Noise-Tolerant Machine Learning Attacks against Physically Unclonable Functions

Fatemeh Ganji, Shahin Tajik, Jean-Pierre Seifert

Security in Telecommunications
Technische Universität Berlin
Germany
{fganji,stajik,jpseifert}@sec.t-labs.tu-berlin.de

Abstract. Along with the evolution of Physically Unclonable Functions (PUFs) as a remedy for the shortcomings of conventional key storage and generation methods, numerous successful attacks against PUFs have been proposed in the literature. Among these are machine learning (ML) attacks, ranging from heuristic approaches to provable algorithms, that have attracted great attention. Nevertheless, the issue of dealing with noise has so far not been addressed in this context. Thus, it is not clear to what extent ML attacks can be launched on real-world, noisy PUFs. This paper aims to clarify this issue by focusing on provable ML algorithms, i.e., Probably Approximately Correct (PAC) learning algorithms. We prove that PAC learning algorithms for known PUF families are effective and applicable even in the presence of noise. Our proof relies heavily on the intrinsic properties of these PUF families, namely arbiter, Ring Oscillator (RO), and Bistable Ring (BR) PUF families. In addition to this proof, we introduce a new style of ML algorithms taking advantage of the Fourier analysis principle. We believe that this type of ML algorithms, and the notions related to it can broaden our understanding of PUFs by offering better measures of security.

Keywords: Physically Unclonable Functions, Boolean Analysis, Noise Sensitivity, Low Degree Algorithm, Machine Learning, PAC Learning.

1 Introduction

The design of PUFs relies on inherent manufacturing process variations, being uncontrollable, but exploitable by a circuitry to generate either a source of randomness or an instance-specific fingerprint [15]. The growing need for using PUFs in IC security stems from two main issues. On the one hand, the ineffectiveness of traditional security measures, e.g., secure key generation/storage, has been widely accepted. On the other hand, the inevitable fact that overbuilt and counterfeit ICs can be used in various important applications further contribute to this need for robust security measures [23]. Since the notion of PUFs has been introduced to address the aforementioned issues, several studies have focused on the advantages and disadvantages of this concept. Designing such circuits and

their respective security assessments, more particularly, cryptanalysis of PUFs are within two ends of the wide spectrum of these studies.

In addition to invasive and semi-invasive attacks, e.g., [17,33,40–42], a broad range of cryptanalysis of PUFs is covered by non-invasive attacks, for instance [38]. A great variety of these frameworks and numerous models have been developed around the principles of linear algebra [9], stochastic optimization [4], and machine learning (ML) [10–13, 20, 38]. When launching the latter attacks, the adversary observes only a small subset of challenges and their corresponding responses (i.e., the inputs and the outputs of the PUF) in order to build a model of the challenge-response behavior of the PUF. Therefore, when compared with invasive and semi-invasive attacks, these attacks are cost-effective and nondestructive, and consequently, attractive for adversaries.

Applying empirical ML algorithms (e.g., [38]) in the assessment of the security of PUFs marked the beginning of an era, after which the well-established concepts and existing algorithms in the field of ML were applied to analyze the security of these primitives. Beyond the early heuristic methods, probably approximately correct (PAC) learning frameworks have been developed to *prove* vulnerabilities for the known families of intrinsic PUFs to ML attacks [10–13,16]. The results of these studies have been acknowledged, and form now a solid basis for the design of PUFs, c.f. [44]. Albeit being useful for this purpose, the question remains open whether practical aspects of the design of PUFs have been adequately reflected by the PAC learning frameworks. More specifically, except the proof provided for XOR-arbiter PUFs [12], PAC learning in the presence of noise has not been discussed in the literature so far.

This issue is of twofold importance. First, the term "noise" in the PUF-related literature refers to the observation that applying the same challenge may result in obtaining different responses due to the environmental changes, see, e.g. [27]. These noisy responses reveal some information about the challenge-response behavior of the PUF, in a similar way to side channel information, which can be beneficial to model the PUF [4, 7, 8]. Understanding the mechanisms of generating noisy responses is therefore essential for designing a PUF that is robust against such hybrid attacks. Secondly, the gap between the existing noise models in the ML- and PUF-related literature should be bridged primarily by a thorough understanding of differences and similarities between these models. Accordingly, a refined model of noisy PUFs should be established, which provides a firm basis for analyzing the security of these primitives against ML attacks. This paper aims at addressing the above issues by providing the following contributions.

Establishing a refined model of noisy PUFs that is in line with models widely accepted in ML theory. In our model, we take into consideration the impact of noise on the final response of a PUF and as well at the inter-stage behavior of a PUF. We demonstrate that this model agrees with the noise models in ML theory, namely, attribute and classification noise.

Introducing a new style of ML algorithms, gaining an advantage by leveraging Fourier analysis. Thanks to the representation of PUFs as

Boolean functions, we explore the properties of PUFs from the Fourier analysis perspective. We introduce the notion of noise sensitivity of Boolean functions representing PUFs as a powerful analysis tool. Moreover, for known and widely-used PUFs a so-called low degree algorithm approximating the Fourier coefficients of the corresponding Boolean functions is presented in this paper.

PAC learning of PUFs in the presence of attribute and classification noise. Eventually, we prove that for known families of noisy PUFs there exist individual PAC learning algorithms that can learn their challenge-response behavior, with prescribed levels of accuracy and confidence.

2 Notation and preliminaries

2.1 PUFs

First, we stress that our paper does not cover the topics of formalization and formal definitions of the PUFs. For more details on these topics see, e.g., [2,3]. Note that hereafter the term "PUF" refers to the most popular, and known families of standalone PUFs: arbiter PUFs, Ring Oscillator (RO) PUFs, and Bistable Ring (BR) PUFs. Here, a standalone PUF means a PUF that is not composed of a combination of some PUFs (e.g., XOR arbiter PUFs) or other means. Generally speaking, PUFs are physical mappings from the inputs to the outputs, i.e., from the given *challenges* to the respective *responses*. These mappings are characterized by physical properties of the platform, on which the PUF is implemented. From among several security properties of PUFs, here we consider solely unclonability. Let the mapping $f_{\mathrm{PUF}} : \mathcal{C} \to \mathcal{Y}$, where $f_{\mathrm{PUF}}(c) = y$, describes a PUF. Ideally, for a given PUF f_{PUF} unclonability reflects the fact that creating a clone, i.e., a (physical) mapping $g_{\mathrm{PUF}} \neq f_{\mathrm{PUF}}$, is virtually impossible, where the challenge-response behavior of g_{PUF} is *similar* to f_{PUF} [2].

