IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2020, No. 3, pp. 97–120. DOI:10.13154/tches.v2020.i3.97-120

Splitting the Interpose PUF: A Novel Modeling Attack Strategy

Nils Wisiol[1,2], Christopher Mühl[2], Niklas Pirnay[1], Phuong Ha Nguyen[3], Marian Margraf[2], Jean-Pierre Seifert[1], Marten van Dijk[3,4] and Ulrich Rührmair[3,5]

[1] Technische Universität Berlin, Germany
{nils.wisiol,jean-pierre.seifert}@tu-berlin.de,niklas.pirnay@campus.tu-berlin.de
[2] Freie Universität Berlin, Germany {christopher.muehl,marian.margraf}@fu-berlin.de
[3] University of Connecticut, USA phuong_ha.nguyen@uconn.edu
[4] CWI Amsterdam, Netherlands marten.van_dijk@uconn.edu
[5] LMU München, Germany ruehrmair@ilo.de

Abstract. We demonstrate that the Interpose PUF proposed at CHES 2019, an Arbiter PUF-based design for so-called Strong Physical Unclonable Functions (PUFs), can be modeled by novel machine learning strategies up to very substantial sizes and complexities. Our attacks require in the most difficult cases considerable, but realistic, numbers of CRPs, while consuming only moderate computation times, ranging from few seconds to few days. The attacks build on a new divide-and-conquer approach that allows us to model the two building blocks of the Interpose PUF separately. For non-reliability based Machine Learning (ML) attacks, this eventually leads to attack times on $(k_{\text{up}}, k_{\text{down}})$-Interpose PUFs that are comparable to the ones against $\max\{k_{\text{up}}, k_{\text{down}}\}$-XOR Arbiter PUFs, refuting the original claim that Interpose PUFs could provide security similar to $(k_{\text{down}} + \frac{k_{\text{up}}}{2})$-XOR Arbiter PUFs (CHES 2019). On the technical side, our novel divide-and-conquer technique might also be useful in analyzing other designs, where XOR Arbiter PUF challenge bits are unknown to the attacker.

Keywords: Physical Unclonable Function · Strong PUFs · Machine Learning · Modeling Attacks · Interpose PUF (iPUF)

1 Introduction

1.1 Motivation and Overview

The Interpose PUF (iPUF) is one of the most recent Strong PUF design proposals [NSJ+19]. It promises two distinctive and noteworthy features: Firstly, it builds on the Arbiter PUF design, and therefore inherits the practicality and CMOS-compatibility of the latter. Secondly, it contains some novel design elements, which successfully thwarted those state-of-the-art modeling attacks that were available at the time of its publication. This turned the iPUF into one of the most promising Strong PUF design proposals to date.

In this paper, however, we now present a *new, tailor-made* modeling attack that can attack Interpose PUFs with high accuracy for security parameter settings of very substantial sizes and complexities. In-depth analysis of our attack method shows that $(k_{\text{down}}, k_{\text{up}})$-iPUFs can be tackled in a time that is similar to attacking a $\max\{k_{\text{down}}, k_{\text{up}}\}$-XOR Arbiter PUF with classical, non-reliability based ML attacks. This refutes any implicit claims [NSJ+19] that the iPUF security against non-reliability based attacks would be roughly equivalent to that of a $(k_{\text{down}} + \frac{k_{\text{up}}}{2})$-XOR Arbiter PUF.

Received: 2020-01-15 Accepted: 2019-03-15 Published: 2020-06-19

The iPUF paper [NSJ+19] gives an overview (which we will not repeat here) of currently known *"classical ML attacks"* and *"reliability based ML attacks"*. A classical ML attack by definition only uses plain challenge-response-pairs (CRPs) without reliability information, and thus is sometimes also simply called *"CRP-based attack"*, while a reliability-based attack uses challenge-*reliability*-pairs. For single-bit responses, the reliability of a response reflects the probability that the outputted response bit is equal to 0 (or 1, respectively), i.e., the reliability corresponds to the stability or error level of the response bit. Here the probability is taken over measurement noise (which depends on environmental parameters such as temperature, voltage, and age of the PUF). The iPUF paper [NSJ+19] aims at mathematical arguments for formally showing the security of the iPUF against the main existing representatives of the two abovementioned classes of attacks. With respect to best known "classical" ML attacks, it analyzed/simulated Logistic Regression (LR) [Söl09, RSS+10, TB15], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) based on CRPs, and Deep Neural Networks (DNN) [SBC19]. As the best known reliability-based attack, it analyzed/simulated Becker's attack [Bec15] which uses CMA-ES based on challenge-reliability pairs. These various existing algorithms are so shown to be unsuccessful in attacking the iPUF [NSJ+19]. This marks a noteworthy step forward in Strong PUF design, as it shows the security of a Strong PUF candidate against many existing algorithms and attack methods. However, the iPUF paper did not (and also for fundamental reasons could not [1]) formally and unconditionally show that there will not be *other*, future ML-techniques that could attack the iPUF with improved efficiency. This is where our new strategy comes into play. While it does not fully break the iPUF for all conceivable sizes and complexities yet, it represents a technically novel and substantially more efficient attack form on the iPUF design than the ones analyzed in the original iPUF work.

