Breaking the Lightweight Secure PUF: Understanding the Relation of Input Transformations and Machine Learning Resistance

Nils Wisiol, Georg T. Becker, Marian Margraf, Tudor A. A. Soroceanu, Johannes Tobisch and Benjamin Zengin

***Abstract*—Physical Unclonable Functions (PUFs) and, in particular, XOR Arbiter PUFs have gained much research interest as an authentication mechanism for embedded systems. One of the biggest problems of (strong) PUFs is their vulnerability to so called machine learning attacks. In this paper we take a closer look at one aspect of machine learning attacks that has not yet gained the needed attention: the generation of the sub-challenges in XOR Arbiter PUFs fed to the individual Arbiter PUFs. Specifically, we look at one of the most popular ways to generate sub-challenges based on a combination of permutations and XORs as it has been described for the "Lightweight Secure PUF". Previous research suggested that using such a sub-challenge generation increases the machine learning resistance significantly.**

Our contribution in the field of sub-challenge generation is three-fold: First, drastically improving attack results by Rührmair *et al.*, we describe a novel attack that can break the Lightweight Secure PUF in time roughly equivalent to an XOR Arbiter PUF without transformation of the challenge input. Second, we give a mathematical model that gives insight into the weakness of the Lightweight Secure PUF and provides a way to study generation of sub-challenges in general. Third, we propose a new, efficient, and cost-effective way for sub-challenge generation that mitigates the attack strategy we used and outperforms the Lightweight Secure PUF in both machine learning resistance and resource overhead.

I. Introduction

Physical Unclonable Functions (PUFs), have gained much research attention since their invention in 2002. The main idea of PUFs is to use the intrinsic process variations of each chip to build an unclonable function that is device specific. PUFs with an exponential challenge space, often denoted as *Strong PUFs*, are particularly well suited for lightweight authentication scenarios. The by far most prominent PUF with an exponential challenge space is the Arbiter PUF [1] and its variant the XOR Arbiter PUF [2]. PUFs can also be used for generation or storage of cryptographic keys, even if their challenge space is of limited size, such as in the case of SRAM PUFs [3], [4]. While PUF-based key storage has matured and is used in commercial products, using PUFs in a challenge-and-response protocol remains challenging due to so-called machine learning attacks [5]. In these attacks, challenge-and-response pairs are collected and machine learning algorithms are used to approximate the PUF using a software model. While these attacks are very efficient for single Arbiter PUFs and small XOR Arbiter PUFs, the machine learning complexity increases exponentially with the number of XORs [5], [6], [7]. However, Becker [8] showed how reliability information can be used for a powerful machine learning attack with only a linear increase in machine learning complexity.

Despite these setbacks, research in PUF based authentication continues and Arbiter PUFs are still one of the most used building blocks. Many of these PUF protocols make use of the fact that Arbiter PUFs can be modeled using machine learning. During initialization, the challenges and responses are directly exposed so that an exact software model of the PUF can be build that is stored at the server. Afterward the direct access to the challenges and responses is disabled and responses are never directly exposed anymore. A good example of a PUF protocol that can withstand traditional machine learning attacks and attacks based on reliability is the lockdown protocol by Yu *et al.*[9]. A random nonce ensures that an attacker cannot collect the response for the same challenge twice. Additionally, the number of authentication requests and hence the number of exposed challenges and responses is limited so that traditional machine learning attacks are not feasible.

However, to determine the number of allowed authentications, it is crucial to fully understand the machine learning resistance of the underlying PUF construction. In this paper, we will take a closer look at an aspect that has not yet gained the required attention: the input transformation generating the sub-challenges. Rührmair *et al.*[5] have shown that attacking an XOR Arbiter PUF in which each arbiter chain has the same challenge is easier than attacking one in which each arbiter chain gets different challenges. This was later verified by other researchers (see e.g. [6], [9]). The results of [5] are based on the input transformation by Majzoobi *et al.* [10] for their Lightweight PUF of 2008, which has also been used in the Slender protocol [11] and, with some variations, in [12]. One design goal of the input transformation was to achieve the avalanche criterion. The impact of the input transformation has recently been investigated in respect to chosen challenge attacks [13]. Yet, in how far such transformations are actually

Nils Wisiol, Marian Margraf, and Tudor A. A. Soroceanu are with the Institute of Computer Science of Freie Universität Berlin, Germany; Nils Wisiol is also with the Chair for Security in Telecommunications of Technische Universität Berlin. Contact: nils.wisiol@fu-berlin.de, marian.margraf@fu-berlin.de, tudor.soroceanu@fu-berlin.de.

Georg Becker is with the Digital Society Institute at the ESMT Berlin, Germany. He is being supported by Rheinmetall. Contact: georg.becker@esmt.org.

Johannes Tobisch is with the Horst Görtz Institute for IT-Security at Ruhr-Universität Bochum, Germany. Contact: johannes.tobisch@ruhr-uni-bochum.de

Benjamin Zengin is with the Fraunhofer Institute for Applied and Integrated Security at Berlin, Germany. Contact: benjamin.zengin@aisec.fraunhofer.de.

optimal to counter current state-of-the-art machine learning attacks has not been studied.

A. Main Contribution

In this paper we close this gap and perform a thorough analysis of the impact of input transformations on state-of-the-art machine learning attacks. Our analysis shows that one has to carefully chose the input transformation to achieve the desired machine learning resistance. In particular, on first sight the often cited Lightweight input transformation from Majzoobi *et al.*[10] seems to be close to the case of random sub-challenges. (In Section III-A we argue random sub-challenges are hardest to learn.) However, we show that by using this input transformation the logistic regression (LR) learner has a significant probability to converge to a local minimum, providing only a partially accurate model of the XOR Arbiter PUF. We show in a novel attack how these local minima can be exploited to efficiently model PUFs based on the Lightweight input transformation. We furthermore discuss the reasons for these local minima and subsequently present an easy-to-implement input transformation that achieves a modeling resistance comparable to the optimal solution based on random sub-challenges.

