabet PUFs (HoA-PUFs) have gained attention [16, 20, 28]. Instead of deriving a single bit per PUF instance, challenge, and position on a device, HoA-PUFs provide symbols from a higher-order alphabet, typically encoded as a bit sequence. For this purpose, special quantization strategies, like equi-probable or equi-distant quantization [22], are used. Such an approach can allow for deriving more entropy per area and thus makes a PUF more efficient. It was suggested, e.g., for a PUF-based secure enclosure [21]. However, current testing strategies have to be adapted or different, and novel ones are necessary in order to test such PUFs.

In [20], the author modifies the *reliability* and *uniqueness* as two state-of-the-art binary metrics for HoA-PUFs. For these two metrics, an adaption is possible because the Hamming distance of two PUF responses is defined not only for binary data but also for HoA-PUF responses. Compression based on Context Tree Weighting (CTW) can also be extended from a binary to a higher-order alphabet setting [32, 42]. Even NIST SP 800-90B tests have the capability to handle symbols from a higher-order alphabet. However, most tests for binary PUFs are not applicable to HoA-PUFs, because they assume a binary probability distribution.

Therefore [13] proposes new tests for statistically sound analysis of bias effects[5] of HoA-PUFs. Note that metrics for HoA-PUFs are also applicable to binary data, but potentially have lower precision, because the underlying mathematical approximations are less exact.

Summarizing the work in [13], we now illustrate current research on finding suitable test schemes for HoA-PUFs. A Loop PUF [12], which is a ring oscillator based PUF primitive with configurable delay stages, serves as an example. Most commonly, a Loop PUF would (i) measure frequencies of the same ring under always two specific configurations by counting the number of periods within a fixed time and (ii) take the sign bit of the counter difference as the binary response. In addition, we emulate a HoA PUF, by mapping the counter difference after step (i) to more than two intervals. This results in more than two distinct response symbols. Figure 2 depicts the measurement of data from 48 Loop PUF instances on 180 BASYS 3 FPGAs implemented according to [38] and quantized to 16 different symbols. For

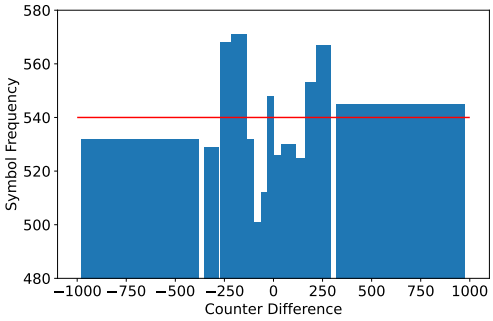


Figure 2: Histogram of symbol frequency of Loop PUF data. The expected symbol frequency for each symbol is 540 [13] .

quantization, intervals were selected so that on a normal distribution approximating the expected counter differences' distribution all symbols are equally likely. As a consequence, the intervals do not have the same size; the intervals toward the tails of the underlying probability distribution are larger than the ones closer to the mean.

Given such a data set, the underlying PUF can be checked for sufficiently low bias using hypothesis testing.[6] This is possible at first glance by interpreting the HoA PUF output as a binary string or as HoA symbols.

Assuming symbols from a higher order alphabet, the following newly proposed tests in [13] evaluate this data: (i) *Pearson's Chi-squared Test* checks the null hypothesis that the measured data corresponds to the expected probability distribution of the counter differences. (ii) The *Multinomial Confidence Intervals* compute the overall confidence interval for all symbols at once. (iii) The *Acceptable Intervals* define intervals for the amount of occurrences for each symbol. The limits of the intervals depend on symbol probabilities and other test parameters for which a user would define the PUF to have a sufficiently high quality. If the measured symbol frequency all are within their respective intervals, then a user can deduce a high quality from the PUF.

For the sake of a comparison between binary bias test and HoA-PUF bias tests in [13] , we now map the symbols in Figure 2 to their binary representation (e.g., symbol 5 would be 0101) and compute the average Hamming weight of the corresponding binary data. The resulting average Hamming weight per bit is 0.499 (also known as bit-alias), which is very close to the ideal value of 0.5. Consequently, the PUF might be considered to have no bias when applying a test for binary data.

In comparison, the evaluation based on symbols and using the new tests in [13] provides more precise statements about bias effects: Given the data set, *Pearson's Chi-squared Test* does not reject the null hypothesis that symbols do not follow a uniform distribution. The *Multinomial Confidence Intervals* contain the expected probability for each symbol, which also indicates a low bias. However, the min-entropy

[5] Bias metrics evaluate if a symbol occurs more (or less) frequently than expected and thus target the unpredictability as a core property of a PUF.

[6] Please note, that "sufficiently low bias" refers to a parameter choice by the designer and implies that subsequent postprocessing of the PUF prevents any predictability issues of a PUF derived secret.

computed based on the received intervals is 3.76 bit (instead of 4 bit), which indicates a slight discrepancy from an ideal HoA-PUF. For the *Acceptable Intervals*, recall that the 16 symbols are ideally uniformly distributed occurring with probability $\frac{1}{16}$. As an example, we allow for the PUF an offset from this ideal value by at most $\frac{1}{80}$, i.e., assuming that some subsequent compression counters such a defect. If we try to guarantee that the bias for each symbol is in such an interval of $\frac{1}{16} \pm \frac{1}{80}$, five out of 16 symbols – namely symbols 3, 4, 5, 6, 15 – fail the test, i.e., based on the test, we do not reject the null hypothesis that the symbol probabilities are outside of $\frac{1}{16} \pm \frac{1}{80}$. So unlike the exemplary test for binary data, such as the average Hamming weight which does not detect any bias effects, the new tests for HoA-PUFs point to several issues, highlighting the benefit of such tests.

2.6. Discussion of Test Strategies

This introduction to testing PUFs and TRNGs shows that testing randomness adds another level of complexity to functional testing in the security domain. The comparison illustrates that PUFs are even harder to test than TRNGs. This and the novelty of PUFs mean that for PUFs no well-established test suite exists today. Thus, further investigation is needed to substantiate the recommendations in the existing PUF standard with a complete set of tests like for TRNGs. In particular, the example given for testing HoA PUFs shows the importance of developing dedicated tests for specific usecases.

The randomness of PUFs and TRNGs constitutes one important part of the overall security of a system and thus deserves dedicated testing strategies. Yet there are additional aspects regarding the security (such as SCA or FIA) of a system which also motivate security-specific testing. These considerations are discussed in the following sections.

3. Assessment of SCA leakage in cryptographic circuits

3.1. Presentation of the problem

Cryptographic algorithms consume keys generated by TRNGs and PUFs. They compute ciphertexts from plaintexts, or generate signatures from hashes of messages. While they compute, they inadvertently leak information on the key, as represented in Fig. 1. As a matter of fact, the intermediate variables within the algorithm incur more or less power consumption. Related to that, the electromagnetic field emitted during the computation is also somehow dependent on the key. For this reason, the RTL description of cryptographic algorithms often leverages "random masking". This is an implementation style whereby a random input is fed to the module, and mixed to the computation. Correct implementations ensure that key-dependent intermediate variables (without mask) are turned into independent variables.

In this context of gate-level masking, not only every net must be duly masked, but also the netlist must be protected against glitches. A glitch is a difference in the evaluation of the netlist, which is likely (or not) to happen, depending on the internal delays while executing the netlist. In this section, we formalize the notions of perfect masking (known since 2014) and perfect masking in the presence of glitches (our contribution). Moreover, we propose efficient methods to verify whether the properties are met. Such methods make up the announced tests of masked logic in the presence of glitches.

3.2. Formalization of correct masking scheme

Let $k \geq 1$, and $F : \mathbb{F}_2^{3k} \to \mathbb{F}_2$ a Boolean function of 3 variables, each of k bits.

The Boolean function F models a net in a netlist, and:

- $a \in \mathbb{F}_2^k$ is the masked information,
- $m_i \in \mathbb{F}_2^k$ is the input random mask, and
- $m_o \in \mathbb{F}_2^k$ is the output random mask.

For example, the masking of a substitution box (also known as an S-box, a permutation from k bits, denoted $S : \mathbb{F}_2^k \to \mathbb{F}_2^k$) is $(a, m_i, m_o) \mapsto S(a \oplus m_i) \oplus m_o$. One coordination of this function is denoted by F.