Our attack has a number of implications for the original iPUF paper [NSJ+19]: First of all, it shows that Definitions 1 and 2 and Theorem 4 of the original iPUF work [NSJ+19], which discuss the *"equivalence"* of certain iPUF and XOR Arbiter PUF sizes, have some limitations: They merely constitute a measure with respect to how many Arbiter PUFs in a given Arbiter PUF based design effectively contribute to the response bit. Only in this (limited) sense, the $(k_{\text{up}}, k_{\text{down}})$-iPUF is *"equivalent"* to an $(k_{\text{down}} + \frac{k_{\text{up}}}{2})$-XOR Arbiter PUF. But this does *not* imply automatically that their security levels against machine learning attacks are exactly equivalent, as our paper and attacks demonstrate. Among other things, we show that for all $1 \leq k' < k$, the effective security levels of a (k, k)-, (k, k')-, and (k', k)-iPUF are essentially the same in the face of our attacks. This has some practical consequences: Since $(1, k)$-iPUFs and $(k, 1)$-iPUFs are most area efficient, they constitute the iPUF design variants of the most future interest.

In a similar vein, Section 6.5 of the original iPUF paper [NSJ+19] analyses LR attacks on the iPUF, showing a general statement on the impossibility of such attacks. While this statement represents a noteworthy step towards proving PUF-security against whole classes of ML-algorithms, our attack illustrates that it only holds for "holistic", standard LR attacks — not for methods that use LR as a subcomponent in new, more complex

[1]We just remark in passing, and without proof, that some thinking suggests the following. Let us consider a Strong PUF with a model $R_i = F_{\mathsf{PUF}}(\mathsf{Disorder}, C_i)$, where F_{PUF} is a polynomial-time computable function, and $\mathsf{Disorder}$ is a bitstring of polynomial length (polynomial in the length of the PUF-challenge in both cases). Unconditionally proving that such a Strong PUF is not learnable in polynomial time would seem to imply a separation between NP and P. The reason is that given a polynomial set of PUF-CRPs, an NP-algorithm can guess the string $\mathsf{Disorder}$ as NP-witness, and can then verify the correctness of this witness by polynomial-time computations, exploiting the equation $R_i = F_{\mathsf{PUF}}(\mathsf{Disorder}, C_i)$ (and its knowledge of the given polynomial number of PUF-CRPs). This suggests that (a decision version of) the problem of learning and predicting a Strong PUF of the above type when given a polynomial number of randomly chosen PUF-CRPs is in NP. If true, this means that apart from certain exceptions [RJB+11], current Strong PUF candidates realistically cannot be proven secure in an unconditional sense, similar to block ciphers or hash functions. Their security needs to be verified or attested every time anew against the current state of the art in attacks (which is a moving target). The same holds for the iPUF, of course [NSJ+19].

modeling algorithms, as we do in this paper. Another consequence of our attack relates to possible reliability-based ML attacks on the iPUF. The currently best-known (and essentially only) reliability-based attack is that of Becker [Bec15]; since it was very effective on XOR Arbiter PUFs, quite naturally little research has been conducted in order to advance it afterwards. Even though this is purely speculative at this point, in light of our new result, an efficient tailor-made reliability based attack may not be out of reach as well. This suggests future research in this direction.

Overall, our results prompt that the design of secure and area-optimal Strong PUFs will remain an active area of research in the years to come. Future designs might utilize design elements from the iPUF [2], and combine it with yet novel ideas in order to yet further improve security and area consumption. We comment that while this back-and-forth game may seem tiring, also other cryptographic primitives have undergone a similar iterative process, before secure solutions were eventually found and accepted. We believe that the same will take place in the area of silicon Strong PUF over the next years.

1.2 Our Contributions

In greater detail, our novel research contributions in this paper are as follows.

- We develop and implement a new tailor-made divide-and-conquer classical ML strategy for the iPUF, and empirically test its concrete and asymptotic performance on the iPUF on very large scales.

- Using this method, we are able to attack $(k_{\text{up}}, k_{\text{down}})$-iPUF structures for up to $k_{\text{up}} = 8$ and $k_{\text{down}} = 8$ and challenge lengths 64 bits with prediction accuracies above 95%. In the hardest-to-learn cases that we studied, i.e., for the $(8, 8)$ and $(1, 9)$-64-bit-iPUF, our attacks require up to 300 million CRPs and 750 million CRPs, respectively[3], and computation times between one and eight weeks on a high-end 8-core machine. For simpler cases, i.e., smaller k_{up} and k_{down}, they take seconds to hours of computation times on the same hardware and far less CRPs. The previously best known attacks on the iPUF had reached up to $(4, 4)$-iPUFs only [SBC19].