II. Background

A. Machine learning attacks on PUFs

It was already shown in 2004 by Lim [14] that Arbiter PUFs can be modeled using Support Vector Machine (SVM) learning techniques and a linear delay model. They proposed the XOR Arbiter PUF and the feed-forward Arbiter PUF to increase the machine learning resistance by increasing the non-linearity of the PUF model. In 2010, Rührmair *et al.* [5] provided an extensive study of the machine learning resistance of the XOR Arbiter PUF and the feed-forward Arbiter PUF. On the one hand, their result showed that these constructions can be modeled quite efficiently for reasonably sized PUFs. On the other hand, they showed that the modeling complexity grows exponentially with the number of XORs. Based on these results it would be theoretically possible to build secure XOR Arbiter PUFs, but also come with quite large area and power requirements. While their initial study was conducted on simulated PUFs, they later confirmed these findings using real silicon data [7]. Other researchers have since then confirmed these results [6], [9]. While Rührmair *et al.* used logistic regression for their attack on XOR Arbiter PUFs, other machine learning techniques such as SVM, evolution strategies, artificial neural networks have been used as well. While there are performance differences between these techniques, no technique fundamentally changed the exponential grow in required number of responses and challenges.

There have been proposals of new Strong PUFs that exhibit inherently more resistance against machine learning attacks due to a non-linear response behavior. An early example is the Bistable Ring PUF [15] which can be implemented on FPGAs but can still be modeled using a linear model [16]. Another research direction looks into using non-linear voltage transfer characteristics to build strong PUFs [17]. However, no breakthrough has been achieved that can prevent modeling attacks on these primitives so that they cannot be used securely on their own [18], [19]. Researchers have also investigated to what extent side-channel information can be used to model Arbiter PUFs, such as power consumption [20], [21], reliability [22], and even optical emissions [23]. While power and optical side-channels require additional measurement capabilities from the attacker and hence increase the attack costs, reliability based machine learning attacks are different in that regard. All they require is the ability to query the PUF via the standard PUF interface multiple times for the same challenge. Hence, they can be performed without any additional measurement requirements and can also be carried out remotely. Such reliability based attacks can be augmented using active fault attacks by operating the PUF device outside of normal conditions [24], [21]. However, due to the unreliability of Arbiter PUFs in practice such active attacks are only necessary if error-correction is used.

The real power of the reliability based machine learning attacks were shown in 2015 by using the reliability information to model XOR Arbiter PUFs with linear complexity [8]. To counter reliability-based machine learning attacks, protocols can be used in which part of the challenge is generated by the PUF device so that the attacker cannot collect responses for the same challenge multiple times to determine the reliability. To be able to do this, a software model is learned during a set-up step by the verifier which can later be used to authenticate the PUF device. A prime example for such an approach is the lockdown technique by Yu *et al.* [9] that was designed to withstand all state-of-the-art machine learning attacks by additionally limiting the number of allowed authentication requests and hence the number of challenges and responses an attacker can collect. Among other PUF protocols that make use of a software model of an Arbiter PUF are [11], [12], [25], [26], [27], [28]. However, it is also noteworthy that many of the proposals do not achieve the claimed security [29]. It should also be noted that Arbiter PUFs or variants thereof have been proposed as primitives for secure key generation [30], [31]. The underlying assumption is that the exposed helper data does not expose enough information that the PUF can be modeled using machine learning. However, this helper data can leak more information than one might assume [32] and hence it is important to fully understand the machine learning resistance of the underlying PUF primitive.

B. Notation

Throughout this paper, we will use natural numbers n, k. Unless specified otherwise, lowercase Latin variables represent vectors, with their elements referenced by a subscript index; lowercase Greek letters and f will represent functions. Note that we also use subscript indices to refer to elements of vectors that are function values, e.g. $\sigma(c)_i$ for the i-th entry of $\sigma(c) \in V^n$ for a function σ and vector space V^n. For two vectors $v, w \in V^n$, we define $\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i$ to be the inner product of v and w. Lists of vectors $v^{(i)}$ will often be denoted as $(v^{(1)}, ..., v^{(k)})$, i.e., vectors of that list are referred to by the superscript index. We denote bits as $-1, 1$ rather

than 0, 1, where −1 is TRUE. Note that the XOR operation is hence represented by the product of two bits. We define sgn x to be the sign of any real number x, and define sgn(0) = 1 arbitrarily. Unless otherwise specified, probabilities are taken uniformly random for independent bits.

C. Modeling XOR Arbiter PUFs

Figure 1 depicts a 2-XOR Arbiter PUF that consists of two individual Arbiter PUFs with their response XORed. Each Arbiter PUF consists of n delay stages consisting of 2-input multiplexers through which two signals are propagated. The multiplexers interchange the two signals depending on the applied challenge and an arbiter at the end measured if a signal arrives first at the top or bottom line to determine the response bit. The delays of the individual Arbiter PUFs are additive, i.e., the final delay difference between the two signals is the sum of the delay difference of the individual stages. A challenge bit $c_i = -1$ swaps the two signals and can be modeled by multiplying the delay difference $\delta(i)$ at stage i with minus one. This way a recursive formula can be constructed to model the delay difference $\delta(i)$ at stage i

$$\delta(i) = \delta(i-1, c) \cdot c_i + s_i(c_i) \tag{1}$$

where $s_i(c_i)$ is the delay difference introduced at stage i for challenge c_i. The sign of the final delay difference $\delta(n)$ at stage n then defines the response bit. The above recursive formula can be simplified into a linear threshold function [14]. We follow the same approach as other researchers (e.g., [5]) and model an n-bit k-XOR Arbiter PUFs based on a product of *linear threshold functions (LTF)* given by

$$f(c) = \prod_{l=1}^{k} \operatorname{sgn}\left\langle w^{(l)}, x^{(l)} \right\rangle = \operatorname{sgn} \prod_{l=1}^{k} \left\langle w^{(l)}, x^{(l)} \right\rangle, \tag{2}$$

where the *weight vectors* $w^{(1)}, ..., w^{(k)} \in \mathbb{R}^{n+1}$ model the physical properties of the k arbiter chains (derived from s), and $x^{(1)}, ..., x^{(k)} \in \{-1, 1\}^{n+1}$ are the *feature vectors* for the given master-challenge $c \in \{-1, 1\}^n$. The feature vectors $x^{(l)}$ can be computed from the k *sub-challenges* $c^{(l)}$ given to the individual arbiter chains using the function ATT : $\{-1, 1\}^n \to \{-1, 1\}^{n+1}$,

$$\begin{gathered} x^{(l)} = \mathrm{ATT}(c^{(l)}) \text{ with} \\ x_i^{(l)} = \prod_{j=i}^{n} c_j^{(l)} \text{ for } 1 \le i \le n \\ x_{n+1}^{(l)} = 1 \end{gathered} \tag{3}$$

In analogy to LTF, we call ATT the *arbiter threshold transform*.

