2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) | 978-1-6654-1609-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/DFT52944.2021.9568297

Testing and Reliability Enhancement of Security Primitives

Md Toufiq Hasan Anik[§], Jean-Luc Danger[*], Omar Diankha,[¶], Mohammad Ebrahimabadi[§], Christoph Frisch[‡], Sylvain Guilley[†], Naghmeh Karimi[§], Michael Pehl[‡], Sofiane Takarabt[†]
[*]LTCI, Télécom Paris, Institut polytechnique de Paris, France
[†]Secure-IC, France
[‡]Technical University of Munich, Germany, Department of Electrical and Computer Engineering
[§]University of Maryland Baltimore County, United States
[¶]Paris 8 University, France

***Abstract*—The test of security primitives is particularly strategic as any bias coming from the implementation or environment can wreck havoc on the security it is intended to provide. This paper presents how some security properties are tested on leading primitives: True Random Number Generation (TRNG), Physically Unclonable Function (PUF), cryptographic primitives and Digital Sensor (DS). 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 side-channel analysis (SCA) of cryptographic primitives, like the substitution box in symmetric cryptography, is generally ensured by masking. But the hardware implementation of masking can be damaged by glitches, which create leakages on sensitive variables. A test method is to search for nets of the cryptographic netlist, which are vulnerable to glitches. The DS is an efficient primitive to detect disturbances and rise alarms in case of fault injection attack (FIA). The dimensioning of this primitive requires a precise test to take into account the environment variations including the aging.**

Keywords: Test, PUF, TRNG, SCA, FIA, Digital Sensor, .

I. 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 long in digital circuits, their suitability for security functions appear to be insufficient. Indeed, those techniques only assess the correct functional behavior, but fail to test security functionalities.

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 attack sensors. Cryptographic keys must be unpredictable, hence estimation of noise in the chip shall be made possible. Besides, cryptographic key management shall be secure against side-channel attacks, such as those exploiting masking countermeasures. But 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.

978-1-6654-1609-2/21/$31.00 ©2021 IEEE

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.

a) Contributions: In this paper, we aim at providing a 360 degree overview of tests related to security chips. Such information is usually only available in specialized publications, or even worst, is not publicly discussed (since it is test labs secret know-how). We detail the nature of the tests for three classes of security functions, namely:

1) Functions managing the randomness, namely TRNGs and PUFs;
2) Cryptographic algorithms, whose resistance to side-channel attacks shall be extensively proven;
3) Physical perturbation sensors, who should keep a constant calibration to take into account aging.

b) Outline: The rest of the paper is structured as follows. The question of entropy testing is tackled in Sec. II. Testing for harmfulness of glitches in side-channel protections is the topic of Sec. III. The validation of digital sensors across aging is discussed in Sec. IV. Eventually, Sec. V concludes the paper.

II. Evaluation of PUFs and TRNGs

Secure devices require different kinds of randomness: On the one hand, a secret key is required, which is random but constant over time, on the other hand, randomness, such as for numbers used once, must differ every time when sampled. The first kind of randomness is in some scenarios nowadays provided through a PUF, which derives a secret

from process variations, the second is normally based on a TRNG, which derives the randomness from noise. Still, both kinds of randomness have to be unpredictable for an attacker. While already testing the pure operation of a module for key storage or to generate random numbers without introducing a backdoor is hard, testing for sufficient randomness adds another level of complexity. In this paper we exclusively focus on offline tests that are based on extracted data; Build In Self Tests (BISTs) and online tests are out of scope for this work.

A. Fundamental Differences in Statistical Properties

On a first glance both, PUFs and TRNGs, output random numbers. But it is a harmful approach to use the same tests for both sorts of randomness since it ignores significant differences in the nature of the random numbers generated from PUFs and TRNGs:

Source of Randomness: The most important difference between PUF and TRNG is from where both retrieve randomness. The randomness of a PUF stems from the manufacturing process and ideally is invariant over time. For a TRNG, noise provides the randomness. Subsequent differences are implications of this aspect.