2.2 Boolean Functions as representations of PUFs

Similar to the most relevant studies on PUFs, we adopt the above mentioned, general definition of PUFs that is the physical mappings (see Section 2.1). This enables us to represent PUFs as Boolean functions over the finite field $\mathbb{F}_2$. To this end, consider $V_n = \{c_1, c_2, \ldots, c_n\}$ that is the set of Boolean attributes or variables, being either ***true*** or ***false*** denoted by "1" and "0", respectively. Moreover, let $C_n = \{0,1\}^n$ be the set of all binary strings with n bits, and an *assignment* be a mapping from V_n to $\{0,1\}$. Therefore, an assignment can be thought of as an n-bits string, where the i^{th} bit associated with the value of c_i (i.e., "0" or "1").

A Boolean formula is a mapping that assigns values from the set $\{0,1\}$ to an assignment. In this regard, each Boolean attribute is also a formula, i.e., c_i is a possible formula. If the Boolean formula assigns "1" to a Boolean assignment, it is a a *positive example* of the concept, otherwise a *negative example*. Furthermore, a Boolean function $f : C_n \to \{0,1\}$ can be defined by a Boolean formula respectively.

In general, Boolean functions can be represented by several different classes of functions, e.g., juntas, Linear Threshold functions (LTFs), and Decision Lists (DLs), cf. [34, 37]. A k-junta is a Boolean function, whose output is determined solely by an unknown set of k variables. A list L containing r pairs $(f_1, v_1), \ldots, (f_r, v_r)$ is called a DL, where the Boolean formula f_i is a conjunction of Boolean attributes, and $v_i \in \{0, 1\}$ with $1 \leq i \leq r-1$. For $i = r$, we have $v_r = 1$. When representing a Boolean function by a decision list, $L(c) = v_j$, where $c \in C_n$ and j is the smallest index in L so that $f_j(c) = 1$. Let k-DL denote the set of DLs, where each f_i is a conjunction of at most k Boolean attributes.

Before defining linear threshold functions, we define the encoding scheme $\chi(0_{\mathbb{F}_2}) := +1$, and $\chi(1_{\mathbb{F}_2}) := -1$. Hence, the Boolean function f can be defined as $f : \{-1, +1\}^n \rightarrow \{-1, +1\}$. Such a function is called a linear threshold function, if there are coefficients $\omega_1, \omega_2, \cdots, \omega_n \in \mathbb{R}$ and $\theta \in \mathbb{R}$ such that

$$f(c) = \operatorname{sgn}\left(\left(\sum_{i=1}^{n} \omega_i c_i\right) - \theta\right).$$

Without loss of generality, we assume that $\sum_{i=1}^{n} \omega_i c_i \neq \theta$ for every $c \in C_n$.

Noise Sensitivity of Boolean Functions This section describes the notion of noise sensitivity of a Boolean function. This should not be mistaken as the notion of noise discussed in the PUF-related literature. The noise sensitivity of the Boolean function $f : \{-1, +1\}^n \rightarrow \{-1, +1\}$ can be defined as follows (see Section 2.2 for more details on the encoding scheme required to define the noise sensitivity). Let c be a string chosen randomly at uniform. By flipping each bit of this string independently with probability ε ($0 \leq \varepsilon \leq 1$) we obtain the string c'. The noise sensitivity of f at ε is

$$NS_\varepsilon(f) := \Pr[f(c) \neq f(c')].$$

When studying the noise sensitivity of Boolean functions, applying methodologies developed for the spectral analysis of Boolean functions can provide a better understanding of this notion. The Fourier expansion of a Boolean function can be written as

$$f(c) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(c),$$

where $[n] := \{1, \ldots, n\}$, $\chi_S(c) := \prod_{i \in S} c_i$, and $\hat{f}(S) := \mathbf{E}_{c \in \mathcal{U}}[f(c)\chi_S(c)]$. Note that $\mathbf{E}_{c \in \mathcal{U}}[\cdot]$ indicates the expectation over examples chosen uniformly at random.

2.3 PAC Learning Model

PAC learning algorithms have attracted a considerable attention since they can guarantee the delivery of the outcome (i.e., the final model) as well as the accuracy of that for prescribed, desired levels of confidence and accuracy. Here we

briefly summarize the remarkable facts about this model, and refer the reader to [21] for more information.

Consider a PAC learner that is a learning algorithm, which is given access to a set of *examples* to generate an approximately correct hypothesis, with high probability. More formally, suppose that $F = \cup_{n\geq 1} F_n$ denotes a target concept class, i.e., a set of Boolean functions over the instance space $C_n = \{0,1\}^n$. In this paper, we are interested in a useful extension of the PAC model, in which each example is drawn from the instance space C_n with regard to the uniform distribution U. The hypothesis $h \in F_n$ that is a Boolean function over C_n is an ε-approximator for $f \in F_n$, if

$$\Pr_{c \in_U C_n} [f(c) = h(c)] \geq 1 - \varepsilon.$$

The complexity of a target concept $f \in F$ is assessed by measuring the size of that under a target representation. In order to define the size of a target concept $f \in F$, $size(f)$, we define the mapping $size : \{0,1\}^n \to \mathbb{N}$, relating a natural number $size(f)$ with a target concept $f \in F$. A polynomial-time algorithm $\mathtt{A}$, i.e., our learner, is provided with labeled examples $(c, f(c))$, where $f \in F_n$, and c is chosen uniformly at random from C_n. Here we concentrate on the strong uniform PAC learning algorithms, defined as follows.

Definition 1 *A strong uniform PAC learning algorithm* $\mathtt{A}$ *for the target concept class* F *is given a polynomial number of labeled examples to generate an* ε*-approximator for* f *under the uniform distribution* U*, with probability at least* $1 - \delta$*. In this regard, for any* $n \geq 1$*, any* $0 < \varepsilon, \delta < 1$*, and any* $f \in F_n$*, the running time of the algorithm* $\mathtt{A}$ *is* $poly(n, 1/\varepsilon, size(f), 1/\delta)$*, where* $poly(\cdot)$ *denotes a polynomial function.*

3 Noise: its Origin and Models

In this section we aim to come up with a model for PUFs enabling us to understand, how the noise can affect the functionality of a PUF. As mentioned in Section 1, in the PUF-related literature the response to a challenge is noisy, if repeated evaluations of the PUF with the respective challenge results in different responses. This is due to environmental variations and their impact on the functionality of physical components forming the PUF, e.g., the stages in an arbiter PUF. Environmental variations cover a wide range of uncontrollable random noise, e.g., thermal noise, uncertainties in measurement, cross talk, and power supply noise [2, 43]. According to the lessons from performance specifications of circuits, these random variations can be conventionally modeled as random variables following Gaussian distributions [1, 32, 39].