III. Input Transformations: Classic vs. Random

When XOR Arbiter PUFs were proposed by Suh and Devadas [2], the first step was to provide all arbiter chains with the same challenge (here called *classic* design). Subsequently, Majzoobi et al. [10] proposed to modify the challenge before feeding it into the individual arbiter chains, to let the PUF fulfill the strict avalanche criterion. Although initially designed to harden XOR Arbiter PUFs against chosen-challenge attacks,

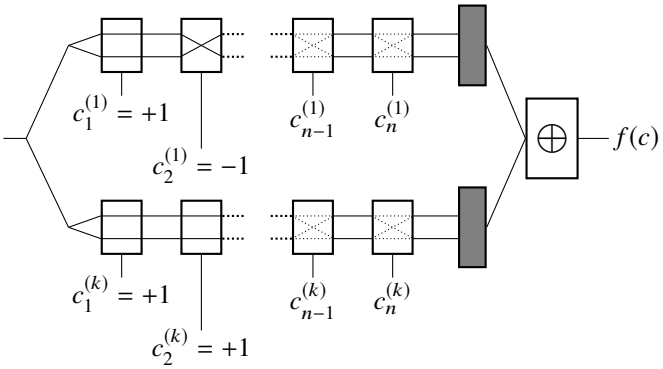

Fig. 1. Schematic representation of an XOR Arbiter PUF. (The case $k = 2$ is shown, but the scheme can easily be extended.) After the challenge is set up, a rising edge is input on the left-hand side, with the arbiters at the end of each chain (gray rectangles) measuring if the top line or bottom line shows the signal first. Duing this process, the $n \cdot k$ challenge bits $c^{(l)} \in \{-1, 1\}^n$, $1 \le l \le k$ decide at each stage (white rectangles), if the signal paths are crossed or not. The arbiter result bits of each line are then xored and output as $f(c)$ on the right-hand side.

it became clear that the design twist also has an impact on the passive (that is, non-adaptive) regression attack introduced by Sölter [33] and Rührmair et al. [5]. In this work, we generalize the idea of transforming challenges for each arbiter chain and call it *input transformation*. To shed some light on how machine learning hardness can be increased using an input transformation, we studied the impact of input transformations on the success rate of logistic regression attacks.[1]

We use the linear model introduced in the background section for modeling the XOR Arbiter PUFs and assume that the sub-challenges $c^{(l)}$ can be computed from a single master-challenge $c \in \{-1, 1\}^n$. We call a list of functions $(\tau^{(1)}, ..., \tau^{(k)})$ with $\tau^{(l)} : \{-1, 1\}^n \to \{-1, 1\}^n$ that transform the master-challenge c into sub-challenges $\tau^{(l)}(c) = c^{(l)}$ the *sub-challenge generators*.

We take the classic design as an example for our notation. As all arbiter chains are fed the master-challenge, we have $\tau^{(1)} = \cdots = \tau^{(k)} = \mathrm{id}$. Hence, we can compute any feature vectors $x^{(l)}$ directly from the master-challenge given,

$$x^{(l)} = \mathrm{ATT}(\tau^{(l)}(c)) = \mathrm{ATT}(c) = (c_1 c_2 \cdots c_n, c_2 \cdots c_n, ..., c_n, 1).$$

It is crucial to distinguish sub-challenges $c^{(i)}$ from feature vectors $x^{(i)}$. Sub-challenges represent the bits that are physically fed into the arbiter chains, whereas features vectors are used to enable a modeling of arbiter chains as linear threshold function (LTF) as given in (2).

For our analysis, the feature vector structure for a given input transformation is crucial. We hence abbreviate the value $\mathrm{ATT}(\tau^{(l)}(c))$ to $\sigma^{(l)}(c)$ and formally define *input transformation* to be the list of functions $(\sigma^{(1)}, ..., \sigma^{(k)})$ that transforms the master-challenge into the feature vectors. Figure 2 summarizes our notation.

Applying our notation to the model given in (2), the model for an XOR Arbiter PUF with input transformation

[1] Attack and analysis implementation can be found at https://github.com/nils-wisiol/pypuf/.

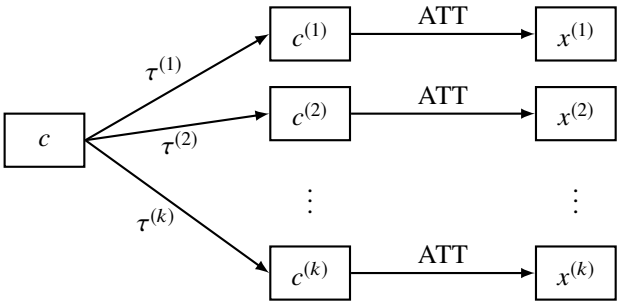

Fig. 2. Generation of sub-challenges $c^{(l)} \in \{-1,1\}^n$ and feature-vectors $x^{(l)} \in \{-1,1\}^{n+1}$ from the master-challenge $c \in \{-1,1\}^n$ using functions $\tau^{(l)} : \{-1,1\}^n \to \{-1,1\}^n$ and $\mathrm{ATT} : \{-1,1\}^n \to \{-1,1\}^{n+1}$. Note that we abbreviate $\mathrm{ATT}(\tau^{(l)}(c)) = \sigma^{(l)}(c)$ for all master-challenges c, where $\sigma : \{-1,1\}^n \to \{-1,1\}^{n+1}$.