- Concerning classical modeling attacks, Nguyen et al. [NSJ+19] claimed that $(k_{\text{up}}, k_{\text{down}})$-iPUFs are comparably secure to an $(k_{\text{down}} + \frac{k_{\text{up}}}{2})$-XOR Arbiter PUF. As explained above, this statement does not generally hold, but only for a specific subset of classical ML attacks. Using our novel attack methods, we now empirically show that the Interpose PUF is, with respect to our attacks, comparably secure to a $\max\{k_{\text{up}}, k_{\text{down}}\}$-XOR Arbiter PUF. [4]

- We make the new technical observation that the logistic regression based PUF modeling algorithm [RSS+10] can fully or partially recover XOR Arbiter PUFs even in the presence of *feature*-noise in the training set, i.e. when some or many challenge bits to an XOR Arbiter PUF are unknown or noisy. This may prove useful in design and attack of future XOR Arbiter PUF-based designs. Previous work only showed that the LR algorithm is robust with respect to label-noise, i.e., when the response bits are noisy [RSS+10].

[2] For example, the so-called interpose trick of the iPUF in fact successfully defeats the existing reliability-based ML attacks by Becker [Bec15], and may hence be useful in future designs.

[3] We comment that this amount of CRPs could be collected from a silicon Strong PUF implementation with a 1MHz CRP-frequency in merely five and 12.5 minutes, respectively.

[4] Based on the study of the reliability, security and hardware footprint of the iPUF design, the authors in [NSJ+19] suggested to work with $(1, k_{\text{down}})$-iPUFs as the best choice of design parameters. This paper provides further, empirical results that support this suggestion.

- Our work is complemented by studying a number of suggestive and straightforward iPUF design variants. For all of these variants, we demonstrate the existence of successful ML-attacks based on straightforward deep learning methods. This suggests that the iPUF security cannot be quickly restored by all too simplistic means, confirming the impact and relevance of our results.

1.3 Related Work

Since the introduction of the first silicon Strong PUF in 2002 [GCvD02], their secure implementation has been intensely investigated, with attacks and countermeasures/new designs quickly superseding each other. It hence makes sense to comparatively summarize their history here, also with special hindsight to the employed machine learning algorithms. To start with, Arbiter PUFs themselves were attacked successfully for the first time already in 2005 [LLG+05] by support vector machines. New variants, such as the XOR Arbiter PUF [SD07], Feed-Forward Arbiter PUF [GLC+04], and Lightweight Secure PUF (LS PUF) [MKP08] remained secure for several years. But they were eventually attacked by so-called evolution strategies and a tailor-made variant of logistic regression with Rprop gradient descent at CCS in 2010 [RSS+10] and at IEEE TIFS in 2013 [RSS+13]; see also [RS14] for an overview. Recently, a specialized correlation attack has further improved the modeling performance on LS PUFs and their special input mapping [WBM+20], and so has the use of strong computing resources [TB15].

Other promising designs, such as the Bistable Ring PUF [CCL+11] and its XORed versions [XRHB15], the Current Mirror PUF [KB14], or the Voltage Transfer Characteristic PUF [VK15], were also tackled successfully, using support vector machines [XRHB15], genetic algorithms [GYG+16], simulated annealing and ant colonies [YHL16], or even attacks without a mathematical PUF-model [GTFS16]. These abovementioned empirical methods were complemented by formal proofs, for example in the PAC-framework [GTS16, GTS15a]. Also side channel information has been used in connection with ML, boosting attack complexity from exponential to polynomial in various cases: For example, certain power and timing side channels for the first time allowed attacking XOR Arbiter PUFs with polynomial complexity [RXS+14]. The leanest and simplest polynomial attack method on XOR Arbiter PUFs today seems the exploitation of the output stability of PUF-CRPs as an additional source of information by the adversary [Bec15]. Finally, also very effective protocol-related attacks on Strong PUFs have been considered [Rv12, RvD13a, RvD13b].

This existing research landscape clearly illustrates the difficult quest for secure Strong PUF designs. It leaves the realization of highly (area-)efficient and secure silicon Strong PUFs as a major open research problem in the field. Two of the recent, most promising designs were the MUX-PUF [SMCN18] and the iPUF [NSJ+19]. Until now, both only had been attacked up to relatively moderate sizes by deep learning methods [SBC19], such as up to $(4, 4)$-iPUFs. Our new attacks strongly improve this outreach up to $(8, 8)$ and $(1, 9)$-iPUFs.

1.4 Organization of this Paper

In Sec. 2, we give required background and outline our employed methodology. Sec. 3 introduces and discusses our attack strategy in detail. We present our empirical results in Sec. 4. In Sec. 5, we study some extensions to the Interpose PUF design, after which we conclude the paper in Sec. 6.