We aim to verify that F protects the value of the sensitive information $x = a \oplus m_i$, leveraging either input mask m_i or output mask m_o.

Notice that the masks are uniformly distributed, that is $P(M_i = m_i) = 2^{-k}$, for all value $m_i \in \mathbb{F}_2^k$, and similarly $P(M_o = m_o) = 2^{-k}$, for all $m_o \in \mathbb{F}_2^k$.

In the sequel, to simplify the analysis, we focus on nets which are balanced. We define two properties.

Property 1 (Perfect masking [9])**.** *The function F is perfectly masked if, for all $x \in \mathbb{F}_2^k$,*

$$P(F(A, M_i, M_o) = 1 | X = x) = P(F(A, M_i, M_o) = 0 | X = x).$$

Property 2 (Perfect masking against glitches)**.** *The function F is perfectly masked against glitches if, for all $x \in \mathbb{F}_2^k$, for all $\delta \in \mathbb{F}_2^{3k} \backslash \{0\}$, denoted $\delta = (\delta_A, \delta_{M_i}, \delta_{M_o})$,*

$$P(F(A \oplus \delta_A, M_i \oplus \delta_{M_1}, M_o \oplus \delta_{M_o}) \oplus F(A, M_i, M_o) = 1 | X = x) = P(F(A \oplus \delta_A, M_i \oplus \delta_{M_1}, M_o \oplus \delta_{M_o}) \oplus F(A, M_i, M_o) = 0 | X = x).$$

It is proven in Appendix A that properties 1 and 2 can be checked efficiently based on computing Walsh transforms. These can be speeded up with butterfly algorithms. Namely, the systematic and automatic masking verification is carried out as shown in Alg. 1. The design is classified as secure if the two lists $\mathcal{L}_u$ and $\mathcal{L}_g$ are empty.

3.3. Emblematic example

One challenge is, for instance, to verify each and every net from Canright's masked S-Box [11] of AES. The netlist can be found in [14], and the function we consider is:

```
module bSbox ( A, M, N, encrypt, Q );
```

at line 234 (see Listing 1).

Algorithm 1: Masking verification method

input : Netlist

output: Lists of unmasked gates and of gates susceptible to glitching unmasked value

$\mathcal{L}_u \leftarrow \emptyset, \mathcal{L}_g \leftarrow \emptyset$ // Unmasked / Glitching nets

for $F \in Netlist$ **do** // Traversal is chosen by the tester

  for $x \in \mathbb{F}_2^k$ **do**

    $w_u \leftarrow 0$

    for $m_i, m_o \in \mathbb{F}_2^k$ **do**

      $w_u \leftarrow w_u + (-1)^{F(x \oplus m_i, m_i, m_o)}$

    if $w_u \neq 0$ **then** // Verification of Prop. 1 leveraging Lem. 1

      $\mathcal{L}_u \leftarrow \mathcal{L}_u \cup \{F\}$

    for $\delta \in \mathbb{F}_2^{3k} \setminus \{0\}$ **do**

      $w_g \leftarrow 0$

      for $m_i, m_o \in \mathbb{F}_2^k$ **do**

        $w_g \leftarrow w_g + (-1)^{F((x \oplus m_i, m_i, m_o) \oplus \delta) \oplus F(x \oplus m_i, m_i, m_o)}$

      if $w_g \neq 0$ **then** // Verification of Prop. 2 leveraging Cor. 2

        $\mathcal{L}_g \leftarrow \mathcal{L}_g \cup \{F\}$

return $\mathcal{L}_u, \mathcal{L}_g$

Listing 1: S-Box definition, in Canright's masked AES implementation.

```
[...]
/* find either Sbox or its inverse in GF(2^8), by Canright Algorithm
   with MASKING: the input mask M and output mask N must be given */
module bSbox ( A, M, N, encrypt, Q );
    input  [7:0] A;
    input  [7:0] M;
[...]
```

The masked information on $k = 8$ bits is A, the input mask m_i is M and the output mask m_o is N. The signal *encrypt* selects whether the S-Box is the direct or inverse function (SubBytes vs. InvSubBytes), and the output is Q. We shall test all 8 bits of Q, and also all internal nets within the netlist.

In this netlist, it is known that all nets are well masked, but also that some nets are vulnerable to glitches. This has motivated to elaborate more complex protections, such as threshold [30], glitch-free [29], or glitch-immune [37] implementations. We recall the list $\mathcal{L}_g$ of glitching gates which disclose the secret here. They consist in the code below the comment [*sic*]:

```
// YO! NEED TO DO SUMMATION BELOW IN SEQUENTIAL
ORDER FOR SECURITY !!!!
```

at lines 74, 96, 100, and 106 of the netlist [14] (see Listing 2).

Listing 2: S-Box implementation, in Canright's masked AES implementation.

```
[...]
// YO! NEED TO DO SUMMATION BELOW IN SEQUENTIAL ORDER FOR SECURITY !!!!
/* optimize section below using NOR gates */
    assign cst = {   /* note: ~| syntax for NOR won't compile */
      ~(a[1] | b[1]) ^ (~(af[2] & bf[2])) ,
      ~(af[2] | bf[2]) ^ (~(a[0] & b[0])) }
        ^ m2 ;
/* end of NOR optimization */
    assign csa = cst ^ an ;
    assign csb = csa ^ mb ;
    assign cm = {   /* this includes mask switch */
      m[1] ^ nf[2] ,
      mf[2] ^ n[0] }
        ^ mn ^ m2 ;
    assign c   = csb ^ cm ;
    assign e = {  /* inverse masked by n (lo input mask) */
      c[0] ,
      c[1] };
    FAC_2 efac(e, ef);
    GF_MULS_2  qmul(ef, af, q);
    GF_MULS_2 emmul(ef, mf, em);
/* NOTE: to maintain masking, the output mask N must be added BEFORE
   p, q are added to other terms */
// YO! NEED TO DO SUMMATION BELOW IN SEQUENTIAL ORDER FOR SECURITY !!!!
    assign qsa = N[1:0] ^ an ;  /* mask terms for q (lo output) */
    assign qsb = qsa ^ em ;     /* mask terms for q (lo output) */
    assign qm = qsb ^ mn ;      /* mask terms for q (lo output) */
// YO! NEED TO DO SUMMATION BELOW IN SEQUENTIAL ORDER FOR SECURITY !!!!
    assign dm = m ^ n;  /* mask adjustment */
    assign d =  e ^ dm;  /* switch masks: n -> m (hi input mask) */
    FAC_2 dfac(d, df);
    GF_MULS_2  pmul(df, bf, p);
    GF_MULS_2 dnmul(df, nf, dn);
// YO! NEED TO DO SUMMATION BELOW IN SEQUENTIAL ORDER FOR SECURITY !!!!
[...]
```

Those lines can be spotted by our method by running Alg. 1. This method is automatic and extends beyond the verification of S-boxes to any masked combinational logic.

3.4. Validation of the methodology by attacks

In this section, we confirm that the glitch-based side-channel generated by Canright's netlist is indeed exploitable by a first-order attack. We recall that the leakage detection method (Alg. 1) returns an empty list $\mathcal{L}_u$ for unmasked nets, but finds some nets which leak unmasked information (list $\mathcal{L}_g$ is not empty). In this respect, we analyze the Canright netlist code mapped on a SPARTAN6 XC6SLX75 FPGA target. Mapping is obtained using ISE tool from within Xilinx Vivado toolchain.

First of all, digital simulation is carried out with Mentor Graphics Modelsim, without considering timing in the signals (except the clock signal). Synthetic traces are built by collecting, at each clock cycle, the *toggle count* over all signals of the netlist. We perform a Correlation Power Analysis (CPA), using as a model the "Hamming distance" between the netlist consecutive inputs. This leakage model indeed reflects the switching of nets within the netlist. This CPA analysis is first performed, and, without surprise, no leakage is reported. All combinational signals are independent of the secret data, and the synthesizer did not make any optimization that may unmask the secret data. This is consistent with our constraints: we have forced the synthesizer to keep all intermediate signals and the hierarchy of each module, using the attribute "keep".