3.1 Impact of the Noise on a Single Stage

To provide a better understanding of the impact of the environmental variations on the internal functionality of a PUF, we focus on a single constitutive physical

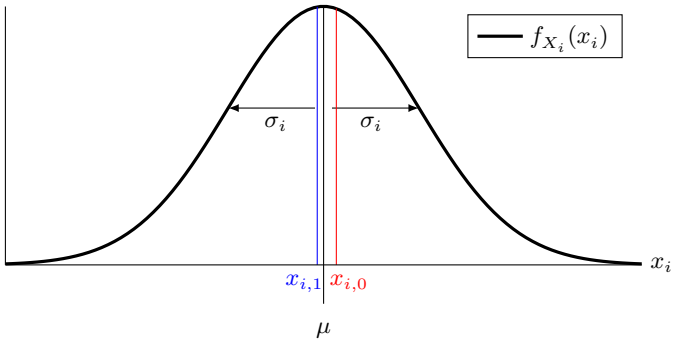

Fig. 1: The Gaussian random variable X_i that corresponds to the i^{th} physical component of the PUF, and its two realizations $x_{i,1}$ and $x_{i,0}$. In a meta-stable state these two realizations will be very close together.

component of the PUF, hereafter called a stage. As carefully formulated in [26], although the consideration of low-level physical details is a tedious task, (measurable) physical processes can be approximated by "hidden variables", namely the process variable and the noise variable. The latter variable corresponds to the effect of random noise on a single stage of the PUF. This variable follows a Gaussian distribution N_i, whose realization n_i differs for each evaluation of the PUF.

The definition of process variable determines the effect of manufacturing process variations on a single stage of the PUF [26]. As discussed before, this variable dented by X_i follows a Gaussian distribution. Similarly and independently, $X_1, \cdots, X_n$ can be defined, where n denotes the number of stages. In this manner, the mean value of the respective distribution (μ) is reported by manufactures as the nominal value, and the standard deviation σ_i is the result of the process variations, cf. [7, 13, 30]. Two realizations of the random variable X_i, namely $x_{i,1}$ and $x_{i,0}$, are generated during manufacturing. Without loss of generality, suppose that the following holds. When $c_i = 1$ is applied to the i^{th} stage, the realization $x_{i,1}$ is chosen to be involved in generating the final response of the PUF, whereas $x_{i,0}$ corresponds to $c_i = 0$. Moreover, suppose that the order relation between these realizations is $x_{i,1} > x_{i,0}$. Now, the *total impact of hidden variables on a stage* can be formulated as $Z_i = X_i + N_i$ $(1 \leq i \leq n)$, where Z is clearly a Gaussian random variable. In addition, the realizations of this random variable are $z_{i,1} = x_{i,1} + n_{i,1}$ and $z_{i,0} = x_{i,0} + n_{i,0}$, relating to the challenge bit applied to the PUF. Since the realizations $z_{i,1}$ and $z_{i,0}$ are related to two different evaluations of the PUF (with $c_i = 1$ and $c_i = 0$, respectively), the noise realizations vary, as indicated by different indices. As defined in [26], the final response of the PUF is determined by these realizations. Obviously, the difference between $z_{i,1}$ and $z_{i,0}$ is the main factor contributing to the final response of the PUF.

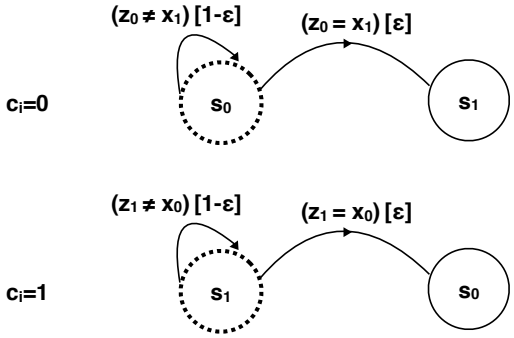

Fig. 2: Our simple Probabilistic (labeled) Transition Systems (PTS) describing how the noise can affect each stage in a PUF. The expressions included in parentheses denote the *labels*, whereas the information given in brackets refers to the probability of the transition between the states.

Now consider the i^{th} stage that is a meta-stable state, see Fig. 1. Here by meta-stable condition we refer to the fact that the realizations of the random variable X_i, i.e., $x_{i,1}$ and $x_{i,0}$ can be very close together so that under the effect of environmental noise one realization can be equal to another: $z_{i,0} = x_{i,1}$ or $z_{i,1} = x_{i,0}$, depending on the value of the challenge bit c_i [6]. To explain this, a simple Probabilistic (labeled) Transition Systems (PTS) can be defined as follows [36].

- There are two processes (i.e., sequences of events) corresponding to the value of the challenge bit applied to the i^{th} stage: $c_i = 1$ and $c_i = 0$, see Fig. 2.
- In both processes, the set of states S contains two states denoted by s_0 and s_1. The state s_0 represented the case that the challenge bit $c_i = 0$ is applied and in an ordinary condition (i.e., not meta-stable) we expect that $x_{i,0}$ would be involved in generating the final response of the PUF. Similarly, the state s_1 can be defined.
- $s_{int} \in S$ is the initial state in each process, shown by dashed circles in each process.
- A transition probability function $T: \quad S \times L \times S \rightarrow [0,1]$ represents, under which circumstances and what degree of probability the system transits from one state to another. Clearly,

$$\sum_{(l_i,s_j)\in L\times S} T(s_{int}, l_i, s_j) = 1.$$

Precisely defining our PTS, the tuple $\mathcal{A}_i$ represents the process related to the case, when the challenge bit c_i is applied:

$$\mathcal{A}_i = (S, L, T).$$

In each of the processes, as illustrated in Fig. 2, the PTS may transit from one state to another with probability ε, otherwise it remains in its initial state. For instance, applying the challenge bit $c_i = 0$, the initial state s_0 indicates that $x_{i,0}$ would contribute to the final response of the PUF. However, if this stage (the i^{th} stage) is in a meta-stable state, i.e., $z_{i,0} = x_{i,1}$, it is not possible to *differentiate* whether $x_{i,1}$ or $z_{i,0}$ would be involved to generate the final response of the PUF[1]. In other words, it can be thought that $c_i = 1$ is applied and the final response is under the influence of $x_{i,1}$, i.e., the PTS is in s_1 state. More precisely, we define a discrete random variable A corresponding to the event of a transition between s_0 and s_1. Formally, let $\Omega := \{\text{transition}, \text{stay}\}$ denote the sample space of the random variable A defined as follows.