Amount of Data: For a thorough statistical testing lots of samples are needed to increase the significance of the results. For TRNGs the amount of data is, despite of latency, unlimited since the source of randomness (the noise) is different at any point in time. In contrast, PUFs can provide only a limited amount of bits, since the source of randomness (process variations) is fixed after production of the chip. The amount of bits derived from these variations depends on the type of PUF. Typically, for PUFs storing a key, the output is one bit per PUF cell, e.g., for an SRAM PUF [1], up to few ten bits per PUF cell, like for a Loop PUF [2], and multiple PUF cells are used per chip[1]. Consequently, many devices must be manufactured to obtain a large enough sample size for testing.

Dimension of data: For PUF data, various dimensions must be tested as opposed to a one-dimensional bitstream of a TRNG. To illustrate this, note that a typical TRNG outputs all bits from a single source. In contrast, the following PUF dimensions exist: (i) For many PUFs multiple cells are combined to derive a key of sufficient length. Thus, already two dimensions must be tested – the predictability of a response of a PUF cell at a fixed position over multiple devices and the dependence of bits on a single device. (ii) Frequently, PUF cells on a single device are arranged in two-dimensional arrays on a chip. When, e.g., considering the PUF responses as bitstream already the decision of how to concatenate bits influences the result [6]. (iii) Some PUFs output multiple bits per challenge or position or (iv) are configurable by a challenge. In the latter case the choice of the challenges adds another dimension. One solution to cover many dimensions is to consider the PUF not as one source that outputs multiple bits, but as a multi-bit source or as multiple one-bit sources. In either case, PUFs require other tests than classic TRNGs.

Impact of Noise: For PUFs the evaluated entropy is extracted from noisy data, while TRNGs use this noise to generate data. This has two consequences: (i) A test for PUFs has to consider that measurement data comes from a joint distribution of noise effects and process variations[2]. (ii) Not only does the variation in the manufacturing process limit the entropy of a PUF, the effective entropy for the secret key is also reduced by noise, which has to be mitigated by error correction or other means.

B. Properties to be Tested

According to the fundamental differences of PUFs and TRNGs, also the properties to be tested differ. For TRNGs, there are two flavors of tests. On the one side tests exist, which try to distinguish the TRNG from a source outputting independent and identically distributed (iid) random numbers. If no distinction is possible, a high-quality TRNG is assumed. Alternatively, the entropy of a sequence of bits is estimated. In either case, the question is if the TRNG outputs are sufficiently random.

Similarly, for PUFs the question is if the response is sufficiently random. However, recall that a PUF has multiple dimensions to be tested. Ultimately, it must be unpredictable independent of any information of any PUF dimension given to an attacker. Hence, various properties of a PUF response have to be checked. Bits generated from different PUF cells on a device and response bits under different challenges have to be unbiased and uncorrelated, as well as bits at the same position over multiple devices.

Besides the analysis of PUF unpredictability by statistical tests, the impact of noise has also to evaluated. The goal is to reach a sufficiently low noise level that requires only a minimum of, e.g., error correction.

Finally, for both PUFs and TRNGs, entropy is an important evaluation metric. But for PUFs the entropy estimator must consider the PUF specific properties, like the different dimensions among others.

C. Test Methods for TRNGs

Due to its probabilistic nature, no single test is able to prove randomness of a TRNG. Hence, all test suites combine multiple tests such that the confidence increases with each passed test.

The NIST SP 800-22 [7] compares the TRNG output against the expected behaviour of an ideal randomness. It involves 15 tests analyzing the input regarding different patterns and checking the null hypothesis H_0 that the TRNG under test is random. If the corresponding p-value for a test is too low, H_0 is rejected and the test indicates

[1]Please note, that multi-challenge PUFs like Arbiter PUFs [3] or SUM PUFs [4] and their relatives, are rarely used for key storage and potentially weak against machine learning [5].