Second, we add the timing information to the netlist (namely, each gate is annotated with a propagation time), and simulations are re-run. Synthetic traces are regenerated by processing the simulation waveforms as follows:

- a simulation step is selected (1 ps), and
- the trace value at each step is set to be the measured *toggle count* within the past step.

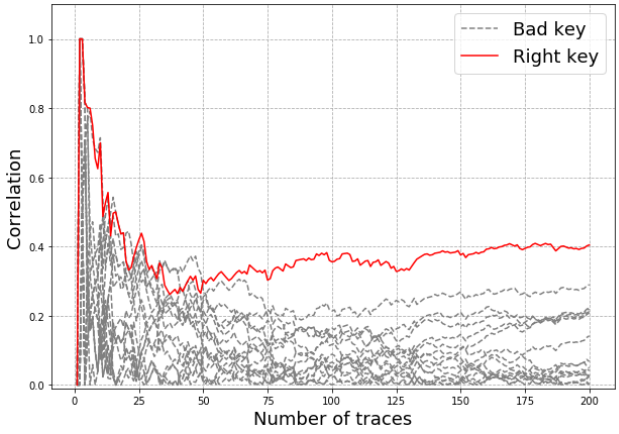


Figure 3: Correlation Power Analysis on the transiently unmasked variables identified within Canright's Sbox's netlist.

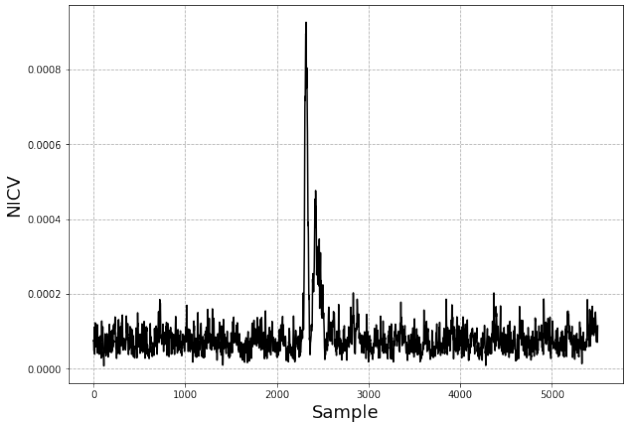


Figure 4: NICV using the unmasked input of the Sbox.

Such traces do exhibit glitches. We subsequently replay the same CPA. As shown in Figure 3, the CPA succeeds in extracting the key, which confirms that Alg. 1 does work. Namely, the working factor for the attack is that the "Hamming distance" model is correlated with the activity of nets belonging to list $\mathcal{L}_g$, when the simulation includes propagation delays. Actually, only 75 traces are sufficient to recover the secret key. This low value can be accounted by the fact the simulation is noise-less. In practice, real measurements bear noise; hence successful key extraction requires more traces.

To check this leakage on real-world traces, we acquired 200,000 electromagnetic captures on an actual FPGA target (SPARTAN6 XC6SLX75 soldered on a SAKURA-G board [18]). We then applied the Normalized Inter-Class Variance (NICV [8]) leakage detection statistical tool using the unmasked Sbox input. The resulting curve is shown in Figure 4; it is clear that the NICV detects a leakage, as there are significant spikes (localized in time, though, around samples $2300 \sim 2500$). This confirms the existence of a leakage. We note that a correlation attack gives the same result, namely, the secret key can be extracted.

4. Aging-Aware Digital Sensor dimensioning to enhance reliability

Cryptographic devices are also vulnerable to fault injection attacks. Referring to Fig. 1, adversaries may perturb the system via injecting faults into the cryptographic devices through environmental changes (or even via injecting targeted faults via laser illuminations) in order to operate the device out of specifications to extract its secret information. To detect such Fault Injection Attacks (FIAs), researchers frequently use Digital Sensors (DS) as a countermeasure [3, 6].

DSs are designed to detect clock / voltage glitches and temperature attacks. Note that they cannot prevent them, but they can detect such attacks and thus prevent leakage of secure data after such detection. DSs are designed based on the designers' preferable range of operation, which covers the whole operating range of the targeted system. In this way, the sensor can ensure system security and integrity even if the system is operated out-of-specification via raising an alarm. For example, let us consider the operating range of voltage and temperature as $[1.0, 1.4]$ V and $[0, 85]$ °C respectively for a cryptographic block. To detect FIAs launched by change of temperature or by a voltage glitch, the deployed DSs should be designed such that the range of operating conditions they can cover is beyond the range in which the system is supposed to operate properly, e.g., $[0.65, 1.5]$ V and temperature as $[-10, 150]$ °C. Note that in practice during their lifetime, devices experience aging-induced changes [5]. As DS are also aged over time, the sensor outcome can be affected during the course of usage. This in turn can introduce security concerns. Thereby as we will discuss below the sensor should be designed such that even if it is aged, its outcome (detecting the faults that affect the cryptographic block it is monitoring) remains accurate. To do so, we need to test the sensor before fabrication (test in its non-conventional form as mentioned earlier) in different operating conditions and aging durations to decide about its dimensions (number of flip-flops and buffers) for fabrication. Additionally, DSs (like other circuits) encounter process variation during the manufacturing process. Given the impact of process variation, it is crucial to calibrate the DSs after fabrication. Thus, we need to consider the impact of process variation by testing the sensor outcome after fabrication, and calibrate it accordingly. In this section, we present our DS dimensioning algorithm and show its validity in real silicon more specifically in FPGAs. We demonstrate the impact of process variation in the DS outcome as well.

4.1. Introduction on Digital Sensor

To break a system by FIA, an adversary may perturb it. Thereby detecting abnormal operating conditions, e.g., change of voltage, temperature, or the frequency at which the

system operates is of utmost importance. To address such security and safety concerns, digital sensors have been broadly deployed in recent years, and have replaced the traditional analog counterparts. Indeed, being designed in full custom layout [34], and accordingly vulnerability to removal attacks due to their identifiability from the intractable sea of gates, the substantial calibration cost, the high power consumption due to their always-on status, and finally, the low failure rate detection due to dealing with physical quantities separately (e.g., voltage alone, temperature alone) make the analog sensors less attractive than the digital opponents [33]. Being part and parcel of the reactive arsenal, DSs must be tested, as they must operate reliably in all corners.

A Digital Sensor (DS) can be realized by inserting a delay chain in the target circuitry. The idea is to implicitly measure the time to propagate a transition (a rising or falling edge) over such a path in different operating conditions. In practice, the propagation time is not really quantified; rather, it is checked if the transition manages to propagate to the end of the delay chain at the considered frequency [15]. Figure 5 shows a sample DS sensor architecture in which a chain of buffers realizes the critical path, and multiple D Flip-Flops (DFFs) sample the delay of the transitions fed from signal a_0, generated by a Toggling DFF (TFF) at the beginning. Based on the operating conditions, i.e., voltage and temperature, as well as clock frequency, the setup time violation occurs in a different sampling DFF. This sensor can be characterized using the so-called Average Flip-Flop Number (AFN) [3], that is extracted based on the flip-flop outputs in each voltage and temperature combination, noted as (V,T) hereafter. What follows discusses the AFN assessment in more detail.

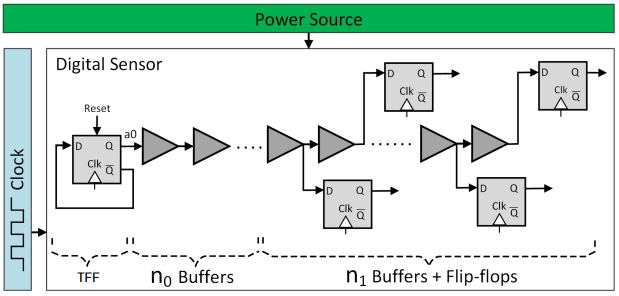


Figure 5: The architecture of the target digital sensor.