$$A(\omega) = \begin{cases} 1, & \text{if } \omega = \text{transition}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Obviously, this random variable follows a Bernoulli distribution with the success probability ε, i.e., $A \sim \text{Bern}(\varepsilon)$.

Furthermore, as described above, we have translated the impact of noise on a single stage to a transition from one state to another state. Consequently, this change in the states can be seen as a probabilistic change of a challenge bit, e.g., the challenge bit $c_i = 0$ is flipped to challenge bit $c_i = 1$ or vice versa. To precisely summarize, with regard to our model, when applying the challenge c that is a Boolean string, the input to the PUF f_{PUF} can be written as $c \oplus a$, where $\oplus$ denotes the bit-wise XOR operator and a is a random string composed of bits generated independently from the distribution $\text{Bern}(\varepsilon)$.

3.2 Impact of the Noise on the Measuring Element of the PUF

In the second phase, we should take into consideration the impact of uncertainty on the generation of the final response. In the literature this issue has been already explored, when discussing the precision of the measuring element (e.g., the arbiter in the case of arbiter PUFs), which makes a decision whether the response of the PUF to the respective challenge is "0" or "1" [7, 13, 28]. Clearly, being limited in the precision, such component may change the output of the PUF. For the purpose of this paper, we are not interested in the real-world distribution of this noise, but in how the effect of this noise on the responses can be precisely described. A useful interpretation of this effect is given in [12], namely that after generating the response of the PUF an unfair coin (Head with probability $1 - \eta$) is flipped. Depending on the outcome, the final response is determined: when the outcome is Head, the response generated by the PUF remains unchanged, or otherwise, the response of the PUF is flipped. Here we follow the same principle to model the uncertainty with regard to the final response. Let a random variable B represent the impact of a limited precision

[1] Note that such transition does not always lead to a change in the response of the PUF.

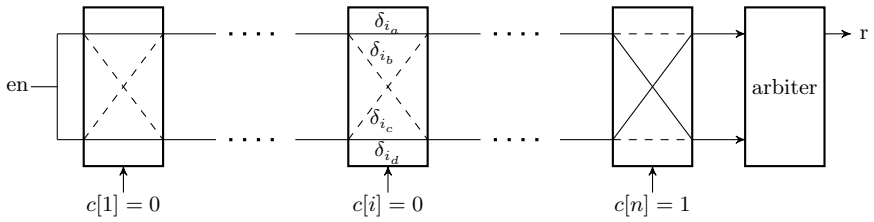

Fig. 3: Schematic of an arbiter PUF composed of n stages and an arbiter terminating the chain. When applying a challenge bit to a respective stage, either the realizations $x_{i,1}$ or the realization $x_{i,0}$ is chosen to be involved in generating the response of the PUF. For instance, we have $x_{i,1} = \delta_{i_a} - \delta_{i_d}$ and $x_{i,0} = \delta_{i_b} - \delta_{i_c}$.

of the physical component that makes the decision about the final answer. As explained above, this random variable also follows a Bernoulli distribution with the success probability $1 - \eta$, i.e., $B \sim \mathrm{Bern}\,(1 - \eta)$. For the sake of readability hereafter we denote $\mathrm{Bern}\,(1 - \eta)$ by R, and $\mathrm{Bern}\,(\varepsilon)$ by D. We have already defined the random string a containing independent random bits drawn according to the distribution D (see Section 3.1), and the random bit b drawn from the distribution R. Hence, the final response of the PUF can be formalized as $y = f_{\mathrm{PUF}}(c \oplus a) \oplus b$.

3.3 Practical Implications of the Noise Model

In this section, we explain, how a relation between the parameters introduced in the previous sections (Section 3.1 and 3.2) and real-world PUFs can be established.

Arbiter PUFs: First we consider the impact of the noise on a single stage of an arbiter PUF. For the arbiter PUF family, the realizations $x_{i,0}$ and $x_{i,1}$ are associated with the difference between the delays of crossed and straight signal paths, namely, $\delta_{i_a} - \delta_{i_d} = x_{i,1}$ and $\delta_{i_b} - \delta_{i_c} = x_{i,0}$, see Fig. 3. When the difference between these variables is small and the challenge bit c_i is applied to the stage, in the presence of the noise it is not possible to make a decision whether $x_{i,0}$ or $x_{i,1}$ has impacted the final response of the PUF.

Moreover, the impact of the noise on the response of the PUF can be explained by considering the limited precision of the arbiter terminating the chain, see Fig. 3 [13]. In this case, if after the final stage the delay difference between the upper and the lower paths is smaller than the precision of the arbiter, the arbiter could enter a metasable state and thus generate a wrong response.

Ring Oscillator (RO) PUFs: The response of this PUF is generated according to the difference between the frequencies of two ROs selected by the challenge. In other words, the challenge determines a pair of ROs that contributes to the final response of the PUF, see Fig. 4. When the frequency differences of ROs in two pairs vary insignificantly, under noisy conditions one of those RO

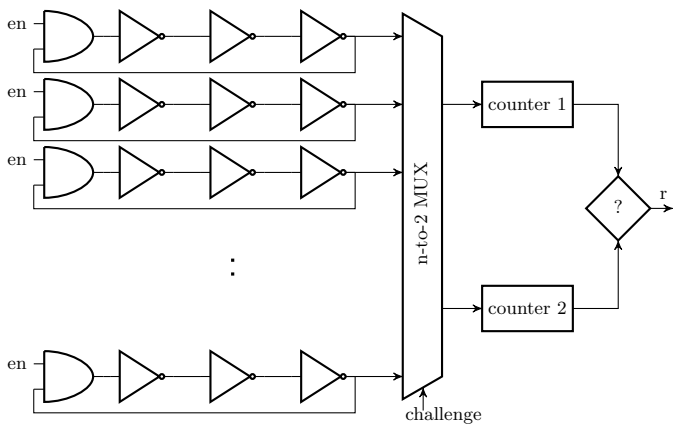

Fig. 4: An RO PUF with n ring-oscillators. By applying a challenge to a n-to-2 multiplexer, two ring-oscillators are selected and their outputs are connected to the clock inputs of 2 binary counters. The counters count the number of the rising edges during a predefined time period. Finally, the state of the counters is compared by the comparator placed at the end of the PUF to generate a binary response.

pairs can mimic another one. Therefore, it can be thought that some of the bits of the challenge applied to the PUF are flipped so that another RO pair makes impact on the final response of the PUF.