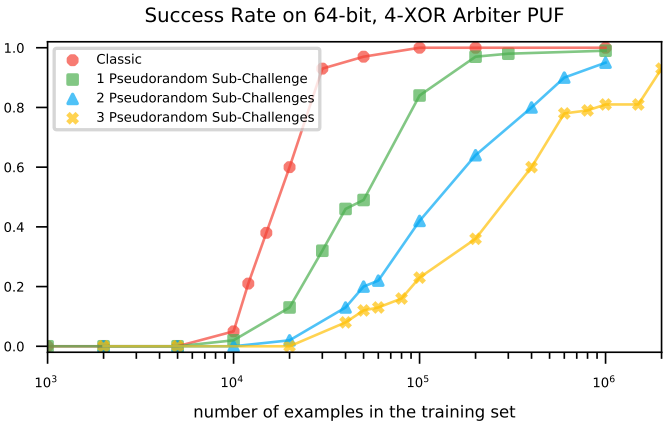

Fig. 3. Success rate of logistic regression attacks on simulated XOR Arbiter PUFs with 64-bit arbiter chains and four arbiter chains each, based on at least 250 samples per data point shown. Accuracies better than 70% are considered success (but cf. Figure 4). Four different designs are shown: of the four arbiter chains in each instance, an input transform is used that transforms zero, one, two, and three challenges pseudorandomly, keeping the remaining challenges unmodified. The success rate decreases when the number of arbiter chains with pseudorandom challenges is increased. The case with 4 pseudorandom sub-challenges is not shown as it coincides with the results for 3 pseudorandom challenges. Note the log-scale on the x-axis.

$(\sigma^{(1)}, ..., \sigma^{(k)})$ is given by

$$f(c) = \mathrm{sgn} \prod_{l=1}^{k} \left\langle w^{(l)}, \sigma^{(l)}(c) \right\rangle \tag{4}$$

where $\sigma^{(1)} = \cdots = \sigma^{(k)}$. We have for all master-challenges c that $\mathrm{ATT}(\tau^{(i)}(c)) = x^{(i)} = \sigma^{(i)}(c)$.

A. Pseudorandom Input Transformation

We demonstrate the influence of input transformations on the learning hardness of logistic regression attacks in Figure 3. To contrast the classic design, where all arbiter chains receive the same challenge, we implemented a simulation of XOR Arbiter PUFs with cryptographically secure pseudorandom sub-challenge generators, where all arbiter chains receive an individual pseudorandom challenge. Assuming security of the pseudorandom generator, we can guarantee statistical independence of the sub-challenges and feature vectors (for all polynomially time-bounded observers).

By the absence of any correlation, the pseudorandom input transformation is, while not being a reasonable real-world design choice, an extremal example among all input transformations. As elaborated in Section IV, the absence of correlation results in a decrease of the number of minima in the logistic regression attack.

The empirical results match this rationale: Figure 3 shows that, compared to the classic design, the required size of the training set to achieve a high success rate increases substantially. Figure 3 also shows designs in which only a subset of arbiter chains receive pseudorandom challenges, whereas the others receive the same unmodified challenge. For those designs, the required size of the training set is, as could be expected, in between the pure classic and the pure pseudorandom case.

B. Local Minima

Logistic regression uses gradient descent over a function f defined by the provided training set to conduct the modeling attack. The algorithm's ability to find a "good" minimum depends, among other parameters, on the algorithm's random initialization. Empirical results obtained by repeatedly attacking the same XOR Arbiter PUF show that the probability to guess successful initializations significantly changes with the input transformation in use (Figure 6).

Whenever an input transformation of an XOR Arbiter PUF sends the same challenge to several arbiter chains, this will be reflected in function f as symmetry. Using the classic input transformation, the attacker has at least $k!$ equally good minima[2] to choose from. This idea of f's symmetry can be generalized to the case where properties of the input transformation allow permutations of the original weights to approximate the XOR Arbiter PUF with mediocre accuracy, as we will show in Section IV-B. The approximating permutations can be observed as local minima in the logistic regression attack. On the contrary, using pseudorandom transformations, we can reduce the symmetries of f down to the minimum, hence increasing machine learning hardness and avoiding any intermediate solutions. This is confirmed by empirical results shown in Figure 4.

IV. Input Transformations: Lightweight Secure

The Lightweight Secure PUF design was introduced by Majzoobi et al. [10] in 2008 before Rührmair et al. [5] published their machine-learning attacks. The design proposes an input transformation presented in two steps.

First, for the generation of the l-th sub-challenge, the master-challenge is rotated by l bits, here denoted by $d^{(l)}$. Second, the sub-challenge $c^{(l)}$ will mostly be computed by xoring bits pairwise, such that it consists of three parts with

[2]Strictly speaking, all models will have an infinite amount of local minima, as all weights in the model can be modified by a small value or scaled by a positive scalar without affecting the model's behavior. To fix above argument we can argue that additional symmetry causes the gradient descent to remain at a local minima with higher probability.

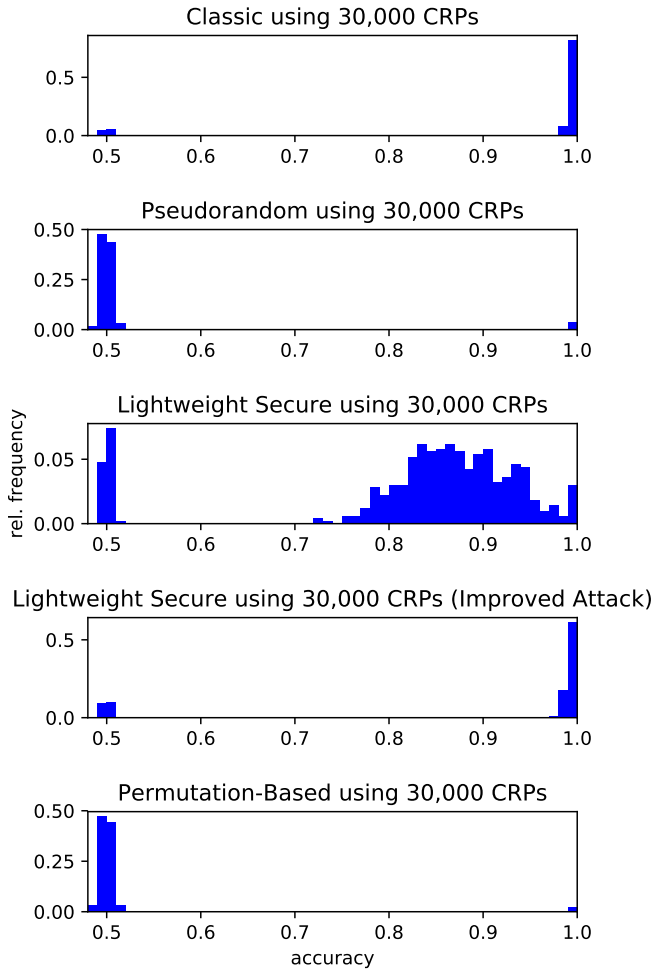

Fig. 4. Accuracy distribution for learning attempts on randomly chosen simulated XOR Arbiter PUF instances with different input transformations. All experiments were run on 64-bit, 4-XOR Arbiter PUFs. When using the Lightweight Secure input transformation, some learning attempts end with an intermediate result, while both classic XOR Arbiter PUF and pseudorandom sub-challenges do not show intermediate solutions. It can be seen that using our new correlation attack, the resulting model accuracy is increased significantly over the plain LR attack.

