Pitfalls in Machine Learning-based Adversary Modeling for Hardware Systems

Fatemeh Ganji, Sarah Amir, Shahin Tajik, and Domenic Forte
Department of Electrical Engineering
University of Florida
Gainesville, USA
{fganji, sarah.amir, stajik}@ufl.edu, dforte@ece.ufl.edu

Jean-Pierre Seifert
Security in Telecommunications
Technische Universitt Berlin
Berlin, Germany
jean-pierre.seifert@external.telekom.de

***Abstract*—The concept of the adversary model has been widely applied in the context of cryptography. When designing a cryptographic scheme or protocol, the adversary model plays a crucial role in the formalization of the capabilities and limitations of potential attackers. These models further enable the designer to verify the security of the scheme or protocol under investigation. Although being well established for conventional cryptanalysis attacks, adversary models associated with attackers enjoying the advantages of machine learning techniques have not yet been developed thoroughly. In particular, when it comes to composed hardware, often being security-critical, the lack of such models has become increasingly noticeable in the face of advanced, machine learning-enabled attacks. This paper aims at exploring the adversary models from the machine learning perspective. In this regard, we provide examples of machine learning-based attacks against hardware primitives, e.g., obfuscation schemes and hardware root-of-trust, claimed to be infeasible. We demonstrate that this assumption becomes however invalid as inaccurate adversary models have been considered in the literature.**

***Index Terms*—Physically Unclonable Functions, Logic Locking, Composed Hardware, Root-of-Trust, Machine Learning.**

I. Introduction

In an era characterized by increasing cybersecurity threats, we have witnessed the ever-continuing competition between system designers/ manufacturers and adversaries that maliciously break the security of systems. Modern systems are usually composed of several hardware components, potentially involved in various protocols. Analyzing the security of such a system in an effective manner is a nontrivial task, even though frameworks for modular security, e.g., Canetti's universally composable (UC) framework, has been devised to support the security evaluation of hardware-assisted protocols [1]. It is even challenging for stand-alone hardware components, despite the existence of mathematical, cryptographic reductions. This can be explained by the fact that assumptions regarding the availability of secure key generation and storage, as well as secure execution, cannot always be valid. In an attempt to resolve this issue, it has been suggested to build a physical root-of-trust (RoT), i.e., a physical primitive that can fulfill the physical security objectives. In this regard, the security in the system depends heavily on the security of the RoT. Nevertheless, it has been demonstrated that the security of an RoT can be compromised through attacks covering the whole spectrum of invasive, semi-invasive, and non-invasive ones.

Non-invasive attacks have become increasingly threatening and much more common as a result of the advancement in machine learning (ML). This is also partially due to the lack of systematic and provable methods, which can assess the security of a system in the design phase. This lack of methods is present, albeit of well-known, and acknowledged frameworks developed in cryptography, and its sister field, i.e., ML. This close relationship has been first well formulated in a seminal work of Rivest [2], which has recognized the similarities between attack types and the queries required by an ML algorithm. Rivest's paper has further highlighted the difference between the exact and approximate inference. The above similarities and differences have been solely partially considered in hardware security-related literature. In addition to these, the significance of selecting the proper ML setting is often underestimated.

To address this, our paper aims at exploring the close relationship between machine learning and cryptography in the context of composed hardware and the RoT. To this end, we discuss the main aspects of an ML framework that are crucial for assessing the security of these primitives, namely the distribution of learning examples, access type, and choices of models (i.e., representations) assigned to an attacker. For this purpose, we provide examples of physical systems and discuss how inappropriate conclusions can be made if the setting of the ML model is applied improperly.

II. Background on Hardware Security and ML

A. Logic Locking Techniques

ICs nowadays become vulnerable to IP piracy, tampering, and counterfeiting while several phases of chip manufacturing, such as design, integration, and fabrication are carried out at various - often untrusted - facilities. Besides, on the end-user side, especially for the devices delivered to malicious entities, IP protection is still an issue. IP logic locking (LL) schemes [3] have been proposed to cope with this issue for not only standalone hardware primitives, but also composed

hardware. In the latter case, the security of the composed IP can be ensured by protecting either all the IPs involved in the composed hardware or solely ones that are security-critical. Regardless of these scenarios, our goal is to observe why and to what extent the existing security analysis of LL schemes through ML can be less precise. Therefore, here, we solely focus on some examples of sequential and combinational LL. Sequential LL mainly refers to the augmentation of the Finite State Machines (FSMs) at, e.g., the gate-level, by adding a new set of states, whereas combinational LL methods aims at adding extra key gates in a design, which are controlled by a key given to it as input bits [3].