Furthermore, the limited precision of the counters measuring the frequencies of the ROs can affect the response of the PUF, see Fig. 4. More precisely, if the difference in the oscillation frequencies of a selected RO pair is not significant, the counters cannot measure the frequencies with high precision. Comparison of uncertain frequency measurements can lead to the generation of a wrong response.

Bistable Ring (BR) PUFs: Although a precise analytical model of the BR PUF is missing in the literature [10], we can still describe the impact of the noise on individual stages. For a given challenge in the BR PUF, n inverters are selected (see Fig. 5), and upon setting the reset signal to low, the created inverter ring starts to oscillate until it settles down to a valid logical state. In this case, the process variables can be intrinsic differences in the propagation delays and electrical gains of each inverter. Therefore, based on the environmental conditions, the noise can be added to the realization of the process variables.

However, in contrast to the arbiter and RO PUFs, there is no *explicit* measuring element in this PUF architecture. But the required additional measurement element which introduces noise for this type of PUF is explicitly discussed in [10, 18, 19].

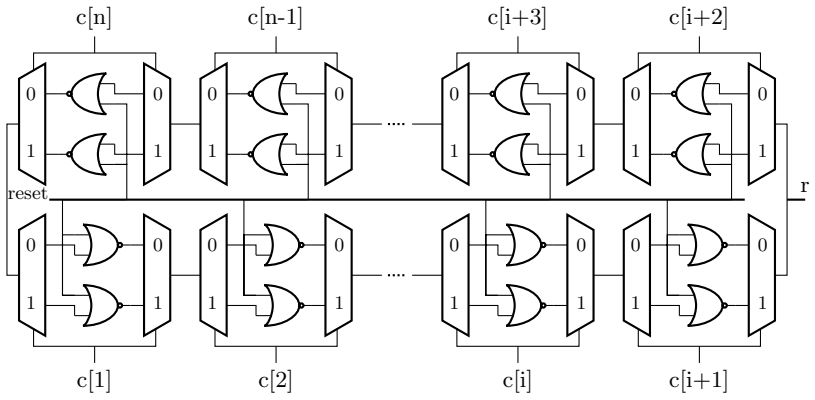

Fig. 5: The schematic of a BR PUF composed of n stages. From between two arbitrary stages, the response of the PUF can be read. When applying a challenge, the reset signal can be set low to activate the PUF. The BR PUF might be settled to an allowed logical state after a transient period.

3.4 Modeling the Noise from the Perspective of Machine Learning

With regard to the discussion from the previous sections 3.1- 3.3, PUFs can be thought of as Boolean functions, whose input-output behavior is determined by random process variations as well as the inevitable impact of random noise. In line with this view, a model of PUFs as illustrated in Fig. 6 can be established. This model can be seen as an extension of a model introduced and evaluated practically in [26]. Our model is composed of two components: random and deterministic components. The random component represents the random environmental noise, whereas the deterministic component accounts for the deterministic Boolean function realized in the chip. In other words, in the absence of noise, the response of the PUF to a challenge applied repeatedly remains the same. The building blocks as well as the parameters related to the model have been introduced previously, although their interpretations in the machine learning context have not yet been considered. As the next step in our framework, this section elaborates on how the functionality of a noisy PUF, as shown in Fig. 6, can be described from the point of view of machine learning. To this end, the following Lemma plays an important role, cf. [5].

Lemma 1 *Consider U, D and R that are a uniform, and two arbitrary distributions*[2] *over the space $\{0,1\}$, respectively. Moreover, let the function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be an arbitrary Boolean function. Let $C \in_U \mathbb{F}_2^n$, $A \in_D \mathbb{F}_2^n$ and $B \in_R \mathbb{F}_2$ be independent random strings and a random variable, respectively. The random variables $(C, f(C \oplus A) \oplus B)$ and $(C \oplus A, f(C) \oplus B)$ follow identical distribution.*

[2] Although regarding the physical properties of noisy PUFs we have defined the distribution D and R precisely (see Section 3.2), in general these distribution can be arbitrary.

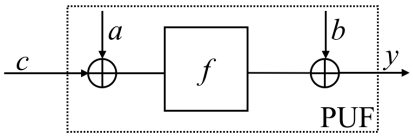

Fig. 6: Our model composed of blocks representing a noisy PUF. The random string a contains independent random bits drawn according to the distribution D (see Section 3.1), and the random bit b is drawn from the distribution R (see Section 3.2). From the machine learning point of view, they refer to attribute noise and classification noise, respectively.

For the proof of this Lemma, we refer to [5]. The conclusions drawn from it are of great importance for us. First, from the machine learning point of view, the noise represented by the random variable A is called the "attribute noise", whereas the random variable B corresponds to the "classification noise". The issue of attribute and classification noise learnability has been well addressed in the relevant literature, see, e.g., [5, 45]. Secondly, thanks to the seminal paper published by Bshouty et al. [5], a relationship between machine learning under noisy conditions and the noise sensitivity of Boolean functions has been established. The consequence of this relation is that efficient algorithms developed to estimate the Fourier coefficients of an unknown function can be applied to learn the respective function *even* under noisy conditions.

4 PAC Learning in the Presence of Noise

In this section, we elaborate on our PAC learning framework relying primarily on the concept introduced in Section 3. First, we introduce an algorithm proposed by Linial, Mansour, and Nisan [25] to estimate the Fourier coefficients of an unknown function (i.e., so-called LMN-style algorithm). The rationale behind the LMN-style algorithm, originally called "low degree" algorithm [31], is that some classes of Boolean functions can be approximated by taking into account solely a small number of their Fourier coefficients (called "low" coefficients), corresponding to small subsets of $[n]$ (see Section 2.2).