2 Background and Methodology

2.1 Arbiter PUF, Linear Additive Delay Model, and Reliability

The behavior of an n-bit Arbiter PUF [SD07] (see Fig. 1a) can be modeled using the *linear additive delay model* [GLC+04]. In this approach, we use $\{-1, 1\}$ to model the bit values[5] of challenge and response and n real values $w \in \mathbb{R}^n$ to model the physics of the production imperfections. By an induction argument, an Arbiter PUF with n stages can be modeled as a function $f : \{-1, 1\}^n \to \{-1, 1\}$ where

$$f(c) = \operatorname{sgn} \sum_{i=1}^{n} w_i \cdot c_i c_{i+1} \cdots c_n.$$

The high accuracy of this model is confirmed unanimously in the Arbiter PUF literature and has become an established standard in the PUF-literature, in particular in PUF modeling attacks on any Arbiter PUF variants [GLC+04, RSS+13].

Defining the bijection $\{-1, 1\}^n \to \{-1, 1\}^n$, $c \mapsto x$ by setting $x_i = c_i c_{i+1} \cdots c_n$, we can write the linear additive delay model as

$$f(c) = \operatorname{sgn} \sum_{i=1}^{n} w_i \cdot x_i = \operatorname{sgn}\langle w, x\rangle,$$

that is, the Arbiter PUF can be expressed as a hyperplane in an n-dimensional space. In this work, we will refer to x as the *transformed challenge* and to c as the *(physical) challenge*. Note that the transformed challenge x does not depend on manufacturing imperfections, and can thus be computed by an attacker from the physical challenge c.

All known Strong PUFs' implementations suffer from unreliability issues. To capture the extent of reliability, we define the *reliability* $r_{\mathcal{P}}$ of a PUF instance $\mathcal{P}$ to be the average probability over all challenges that we get a noise-free response (i.e, not disturbed by noise). Formally, we have

$$r_{\mathcal{P}} = \mathop{\mathrm{E}}_{c \in \{-1,1\}^n} \left[\Pr_{\text{noise}} \left(\mathcal{P}(c) = \mathcal{P}_{\text{noise-free}}(c) \right) \right].$$

Although $\mathcal{P}_{\text{noise-free}}$ cannot easily be determined in practice, it is easy to handle in simulations. An alternative notion for reliability is *stability*, defined as the average probability that two evaluations of the same challenge will show the same response.

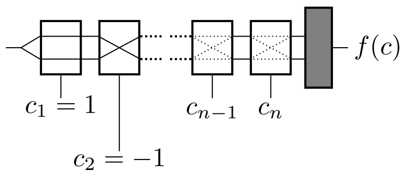
In this work, a reliability of 1.0 is referring to simulation with zero noise; other values of reliability are rounded to the nearest $1/10$.

2.2 XOR Arbiter PUF and Interpose PUF

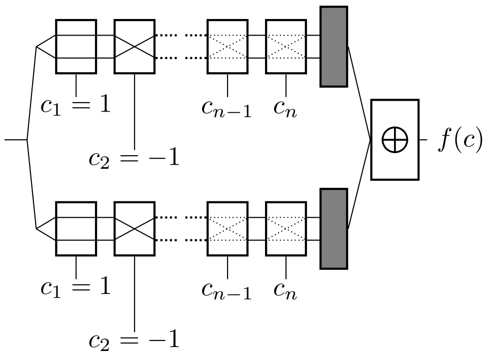
To overcome the vulnerabilities of plain Arbiter PUFs (see Sec. 1.3), it was proposed [SD07] to evaluate several Arbiter PUFs in parallel, and to only output the XOR of their individual response bits. The resulting *XOR Arbiter PUF* has two security parameters, namely the number of challenge bits, in our work usually denoted n, and the number of employed arbiter chains, commonly called k, which enter the final XOR, generating the overall response bit. The case of $k = 2$, i.e., a 2-XOR Arbiter PUF, is shown in Fig. 1b.

Nguyen et al. [NSJ+19] aimed at mitigating the known weaknesses of XOR Arbiter PUFs (see Sec. 1.3) in a new design attempt. Their Interpose PUF or iPUF architecture

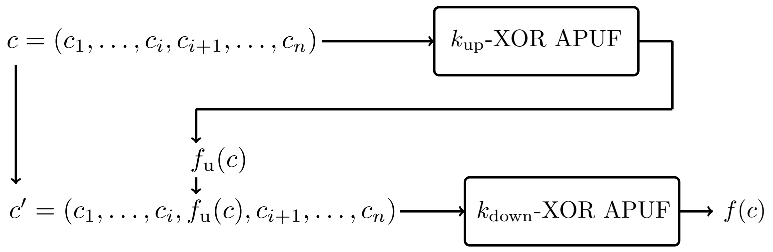
[5] In $\{-1, 1\}$ notation, the XOR operation is represented by multiplication. Note that in order to obtain an isomorphism φ to the group $\mathbb{F}_2 = (\{0, 1\}, +)$, we must assign $\varphi(0) = 1$ and $\varphi(1) = -1$, i.e. representing True by -1 and False by 1. We can then write $\varphi(a) = (-1)^a$, which corresponds to notation of the linear additive delay model used in some related works.