Prime examples of employing (provable) ML algorithms to assess the security of LL methods have been suggested in [4], [5]. More precisely, it has been observed that this problem can be reduced to the Boolean satisfiability (SAT) problem and solved by applying off-the-shelf SAT-solvers. When formulated in the probably approximately correct (PAC) learning framework, the approximation-based versions of SAT attacks have been applied to deobfuscate circuits effectively [5]. Furthermore, it has been attempted to prove the impossibility of building robust LL schemes, when the adversary is given access to an oracle providing her with the outputs of an unlocked design [4]. In our paper, we mainly consider these results to put emphasis on different angles of the problem, which are crucial for conducting thorough PAC learning-based security assessments.

B. Physically Unclonable Functions

In practice, secure key generation and storage have been the main challenges to ensure the security of a cryptographic primitive. As the conventional methods for this failed to reach the desired level of security, Physically Unclonable Functions (PUFs) have been considered as a promising solution. PUFs can be thought of as mappings, where the input/output (so-called, challenge-response) behavior is determined by the physical characteristics of the system embodying the PUF. Among all PUFs, Arbiter PUFs are one of the most celebrated types of PUF, where the delays of two symmetrically-designed paths are used to generate an instance-specific response to a given challenge. Shortly after the introduction of Arbiter PUFs, it was demonstrated that these circuits are "not difficult enough to model," cf. [6].

XORing several chains of Arbiter PUFs has been suggested to overcome the above shortcoming of Arbiter PUFs [7]. It is clearly an example of a composition of several instances, aiming at achieving robustness against ML attacks. Nonetheless, this new type of PUF, called XOR Arbiter PUF, has come under non-invasive, machine learning (ML) attacks, ranging from empirical to provable [8], [9]. In the latter case, it has been proven that if the number of chains exceeds an upper bound, a class of provable ML algorithms cannot be applied against XOR Arbiter PUFs [9]. This result has been applied to design new PUF constructions [10]. However, it seems that the essential aspects of the framework applied to establish this bound have been overlooked, which we discuss in this paper.

Besides XOR Arbiter PUFs, we concentrate our attention on security assessments of Bistable Ring (BR) PUFs as one of the most interesting families of PUF [11]. This is indeed tempting, since for BR PUFs, no concrete, mathematically precise model is known to model the internal functionality of PUF instances.

C. Probably Approximately Correct Learning

This framework deals with the problem of generating a hypothesis that is a good approximation of the unknown target, with a given level of probability [12]. More formally, a set of labeled *examples* (i.e., a set of input/output pairs) is given to a learning algorithm A that generates, with high probability, an approximately correct hypothesis. The examples are drawn according to a fixed, arbitrary probability distribution D on the instance space $C_n = \{0,1\}^n$. Our *target concept class* is a collection of Boolean functions $F = \cup_{n\geq 1} F_n$, defined over C_n. Similarly, a hypothesis $h \in H_n$ is a Boolean function over C_n, which is called an ε-approximator for $f \in F_n$, if

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

Definition 1 formulates how the algorithm A works:

Definition 1. *For the target concept class F, an algorithm A is called a PAC learning algorithm , if for any $n \geq 1$, any distribution D, any $0 < \varepsilon, \delta < 1$, and any $f \in F_n$ the following holds. When A is given a polynomial number of labeled examples, it runs in time polynomial in n, $1/\varepsilon$, $VC_{dim}(F)$, $1/\delta$, and returns an ε-approximator for f under D, with probability at least $1 - \delta$.*

For this definition, the Vapnik-Chervonenkis (VC) dimension $\mathrm{VC_{dim}}(F)$ is used to provide the measure of the class.

III. Learning under Uniform Distribution

According to Definition 1, one of the most critical parameters is the *target distribution* D, on which no restriction is imposed rather than being fixed. Specifically, the PAC learning algorithm A - run by an attacker - is needed to perform well (i.e., in terms of the accuracy and the confidence levels) with respect to *any* distribution D. Despite the fact that this general setting is considered interesting for ML community, for complexity theory and cryptography, another variant of PAC learning has become more common, namely, uniform-distribution PAC learning. The reason for this shift in the focus of relevant studies is two-fold: (1) the development of efficient algorithms in original PAC learning is more challenging [13], and (2) the requirement regarding distribution-independency makes PAC learning of relatively simple concept classes challenging [14]. In this regard, interestingly enough, when the assumption regarding the distribution-independent model is relaxed, different, often positive results (i.e., a target concept class is PAC learnable) can be obtained. Next, we will discuss such results in the context of PUFs and LL schemes.