Theorem 1. (**Low degree algorithm**) [25, 31, 35] *Assume that an algorithm can determine a set* $\mathcal{S} \subseteq 2^{[n]}$ *containing subsets of* $[n]$ *so that* $\sum_{S \in \mathcal{S}} \hat{f}(S)^2 \geq 1-\varepsilon$. *The algorithm is given a pre-defined confidence level* δ *and access to a polynomial number of input-output pairs of the Boolean function* f *that are chosen uniformly at random. With probability* $1-\delta$ *the algorithm delivers a Boolean function* h *that is an* ε*-approximator of the Boolean function* f *such that*

$$\sum_{S \subseteq [n]} \left(\hat{f}(S) - \hat{h}(S) \right)^2 \leq \varepsilon.$$

The running time of the algorithm is poly $(|\mathcal{S}|, n, 1/\varepsilon, \log_2(1/\delta))$.

For the proof of this theorem, we refer the reader to [25, 31], in which the mechanism for determining the set $\mathcal{S}$, and the lower bound on the number of input-output pairs required by the algorithm has been discussed extensively.

Now, in order to prove the PAC learnability of PUFs under noisy conditions, the following steps are executed.

1. On the basis of certain properties of representations of arbiter PUFs, RO PUFs, and BR PUFs, we prove the existence of LMN-style algorithms for them.
2. Finally, we demonstrate that the existence of an LMN-style algorithm for an unknown Boolean function is sufficient and necessary to PAC learn that function under the uniform challenge distribution even in the presence of noise.

4.1 An LMN-style Algorithm for RO PUFs

Although the security of these PUFs can be easily broken by simply reading out all CRPs, launching a machine learning attack in the specific circumstance of having limited access to the CRPs (e.g., eavesdropping them) has been addressed in the literature, see for instance [11, 38]. In the present case, learning of noisy RO PUFs has not been discussed.

Our proof of the existence of an LMN-style algorithm for RO PUFs heavily relies on the key result presented in [11], i.e., RO PUFs can be represented by k-DLs.

Theorem 2. [24, 31] *An LMN-style algorithm can be employed that with probability* $1 - \delta$ *delivers a Boolean function* h *approximating a decision list* L*, which represents an RO PUF. The running time of this algorithm is* $poly\,(n, \log_2(1/\varepsilon), \log_2(1/\delta))$.

The proof sketch can be summarized as follows. According to results presented in [11], an RO PUF can be represented by a DL. Furthermore, Mansour proved that a DL could be approximated by a Boolean function h, whose Fourier coefficients concentrate only on a small set of variables, namely, $\log_2(1/\varepsilon)$ variables [31][3]. To find this set of variables, the low degree algorithm can be applied to deliver h and the running time of that is $poly\,(n, \log_2(1/\varepsilon), \log_2(1/\delta))$.

4.2 An LMN-style Algorithm for Arbiter PUFs

To prove the existence of an LMN-style algorithm for arbiter PUFs we argue as follows. It is known that if a Boolean function exhibits a bounded, small noise sensitivity, its Fourier coefficients are mainly low coefficients. More precisely, the following Corollary can be proved (for the proof see Corollary 2.3.3 in [35]).

[3] Here we do not discuss the details of the proof and refer the reader to [31] for more information.

Corollary 1 [35] *Consider* $\alpha : [0,1/2] \rightarrow [0,1]$ *that is a strictly increasing continuous function so that* $NS_{\varepsilon}(f) \leq \alpha(\varepsilon)$. *We have*

$$\sum_{|S| \geq m} \hat{f}(S)^2 \leq \varepsilon,$$

where $m = \alpha(\varepsilon/2.32))$.

It is clear that before proving the existence of an LMN-style algorithm for arbiter PUFs, we must elaborate on the noise sensitivity of Boolean functions representing these PUFs. Thanks to the results reported in [15, 29, 38], LTFs are appropriate representations of arbiter PUFs. Moreover, Corollary 2 has been proved by Klivans et al. [22].

Corollary 2 *For any LTF* f *its noise sensitivity is a bounded, small value depending only on* ε. *Precisely, we have:*

$$NS_{\varepsilon}(f) \leq 8.54\sqrt{\varepsilon}.$$

Corollary 3 states how Corollary 1 in conjunction with Corollary 2 can be applied to prove the existence of an LMN-style algorithm for arbiter PUFs.

Corollary 3 *Representing an arbiter PUF by an LTF, a Boolean function* h *approximating this LTF can be delivered by an LMN-style algorithm, whose running time is polynomial in* n, $1/\varepsilon^2$, *and* $\log_2(1/\delta)$.

Proof: Fix $\alpha(\varepsilon) = \sqrt{\varepsilon}$. The running time of the LMN-style algorithm is polynomial in $O(n^m)$, where $m = 1/\alpha^{-1}(\varepsilon/2.32)$ (note that $\alpha^{-1}(\cdot)$ denotes the inverse of the function $\alpha(\cdot)$). ■

4.3 An LMN-style Algorithm for BR PUFs

Similar to the proof of the existence of an LMN-style algorithm for arbiter PUFs, we take advantage of the properties of the Boolean functions representing BR PUFs. More specifically, we rely on the fact that a BR PUF can be represented by k-junta, where k is a (relatively) small constant value for practical values of n, as demonstrated in [10]. We first show that the noise sensitivity of BR PUFs is a bounded, small value.

Theorem 3. *For any Boolean function* f *represented by a* k*-junta the noise sensitivity is*

$$NS_{\varepsilon}(f) \leq k\varepsilon/2.$$

The proof is straightforward; for more details see, e.g., [14]. Now the following corollary of Theorem 3 and Corollary 1 can be formulated as follows.

Corollary 4 *An LMN-style algorithm can be applied to deliver an* ε*-approximator for a* k*-junta representing a BR PUF. The running time is polynomial in* n, $1/\varepsilon$, *and* $\log_2(1/\delta)$.

4.4 Existence of PAC Learning Algorithms in the Presence of Noise

The low degree algorithm mainly aims to provide an approximator of a Boolean function with a given probability, when it is given a polynomial number of input-output pairs of the Boolean function that are chosen uniformly at random. However, its existence has a serious consequence. More specifically, if the set $\mathcal{S}$ is composed of all the subsets of low degree, Theorem 1 introduces a PAC learning algorithm under the uniform distribution [35]. Before formulating this precisely, we first shift our focus to the issue of dealing with noise.