[2]Other effects like temperature shifts, changes in the supply, or aging might be statistically modelled on top.

that the data is not sufficiently random. An interpretation of the overall result is also part of the test suite.

Another standardized test suite is the BSI AIS 31 [8]. Its nine tests (partly similar to [7]) evaluate aspects of the random sequence. Furthermore, information about the TRNG's structure has to be provided by the applicant to strengthen the confidence in the randomness.

Additional non-standardized test-suits have been developed such as TESTU01 [9] which includes six test batteries. Overall, these test-suits follow a similar principle as already described.

Whereas the previous methods try to distinguish the actual TRNG output from an ideal one, the NIST SP 800-90B [10] estimates the min-entropy. Thus, the user first assigns the TRNG to an iid or a non-iid track, which have up to ten estimators. The overall estimated min-entropy is the lowest value of any of the tests.

D. Test Methods for PUFs

Currently, ISO/IEC 20897-2 [11] is the only standard evaluating PUFs. It demands randomness and reliability tests. The exemplified test set include PUF-specific tests, e.g, [12], [13], and methods from standardized TRNG test suits. It also uses the NIST SP 800-90B in order to approximate the entropy of PUFs.

However, other research in the PUF domain complements the methods in [11]. As a first example, [14] show that entropy tests for PUFs benefit from considering spatial effects. In addition, in the PUF domain a large variety of qualitative tests exists, whose plots visualize problems in the PUF design. One example is the application of Principal Component Analysis (PCA) to show gradients caused by manufacturing variations [15]. A new trend in PUF designs are tests, which ensure the quality of a PUF with some confidence or even try to prove the PUF quality through hypothesis testing [16]. Such tests can also be aware of spatial effects [17], which is not the case for the test defined for TRNGs. In addition, PUFs are frequently used to store a secret key, so that effectively not only the PUF's entropy but also the entropy in the key should be estimated [18].

E. Discussion of Test Strategies

This introduction into 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.

III. Assessment of SCA leakage in cryptographic circuits

A. 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 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 leverage "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 of 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.

B. 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 coordinate 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 [19])**.** *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 $\delta = (\delta_A, \delta_{M_i}, \delta_{M_o})$,*

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

C. 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:

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

Symmetrically,

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

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:*

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

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,

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

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. They can be speeded-up with butterflies algorithms. Namely, the systematic and automatic masking verification is carried out as shown in Alg. 1.

Algorithm 1: Masking verification method

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

$\mathcal{L}_u \leftarrow \varnothing, \mathcal{L}_g \leftarrow \varnothing$ `// 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 $\delta \in \mathbb{F}_2^{3k} \backslash \{0\}$ **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 $\mathcal{L}_u, \mathcal{L}_g$

D. Emblematic example

One challenge is for instance to verify each and every net from Canright's masked S-Box [20] of AES. The netlist can be found in [21] and the function we consider is:

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

at line 234. 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 [22] or glitch-free [23] 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 [21]. 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.

IV. Aging-Aware Digital Sensor dimensioning to enhance reliability

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 the recent years, and have replaced the traditional analog counterparts. Indeed, being designed in full custom layout [24] 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 [25].

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 [26]. Fig. 1 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 a_0. 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) [27], 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.

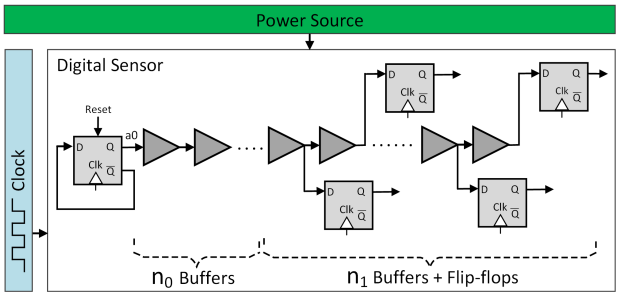


Figure 1. The architecture of the target digital sensor.