length $n/2$, 1, and $n/2-1$, respectively. More specifically, we have

$$\begin{aligned}\left(c_1^{(l)},\ldots,c_{n/2}^{(l)}\right) &= \left(d_1^{(l)}d_2^{(l)},\ d_3^{(l)}d_4^{(l)},\ \ldots,\ d_{n-1}^{(l)}d_n^{(l)}\right),\\ \left(c_{n/2+1}^{(l)}\right) &= \left(d_1^{(l)}\right),\\ \left(c_{n/2+2}^{(l)},\ldots,c_n^{(l)}\right) &= \left(d_2^{(l)}d_3^{(l)},\ d_4^{(l)}d_5^{(l)},\ \ldots,\ d_{n-2}^{(l)}d_{n-1}^{(l)}\right).\end{aligned} \tag{5}$$

In this section, we will refer to the sub-challenges as defined in (5) with $\tau^{(1)}(c),\ldots,\tau^{(k)}(c)$ and to the feature vectors they induce with $\sigma^{(1)}(c),\ldots,\sigma^{(k)}(c)$.

The transformation is chosen such that the Strict Avalanche Criterion is (almost) satisfied [10], i.e., a single bit flip in the master challenge will result in bit flips in about 50% of the elements of each *feature* vector for each arbiter chain. If 50% of the *feature* vector bits flip, then the PUF output also flips with probability 50%.

In this work, we will not consider weaker versions of the Lightweight Secure PUF with multiple output bits.

A. Feature Vector Correlation

In a typical machine learning attack on an XOR Arbiter PUF, we expect that a call of the LR algorithm either yields a near optimal model that has a predictive accuracy of around 99% or yields a model that performs poorly in prediction, barely exceeding an accuracy of 50%, i.e., random guessing. This can be observed in Figure 4, which shows a histogram of achieved model accuracies for individual calls of the machine learning algorithm for the XOR Arbiter PUF with random inputs. Two cases can be seen, one centered around 98% and one around 51% with no run resulting in a machine learning accuracy between 55%-95%. Interestingly, this is not the case for the Lightweight PUF. We found that the machine learning algorithm yielded models that performed clearly better than random guessing but did not achieve the desired accuracy of around 99% as can be seen in Figure 4.

In empirical results, we found that weight vectors of the intermediate solutions consisted mostly of a permutation of the weight values of the original PUF model. In fact, by permuting the individual weight vectors of the arbiter chains and rotating them for certain but distinct amounts, a close approximation of the original weight vectors could be constructed. Furthermore, we learned, that if weight vector $w^{(a)}$ was at position b it was rotated by π, then if $w^{(b)}$ was at position a it was rotated by π^{-1}.

To give a theoretical basis to our attack, we formalize this observation by examining the impact of swapping and rotating two different weight vectors. Let $w^{(1)},\ldots,w^{(k)}$ be the weight vectors of a Lightweight Secure XOR Arbiter PUF. Our observations suggest that this PUF can be approximated when weight vectors are swapped and shifted in a characteristic way. We call the weight vectors to be swapped w and v and the corresponding input transformation functions λ and μ. Note that this argument uses feature vectors, not sub-challenges. Consider the relevant part of the product in the XOR Arbiter PUF model (cf. (4) and (5)):

$$\langle w,\lambda(c)\rangle\cdot\langle v,\mu(c)\rangle=\sum_{i,j}w_i\cdot v_j\cdot\lambda(c)_i\cdot\mu(c)_j$$

In the following, we compare this to the model where the weight vectors v and w are swapped and rotated by π and π^{-1}, respectively. That is, we replace w by $\pi^{-1}(v)$ and replace v by $\pi(w)$:

$$\begin{aligned}&\left\langle\pi^{-1}(v),\lambda(c)\right\rangle\cdot\langle\pi(w),\mu(c)\rangle\\ &=\sum_{i,j}\pi(w)_i\cdot\pi^{-1}(v)_j\cdot\mu(c)_i\cdot\lambda(c)_j\\ &=\sum_{i,j}w_i\cdot v_j\cdot\pi^{-1}(\mu(c))_i\cdot\pi(\lambda(c))_j \qquad \text{(re-numbering i,j)}\end{aligned}$$

To prove that the latter is an approximation of the original model, we studied the relationship of $\pi^{-1}(\mu(c))_i\cdot\pi(\lambda(c))_j$ and $\lambda(c)_i\cdot\mu(c)_j$ and found that for most pairs i,j, we have equality with significant probability for a uniformly random master-challenge c. The higher this probability, the better is

TABLE I
OVERVIEW OF CORRELATIONS FOR A 64-BIT 6-XOR LIGHTWEIGHT SECURE ARBITER PUF. AS AN EXAMPLE, THE FEATURE VECTORS OF THE FIRST AND SECOND ARBITER CHAIN SHOW A CORRELATION OF 0.98 AS DEFINED IN (6) WITH A ROTATION BY 32 AND 33 POSITIONS, RESPECTIVELY. HENCE, THE CORRESPONDING WEIGHT VECTORS CAN BE SWAPPED IF THEY ARE ROTATED ACCORDINGLY WITHOUT SIGNIFICANT CHANGE IN THE MODEL ACCURACY.

	1	2	3	4	5	6
1	-/-	32/0.98	64/0.97	31/0.95	63/0.94	30/0.92
2	33/0.98	-/-	32/0.98	64/0.97	31/0.95	63/0.94
3	1/0.97	33/0.98	-/-	32/0.99	64/0.97	31/0.95
4	34/0.95	1/0.97	33/0.99	-/-	32/0.98	64/0.97
5	2/0.94	34/0.95	1/0.97	33/0.98	-/-	32/0.98
6	35/0.92	2/0.94	34/0.95	1/0.97	33/0.98	-/-

the approximation of the original model by the swapped and rotated version.