Uniform PAC learning of LL schemes: First and foremost, we note that for the class $\mathtt{AC}^0$ containing $poly(n)$-size depth-d circuits, considered in the context of LL, the running time of a non-trivial distribution-free learning algorithm cannot

be better than $2^{n-n^{\Omega(1/d)}}$ [15]. On the contrary, when the uniform variant of the PAC learning framework is taken into account, to learn the circuits in $\mathtt{AC}^0$, a polynomial-time algorithm has been devised [16]. Hence, when it comes to the application of PAC learning for LL, this variant must have been considered, although being not explicitly stated in the literature, see, e,g., [4], [5]. In other words, by the term "random" input/output pairs widely used in the LL-related literature, *uniformly* distributed examples are meant; however, this term is used to refer to *arbitrarily* distributed examples in the ML-related literature.

A. Uniform PAC Learning of XOR Arbiter PUFs

As mentioned in Section I, PAC learning of XOR Arbiter PUFs, more specifically, the bound proved in [9] has been attracted a great deal of attention in the hardware security community. This upper bound relies on an observation made in the literature that is, Arbiter PUFs can be represented by linear threshold functions (LTFs) [6], [8]. In this respect, an n-bit XOR Arbiter PUF composed of k chains of Arbiter PUFs can be represented by the function $f : C_n = \{-1, +1\}^n \rightarrow Y = \{-1, +1\}$, where $y = f(c) = \mathrm{sgn}\left((\sum_{i=1}^{n} \omega_i c_i) - \theta\right)$ with the coefficients $\omega_1, \omega_2, \cdots, \omega_n, \theta \in \mathbb{R}$. In this formulation, the set of challenges are denoted by C_n, whereas the set Y includes the responses. Note that to define this function the following encoding scheme is performed $\chi(0_{\mathbb{F}_2}) := +1$, and $\chi(1_{\mathbb{F}_2}) := -1$. It has been shown that the coefficients associated with the LTFs representing XOR Arbiter PUFs are functions of k and n. Based on this result, the upper bound has been established (see, Corollary 2 in [9], and Table I), based on the upper mistake-bound of the Perceptron algorithm. Two aspects of this result are particularly important.

1. Algorithm-independent and uniform PAC learning: First, the bound in [9] is algorithm-specific, i.e., for other algorithm this bound may not hold. Here we go one step further and explore the change in the upper bound in the algorithm-independent setting. In doing so, we apply the general bound proposed in [12], which depends on the levels of accuracy and confidence as well as the $\mathrm{VC}_{\mathrm{dim}}$. For an XOR Arbiter PUF, $h(c)$, $\mathrm{VC}_{\mathrm{dim}} h(c) = O(k(n+1)(1+\log(kn+k)))$ cf. [17]. Hence, it is straightforward to achieve the so-called general bound reported in Table I, summarizing our new results along with the previous ones. This bound indicates that if there is an algorithm (without specifying that) to learn the target concept, at most, how many examples is required.

2. Algorithm-dependent and uniform PAC learning: The bound in [9] has been established in the original, distribution-independent PAC learning framework. Let us examine whether the bound can vary if we consider the uniform-distribution PAC learning. To this end, among various uniform-distribution PAC learning algorithms, we take the low degree algorithm, so-called LMN algorithm, into account. This combines two advantages: (1) the LMN algorithm can tolerate the noise in its given examples (for a discussion on the inherent noise in the XOR Arbiter PUFs, see [17]) , and (2) for this algorithm, no limitation is imposed on the hypothesis class. By taking advantage of the second property, we provide a stronger security assurance as the freedom given to the learning algorithm can make it potentially more powerful [18].