(a) Schematic of an Arbiter PUF with n-bit challenge c (represented with bit values from $\{-1, 1\}$) and final response $f(c)$.



(b) Schematic representation of a 2-XOR Arbiter PUF with n-bit challenge c and final response $f(c)$.



$c = (c_1, \dots, c_i, c_{i+1}, \dots, c_n)$
k_{up}-XOR APUF
$f_{\text{u}}(c)$
$c' = (c_1, \dots, c_i, f_{\text{u}}(c), c_{i+1}, \dots, c_n)$
k_{down}-XOR APUF
$f(c)$

(c) Schematic representation of a n-bit Interpose PUF (iPUF). Both upper layer and lower layer consist of k_{up} and k_{down} arbiter chains, respectively. The response of the upper layer $f_{\text{u}}(c)$ is interposed in the middle input bit of lower layer the lower layer; the final response $f(c)$ is subsequently determined by the lower layer.

Figure 1: Schematics of Arbiter PUF, XOR Arbiter PUF, and Interpose PUF.

consists of a novel combination of two XOR Arbiter PUFs. It is defined by a challenge length n, the number k_{up} of XORs in the first XOR Arbiter PUF, and the number k_{down} of XORs in the second XOR Arbiter PUF. The first XOR Arbiter PUF, called *upper layer*, has challenge length n and consists of k_{up} arbiter chains. When challenged with an input, its 1-bit response is interposed in the middle bit position of the second XOR Arbiter PUF (in the *lower* or *downward layer*), resulting in a challenge length of $n + 1$ bit for the lower layer and in k_{down} independent arbiter chains there. A schematic representation of the iPUF is depicted in Fig. 1c.

By virtue of the interposed bit on the lower layer, LR attacks on the iPUF cannot directly be conducted, as information is missing from the training set. Applying LR naively with omission of the interpose bit, constant or a random interpose bit (called *"linearization attack"* by Nguyen et al. [NSJ+19]) will result in a maximum accuracy of 75%. Furthermore, as the response bit of the upper layer will influence the bottom layer response for approximately half of all challenges, the existing reliability-based attack on XOR Arbiter PUFs [Bec15] is also mitigated by the iPUF design [NSJ+19]. Overcoming the existing reliability-based, polynomial attacks on XOR Arbiter PUFs [Bec15] by virtue of the interpose bit is an achievement of the iPUF design that remains valid, also in the face of our novel attacks.

2.3 Numeric Generation of (Noisy) CRP-Sets

All results presented in this work are based on numerically simulated CRPs. Our simulations use the linear additive delay model (see Sec. 2.1). For a single arbiter chain instance, $w \in \mathbb{R}^n$ (as defined in Eq. 2.1) is drawn independently from a Gaussian distribution. All arbiter chains are assumed to be unbiased. More complex XOR Arbiter PUFs and Interpose PUFs are constructed out of single arbiter chains in the straightforward manner. Any real-world implementation of Arbiter PUFs, including implementations of Interpose PUFs, will suffer from noise. This noise stems from temperature and voltage variations, circuit aging, and other effects and leads to unreliable PUF responses, i.e. sometimes, the response to the same challenge will vary. Effect, distribution, and modelling of noise in Arbiter PUF have been studied [DV13, WM19]. Our generation of "noisy" CRPs follows the approach accepted in the literature by evaluating the linear additive delay model of each Arbiter PUF and then adding an amount of noise, chosen for each CRP-evaluation independently. For an Arbiter PUF as defined above, on a given challenge c the response bit is thus computed as $\operatorname{sgn}\left(X + \sum_{i=1}^{n} w_i x_i\right)$, where x is as defined above and the noise X is drawn from a Gaussian distribution of zero mean and given variance; the variance is chosen such that the final responses have a realistic level of reliability (see Table 1). We remark in passing that the precise reason for the occuring noise (whether it is aging or temperature or voltage fluctuations) is not relevant in this context; we demonstrate that our attacks can cope well with noise, irrespective of its origin. We used a seeded pseudo-random number generator (numpy's implementation of the Mersenne Twister) for generating any required randomness, e.g., in the random choice of challenges, Gaussian weights, and Gaussian noise. Our entire software for CRP-simulation, as well as for our attacks and managing our ML-experiments, is open-source, and made available from `https://github.com/nils-wisiol/pypuf/tree/2020-split-ipuf`.

2.4 Adversarial Model, Training Set and Test Set

As standard in Strong PUF attack scenarios [RBK10, RSS+10, HYKD14], we assume that adversaries are not restricted to eavesdropping CRPs that are sent in the clear in communication protocols, but that they also have physical access to the Strong PUF for significant amounts of time. During this period, they can freely apply challenges and collect responses, since a Strong PUF's CRP-interface is public and unprotected by definition [RSS+10, RH14, HYKD14]. Assuming a 1 MHz CRP-rate of the PUF, as done in earlier works [RSS+10], this means that training sets on the order of 750 million CRPs (which is the largest number of CRPs that we use in this paper) could be collected in around 12.5 minutes from the PUF. Note that after this brief CRP-collection, no further physical PUF-access is required any more.