In Table I, we display this correlation of any pair λ, μ of the Lightweight Secure input transformation $\sigma^{(1)}, ..., \sigma^{(6)}$ as measured by

$$\frac{1}{(n+1)^2} \sum_{i=1}^{n+1} \sum_{j=1}^{n+1} \Pr_c \left[\lambda(c)_i \cdot \mu(c)_j = \pi^{-1}(\mu(c))_i \cdot \pi(\lambda(c))_j \right], \tag{6}$$

where c is chosen uniformly at random. For each pair, there is exactly one rotation π which produces a significant correlation. The rotation values for π and the resulting correlations are stated in Table I.

For example, consider the 64-bit 4-XOR Lightweight Secure PUF, where we write $(\sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)}, \sigma^{(4)})$ for the input transformation and denote $\sigma^{(1)}$ as λ and $\sigma^{(2)}$ as μ. If we rotate the first feature vector $\lambda(c)$ by 32, say $\pi(\lambda(c))$, and the second feature vector $\mu(c)$ by the inverse of 33 positions to the right, say $\pi^{-1}(\mu(c))$, then we have high correlation as defined by (6).

As we can see, the correlation of the feature vectors leads to the fact that an approximation of the original model can be constructed by swapping two weight vectors and rotating them accordingly. Using this concept iteratively, any permutation of the weight vectors can be achieved.

Our empirical results in Figure 4 suggest that those partial solutions also generate local minima to which the regression algorithm converges. The combination of the information on the local minimum along with the correlation as outlined above can be used to stage an attack on the input transformation by Majzoobi et al. This must be considered a key weakness of the Lightweight Secure transformation, as our empirical attack results show.

The cause for this symmetry lies in the definition of the input transformation and in the fact that results are xored. In Table I, the length of rotations and the correlation is given for 64-Stage PUFs for up to 6 stages. There is a clear pattern and essentially every pair of PUFs can be exchanged by a rotated version, although the correlation decreases the further the PUF positions are apart from each other.

B. Improved Attack

As seen in the previous section, the LR machine learning attack on the Lightweight Secure PUF often leads to local minima that model the PUF behavior only with a limited accuracy. In some cases, a higher accuracy may be needed to impersonate a PUF device. In this section, we show how a local minimum can be used to find a high-accuracy model.

If the logistic regression attack has found a model with an intermediate accuracy in the range of 65%-98%, we assume that the initialization values for the attack lead to a swapped and rotated version of the high-accuracy version of the weights. Instead of restarting the machine learning algorithm with new initializations until we find a high-accuracy solution and hence the correct ordering, the *correlation attack* tries to generate the correct ordering of the weights. To that end, we first generate the rotated weights for each possible permutation of the weight vectors in a brute-force manner and check their accuracy on a validation set. As a second step, the $2k$ most accurate rotated weights are used to restart the logistic regression attack and refine the weights.

Although the first step has run time $O(k! \cdot nk \cdot V)$ with validation set size V, this procedure can outperform the simple restarting of the LR attack (Table II) for practical values of k, as $10! = 3\,628\,800$. Furthermore, the restarted logistic regression algorithm can use a much lower bound on the maximum number of iterations, discarding low-accuracy solutions rapidly. To achieve fast run times, we used a small validation set for the $k!$ accuracy computations. We empirically found that rotations with high initial accuracy have a higher chance to yield a high-accuracy solution, hence the ordering by initial accuracy helps speed up the attack.

More specifically, we examined the ranking of the permutation that resulted in the highest accuracy solution for 1000 instances of 64-bit 6 XOR Lightweight Secure PUFs. In most cases, the best permutation was within the first 10 candidates (Figure 5).

Three improvements to this strategy are available. First, it is possible to use a greedy algorithm that chooses the permutation by iteratively evaluating swaps of a previously selected permutations, starting with the identity permutation. Although the greedy approach is not guaranteed to work, this still yields usable results while removing the $O(k!)$ runtime and should be considered for very large k. Second, in Table I it can be seen that the approximation accuracy decreases as the "distance" of the swapped arbiter chain increases. Based on the swap accuracy, it is possible to pre-compute a list of interesting permutations in order to skip permutations that are not promising. Third, the attack can be conducted iteratively. We observed attacks that, while improving the initial accuracy significantly, did not reach the 98% success threshold. In those cases, it is possible to restart the search for an apt permutation.

In Table II we have compared the expected time until first result with accuracy better than 98%, computed as the quotient of mean attack time and success probability, for attacking the classic XOR Arbiter PUF and the classic and improved attack on the Lightweight Secure Arbiter PUF. It can be seen that the Lightweight Secure PUF can be learned with much higher accuracy in less time than previously believed, with the security in some instances reduced to what the classic XOR Arbiter PUF provides. In contrast, the XOR Arbiter PUF with the permutation-based input transformation defined in

TABLE II

EXPECTED TIME UNTIL THE FIRST SUCCESS FOR LR AND CORRELATION ATTACKS ON CLASSIC XOR ARBITER PUF, LIGHTWEIGHT SECURE XOR ARBITER PUF, AND PERMUTATION-BASED XOR ARBITER PUF. A PREDICTION ACCURACY OF AT LEAST 98% IS CONSIDERED SUCCESS. RUN TIMES REFER TO SINGLE-THREADED RUNS ON INTEL® XEON® GOLD 6130 CPUS. ALL ENTRIES ARE BASED ON AT LEAST 1000 AND 600 SAMPLES, FOR $n = 64$ AND $n = 128$, RESPECTIVELY. SOURCE CODE AVAILABLE AT HTTPS://GITHUB.COM/NILS-WISIOL/PYPUF.

n	k	# CRPs	LR on Classic	LR on Lightweight Secure	Correlation Attack on Lightweight Secure	LR on Permutation-Based
64	4	12,000	0m 33s	10m 11s	0m 58s	24m 50s
64	4	30,000	0m 31s	3m 57s	0m 44s	4m 45s
64	5	300,000	7m 03s	3h 03m	11m 07s	13h 59m
64	6	1,000,000	42m 30s	8 days	1h 42m	longer than 96h 00m
64	7	2,000,000	75h 07m	longer than 20 days	8 days	longer than 16 days
128	4	1,000,000	20m 31s	2h 53m	51m 23s	58m 38s
128	5	2,000,000	1h 35m	35h 20m	3h 17m	longer than 16 days