In the sensor shown in Figure 5, in each clock cycle C_i, when this sensor is fed with a_0, the first FN_i flip-flops are in phase A (say $0 \rightarrow 1 \rightarrow 0$), and the next flip-flops are in phase $\bar{A}$ (say $1 \rightarrow 0 \rightarrow 1$) where $1 \leq i \leq n_1$, n_1 is the number of DFFs. Here FN_i denotes to the index of the first DFF whose phase is different from its predecessors. For example, the waveform in Figure 6 shows the values of different DFFs of the sensor of Figure 5 with n_0=9 leading buffers followed by n_1=43 buffers and DFFs when operating under (V,T)=(1.2V, 27°C). In this case FN_i is 31 in all clock cycles and accordingly AFN which is considered as the average of the FN_i values would be 31. Indeed averaging FN values over a number of clock cycles is pursued to reduce the effect of unwanted noise. In practice, the AFN value is found to be an appropriate representative of the operating condition. Note that for the conditions under which the circuit operates faster (lower temperature and higher voltage) the AFN gets higher values, while the AFN value is lower when the circuit operates slower.

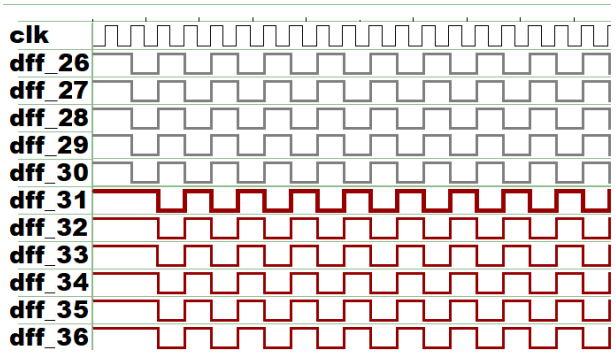


Figure 6: Waveforms of Fig. 5 in (V, T) = (1.2V, 27°C), where $n_0 = 9$ and $n_1 = 43$.

Figure 7 depicts the AFN values in different operating conditions for the sensor shown in Figure 5 with 9 leading buffers and 43 following buffers and DFFs. Note that for the experiments presented in this section, the sensors were implemented at the transistor level using 45 nm NANGATE technology [1]. As clearly shown, the AFN value depends on both voltage and temperature altogether. As expected, the impact of temperature increase can be compensated with the increase of voltage and vice-versa. This can be observed in the trend of AFN value change in different voltage and temperature combinations as well, thus confirming the applicability of the AFN metric in sensing operating conditions. Indeed analog sensors miss this capability by making decisions on raising alarms based on monitoring one physical quantity at a time.

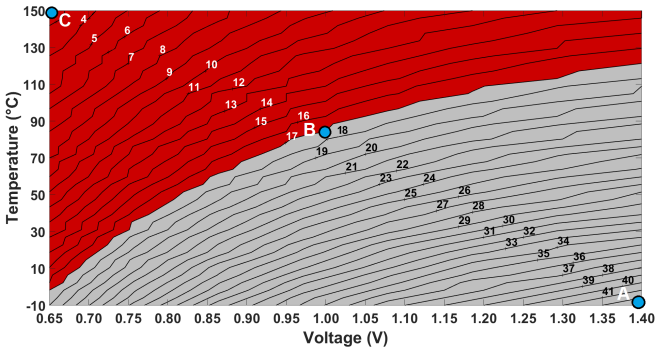


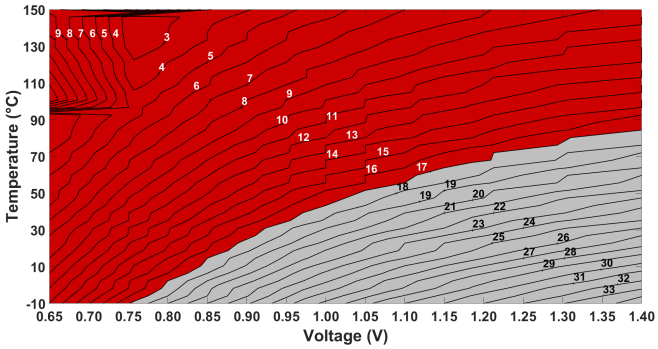
Figure 7: Contour graphs depicting AFN values in different (V,T) conditions for the fresh (age: 0) sensor shown in Fig. 5 where n_0=9 and n_1=43.

We benefit from the sensor's AFN quantity for system's failure detection, and to predict whether the system works properly or not based on the operating conditions. To do so the sensor's AFN value is compared with a pre-defined

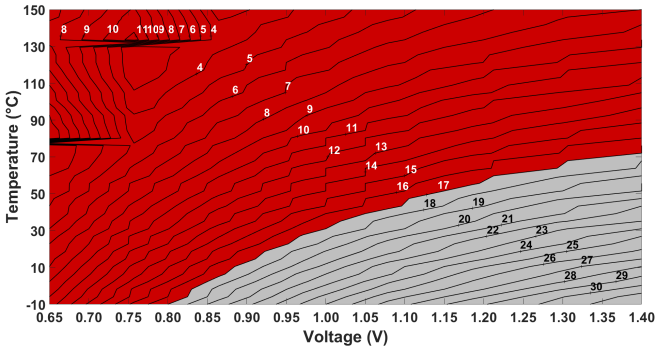
threshold value determined based on the worst-case condition in which the system is expected to work properly, and an alarm is raised in cases that extracted AFN is lower than the threshold value relates to the worst-case condition. We assume the worst-case condition as (V,T)=(1.0V, 85°C) for the sensor we implemented here. As Figure 7 shows, the AFN in this condition is 17. Thus an alarm is raised for the cases where $AFN < 17$; shown in red in the figure depicting that the circuit operates slower than expected, while the grey area shows the conditions considered as safe. It is noteworthy to mention that this threshold is tuned based on the application and user's configuration.

Indeed chips are designed in different temperature grades (e.g., commercial, industrial, military, etc.), i.e., a different range of temperatures under which it is expected to work properly. Thereby to realize a sensor (similar to the one shown in Figure 5) that can cover the whole expected range of operating conditions, it is required to have a well-defined architecture in terms of the number of buffers and DFFs that the sensor includes what we call the sensor dimension hereafter.

Note that although digital sensor's data is sensitive, and protection is needed to prevent side-channel analysis attacks using this data, such data is not publicly available. From a system-level point of view, this data is available to the system bus, typically addressed by privileged instructions, and unless the privileges are escalated by the attacker, it will not be possible to access such data from a digital sensor.



(a) 4-year old sensor



(b) 7-year old sensor

Figure 8: Contour graphs depicting AFN values in different (V,T) conditions for the 4- and 7- year old sensors shown in Figure 5 where n_0=9 and n_1=43.

4.2. Digital Sensor dimensioning

We have presented an algorithm for sensor dimensioning in our prior work (Algorithm 1 in [3]), which determines the number of DFFs and buffers embedded in the sensor based on the "Best" and the "Worst" Case conditions the circuit is supposed to work properly (points *A* and *B* in Figure 7 in our case). Here *A* and *B* are examples of "Best" and "Worst" case points. Note that, without loss of generality, based on any range of operating conditions, we can dimension the sensor (using Alg. 2) such that it detects the fault attacks accurately in such a range. Deploying our prior algorithm (refer to [3] for more details) for dimensioning the sensor in Figure 5 realized using 45 nm NANGATE technology while considering the "Best" and "Worst" conditions as (1.0V,85°C) and (1.4V,-10°C), respectively recommends embedding n_0=9 leading buffers followed by n_1=43 buffers and DFFs. Although such dimensioning fits the sensor's expected operating range well, it fails to consider aging effects occurring during the circuit lifetime.

In practice, the electrical behavior of the transistors embedded in the deployed DS (similar to other CMOS circuits) deviates from the original one during the sensor lifetime. This deviation, so called aging, results in the delay increase for the gates embedded in the sensor. To show the necessity of considering aging degradation when dimensioning the sensor, Figure 8(a) and Figure 8(b) depict the AFN evolution for the same sensor after 4 and 7 years of aging, respectively. As expected, the sensor circuitry becomes slower with aging, thus the AFN value decreases over time for the same operating condition. This can be observed as a shift of the red zone in Figure 8(b) compared to Figure 8(a) and Figure 7. Another important observation is the trend of AFN value change in the aged sensors shown in Figure 8(a) and Figure 8(b) when operating under high temperature and low voltage combinations. In these cases, as the sensor becomes slower and slower with aging, the AFN value may not be reliable, i.e., the sensor may need more DFFs to be able to correctly sample the setup time violation occurring in the buffer chain. To alleviate this problem, we improved the dimensioning algorithm presented in [3] by considering aging effects. The new algorithm is depicted below as Algorithm 2. As shown, the number of buffers and DFFs is decided based on the "Best" operating condition (point *A* when the sensor is fresh) along with the "Worst Non-Functional" condition (point *C* for the L-year old sensor where *L* is the expected lifetime; *L* is assumed to be 7 in this paper). Note that point *C* denotes the worst operating condition that the circuit may experience but is beyond its range of proper operation. Dimensioning the sensor based on the AFN value it experiences in point C results in a reliable and accurate outcome over the course of usage. In other words, sensors' results remain accurate even when aged.