Besides these training sets, we supply the attack algorithm with an additional test set of 10^4 CRPs. This test set is used by our attack algorithm for early stopping whenever a model with high accuracy has been detected. Its size remains the same for all of our experiments, and is not included in the total number of challenges given in Tab. 1. Note that all model accuracies reported in this paper are computed on a fresh and uniformly chosen CRP set, which is independent of the training and test set, and hence was not seen before by the attacker. Its size again consists of 10^4 CRPs.

3 Modeling the Interpose PUF

This section details our machine learning-based modeling algorithm for the iPUF. First, we provide an intuition of the employed divide-and-conquer algorithm that separately models upper and lower layer of the iPUF. Sec. 3.1 describes how we obtain an initial high-accuracy model for the lower layer of the iPUF. Afterwards, Sec. 3.2 and Sec. 3.3

show how this paves the way to obtain a complete model of the full iPUF. Empirical results of our attack are relegated to Sec. 4.

Our proposed attack technique on the Interpose PUF uses a divide-and-conquer approach and is based on two crucial observations.

- First, when conducting the linearization attack proposed by Nguyen et al., the resulting model will not only predict PUF responses with an accuracy up to 75%, but will contain all secret information about the lower layer of the Interpose PUF. That is, the reason for the relatively low accuracy of the linearization attack is not the missing information about the lower layer, but almost exclusively the missing challenge bit information.
- Second, the response bits of the upper layer can be heuristically guessed by the attacker for about half the known challenge-response pairs with high accuracy by Alg. 1.

Both observations will be detailed in the following.

3.1 Initial Modeling of the Lower Layer via Random Interpose Bits

This section describes the initial modeling of the lower layer of the given Interpose PUF. Using a given challenge-response set (C, R), we argue why an attacker is capable of obtaining a high-accuracy model of the lower layer of the Interpose PUF. Possession of such a model subsequently enables the attacker to conduct the divide-and-conquer attack as described in the following sections.

Any challenge-response set (C, R) of the full Interpose PUF contains already n out of the $n+1$ challenge bits to the lower layer as well as the lower layer responses. Hence, the only information hidden from the attacker aside from the manufacturing imperfections are the challenge bits in the interpose position. However, our results show that this information is not required to obtain a high-accuracy model of the lower layer. Instead, the attacker can randomly guess the interpose bits, i.e., create the challenge-response set (C_d, R) by himself where C_d is simply interposed with uniformly chosen random bits, i.e.

$$C_d = \left\{ \left(c_1, \ldots, c_{n/2}, \mathbf{c_r}, c_{n/2+1}, \ldots, c_n\right) \mid (c_1, \ldots, c_n) \in C, \mathbf{c_r} \sim_{\mathrm{u}} \{0,1\} \right\}.$$

As previous research has shown [MKP08, NSJ+19], the influence of the middle challenge bit of any XOR Arbiter PUF on the response bit is about 50%, i.e., in about half of the challenges, the response bit will flip if the middle challenge bit is flipped. Applied to our situation, the response of the upper layer of the Interpose PUF will be irrelevant for the PUF's response in about 50% of cases. It follows that the information in (C_d, R) for that half of challenges – where the middle bit does not have an influence on the response – is correct. Furthermore, for the other half of challenges that do have an influence on the response bit, the attacker's guess will be correct with probability 50%, resulting in a total accuracy of 75% for the self-created CRPs (C_d, R) for the lower layer XOR Arbiter PUF.

A refined analysis will then show the following surprising result. Although the training set (C_d, R) has only an anticipated 75% accuracy on the lower layer XOR Arbiter PUF of the target Interpose PUF, the trained model obtained from learning with Logistic Regression will model the lower layer with very high accuracy. This is a crucial result, as the accuracy of the trained model surpasses the estimated accuracy of the training set.

Recall that the Logistic Regression learning algorithm for XOR Arbiter PUFs uses a gradient descent algorithm to train an XOR Arbiter PUF model that agrees with the training set on as many as possible challenges. As stated above, a model of the lower layer of the Interpose PUF will agree with the training set on 75% of challenges, including models where the order of arbiter chains is permuted. Furthermore, models where one