LMN algorithm: This algorithm relies on the spectral properties of Boolean functions by analyzing the Fourier expansion of them [16]. To define this expansion of a Boolean function, we use the encoding scheme defined above to write

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

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

The notion underlying the LMN algorithm is that the Fourier expansion of some Boolean functions features *low* coefficients, sufficient to approximate the respective functions [16]. One class of function with this property include functions, whose *noise sensitivity* is small and bounded[1]. Informally, for PUFs, the noise sensitivity of a Boolean function indicates the probability of receiving a new response, when the challenge bits are flipped independently, with some probability. More precisely, the noise sensitivity of f at ε is $\mathrm{NS}_\varepsilon(f) := \Pr[f(c) \neq f(c')]$, where c is a uniformly chosen challenge and c' is obtained by flipping *each bit* of the string c independently, with probability ε ($0 \leq \varepsilon \leq 1$). For some classes of Boolean function, the noise sensitivity can be analyzed thoroughly and upper bounds for that can be established accordingly. LTFs are examples of such function, where it has been proven that for any LTF f, we have $\mathrm{NS}_\varepsilon(f) = O(\sqrt{\varepsilon})$. More importantly, for any function of k LTFs, $h : \{-1, +1\}^n \rightarrow \{-1, +1\}$ with $h = g(f_1, \cdots f_k)$, $\mathrm{NS}_\varepsilon(h) = O(k\sqrt{\varepsilon})$ [20]. This can be now used to derive the upper bound on the number of CRP for the LMN algorithm:

Corollary 1. *For an XOR arbiter PUF represented by the Boolean function h, the LMN algorithm can deliver a set of Fourier coefficients approximating h, where the number of examples is polynomial in n, k^2/ε^2, and $\ln(1/\delta)$.*

Proof: As the noise sensitivity of h is $\mathrm{NS}_\varepsilon(h) = O(k\sqrt{\varepsilon})$, fix $\alpha(\varepsilon) = k\sqrt{\varepsilon}$. According to theorem underlying the notion of the LMN algorithm, the set of low coefficients S that can approximate h with the accuracy ε fulfills the inequality $\sum_{|S| \geq m} \hat{f}(S)^2 \leq \varepsilon$, where $m := 1/\alpha^{-1}(\varepsilon/2.32))$, and $\alpha^{-1}(\cdot)$ denotes the inverse of the function $\alpha(\cdot)$ cf. [16]. To compute S, the LMN algorithm requires $n^{O(m)} \ln(\delta^{-1})$ example. For our XOR Arbiter PUFs, represented by h, it is straightforward to show $m = 2.32k^2/\varepsilon^2$. Note that here we implicitly assume that $\varepsilon \leq 1/k^2$ to make the function $\alpha(\cdot)$ strictly increasing continuous over the range $[0, 1]$. ■

The implication of Corollary 1 is that, when $k = O(1)$, i.e., k is a constant value, the LMN algorithm can PAC learn the XOR Arbiter PUF. On the other hand, if $k \gg \sqrt{\ln n}$, applying this algorithm becomes infeasible. This is in line

[1]The noise here refers to the attribute noise studied in ML. It is related to impact of "hidden" factors that can influence the response of the PUF to a given challenge, e.g., meta-stability of a PUF stage, aging, etc. [17], [19]

TABLE I: Summary of the upper bounds on the number of CRPs required to PAC learn XOR Arbiter PUFs

Bound of Learning	Setting			
	Bound	***Distribution***	***Algorithm***	***Attackers' Access***
[9] [a]	$O\big((n+1)^k/\varepsilon^3 + \ln(1/\delta)/\varepsilon\big)$	Arbitrary	Perceptron	Random examples
General	$O((k(n+1)(1+\ln(kn+k)\ln(1/\varepsilon)+\ln(1/\delta)))/\varepsilon)$	Uniform	Independent	Uniformly-distributed examples
Corollary 1	$O\big(n^{k^2/\varepsilon^2}\ln(1/\delta)\big)$	Uniform	LMN [16]	Uniformly-distributed examples
Corollary 2	$poly(n, k, (1/\varepsilon), \log(1/\delta))$	Uniform	LearnPoly [21]	Membership queries

[a] Note that this bound does not depend on the VC_{dim} of XOR Arbiter PUFs, but the mistakes made by the Perceptron algorithm.

with what has been reported in the original, distribution-free PAC learning framework, namely when $k = O(\ln n)$, the PAC learner fails. Note that this result does not contradict what has been achieved in [17], where XOR Arbiter PUFs with a large number of arbiter chains have been modeled through applying the LMN algorithm. In this respect, the difference is two-fold: (1) here we assume that the chains of an XOR Arbiter PUF are uncorrelated, whereas in [17], the chains are made correlated intentionally, (2) we do not consider the impact of the inherent bias of XOR Arbiter PUFs here, while in [17], the expected bias (i.e., the bias in the presence of the attribute noise) has been introduced to evaluate the hardness of PUFs.