In the sensor shown in Fig. 1, 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 Fig. 2 shows the values of different DFFs of the sensor of Fig. 1 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 for 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.

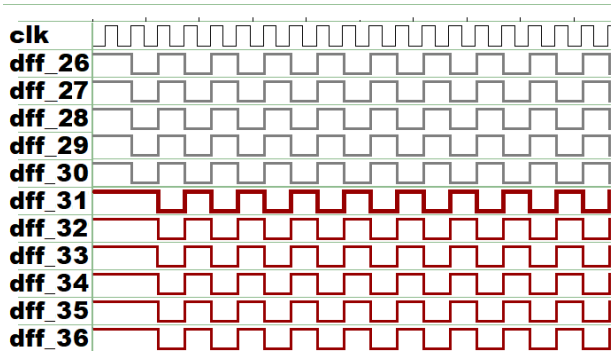


Figure 2. Waveforms of Fig. 1 in (V,T) = (1.2V,27°C), where n_0 = 9 and n_1 = 43.

Fig. 3 depicts the AFN values in different operating conditions for the sensor shown in Fig. 1 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 [28]. 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.

We benefit from the sensor's AFN quantity for system's failure detection, and to predict if 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 Fig. 3 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

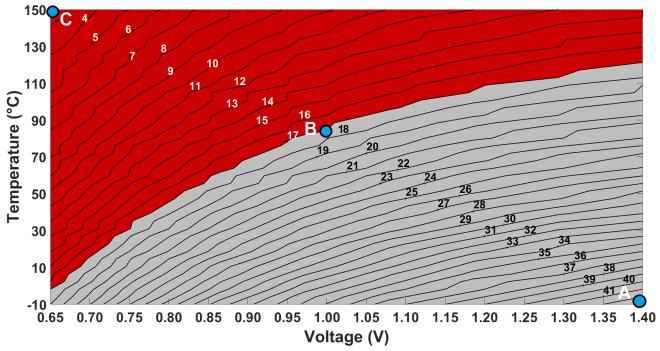


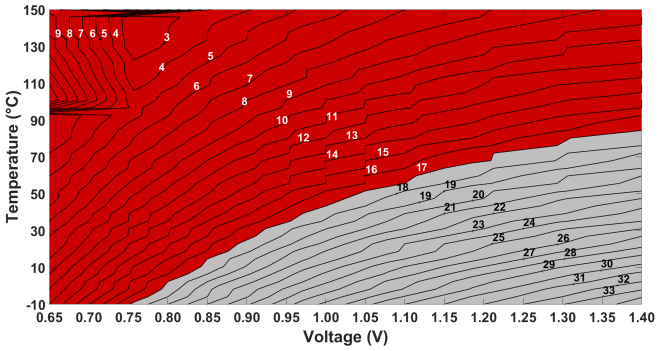
Figure 3. Contour graphs depicting AFN values in different (V,T) conditions for the fresh (age:0) sensor shown in Fig. 1 where n_0=9 and n_1=43.

considered as safe. It is noteworthy to mention that this threshold is tuned based on the application and user's configuration.

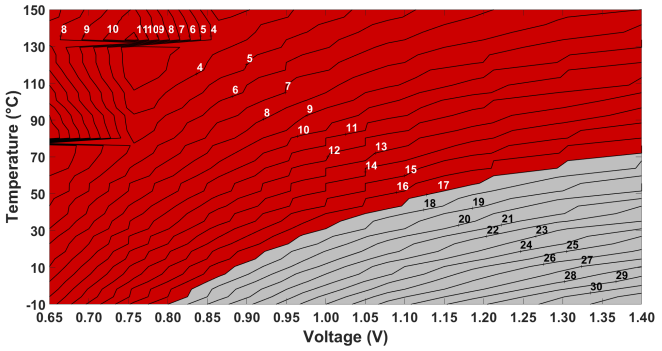
Indeed chips are designed in different temperature grades (e.g., commercial, industrial, military, etc), i.e., a different range of temperatures under which it is expected to work properly. Thereby, to realize a sensor (similar to the one shown in Fig. 1) that can cover the whole expected range of operating 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.