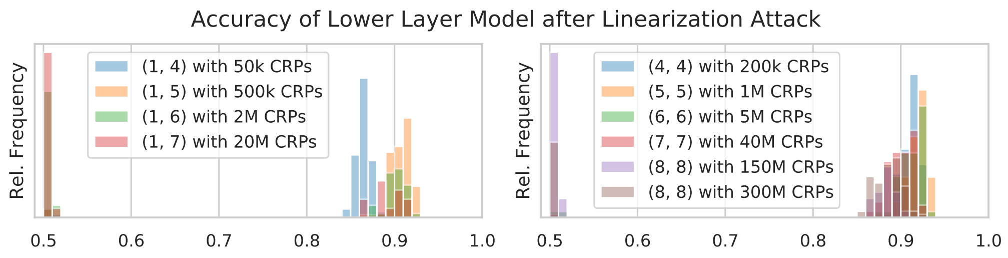


Figure 2: Accuracy of the lower layer model $\hat{f}_\text{d}$ after training using the CRP set (C_d, R) with randomly guessed interpose bits for $(1, k)$ and (k, k)-Interpose PUFs, as captured after execution of line 3 of Alg. 2. Results shown are using the estimated best number of challenge-response pairs (see also Table 1); accuracy is relative to the PUF simulation's reliability. As with ordinary learning of XOR Arbiter PUFs, the probability to obtain a high-accuracy model of the lower layer depends on the size of the Interpose PUF and the training set size. (Just for completeness, our results are artificially capped at 95% due to termination-criteria of the algorithm which increase performance. For models where the initial modeling resulted in the model for a half-inverted lower layer, the accuracy on this is shown; for a justification see Sec. 3.3.)

half of the weights are inverted also agree with the training set on 75% of challenges and model the PUF as if the interpose bit was negated. We will refer to these classes of models as *non-inverted* and *half-inverted*, respectively. For both classes, small variations of the models will agree with the training set on close to 75% of challenges.

On the other hand, with overwhelming probability, no unrelated XOR Arbiter PUF model will agree with this training set on a portion larger than 75% of its CRPs. This is due to the fact that the above constructed training set will very likely contain values that *cannot* be described with an XOR Arbiter PUF model[6]. Hence, the non-inverted model and the model with half-inverted weights of the lower layer both constitute global minima in the Logistic Regression's loss function.

Fig. 2 gives an overview of the lower layer's model accuracy when trained on the randomly interposed challenge-response set, which confirms that a high-accuracy model can be obtained from a partially guessed training set (C_d, R). The accuracy shown is with respect to both half-inverted weights and non-inverted weights, whichever is better. As we will see below, the random choice of a model class will not affect the final accuracy or run time of the attack in any way.

For variations of the Interpose PUF design it is important to note that this observation can (to some extend) be generalized to the case of multiple interposed bits and several layers of interposing[7]. In some extreme cases, we observed that the Logistic Regression algorithm is capable of recovering a significant proportion of the secret information of an XOR Arbiter PUF even if half of all challenge bits in the training set were replaced with random bits. We hence recommend future PUF designs to be tested against this particular vulnerability by analyzing the correlation of the learned model with the simulation under test. We summarize again our above key points: *A low accuracy of some training result (set) is not sufficient to even prove resilience against the LR machine learning algorithm.*

[6]To see this, recall that Arbiter PUFs can be modeled with linear threshold functions (LTFs). LTFs are monotone in all input bits, but the above randomized challenge-response set (C_d, R), is likely not. Although the monotonicity argument gets weaker for products of k LTFs, randomized values are still likely to violate it.

[7]For a more rigorous treatment of feature and label noise in PUF modeling, we refer to Ganji et al. [GTS18].

3.2 Modeling of the Upper Layer

Algorithm 1 Heuristic for creating upper-layer training sets

```
1: procedure Heuristic($C, R, \hat{f}_{\mathrm{d}}$)
2:     initialize empty training set $(C_H, R_H)$
3:     for $c, r$ in $C, R$ do
4:         $c^{(+)} \leftarrow (c_1, \ldots, c_{n/2}, +1, c_{n/2+1}, \ldots, c_n)$
5:         $c^{(-)} \leftarrow (c_1, \ldots, c_{n/2}, -1, c_{n/2+1}, \ldots, c_n)$
6:         if $\hat{f}_{\mathrm{d}}(c^{(+)}) = \hat{f}_{\mathrm{d}}(c^{(-)})$ then
7:             continue
8:         end if
9:         if $\hat{f}_{\mathrm{d}}(c^{(+)}) = r$ then
10:            add $(c, 1)$ to $(C_H, R_H)$
11:        else
12:            add $(c, -1)$ to $(C_H, R_H)$
13:        end if
14:    end for
15:    return $(C_H, R_H)$
16: end procedure
```

Algorithm 1 can construct a training set for the upper layer when given a model with decent accuracy for the lower layer and as well a training set for the complete Interpose PUF. Intuitively, the algorithm first filters all challenges for the complete Interpose PUF where the response of the upper layer does not matter for the final response, as those challenges contain no information about the upper layer. Second, for all remaining challenges, the model for the lower layer is evaluated on both possibilities, and the interpose bit producing the correct response is added to the training set of the upper layer. For all challenges where the model's prediction for the lower layer is correct, the heuristic will correctly determine the upper layer's response bit. We formally show the correctness, effectiveness and accuracy of this heuristic in the following.