As explored in Section 3.4, we take the attribute and the classification noise into account. The question is how these processes affect the functionality and the efficiency of an LMN-style algorithm. This issue is well addressed by Bshouty et al., [5], and here we briefly summarize their results. They have shown that the attribute and the classification noise attenuate the Fourier coefficients, which the LMN-style algorithm aims to estimate from the uniformly random examples. To be exact, assume that an LMN-style algorithm attempts to estimate the Fourier coefficient $\hat{f}(S)$. Under the noisy conditions, it delivers $\hat{f}(S)(1-2\eta)\alpha_S$, where $(1-2\eta)$ and α_S are attenuation factors corresponding to the classification and attribute noise, respectively. While the former factor is known, see e.g., [12], the latter requires more attention. The attenuation factor α_S is the defined as follow.

$$\alpha_S := \mathbf{E}_{a \in D}[\chi_S(a)],$$

where $\mathbf{E}_{a \in D}[\cdot]$ denotes the expectation over random examples drawn from the known distribution D. As discussed in Section 3.4, here we consider a that is a random string, whose bits are independently generated following a Bernouli distribution Bern (2ε) with $\varepsilon \in (0,1/2]$. Hence,

$$|\alpha_S| = \prod_{i \in S}(1-2\varepsilon) = (1-2\varepsilon)^{|S|}.$$

Note that the practical implication of the attenuation factors ($(1-2\eta)$ and α_S) is that after running the LMN-style algorithm each Fourier coefficient delivered by the algorithm should be multiplied by $(1-2\eta)^{-1}$ and α_S^{-1} to eliminate the impact of the noise.

The existence of an LMN-style algorithm for a PUF, in conjunction with this discussion on the estimation of the Fourier coefficients in the presence of noise implies the existence of a PAC learning algorithm for this PUF under this condition. The following Corollary concludes this section by summarizing the above discussion (for more details, cf. [5, 35]).

Corollary 5 *Consider a given PUF that is represented by a Boolean function f_{PUF} which allows for an LMN-style learning algorithm. Then the PUF is PAC learnable under the uniform challenge distribution in the presence of known*[4] *attribute and classification noise. The running time of the PAC learner is poly* $(|\mathcal{S}|, n, 1/\varepsilon, \log_2(1/\delta), (1/1-2\eta))$.

[4] Although we assume that the distributions D and R are known, this assumption can be eliminated as discussed in [5].

5 Conclusion

The main aim of this paper is to address the issue of machine learning of noisy PUFs. To fulfil this aim, at the first stage we provide an in-depth discussion of noise origins and its impact on each and every component of a PUF, which leads to the development of a refined model of PUFs. This model enables us to bridge the gap between the models of noise proposed in machine learning- and PUF-related literature, and consequently, reflects the physical characteristics of PUFs appropriately. Furthermore, looking at this model from the angle of machine learning, we relate the issue of learning a noisy PUF to the well-known problem in this field that is learning in the presence of the attribute and the classification noise. As the next step in our approach, we explore the noise sensitivity of Boolean functions representing known families of PUFs. This step is extremely delicate since it implies the existence of a so-called Linial-Mansour-Nisan algorithm relying on the spectral properties of the Boolean functions associated with the respective PUFs. Moreover, we show how these spectral properties can be slightly changed under the noisy conditions. Accordingly and finally, the last step allows proving the PAC learnability of commonly used and known families of PUF in the presence of noise.

We believe that in addition to the proof of PAC learnability in the presence of noise, each and every step mentioned above provides many interesting insights into not only the assessment of the security of PUFs, but also the design of PUFs with better security-related characteristics. A prime example of these insights is the concept of noise sensitivity of a Boolean function corresponding to a PUF as a measure of learning complexity.

References

1. Alkabani, Y., Koushanfar, F., Kiyavash, N., Potkonjak, M.: Trusted Integrated Circuits: A Nondestructive Hidden Characteristics Extraction Approach. In: Information Hiding. pp. 102–117. Springer (2008)
2. Armknecht, F., Maes, R., Sadeghi, A., Standaert, O.X., Wachsmann, C.: A Formalization of the Security Features of Physical Functions. In: Security and Privacy (SP), 2011 IEEE Symp. on. pp. 397–412 (2011)
3. Armknecht, F., Moriyama, D., Sadeghi, A.R., Yung, M.: Towards a Unified Security Model for Physically Unclonable Functions. In: Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conf. vol. 9610, p. 271. Springer (2016)
4. Becker, G.T.: The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In: Cryptographic Hardware and Embedded Systems–CHES 2015, pp. 535–555. Springer (2015)
5. Bshouty, N.H., Jackson, J.C., Tamon, C.: Uniform-Distribution Attribute Noise Learnability. Information and Computation 187(2), 277–290 (2003)
6. Danger, J.L.: Metastable Latches: a Boon for Combined PUF/TRNG Designs. Hardware Security (Dagstuhl Reports on Seminar 16202) 6(5), 78 (2016), `http://drops.dagstuhl.de/opus/volltexte/2016/6721`