IV. Access Models of Attackers

An important aspect of an attack against cryptosystems is the access given to an attacker to gather information about the system, more precisely, the unknown function describing the internal functionality of the system. While in cryptography, the attacker's access to the inputs/outputs of the system has been accurately classified, in the context of machine learning, membership and equivalence queries have been defined to deal with the learning scenarios. A membership query refers to the case where the attacker can ask the value of the unknown function on some specified inputs, similar to chosen-plaintext attacks. By providing access to the equivalence queries, the attacker can go one step further by asking whether the hypothesis determined by her is equivalent to the unknown target hypothesis. Although one may think that this type of queries may not be available for security evaluation of hardware primitives, cf. [4], according to a well-known result provided by Angluin [22], equivalence queries can be simulated using random examples in a straightforward manner. Hence, here, we consider solely access to the membership queries that can have a significant impact on the evaluation results.

A. Membership Queries for Learning LL Schemes

As explained before, for LL methods, one must consider the uniform PAC learning framework. Interestingly enough, when membership queries are also allowed, the analysis of the learnability leads to a drastically different result. More concretely, the distinction between the exact and approximate learning of obfuscated circuits has been made in [4]. As stated by Rivest [2], in cryptography, an attacker aims at exactly identifying the unknown function, whereas ML concerns not only this but also an approximation of the unknown function. In this regard, what has been suggested in [4] is that although the approximation-resiliency of obfuscated circuits cannot be guaranteed, for some obfuscated circuits, it is possible to ensure the exact inference-resiliency. To formulate this in the learning framework, a key aspect has not received sufficient attention; that is, uniform PAC learning (i.e., approximate learning) algorithms can be straightforwardly converted to *exact* learning one with membership queries [15]. Hence, the impossibility of exact learning is not relevant.

B. Membership Queries for Learning PUFs

To provide an example of how the ML results can be different, when the attacker is given the access the the membership queries, let us consider again XOR Arbiter PUFs as stated in the following corollary.

Corollary 2. *For an XOR arbiter PUF with* $\log(n)$ *chains, if the ML algorithm is given access to the membership queries, the runtime of the algorithm is* $poly(n)$.

Proof: As each Arbiter PUF can be modeled by an LTF, according to the noise sensitivity of LTFs (see Section III) and the Bourgain's theorem [23], it is close to an $O(\varepsilon^{-3/2})$-junta, i.e., small juntas. Moreover, it is known that the class of r-juntas is a subset, or equal, to the class of r-XT that is the class of XOR of terms, $T_1 + T_2 + \cdots + T_s$, where T_i's are the terms with the size at most r (i.e., the conjunction of r Boolean variables), and s is the number of terms. Hence, our $O(\varepsilon^{-3/2})$-junta can be equivalent to some r-XT functions, where r is small and $r = O(\varepsilon^{-3/2})$ [24]. Consequently, when we XOR our k Arbiter PUFs, the combined function is of the form $O(k)$-term r-XT that can be represented as $O(2^r k)$-monomial r-Boolean Multivariate Polynomial, i.e., sparse multivariate polynomial of degree r over $\mathbb{F}_2$, cf. [24]. The immediate result of this is that the polynomial-time algorithm suggested in [21] can be applied to learn the combined function. For this, the algorithm requires $poly(n, (1/\varepsilon), \log(1/\delta))$ uniformly chosen membership queries to learn $\log(n)$-XOR Arbiter PUFs with accuracy ε and confidence δ. ■

We put emphasize on the important consequence of this result that is, XOR Arbiter PUFs constructed upon the difficulty of learning $O(\log(n))$-XOR LTFs cannot be secure against attackers given access to the membership queries.

V. Choice of Representations for ML Attacks

When assessing the security of hardware systems through ML, two issues should be tackled: (1) the internal functionality of the system should be (at least) approximated accurately and, (2), the algorithm running to learn this approximated model should deliver a hypothesis with pre-defined levels of accuracy and confidence. These two issues are discussed below.

A. Representation of the Concept Class

Regarding this case, it is known that there is a fundamental difference between a concept (i.e., a set of Boolean functions) and its representation. As explained below, this difference is

not only reflected in the size of the representations, but also on the effectiveness of learning in terms of accuracy.