We have presented an algorithm for sensor dimensioning in our prior work (Algorithm 1 in [27]) 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 on properly (points A and B in Fig. 3 in our case). Deploying our prior algorithm (refer to [27] for more details) for dimensioning the sensor in Fig. 1 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, Fig. 4(a) and Fig. 4(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 Fig. 4(b) compared to Fig. 4(a) and Fig. 3. Another important observation is the trend of AFN value change in the aged sensors



(a) 4-year old sensor



(b) 7-year old sensor

Figure 4. Contour graphs depicting AFN values in different (V,T) conditions for the 4- and 7- year old sensors shown in Fig. 1 where n_0=9 and n_1=43.

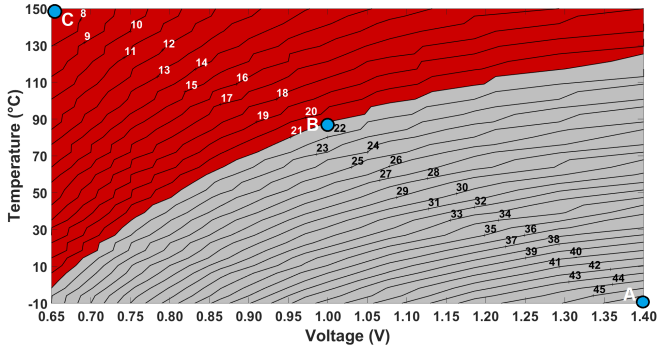
shown in Fig. 4(a) and Fig. 4(b) when operating under high temperature and low voltage combinations. In these cases as the sensor becomes slower and slower with aging, the AFN value may not be reliable, i.e., the sensor may need more DFFs to be able to correctly sample the setup time violation occurring in the buffer chain. To alleviate this problem, we improved the dimensioning algorithm presented in [27] 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.

In Alg. 2 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 necessary 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

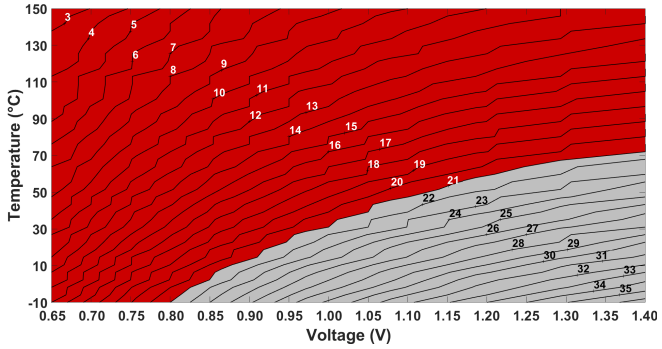
Algorithm 2: Aging Aware DS Dimensioning algorithm

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 Fig. 3
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 Fig. 3
6 Determine the position (AFN_high) of first sampling inversion error
7 **return** $(n_0 = \mathrm{N} - K, n_1 = \mathrm{AFN\_high} - n_0 + K)$



(a) Fresh (Age:0) sensor



(b) 7-year old sensor

Figure 5. Contour graphs depicting AFN values in different (V,T) conditions for the fresh and 7-year old sensors shown in Fig. 1 where $n_0 = 4$ and $n_1 = 48$.

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 Fig. 1 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 Fig. 5(a) and 5(b), respectively. As illustrated, by considering the aging effects in Alg. 2, the trend of AFN values are as expected even when the circuit is aged.