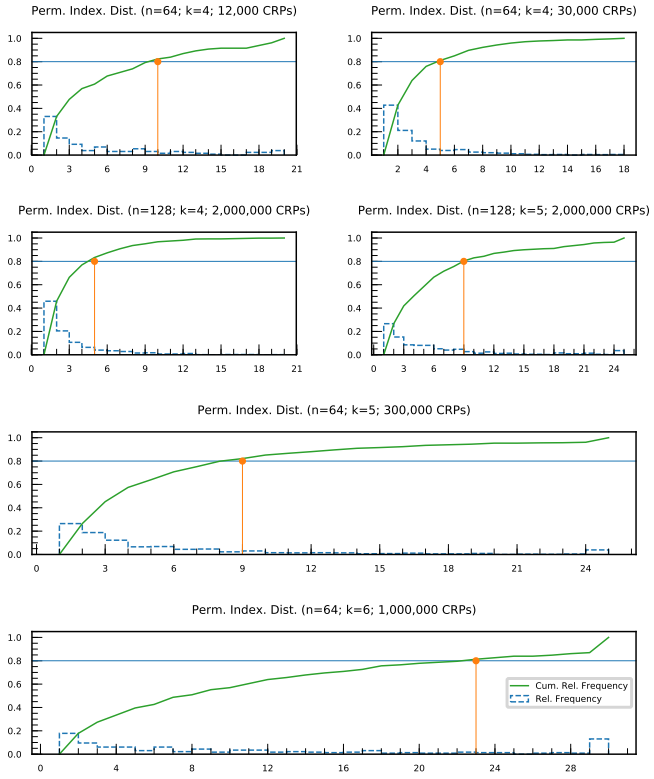

Fig. 5. When ordering possible permutations for the correlation attack by validation set accuracy, the target permutation often appears within the first few permutations. To obtain 80% probability of finding the permutation among the first tested, we chose to check the first four permutations for $k = 4$, the first ten permutations for $k = 5$ and the first 18 permutations for $k = 6$. This justifies restarting the logistic regression learner for only a couple possible permutations instead of all $k!$ many.

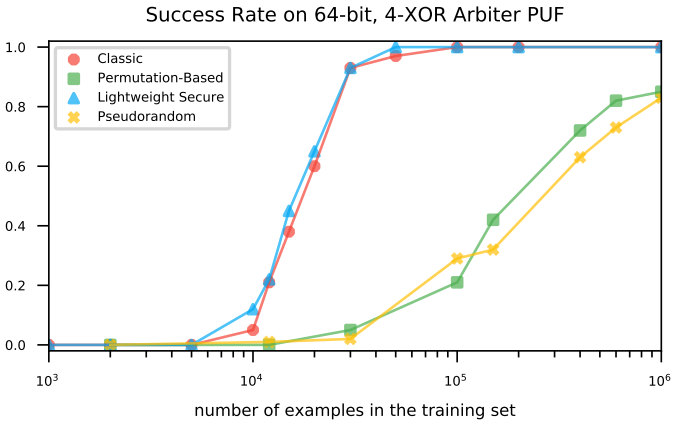

Fig. 6. Success rate of logistic regression attacks on simulated XOR Arbiter PUFs with 64-bit arbiter chains and four arbiter chains each. Four different input transformations are shown: classic, Lightweight Secure by Majzoobi et al. [10], the permutation-based input transformation proposed in this work, and a pseudorandom input transformation used as comparison. All data points are based off at least 80 samples. For a success threshold of 70%, Lightweight Secure and classic are equally hard to attack, whereas permutation-based and pseudorandom require significantly more CRPs.

Section V is considerably harder to attack and does not posses the attack surface we used in the correlation-based attack (cf. Figure 4).

V. SOLUTION

The previous sections show that input transformations have impact on the machine learning resistance. When using the same challenges for all arbiter chains as done in the classic XOR Arbiter PUF, there are multiple equivalent solutions as the order of the weight vectors $(w^{(1)}, ..., w^{(k)})$ does not matter. Using pseudorandom sub-challenges ensures that only one order is valid and hence reduces the number of global minima in the gradient descent of the LR attack. Hence, from a security perspective using pseudorandom sub-challenges is a good approach. However, it should be noted that this comes with quite some overhead in terms of area as well as power/energy since at least $k \cdot n$ registers are needed to store the pseudo random bits together with some logic to generate them. For example, Yu *et al.* [9] use a 256 bit LFSR to feed the four arbiter chains in the used 64-Stages 4-XOR Arbiter PUF. The area overhead of their 4-XOR Arbiter PUF is given by Yu *et al.* as 1024 Gate Equivalents (GE). The size of the LFSR is not provided in [9], but assuming 4.5 GE for a flip-flop, the size of a 256 stage LFSR is comparable to that of the PUF circuitry[3]. Implementing a cryptographically secure pseudorandom generator will consume even more resources.

In the Lockdown Protocol [9] the LSFR is an essential part of the authentication protocol and hence needed anyways. But for other designs, especially if larger PUF instances are used with 128 stages, a more efficient input transformation

[3]Although the Gate Equivalents for a PUF circuit can be a bit misleading as PUFs need special isolated routing compared to conventional digital circuits such as LFSRs.

is advised as the overhead is not negligible. However, our analysis of the Lightweight Secure PUF shows that this input transformation suffers a significant weakness and must be considered insecure. The fact that feature vectors correlate in a certain way simplifies the machine learning attack to the point where no relevant advantage over the classic XOR Arbiter PUF is achieved.

A. Permutation-Based Input Transformations

We propose an input transformation that is actually even more lightweight than the Lightweight Secure PUF solution but does not show any indication of local minima. The idea is to use k different, fix-point-free permutations as sub-challenge generators[4]. As this input transformation can be implemented in wiring, no additional gate is used in the PUF design. A permutation of the challenges does not result in a permutation of the feature vectors due to the nature of the ATT. To be more precise, the multiplications in (3) ensure that if the challenge vectors are permuted, different bits are being multiplied. Therefore pairs of feature vectors do not show significant correlation according to (6) even if they are permuted. We call this family of input transformations the *permutation-based input transformations*.

We empirically confirmed that this approach does not show any of the local minima we observed for the Lightweight Secure PUF, as can be see in Figure 4. The machine learning resistance was instead comparable to the results of pseudorandom inputs (Figure 6), which represent and upper bound on input transformation quality as argued in Section III. Without observing local minima or correlations, the attack described in Section IV-B cannot be applied.