Concept representations for obfuscated circuits: When analyzing the security of LL methods, the class $\mathtt{AC}^0$ is often considered, which is composed of all families of circuits of depth $O(1)$ and polynomial size, with mainly unlimited fan-in AND gates and OR gates. In this manner, the size of the circuit has a drastic impact on both of the number of examples (i.e., the input/output pairs), and the running time of the algorithm. It may be interesting to examine why this factor is not taken into account in one of the most relevant studies presented in [5]. The point is that the ML model considered in [5] is the online-ML, which can be converted to a PAC learning model. Accordingly, the impact of the size of the concept representation is reflected by the number of mistakes that the algorithm is allowed to make for a given level of accuracy.

Concept representations for PUFs: In contrast to LL scenarios, for PUFs, various representation classes have been considered. As an example of how choosing an improper representation can affect the accuracy of learning PUFs, we shift our focus to the problem of PAC learning BR PUFs. In an attempt to approximate the functionality of these PUFs, it has been suggested that LTFs can represent BR PUFs [11]. As an inevitable result of this simplified representation, according to the results presented in [11], it is not possible to arbitrarily increase the learning accuracy. In other words, the accuracy level reaches the maximum possible value (approximately 95% for various BR PUF instances) and remains the same even after increasing the number of CRPs fed into the ML algorithm. Seen from the angle of ML, it is known that to tackle this problem, a more expressive model could be helpful. For this purpose, two main directions can be explored.

1. Regularity of LTFs representing BR PUFs: As the regularity of a representation (i.e., having only a few large coefficients) plays an important role in its learnability, it could be thought that if the representation of a BR PUF (i.e., LTFs) can be made regular, the accuracy of learning can be increased to an arbitrarily high level. This is according to a well-known result in ML, which states that for any LTF f defined over $\{0,1\}^n$, there is a linear threshold function f' so that it is ε-close to f and all of its weights are integers of magnitude at most $\sqrt{n}(1/\varepsilon)^{O(log^2(1/\varepsilon))}$ [25]. Such a function f' can be built upon the Chow parameters of f, i.e., $n+1$ degree-0 and degree-1 Fourier coefficients of f, namely $\hat{f}(0) = \mathbf{E}[f(x)]$, and $\hat{f}(i) = \mathbf{E}[f(x)x_i]$ for $i = 1, \cdots, n$. To approximate the Chow parameters and construct f', a minimum number of $\Omega(n)$ labeled examples should be given to a polynomial-time algorithm that outputs f' for a constant ε, e.g., $\varepsilon = 0.01$.

Note that if a BR PUFs can be approximated by LTFs, then f' must approximate it with a constant, arbitrarily small, predefined level of ε. To investigate this, we conduct experiments on BR PUFs implemented on an Intel/Altera Cyclone IV FPGA, manufactured on a 60nm technology [26]. In this regard, by using a small set of noiseless CRPs, we approximate the Chow parameters and follow the algorithm suggested in [25] to construct f'. Afterward, we apply the Perceptron algorithm embedded in Weka [27] to learn the CRPs obtained from f', i.e., for each challenge, the response is computed and inserted in the training set. The remaining CRPs are involved in the test set, composed of 44834, 35876, and 31375 noiseless and stable CRPs collected from 16-, 32-, and 64-bit BR PUF, respectively. We expect that if BR PUFs can be represented by LTFs, the output/input relationship of f' can be learned and generalized to the training set. The results of our experiments are presented in Table II, where the key insight is that the above does not hold. Thus, our assumption regarding representing BR PUFs by LTFs is not valid.

TABLE II: Results of learning an LTF f' built upon Chow parameters approximated by using the CRPs collected from BR PUFs. Even with an increase in the number of CRPs in the training set, the accuracy cannot be increased arbitrarily.