V. Conclusion

This paper demonstrates that security primitives requires specific tests to ensure a high level of security. Emblematic examples of properties to test are related to hostile environment and threats: randomness quality, leakage level, 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 minimum level 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 is necessary to avoid glitches which can unmask the sensitive values. This paper proposes a 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. 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.

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] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA Intrinsic PUFs and Their Use for IP Protection," in *CHES*, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer, Heidelberg, 2007, pp. 63–80.

[2] Z. Cherif, J.-L. Danger, S. Guilley, and L. Bossuet, "An easy-to-design puf based on a single oscillator: the loop puf," in *Euromicro - Digital System Design*. IEEE, 2012, pp. 156–162.

[3] D. Lim, J. Lee, B. Gassend, G. Suh, M. Van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

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

[5] 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 Trans. on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 2, 2021.

[6] M. Pehl, A. Punnakkal, M. Hiller, and H. Graeb, "Advanced performance metrics for physical unclonable functions," in *Int'l Symp. on Integrated Circuits (ISIC)*. IEEE, 2014.

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

[9] P. L'Ecuyer and R. Simard, "Testu01: Ac library for empirical testing of random number generators," *ACM TOMS*, vol. 33, no. 4, pp. 1–40, 2007.

[10] 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.

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

[12] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs," in *Int'l Conf. on Reconfigurable Computing and FPGAs*, ser. RECONFIG '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 298–303.

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

[14] M. Pehl, T. Tretschok, D. Becker, and V. Immler, "Spatial context tree weighting for physical unclonable functions," in *2020 ECCTD*. IEEE, 2020, pp. 1–4.

[15] F. Wilde, M. Hiller, and M. Pehl, "Statistic-based security analysis of ring oscillator pufs," in *2014 ISIC*. IEEE, 2014, pp. 148–151.

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

[17] F. Wilde, B. Gammel, and M. Pehl, "Spatial correlation analysis on physical unclonable functions," *IEEE Trans. on Information Forensics and Security*, vol. 13, no. 6, pp. 1468–1480, 2018.

[18] F. Wilde, C. Frisch, and M. Pehl, "Efficient bound for conditional min-entropy of physical unclonable functions beyond iid," in *2019 IEEE Int'l Workshop on Information Forensics and Security (WIFS)*, 2019, pp. 1–6.

[19] J. Blömer, J. Guajardo, and V. Krummel, "Provably Secure Masking of AES," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, H. Handschuh and M. A. Hasan, Eds., vol. 3357. Springer, 2004, pp. 69–83.

[20] D. Canright, "A Very Compact S-Box for AES," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 441–455. [Online]. Available: https://doi.org/10.1007/11545262_32

[21] [Online]. Available: https://github.com/coruus/canright-aes-sboxes/blob/master/verilog/sboxmaskcorr.verilog

[22] S. Nikova, V. Rijmen, and M. Schläffer, "Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches," *J. Cryptology*, vol. 24, no. 2, pp. 292–321, 2011.

[23] A. Moradi and O. 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,*. IEEE, 2012, pp. 89–95. [Online]. Available: https://doi.org/10.1109/HST.2012.6224326

[24] D. Shahrjerdi, J. Rajendran, S. Garg, F. Koushanfar, and R. Karri, "Shielding and securing integrated circuits with sensors," in *ICCAD*, 2014, pp. 170–174.

[25] N. Selmane, S. Bhasin, S. 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*, vol. 5, no. 4, pp. 181–190, 2011.

[26] M. Ebrahimabadi, M. T. H. Anik, J.-L. Danger, S. Guilley, and N. Karimi, "Using digital sensors to leverage chips' security," in *Physical Assurance and Inspection of Electronics (PAINE)*, 2020, pp. 1–6, DOI: 10.1109/PAINE49178.2020.9337730.

[27] M. T. H. Anik, J.-L. Danger, S. Guilley, and N. Karimi, "Detecting failures and attacks via digital sensors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, pp. 1315–1326, 2021.

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