Algorithm 2 shows how we dimension the DS by testing its outcome (via monitoring its included flip-flops' values) in different operating conditions (voltage, temperature) and

Algorithm 2: Aging Aware DS Dimensioning algorithm

input : Design kit for the target technology, desired clock period, safety margin of K buffers

output: Sensor dimensions n_0 and n_1; values to be used for architecturing the sensor aiming at failure detection during run time

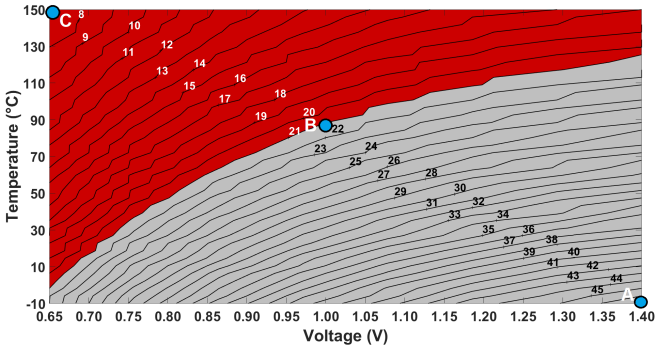
1 Build a netlist consisting of a DFF which samples its inverted output, and feeding an infinite chain of buffers; each buffer feeds also a separate flip-flop
2 Set the conditions to `Non-Functional worst case` (e.g., slow process, high temperature, low voltage, maximum expected age) — point, **C** in Figure 7
3 Determine the position (N) of first sampling inversion error by aging simulation for maximum expected lifetime
4 Remove the Flip-flops connected to the first N buffers
5 Set the conditions to `best case` (e.g., fast process, low temperature, high voltage, No age (i.e., age:0)) — point **A** in Figure 7
6 Determine the position (AFN_high) of first sampling inversion error
7 **return** (n_0 = N − K, n_1 = AFN_high − n_0 + K)

expected lifetime. In this algorithm, we first consider a chain of infinite number of buffers each feeding a flip-flop and then trim the circuit based on the operating conditions that the circuit may experience. By "aging" simulation of this chain of buffers and flip-flops under the "Worst" case condition that the circuit may experience (not necessarily working properly at this condition; called "Worst Non-Functional" condition earlier) the number of leading buffers is decided. Note that the "aging" simulation is performed assuming the longest expected lifetime (e.g., 7 years under a high aging stress). We use the HSpice MOSRA for aging simulations. Then we decide about the number of following DFFs and buffers by considering the "Best" case operating condition for the sensor. Note that the calculations are done based on simulation using the same technology libraries that will eventually realize the sensor. Thereby, we consider a safety margin including "K" to account for process variations.

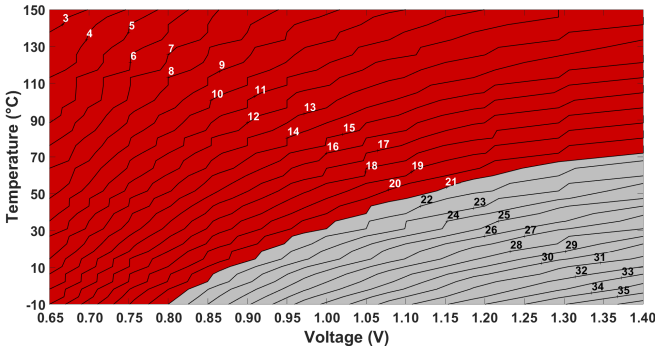
Applying Alg. 2 to the sensor shown in Figure 5 recommends embedding n_0=4 leading buffers followed by n_1=48 buffers and DFFs. The related contour graphs for the fresh (age:0) and the 7-year old sensors with this dimension are shown in Figure 9(a) and 9(b), respectively. As illustrated, by considering the aging effects in Alg. 2, the trend of AFN values is as expected even when the circuit is aged.

4.3. Validation on FPGA: Proof of Concept

We validated the dimensioning algorithm of the digital sensor based on the proposed algorithm on two SPARTAN6 XC6SLX75 FPGAs soldered on a SAKURA-G board [18], with Xilinx ISE 14.7 software. The goal is to investigate if calibration is needed after dimensioning the sensor in the design phase when the sensor is implemented using the same mask design to realize different chips (here on FPGA). In



(a) Fresh (Age:0) sensor



(b) 7-year old sensor

Figure 9: Contour graphs depicting AFN values in different (V,T) conditions for the fresh and 7-year old sensors shown in Figure 5 where n_0 = 4 and n_1 = 48.

other words, we opt to show the impact of process variation on the sensor outcome.

4.3.1. FPGA Implementation of Digital Sensors

To design a digital sensor on FPGA, a manual place and route is employed. This is crucial as the sensor outcome relates to the delay of the buffer chain included in the sensor thus having almost the same delay between each buffer and its related DFF is required. As shown in Figure 5 a digital sensor includes three basic components: A Toggling DFF (TFF), a set of DFF, and a set of Buffers. In the first step of implementing a digital sensor in FPGA, three different hard-macros (a circuit which is already placed and routed on FPGA) need to be designed; a TFF for generating $a0$ signal, a Buffer to be used in the initial chain, and a Buffer-DFF to be used in sampling chain. Indeed, using a hard-macro ensures that specific DFF and LUTs of each slice are used to realize one buffer and its related flip-flop in the digital sensor; thereby following a balanced place & route for the whole sensor. Then in the next step, by instantiating a TFF hard-macros, n_0 Buffer hard-macros, and n_1 Buffer-DFF hard-macros the hard-macro of the digital sensor is generated. This hard-macro is called *main hard-macro* hereafter.

While designing the sensor, we followed the steps from Algorithm 2. In these experiments, we considered the voltage range from $0.8V$ to $1.3V$ and temperature as room temperature for our system (we do not have temperature change in our FPGA experiments). For this range of voltage,

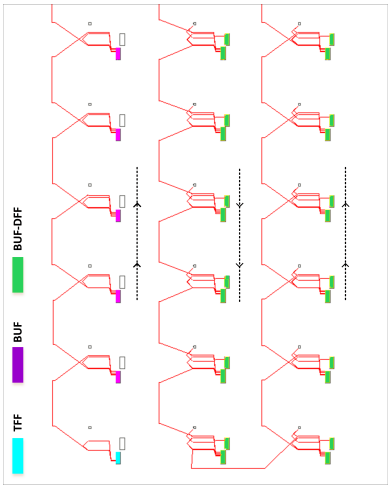


Figure 10: Manual placement & routing of the digital sensor using the Xilinx ISE 14.7 FPGA Editor.

we needed n_o=7 leading buffers and n_1=54 sampling DFFs and related buffers. Note that this dimensioning is different from what we showed for simulation results earlier, as the technology is different, and also the range of operating conditions is different.

Figure 10 shows the partial floor plan of our sensor. The TFF hard-macro is shown in red, followed by the leading buffers depicted in green. Each green slice implements one buffer realized via two back-to-back inverters. In our implementation, all leading buffers are placed such that we have the same routing from one buffer to the next; therefore, the same routing delays between each two consecutive buffers. After implementing the leading buffer chain, the sampling chain is inserted by using the Buffer-DFF hard-macros. The Buffer-DFF hard-macro (a pair of blue slices) is implemented using two back-to-back slices where the left slice includes the Buffer and the right one implements the related DFF. Similar to the leading buffers, in the sampling chain, we make sure that each DFF-Buffer combination has the same distance from the one it feeds; thus similar routing delays between them. In our implementation, each sensor spans in local zones to ensure that each component residing in each sensor experiences the same power and clock variations. Note that different sensors may experience a slightly different IR drop as located in different parts of the FPGA.