# Noiseless CRPs for computing the Chow parameters and in training set	Accuracy		
	16	32	64
1000	71.93	91.52	92.55
2500	81.02	92.04	93.80
5000	84.94	91.45	93.57
10000	88.65	91.85	93.69

TABLE III: Results of testing how far BR PUFs are to LTFs.

n	# CRPs	How far from any halfspace (min.) [%]
16	100	20
32	1339	40
64	63434	50

2. Testing halfspaces: It can be argued that since we first represent a BR PUF by an LTF and further approximate its corresponding Chow parameters, the error of the whole process could have increased so that the accuracy of learning cannot be arbitrarily high. In another attempt to confirm that an inappropriate representation, namely, LTFs, results in this, we employ a property testing algorithm. More specifically, a property tester can with high probability examine how close an unknown function can be to a class of Boolean functions. To test if BR PUFs are close to LTFs (or so-called, halfspaces), we run a halfspace tester proposed in [28]. In brief, with high probability δ, this tester distinguishes halfspaces from other Boolean functions, which are ε-far from *any* halfspaces, when given $poly(1/\varepsilon)$ uniformly chosen examples - noiseless CRPs in our case. We implement the algorithm in MATLAB, into which we feed the CRPs collected from our BR PUFs. The results of this experiment are summarized in Table III, where $\delta = 0.99$. As can be understood from these results, BR PUFs are not close to halfspace, which confirms our results discussed before.

B. Representation of the Hypothesis Class

Another crucial aspect of an ML-based analysis is to stress how the representation of what should be delivered by the machine (i.e., the hypothesis) can have an impact on the results. In view of the fact that an unknown concept class can be learned under a hypothesis class, but not others, special care must be taken. In other words, if more expressive hypothesis representations are allowed, the analysis of the learnability yields different results. Therefore, it can be desired to remove the dependency on the hypothesis representation and give the learner the freedom to return any hypothesis. In PAC learn-

ing framework, this refers to *improper* learning. Ironically, although being called improper, ML algorithms categorized in this class are more powerful than proper learners, which output a specific hypothesis representation.

Improper learning of PUFs: In the context of PUFs, the significance of improper learning has been already recognized in the literature, see, e.g., [17], [19]. In line with that result, as the LMN algorithm is an improper ML algorithm, the bound that we have derived in Corollary 1 serves as another example of applying improper learning algorithms against PUFs. Specifically, we put emphasis again on the results of learning XOR arbiter PUFs presented in [17], where XOR Arbiter PUFs with a large number of chains ($k \gg \ln n$) have been learned with a reasonable level of accuracy (approximately 75%). This result is obtained by employing the LMN algorithm within the uniform PAC learning framework. At first sight, it could seem to be contrary to what has been proved in [9]; however, now after explaining the factors contributing to that (the distribution of the example and the type of the algorithm), it should be clear that those results are not comparable.

Hypothesis representation for obfuscated circuits: To provide an example of what may constitute a reason for deciding whether an obfuscated circuit can be learned, we explore a representation-dependent result considered in [4]. In a nutshell, it has been discussed if an obfuscated sequential circuit would be vulnerable to PAC learning-based attacks. To answer this question, the authors of [4] have suggested that deterministic finite automaton (DFA) representation of FSMs can be learned through Angluin's method [22], if the number of possible input patterns to the FSM would not be exponential. It should not be overlooked, however, that Angluin's algorithm delivers DFAs, and thus, improper ML algorithms can be further taken into account.

VI. Conclusion

This paper aims to highlight the impact of ML settings on the security assessment of standalone and composed hardware. To this end, we demonstrate the relationship between these settings and the freedom given to an adversary in terms of having access to learning examples and models. In addition to providing theoretical insight into why the ML setting is critical, we discuss examples of real-world hardware primitives, where inappropriate choices of the parameters can result in less accurate security assessment.

Acknowledgment

We acknowledge the effort made by Mr. Antonio Cavotta to compute the Chow parameters, when he was doing his master thesis under the supervision of Profs. Seiferet and Neitzert. This work is supported by National Science Foundation under grant agreement No. CNS 1651701, National Institute of Standards and Technology under grant No. 60NANB16D248 and AFOSR under award No. FA9550-14-1-0351.

References

[1] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," in *Proc. of Symp. on Foundations of Comp. Sci.*, pp. 136–145, IEEE, 2001.

[2] R. L. Rivest, "Cryptography and Machine Learning," in *Intrl. Conf. on the Theory and Application of Cryptology*, pp. 427–439, Springer, 1991.

[3] S. Dupuis and M.-L. Flottes, "Logic locking: A survey of proposed methods and evaluation metrics," *J. of Elec. Testing*, pp. 1–19, 2019.

[4] K. Shamsi, D. Z. Pan, and Y. Jin, "On the Impossibility of Approximation-Resilient Circuit Locking," in *Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*, pp. 161–170, IEEE, 2019.

[5] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *Intrl. Symp. on Hardware Oriented Security and Trust (HOST)*, pp. 95–100, IEEE, 2017.