7. Delvaux, J., Verbauwhede, I.: Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise. In: Hardware-Oriented Security and Trust (HOST), 2013 IEEE Intl. Symp. on. pp. 137–142 (2013)
8. Delvaux, J., Verbauwhede, I.: Fault Injection Modeling Attacks on 65 nm Arbiter and RO Sum PUFs via Environmental Changes. Circuits and Systems I: Regular Papers, IEEE Transactions on 61(6), 1701–1713 (2014)
9. Ganji, F., Krämer, J., Seifert, J.P., Tajik, S.: Lattice Basis Reduction Attack against Physically Unclonable Functions. In: Proc. of the 22nd ACM Conf. on Computer and Communications Security. pp. 1070–1080. ACM (2015)
10. Ganji, F., Tajik, S., Fäßler, F., Seifert, J.P.: Strong machine learning attack against pufs with no mathematical model. In: Intl. Conf. on Cryptographic Hardware and Embedded Systems. pp. 391–411 (2016)
11. Ganji, F., Tajik, S., Seifert, J.P.: Let Me Prove it to You: RO PUFs are Provably Learnable, The 18th Annual Intl Conf. on Information Security and Cryptology (2015)
12. Ganji, F., Tajik, S., Seifert, J.P.: Why Attackers Win: On the Learnability of XOR Arbiter PUFs. In: Trust and Trustworthy Computing, pp. 22–39. Springer (2015)
13. Ganji, F., Tajik, S., Seifert, J.P.: PAC Learning of Arbiter PUFs. Journal of Cryptographic Engineering Special Section On Proofs 2014, 1–10 (2016)
14. Garban, C., Steif, J.E.: Noise sensitivity of Boolean functions and Percolation, vol. 5. Cambridge University Press (2014)
15. Gassend, B., Lim, D., Clarke, D., Van Dijk, M., Devadas, S.: Identification and Authentication of Integrated Circuits. Concurrency and Computation: Practice and Experience 16(11), 1077–1098 (2004)
16. Hammouri, G., Öztürk, E., Sunar, B.: A Tamper-proof and Lightweight Authentication Scheme. Pervasive and mobile computing 4(6), 807–818 (2008)
17. Helfmeier, C., Boit, C., Nedospasov, D., Seifert, J.P.: Cloning Physically Unclonable Functions. In: Hardware-Oriented Security and Trust (HOST), 2013 IEEE Intl. Symp. on. pp. 1–6 (2013)
18. Hesselbarth, R., Manich, S., Sigl, G.: Modeling and Analyzing Bistable Ring Based PUFs. `http://futur.upc.edu/16918245` [accessed 15 March 2017] (2015)
19. Hesselbarth, R., Sigl, G.: Fast and Reliable PUF Response Evaluation from Unsettled Bistable Rings. In: Digital System Design (DSD), 2016 Euromicro Conference on. pp. 82–90 (2016)
20. Hospodar, G., Maes, R., Verbauwhede, I.: Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling Poses Strict Bounds on Usability. In: WIFS. pp. 37–42 (2012)
21. Kearns, M.J., Vazirani, U.V.: An Introduction to Computational Learning Theory. MIT press (1994)
22. Klivans, A.R., O'Donnell, R., Servedio, R.A.: Learning Intersections and Thresholds of Halfspaces. In: Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on. pp. 177–186 (2002)
23. Koushanfar, F.: Hardware Metering: A Survey. In: Introduction to Hardware Security and Trust, pp. 103–122. Springer (2012)
24. Kushilevitz, E., Mansour, Y.: Learning Decision Trees Using the Fourier Spectrum. SIAM Journal on Computing 22(6), 1331–1348 (1993)
25. Linial, N., Mansour, Y., Nisan, N.: Constant Depth Circuits, Fourier Transform, and Learnability. Journal of the ACM (JACM) 40(3), 607–620 (1993)
26. Maes, R.: An Accurate Probabilistic Reliability Model for Silicon PUFs. In: Cryptographic Hardware and Embedded Systems-CHES 2013, pp. 73–89. Springer (2013)

27. Maes, R.: Physically Unclonable Functions: Constructions, Properties and Applications. Springer Berlin Heidelberg (2013)
28. Majzoobi, M., Koushanfar, F., Devadas, S.: FPGA PUF Using Programmable Delay lines. In: Information Forensics and Security (WIFS), 2010 IEEE Intl. Workshop on. pp. 1–6 (2010)
29. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight Secure PUFs. In: Proc. of the 2008 IEEE/ACM Intl. Conf. on Comp.-Aided Design. pp. 670–673 (2008)
30. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Techniques for Design and Implementation of Secure Reconfigurable PUFs. ACM Trans. on Reconfigurable Technology and Systems (TRETS) 2 (2009)
31. Mansour, Y.: Learning Boolean Functions via the Fourier Transform. In: Theoretical Advances in Neural Computation and Learning, pp. 391–424. Springer (1994)
32. Mehrotra, V.: Modeling the Effects of Systematic Process Variation on Circuit Performance. Ph.D. thesis, Massachusetts Institute of Technology (2001)
33. Merli, D., Schuster, D., Stumpf, F., Sigl, G.: Semi-invasive EM Attack on FPGA RO PUFs and Countermeasures. In: Proc. of the Workshop on Embedded Systems Security. p. 2 (2011)
34. O'Donnell, R.: Analysis of Boolean Functions. Cambridge University Press (2014)
35. O'Donnell, R.W.: Computational Applications of Noise Sensitivity. Ph.D. thesis, Massachusetts Institute of Technology (2003)
36. Panangaden, P.: Labelled Markov Processes. Imperial College Press (2009)
37. Rivest, R.L.: Learning Decision Lists. Machine learning 2(3), 229–246 (1987)
38. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling Attacks on Physical Unclonable Functions. In: Proc. of the 17th ACM Conf. on Comp. and Communications Security. pp. 237–249 (2010)
39. Srivastava, A., Sylvester, D., Blaauw, D.: Statistical Analysis and Optimization for VLSI: Timing and Power. Springer Science & Business Media (2006)
40. Tajik, S., Dietz, E., Frohmann, S., Dittrich, H., Nedospasov, D., Helfmeier, C., Seifert, J.P., Boit, C., Hübers, H.W.: Photonic Side-Channel Analysis of Arbiter PUFs. Journal of Cryptology pp. 1–22 (2016)
41. Tajik, S., Dietz, E., Frohmann, S., Seifert, J.P., Nedospasov, D., Helfmeier, C., Boit, C., Dittrich, H.: Physical Characterization of Arbiter PUFs. In: Cryptographic Hardware and Embedded Systems–CHES 2014, pp. 493–509. Springer (2014)
42. Tajik, S., Lohrke, H., Ganji, F., Seifert, J.P., Boit, C.: Laser Fault Attack on Physically Unclonable Functions. In: 2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 85–96 (2015)
43. Wang, X., Tehranipoor, M.: Novel Physical Unclonable Function with Process and Environmental Variations. In: Proc. of the Conf. on Design, Automation and Test in Europe. pp. 1065–1070 (2010)
44. Yu, M.D., Hiller, M., Delvaux, J., Sowell, R., Devadas, S., Verbauwhede, I.: A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication. IEEE Transactions on Multi-Scale Computing Systems PP(99) (2016)
45. Zhu, X., Wu, X.: Class Noise vs. Attribute Noise: A Quantitative Study. Artificial Intelligence Review 22(3), 177–210 (2004)