As devised in Alg. 2, we considered a safety margin (here K=4) in our dimensioning to warrant that the DS works properly even in the presence of process variations.

4.3.2. Intra- vs inter-die variation of Digital Sensor

To analyze the impact of both intra-die and inter-die variations on the sensor outcome, we deployed 2 FPGAs and implemented 8 digital sensors with identical *main hard-macro* in each as depicted in Figure 11. Note that we used the same bitstream to program both FPGAs, thus the sensors are placed in the same locations in both FPGAs. As shown, these 8 identical sensors are implemented in two columns

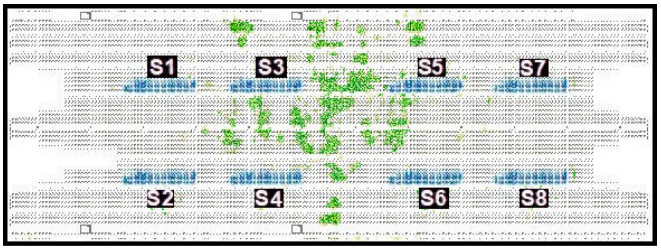
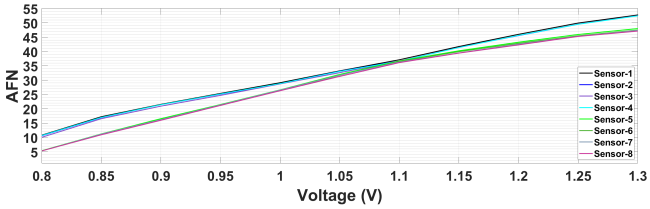
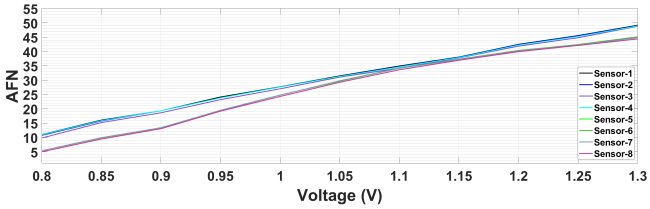


Figure 11: On-chip implementation of Digital sensors in FPGA.

and four rows (the figure has been rotated for the sake of space). Here the 8 identical sensors are implemented solely to demonstrate the impact of intra-die process variation in our sensor outcome. Note that, the number of sensors, and the location they are resided in a chip (to detect FIAs) depend on the number of *sensitive blocks* we want to monitor and their placement in the chip (refer to fig. 19 of [4]).



(a) FPGA 1

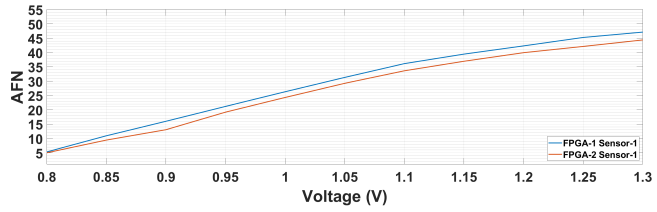


(b) FPGA 2

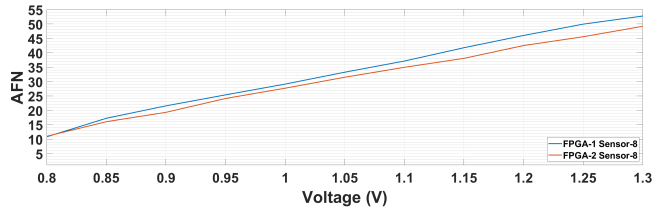
Figure 12: AFN for the 8 digital sensors implemented in two different FPGAs using the same bitstream. All sensors in each FPGA were implemented via the same hard-macros.

Figure 12(a) shows the AFN index for each of the 8 implemented sensors when running the sensor for 16 clock cycles under different voltages, mainly in the range of 0.8V to 1.3V with steps of 0.05V. As expected, the AFN index increases in higher voltages. Another observation that can be made from this figure is that sensors 1 to 4 mainly follow a similar voltage-induced AFN change, and sensors 5 to 8 follow another trend. However, for V=1.1V, the AFN is very similar for all sensors. This is because FPGAs mainly experience low IR drop variations when operating under their designed typical voltage. This trend is slightly different for lower/higher voltages where the IR drop induced changes can vary in different zones of FPGA. This can be the reason for sensors 1 to 4 behaving similarly and different from sensors 5 to 8. Similar observations can be made for FPGA-2 (Figure 12(b)).

Figure 13 compares the variation of sensors' AFN index in various voltage quantities vis-a-vis for two sensors in each FPGA, in particular Sensor-1 and Sensor-8. As depicted in



(a) Sensor 1



(b) Sensor 8

Figure 13: Inter-die variations of sensors' AFN index in various voltages.

Table 2
Average intra- and inter-process variation induced change of AFN in different implemented sensors.

Category	Intra		Inter
	FPGA-1	FPGA-2	FPGA-(1 & 2)
V∈[0.8V,1.3V]	2.1	2.0	3.0
V = 1.1V	0.3	0.5	2.3

both cases, FPGA-1 AFN Index is higher than the FPGA-2. This is also true for the other 6 sensors (not shown for the sake of space). This truly shows the deterministic trend of inter-die process variations.

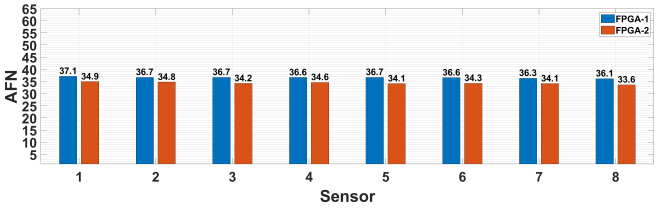


Figure 14: Intra- and Inter-die variations of AFN Index in voltage of 1.1V in 2 FPGAs.

Figure 14 shows the AFN index of all 8 sensors in each FPGA for the voltage of 1.1V. As depicted in this voltage, the *maximum* AFN index variations among different sensors in FPGA-1 is 1 related to the difference of sensor-1 and sensor-8 (due to the intra-die variations). This value is 1.3 for FPGA-2. This confirms the negligible impact of intra-die process variations in the AFN index when operating at the typical voltage of 1.1V. Moreover, the maximum inter-die variation is ≈2.5 related to the difference of sensor-8 in FPGA-1 and FPGA-2 (the same for sensor-3).

Table 2 assesses the effect of process variations in the sensors' outcome in more detail. The results are shown for the voltage of 1.1V as well as the whole range of voltage we considered in this study. On average, the AFN index changes 2.1 unit for FPGA-1 and 2.0 for FPGA-2 when considering the whole voltage pane. However, the process variation effect is very low for voltage 1.1V, where on average intra-die variation for AFN index is 0.3 for FPGA-1 and 0.5 for FPGA-2. For inter-die process variation, these values are changed to 2.3 and 3.0 on average for the voltage of 1.1V and the whole voltage range, respectively.

The takeaway points from these experiments are that process variation effect on the sensor's outcome is not high, thus we need a low-cost calibration after the fabrication. Indeed the effect of process variation on AFN index is low while the chip is designed with symmetric power lines.

In addition, as discussed earlier, considering the safety margin K in sensor dimensioning (recall Algorithm 2) is highly crucial as without such consideration, the process variation may result in an incorrect capture of AFN index in harsh environments (very high/low temperatures or voltage). Finally, the results confirm that dimensioning algorithm (Algorithm 2) is valid for the real-silicon implementation. Note that the FPGA results were extracted for a new device, and we leave the impact of aging on FPGA implementations of sensor for our future work. We assume that ASIC follows the very same trend as FPGA regarding our sensor dimensioning. We will implement the DS in ASIC to validate our findings in our future project.