[6] B. Gassend, D. Lim, D. Clarke, M. Van Dijk, and S. Devadas, "Identification and Authentication of Integrated Circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

[7] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Proc. of the Design Automation Conf.*, pp. 9–14, 2007.

[8] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling Attacks on Physical Unclonable Functions," in *Proc. of the ACM Conf. on Comp. and Comm. Security*, pp. 237–249, 2010.

[9] F. Ganji, S. Tajik, and J.-P. Seifert, "Why Attackers Win: On the Learnability of XOR Arbiter PUFs," in *Trust and Trustworthy Comp.*, pp. 22–39, Springer, 2015.

[10] M. D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication," *IEEE Trans. on Multi-Scale Comp. Sys.*, vol. PP, no. 99, 2016.

[11] X. Xu, U. Rührmair, D. E. Holcomb, and W. P. Burleson, "Security Evaluation and Enhancement of Bistable Ring PUFs," in *Radio Frequency Identification*, pp. 3–16, Springer, 2015.

[12] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the Vapnik-Chervonenkis Dimension," *J. of the ACM*, vol. 36, no. 4, pp. 929–965, 1989.

[13] R. O'Donnell and R. A. Servedio, "Learning Monotone Decision Trees in Polynomial Time," *SIAM J. on Computing*, vol. 37, no. 3, pp. 827–844, 2007.

[14] M. Kharitonov, "Cryptographic Hardness of Distribution-specific Learning," in *Proc. of ACM Symp. on Theory of Computing*, pp. 372–381, ACM, 1993.

[15] R. A. Servedio and L.-Y. Tan, "What Circuit Classes Can Be Learned with Non-trivial Savings?," in *8th Innovations in Theoretical Comp. Sci. Conf.*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[16] N. Linial, Y. Mansour, and N. Nisan, "Constant Depth Circuits, Fourier Transform, and Learnability," *J. of the ACM*, vol. 40, no. 3, pp. 607–620, 1993.

[17] F. Ganji, S. Tajik, P. Stauss, J.-P. Seifert, D. Forte, and M. Tehranipoor, "Rocknroll PUFs: Crafting Provably Secure PUFs from Less Secure Ones," in *Proc. of Intrl. Workshop on Security Proofs for Embedded Systems*, vol. 11, pp. 33–48, 2019.

[18] A. Daniely, N. Linial, and S. Shalev-Shwartz, "From Average Case Complexity to Improper Learning Complexity," *arXiv preprint arXiv:1311.2272*, 2013.

[19] F. Ganji, S. Tajik, and J.-P. Seifert, "A Fourier Analysis Based Attack against Physically Unclonable Functions," in *Intl. Conf. on Financial Crypto. and Data Security*, Springer, 2018.

[20] A. R. Klivans, R. O'Donnell, and R. A. Servedio, "Learning Intersections and Thresholds of Halfspaces," in *Foundations of Comp. Sci., Annual IEEE Symp. on*, pp. 177–186, 2002.

[21] R. E. Schapire and L. M. Sellie, "Learning Sparse Multivariate Polynomials over a Field with Queries and Counterexamples," *J. of Comp. and System Sciences*, vol. 52, no. 2, pp. 201–213, 1996.

[22] D. Angluin, "Learning Regular Sets from Queries and Counterexamples," *Information and computation*, vol. 75, no. 2, pp. 87–106, 1987.

[23] J. Bourgain, "On the Distribution of the Fourier Spectrum of Boolean Functions," *Israel J. of Mathematics*, vol. 131, no. 1, pp. 269–276, 2002.

[24] N. H. Bshouty, "Exact Learning from an Honest Teacher that Answers Membership Queries," *Theoretical Comp. Sci.*, vol. 733, pp. 4–43, 2018.

[25] A. De, I. Diakonikolas, V. Feldman, and R. A. Servedio, "Nearly Optimal Solutions for the Chow Parameters Problem and Low-weight Approximation of Halfspaces," *J. of the ACM*, vol. 61, no. 2, p. 11, 2014.

[26] Altera, "Cyclone IV Device Handbook," *Altera Corp., San Jose*, 2014.

[27] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.
[28] K. Matulef, R. O'Donnell, R. Rubinfeld, and R. A. Servedio, "Testing Halfspaces," *SIAM J. on Comp.*, vol. 39, no. 5, pp. 2004–2047, 2010.