Additionally, this input transformation comes at nearly zero resource overhead. Compared to using a pseudorandom input transformation, the permutation-based transformation is more efficient in terms of area and power and is also more efficient than the input transformation proposed for the Lightweight Secure PUF.

VI. Conclusion

In this paper we revisited the topic of input transformations for Arbiter PUFs, which were introduced to make XOR Arbiter PUFs more resilient against machine learning attacks. We showed that the Lightweight Secure PUF, which was believed to be hard to attack, can in fact be learned with much higher success rate than previously believed. With the same training set size, we were able to achieve attack results comparable to attacking the classic XOR Arbiter PUF. This refutes the assumption that the Lightweight Secure PUF provides significantly better security than the classic XOR Arbiter PUF.

The main reason for this is that the input transformation of the Lightweight Secure PUF produces local minima which can be learned via machine learning algorithms. Our research shows that the input transformation can play an important role when determining the machine learning resistance of a PUF construct. In particular, one needs to ensure that the input transformation does not result in local minima that can be exploited using our two-stage machine learning attack. We therefore advice researchers to carefully look at the distribution of achieved model accuracies as we have done in Figure 4 and not only measure the percentage of "successful" modeling runs. Based on these findings we presented an alternative input transformation using fix-point-free permutations. Our results show that PUFs using this input transformation are nearly as hard to learn as pseudorandom inputs, which we argue is the most resilient input transformation in regards to machine learning attacks. The proposed design has a very low hardware-overhead as it simply consists of a fixed routing of challenge bits to the individual arbiter chains and does not feature any obvious feature vector correlations that could be used to launch an improved attack.

Finally, it should be noted that while our results focus on XOR Arbiter PUF, the results can be generalized to other constructs such as the multiplexer PUF [34] or PUF constructs not based on Arbiter PUFs such as an XOR Bistable Ring PUF [35] or XOR Voltage PUFs [18].

[4] Additionally, we chose the permutations such that no pair always shows the same value on the same output coordinate.

References

[1] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security (CCS)*. ACM, 2002, pp. 148–160.

[2] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual design automation conference (DAC)*. ACM, 2007, pp. 9–14.

[3] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *CHES*, vol. 4727. Springer, 2007, pp. 63–80.

[4] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[5] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security (CCS)*. ACM, 2010, pp. 237–249.

[6] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on pufs with application to noise bifurcation," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2015, pp. 17–31.

[7] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. P. Burleson, and S. Devadas, "Puf modeling attacks on simulated and silicon data," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[8] G. T. Becker, "The gap between promise and reality: On the insecurity of xor arbiter pufs," in *Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2015, pp. 535–555.

[9] 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 Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, July 2016.

[10] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure pufs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2008)*. IEEE, 2008, pp. 670–673.

[11] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 37–49, March 2014.

[12] M. D. Yu, I. Verbauwhede, S. Devadas, and D. M'Raïhi, "A noise bifurcation architecture for linear additive physical functions," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2014, pp. 124–129.

[13] P. H. Nguyen, D. P. Sahoo, R. S. Chakraborty, and D. Mukhopadhyay, "Security analysis of arbiter puf and its lightweight compositions under predictability test," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 2, p. 20, 2017.
[14] D. Lim, "Extracting secret keys from integrated circuits," Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, 2004.
[15] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring puf: A new architecture for strong physical unclonable functions," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 134–141.
[16] D. Schuster and R. Hesselbarth, "Evaluation of bistable ring pufs using single layer neural networks," in *International Conference on Trust and Trustworthy Computing*. Springer, 2014, pp. 101–109.
[17] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant puf design based on non-linear voltage transfer characteristics," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 653–658.
[18] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong puf: Possible or a pipe dream?" in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 19–24.
[19] F. Ganji, S. Tajik, F. Fäßler, and J.-P. Seifert, "Strong machine learning attack against pufs with no mathematical model," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 391–411.
[20] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 476–492.
[21] G. T. Becker, R. Kumar *et al.*, "Active and passive side-channel attacks on delay based puf designs." *IACR Cryptology ePrint Archive*, vol. 2014, p. 287, 2014.
[22] J. Delvaux and I. Verbauwhede, "Side channel modeling attacks on 65nm arbiter pufs exploiting cmos device noise," in *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 137–142.
[23] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich, "Physical characterization of arbiter pufs," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 493–509.
[24] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and ro sum pufs via environmental changes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1701–1713, 2014.
[25] Y. Gao, M. van Dijk, L. Xu, S. Nepal, and D. C. Ranasinghe, "Treverse: Trial-and-error lightweight secure reverse authentication with simulatable pufs," *arXiv preprint arXiv:1807.11046*, 2018.
[26] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, and M. van Dijk, "Mxpuf: Secure puf design against state-of-the-art modeling attacks," Cryptology ePrint Archive, Report 2017/572, 2017, https://eprint.iacr.org/2017/572.
[27] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe, "Puf-fsm: A controlled strong puf," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 1104–1108, 2018.
[28] S. T. C. Konigsmark, D. Chen, and M. D. Wong, "Polypuf: Physically secure self-divergence," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 7, pp. 1053–1066, 2016.
[29] J. Delvaux, "Machine learning attacks on polypufs, ob-pufs, rpufs, lhs-pufs, and puf–fsms," Cryptology ePrint Archive, Report 2017/1134, 2017, https://eprint.iacr.org/2017/1134.
[30] M.-D. M. Yu, D. M'Raihi, R. Sowell, and S. Devadas, "Lightweight and secure puf key storage using limits of machine learning," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 358–373.
[31] Z. Paral and S. Devadas, "Reliable and efficient puf-based key generation using pattern matching," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 128–133.
[32] G. T. Becker, A. Wild, and T. Guneysu, "Security analysis of index-based syndrome coding for puf-based key generation," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 20–25.
[33] J. Sölter, "Cryptanalysis of electrical pufs via machine learning algorithms," MSc thesis, Technische Universität München, 2009.
[34] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen, "A multiplexer-based arbiter puf composition with enhanced reliability and security," *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 403–417, 2018.
[35] X. Xu, U. Rührmair, D. E. Holcomb, and W. Burleson, "Security evaluation and enhancement of bistable ring pufs," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2015, pp. 3–16.