5. Conclusion

This paper demonstrates that security primitives require specific tests to ensure a high level of security. Emblematic examples of properties to test are related to hostile environment and threats, e.g., randomness quality, information leakage level, and aging mitigation. The random variable generation, as provided by the TRNG for dynamic variable, and PUF for device fingerprint, requires a validation by statistical tests to ensure a sufficiently large lower bound on the amount of entropy. PUF requires more complex tests, as it can be biased by the circuit layout and damaged by dynamic noise. The masking countermeasure is an efficient method to protect hardware implementation of cryptographic blocks against SCA. But it is necessary to avoid glitches which can unmask the sensitive values. This paper proposes a netlist-level test algorithm to automatically detect nets which could leak secret information via glitches. The detection of FIA by DS requires an accurate test to dimension the sensor(s). It is shown that it is important to take into account the aging when dimensioning the DS, in order to enhance the reliability of detection over time. All these tests have been carried out and validated on real-silicon (FPGAs). They prove that the proposed methodology is applicable to these three security primitives.

Acknowledgment

This work was partly funded by the Federal Ministry of Education and Research (BMBF) under grant no. 16KIS1389K and the Agence Nationale de la Recherche (ANR) under grant ANR-20-CYAL-0007 in the project APRIORI.

References

[1] Nangate 45nm open cell library. "http://www.nangate.com".

[2] Information security, cybersecurity and privacy protection - Physically unclonable functions - Part 2: Test and evaluation methods. Standard, ISO / IEC, March 2021.

[3] Md Toufiq Hasan Anik, Jean-Luc Danger, Sylvain Guilley, and Naghmeh Karimi. Detecting failures and attacks via digital sensors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(7):1315–1326, 2021.

[4] Md Toufiq Hasan Anik, Mohammad Ebrahimabadi, Jean-Luc Danger, Sylvain Guilley, and Naghmeh Karimi. Reducing aging impacts in digital sensors via run-time calibration. *Journal of Electronic Testing*, 37(5):653–673, 2021.

[5] Md Toufiq Hasan Anik, Sylvain Guilley, Jean-Luc Danger, and Naghmeh Karimi. On the effect of aging on digital sensors. In *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, pages 189–194, 2020.

[6] Toufiq Hasan Anik, Jean-Luc Danger, Omar Diankha, Mohammad Ebrahimabadi, Christoph Frisch, Sylvain Guilley, Naghmeh Karimi, Michael Pehl, and Sofiane Takarabt. Testing and reliability enhancement of security primitives. In *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–8. IEEE, 2021.

[7] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Hecker, and D. Banks. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010.

[8] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. In *International Symposium on Electromagnetic Compatibility (EMC '14 / Tokyo)*. IEEE, May 12-16 2014. Session OS09: EM Information Leakage. Hitotsubashi Hall (National Center of Sciences), Chiyoda, Tokyo, Japan.

[9] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably Secure Masking of AES. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.

[10] Nicolas Bruneau, Jean-Luc Danger, Adrien Facon, Sylvain Guilley, Soshi Hamaguchi, Yohei Hori, Yousung Kang, and Alexander Schaub. Development of the Unified Security Requirements of PUFs During the Standardization Process. In Jean-Louis Lanet and Cristian Toma, editors, *Innovative Security Solutions for Information Technology and Communications - 11th International Conference, SecITC 2018, Bucharest, Romania, November 8-9, 2018, Revised Selected Papers*, volume 11359 of *Lecture Notes in Computer Science*, pages 314–330. Springer, 2018.

[11] David Canright. A Very Compact S-Box for AES. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.

[12] Z. Cherif, J-L. Danger, S. Guilley, and L. Bossuet. An easy-to-design puf based on a single oscillator: the loop puf. In *2012 15th Euromicro Conference on Digital System Design*, pages 156–162. IEEE, 2012.

[13] Frisch Christoph and Michael Pehl. Beware of the bias – statistical performance evaluation of higher-order alphabet pufs. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2022.

[14] David Leon Gil (coruus). AES S-Box. "https://github.com/coruus/canright-aes-sboxes/blob/master/verilog/sboxmaskcorr.verilog".

[15] Mohammad Ebrahimabadi, Md Toufiq Hasan Anik, Jean-Luc Danger, Sylvain Guilley, and Naghmeh Karimi. Using digital sensors to leverage chips' security. In *Physical Assurance and Inspection of Electronics (PAINE)*, pages 1–6, 2020. DOI: 10.1109/PAINE49178.2020.9337730.

[16] Kathrin Garb, Marvin Xhemrishi, Ludwig Kürzinger, and Christoph Frisch. The wiretap channel for capacitive puf-based security enclosures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022.

[17] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727, pages 63–80. Springer, Heidelberg, 2007.

[18] Hendra Guntur, Jun Ishii, and Akashi Satoh. Side-channel AttacK User Reference Architecture board SAKURA-G. In *IEEE 3rd Global Conference on Consumer Electronics, GCCE 2014, Tokyo, Japan, 7-10 October 2014*, pages 271–274. IEEE, 2014.

[19] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh. Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs*, RECONFIG '10, pages 298–303, Washington, DC, USA, 2010. IEEE Computer Society.

[20] Vincent Charles Immler. *Higher-order alphabet physical unclonable functions*. PhD thesis, Technische Universität München, 2019.

[21] Vincent Charles Immler, Johannes Obermaier, Kuan Kuan Ng, Fei Xiang Ke, JinYu Lee, Yak Peng Lim, Wei Koon Oh, Keng Hoong Wee, and Georg Sigl. Secure physical enclosures from covers with tamper-resistance. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):51–96, Nov. 2018.

[22] Vincent Charles Immler and Karthik Uppund. New insights to key derivation for tamper-evident physical unclonable functions. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):30–65, May 2019.

[23] W. Killmann and W. Schindler. A proposal for: Functionality classes for random number generators version 2.0, 2011.

[24] W. Killmann and W. Schindler. A proposal for: Functionality classes for random number generators version 2.35 – draft, 2022.

[25] P. L'Ecuyer and R. Simard. Testu01: Ac library for empirical testing of random number generators. *ACM TOMS*, 33(4):1–40, 2007.

[26] D. Lim, J. Lee, B. Gassend, G.E. Suh, Ma. Van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.

[27] A. Maiti, V. Gunreddy, and P. Schaumont. *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*, pages 245–267. Springer New York, 2013. DOI: 10.1007/978-1-4614-1362-2_11.

[28] Holger Mandry, Andreas Herkle, Sven Müelich, Joachim Becker, Robert FH Fischer, and Maurits Ortmanns. Normalization and multi-valued symbol extraction from ro-pufs for enhanced uniform probability distributions. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(12):3372–3376, 2020.

[29] Amir Moradi and Oliver Mischke. Glitch-free implementation of masking in modern fpgas. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2012, San Francisco, CA, USA,*, pages 89–95. IEEE, 2012.

[30] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.

[31] M. Pehl, A. Punnakkal, M. Hiller, and H. Graeb. Advanced performance metrics for physical unclonable functions. In *International Symposium on Integrated Circuits (ISIC)*. IEEE, 2014.

[32] Michael Pehl, Tobias Tretschok, Daniel Becker, and Vincent Charles Immler. Spatial Context Tree Weighting for Physical Unclonable Functions. In *2020 ECCTD*, pages 1–4. IEEE, 2020.

[33] Nidhal Selmane, Shivam Bhasin, Sylvain Guilley, and J-L Danger. Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks. *IET information security*, 5(4):181–190, 2011.

[34] D. Shahrjerdi, J. Rajendran, S. Garg, F. Koushanfar, and R. Karri. Shielding and securing integrated circuits with sensors. In *ICCAD*, pages 170–174, 2014.

[35] M. Sonmez, E. Barker, J. Kelsey, K. McKay, M. Baish, and M. Boyle. Recommendation for the entropy sources used for random bit generation, 2018-01-10 2018.

[36] E. Strieder, C. Frisch, and M. Pehl. Machine learning of physical unclonable functions using helper data - revealing a pitfall in the fuzzy commitment scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2), 2021.

[37] Sofiane Takarabt, Sylvain Guilley, Youssef Souissi, Khaled Karray, Laurent Sauvage, and Yves Mathieu. Formal Evaluation and Construction of Glitch-resistant Masked Functions. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2021, Tysons Corner, VA, USA, December 12-15, 2021*, pages 304–313. IEEE, 2021.

[38] Lars Tebelmann, Jean-Luc Danger, and Michael Pehl. Self-secured puf: protecting the loop puf by masking. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 293–314. Springer, 2020.

[39] E. Barker. NIST FIPS SP 800-57 Part 1 Rev. 5: Recommendation for Key Management: Part 1 – General, May 2020. DOI: https://doi.org/10.6028/NIST.SP.800-57pt1r5.

[40] Marc Joye and Michael Tunstall, editor. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012. ISBN: 978-3-642-29655-0; DOI: 10.1007/978-3-642-29656-7.

[41] Riehm, Carl and Frisch, Christoph and Burcea, Florin and Hiller, Matthias and Pehl, Michael and Brederlow, Ralf. Structured Design and Evaluation of a Resistor-Based PUF Robust Against PVT-Variations. In *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. IEEE, 2023.

[42] Paulus Adrianus Jozef Volf. *Weighting techniques in data compression: Theory and algorithms*. Technische Universiteit Eindhoven, 2002.

[43] F. Wilde, C. Frisch, and M. Pehl. Efficient bound for conditional min-entropy of physical unclonable functions beyond iid. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2019.

[44] F. Wilde, B. Gammel, and M. Pehl. Spatial correlation analysis on physical unclonable functions. *IEEE Transactions on Information Forensics and Security*, 13(6):1468–1480, 2018.

[45] F. Wilde, M. Hiller, and M. Pehl. Statistic-based security analysis of ring oscillator pufs. In *2014 ISIC*, pages 148–151. IEEE, 2014.

[46] F. Wilde and M. Pehl. On the confidence in bit-alias measurement of physical unclonable functions. In *2019 17th IEEE NEWCAS*, pages 1–4. IEEE, 2019.

[47] M. Yu and S. Devadas. Recombination of physical unclonable functions. *35th Annual GOMACTech Conference*, 2010.

A. Equivalent formulation of the properties 1 and 2

Proposition 1 expresses the security requirements in statistical terms. We can reformulate it using Boolean functions:

Lemma 1 (Mathematical formulation of Prop. 1)**.** *Let* $F : \mathbb{F}_2^{3k} \to \mathbb{F}_2$. *$F$ satisfies Property 1 if and only if:*

$$\forall x \in \mathbb{F}_2^k, \sum_{m_i \in \mathbb{F}_2^k} \sum_{m_o \in \mathbb{F}_2^k} (-1)^{F(x \oplus m_i, m_i, m_o)} = 0.$$

Proof. The three random variables are M_i, M_o and X. We know that M_i and M_o are independent and uniformly distributed.

Notice that for a Boolean variable Y, $\mathbb{P}(Y = 1) = E(Y)$.

Let one value of x. We have:

$$\begin{aligned}
&\mathbb{P}(F(X \oplus M_i, M_i, M_o) = 1 | X = x) \\
&= \mathbb{P}(F(x \oplus M_i, M_i, M_o) = 1) \\
&= E_{M_i, M_o}(F(x \oplus M_i, M_i, M_o)) \\
&= \frac{1}{2^{2k}} \sum_{m_i, m_o} F(x \oplus M_i, M_i, M_o). \qquad (1)
\end{aligned}$$

Symmetrically,

$$\begin{aligned}
&\mathbb{P}(F(X \oplus M_i, M_i, M_o) = 0 | X = x) \\
&= \mathbb{P}(F(x \oplus M_i, M_i, M_o) = 0) \\
&= E_{M_i, M_o}(1 - F(x \oplus M_i, M_i, M_o)) \\
&= \frac{1}{2^{2k}} \sum_{m_i, m_o} (1 - F(x \oplus M_i, M_i, M_o)). \qquad (2)
\end{aligned}$$

Now, (1) is equal to (2) if and only if (iff):

$$\sum_{m_i, m_o} F(x \oplus M_i, M_i, M_o) = \sum_{m_i, m_o} (1 - F(x \oplus M_i, M_i, M_o))$$

i.e., iff

$$\sum_{m_i, m_o} (1 - 2F(x \oplus M_i, M_i, M_o)) = 0.$$

We can note also that:

$$\sum_{m_i, m_o} (1 - 2F(x \oplus M_i, M_i, M_o) = \sum_{m_i, m_o} (-1)^{F(x \oplus M_i, M_i, M_o)}.$$

□

Corollary 1. *Let us note F as the sum of monomials f_j.*
$F(a, m_i, m_o) = \sum_{j=1}^{p} f_j(a, m_i, m_o)$, since $f_j \in \mathbb{F}_2$ then F can be written as
$F(a, m_i, m_o) = \sum_{j=1}^{p} \frac{1}{2}(1-(-1)^{f_j(a, m_i, m_o)}) = \frac{1}{2}(p - \sum_{j=1}^{p}(-1)^{f_j(a, m_i, m_o)})$ and satisfies the Property 1 iff

$$\sum_{j=1}^{p} (\sum_{m_i, m_o} (-1)^{f_j(a, m_i, m_o)}) = 2^{2k}(p-1).$$

Proof. F satisfies the Property 1 iff $\sum_{m_i, m_o} F(x \oplus m_i, m_i, m_o) = 2^{2k-1}$ i.e $\sum_{m_i, m_o} (p - \sum_{j=1}^{p} (-1)^{f_j(m_i, m_o)}) = 2^{2k}$. □

Remark 1. *If F satisfies the Property 1 then the Walsh transformation of F is null at zero: $W_F(0) = 0$ where*

$$W_F(u, v, w) = \sum_{a, m_i, m_o} (-1)^{u \cdot a + v \cdot m_i + w \cdot m_o + F(a, m_i, m_o)}.$$

Indeed,

$$\begin{aligned}
W_F(0) = W_F(0, 0, 0) &= \sum_{a, m_i, m_o} (-1)^{F(a, m_i, m_o)} \\
&= \sum_{a} \underbrace{\sum_{m_i, m_o} (-1)^{F(a, m_i, m_o)}}_{=0} = 0.
\end{aligned}$$

Corollary 2 (Mathematical formulation of Prop. 2)**.** *Let* $F : \mathbb{F}_2^{3k} \to \mathbb{F}_2$. *$F$ satisfies Property 2 if and only if:*

$$\begin{aligned}
&\forall x \in \mathbb{F}_2^k, \forall \delta \in \mathbb{F}_2^{3k} \backslash \{0\}, \\
&\sum_{m_i \in \mathbb{F}_2^k} \sum_{m_o \in \mathbb{F}_2^k} (-1)^{F((x \oplus m_i, m_i, m_o) \oplus \delta) \oplus F(x \oplus m_i, m_i, m_o)} = 0.
\end{aligned}$$

Proof. Pose $G_\delta(A, M_i, M_o) = F((x \oplus m_i, m_i, m_o) \oplus \delta) \oplus F(x \oplus m_i, m_i, m_o)$
One has,

$$\begin{aligned}
&\mathbb{P}(G_\delta(A, M_i, M_o) = 1 | X = x) \\
&= \mathbb{P}(G_\delta(X \oplus M_i, M_i, M_o) = 1 | X = x) \\
&= \mathbb{P}(G_\delta(x \oplus M_i, M_i, M_o) = 1) \\
&= E_{M_i, M_o}(G_\delta(x \oplus M_i, M_i, M_o)
\end{aligned}$$

$$= \frac{1}{2^{2k}} \sum_{m_i, m_o} G_\delta(x \oplus M_i, M_i, M_o).$$

F satisfies Property 2 iff

$$\mathbb{P}(G_\delta(A, M_i, M_o) = 1 | X = x) =$$
$$\mathbb{P}(G_\delta(X + M_i, M_i, M_o) = 1 | X = x) = \frac{1}{2}$$

iff

$$\frac{1}{2^{2k}} \sum_{m_i, m_o} G_\delta(x \oplus M_i, M_i, M_o) = \frac{1}{2}$$

iff

$$\sum_{m_i, m_o} (\frac{1 - (-1)^{G_\delta(x \oplus M_i, M_i, M_o)}}{2}) = 2^{2k-1}$$

iff

$$\sum_{m_i, m_o} (\frac{1}{2}) - \frac{1}{2} \sum_{m_i, m_o} (-1)^{G_\delta(x \oplus M_i, M_i, M_o)} = 2^{2k-1}$$

that means

$$\sum_{m_i, m_o} (-1)^{G_\delta(x \oplus M_i, M_i, M_o)} = 0.$$

□

Therefore, proving the security of masked cryptographic circuits in the presence of glitches amounts to computing Walsh transforms.