Theorem 1. *Given an n-bit (k_u, k_d)-Interpose PUF f, a set of challenges C with corresponding response set R, and an ε-accuracy model $\hat{f}_d$ of the lower layer with $\varepsilon \geq 1/2$, Algorithm 1 will return a training set (C_H, R_H) for the upper layer with accuracy at least $2\varepsilon - 1$ and size expected to be at least $(\varepsilon - 1/2) \cdot |C|$.*

If $\hat{f}_d$ instead has accuracy ε on the half-inverted lower layer, the training set is expected to have accuracy at most $2\varepsilon - 2$, i.e., it models the inversion of the upper layer with accuracy at least $2\varepsilon - 1$; the expectation of the size remains the same.

Proof. For any given challenge $c \in \{-1, 1\}^n$, let $c^{(+)}, c^{(-)} \in \{-1, 1\}^{n+1}$ be defined as in Algorithm 1. We first give a lower bound for the probability that the learned model $\hat{f}_{\mathrm{d}}$ for the lower layer will predict both $c^{(+)}$ and $c^{(-)}$ correctly, based on a pigeon-hole-principle argument. Subsequently, we deduce the accuracy and expected size of the returned challenge-response set that is returned from Heuristic$(C, R, \hat{f}_{\mathrm{d}})$.

For any challenge $c \in \{-1, 1\}^n$, we associate with $c^{(+)}$ and $c^{(-)}$ a pair $(c^{(+)}, c^{(-)})$. By construction, there are 2^n possible pairs containing all 2^{n+1} challenges of $n + 1$ bits each. By prerequisite, we have that $\Pr_{c \in \{-1,1\}^{n+1}}[\hat{f}_{\mathrm{d}}(c) = f_{\mathrm{d}}(c)] = \varepsilon = 1/2 + \alpha$ with $0 \leq \alpha \leq 1/2$. That is, $\hat{f}_{\mathrm{d}}$ models at least $2^n + 2\alpha \cdot 2^n$ challenges correctly. Hence, by the pigeon hole principle, all correctly predicted challenges require at least $2\alpha \cdot 2^n$ pairs $(c^{(+)}, c^{(-)})$. That is,

$$\Pr_{c \in \{-1,1\}^n} \left[\hat{f}_{\mathrm{d}}(c^{(+)}) = f_{\mathrm{d}}(c^{(+)}) \text{ and } \hat{f}_{\mathrm{d}}(c^{(-)}) = f_{\mathrm{d}}(c^{(-)})\right] \geq 2\alpha = 2\varepsilon - 1.$$

I.e., if the model predicts $\hat{f}_\text{d}(c^{(+)}) \neq \hat{f}_\text{d}(c^{(-)})$, then with probability at least $2\varepsilon - 1$ we have $f_\text{d}(c^{(+)}) \neq f_\text{d}(c^{(-)})$ and let r' denote the unique bit that will produce the correct response, which is then indeed the correct response bit of the upper layer of the Interpose PUF. Hence, for each (c, r') added to the training set, the probability that the added example is correct is at least $2\varepsilon - 1$. If $\hat{f}_\text{d}$ is instead an ε-accuracy model for the half-inverted lower layer, then r' is the uniquely inverted interpose bit that will give the correct response and the same argument applies.

As f_d is an XOR Arbiter PUF, we expect $\Pr[f_\text{d}(c^{(+)}) \neq f_\text{d}(c^{(-)})]$ to be $1/2$ on average (with little variance). Our model will thus predict this situation correctly on at least a $2\varepsilon - 1$ fraction of the cases, hence we expect the total number of challenge-response pairs returned by Algorithm 1 to be at least $(\varepsilon - 1/2) \cdot |C|$. $\square$

3.3 Divide-and-Conquer Attack

Algorithm 2 Divide-and-Conquer Interpose PUF Attack

```
1:  procedure ATTACK($n, k_\text{u}, k_\text{d}, C, R$)
2:      $C_d \leftarrow$ INTERPOSE($C$, random bits)          ▷ Guess training set for lower layer
3:      $\hat{f}_\text{d} \leftarrow \text{LR}^{k_\text{d}}_{n+1}(C_d, R)$          ▷ Train model for lower layer
4:      while test accuracy below target do
5:          $C_u, R_u \leftarrow$ HEURISTIC($C, R, \hat{f}_\text{d}$)          ▷ Create training set for upper layer
6:          $\hat{f}_u \leftarrow \text{LR}^{k_\text{u}}_{n}(C_u, R_u)$          ▷ (Re-)train upper layer
7:          $C_d \leftarrow$ INTERPOSE($C, \hat{f}_u(C)$)          ▷ Create training set for lower layer
8:          $\hat{f}_\text{d} \leftarrow \text{LR}^{k_\text{d}}_{n}(C_d, R)$          ▷ Re-train lower layer
9:      end while
10:     return $\hat{f} : c \mapsto \hat{f}_\text{d}(c_1, \ldots, c_{n/2}, \hat{f}_u(c), c_{n/2+1}, \ldots, c_n)$   ▷ Final Interpose PUF model
11: end procedure
```

This section summarizes first our attack strategy and details then how we combine the algorithms outlined in Sec. 3.1 and Sec. 3.2 to form our novel attack against the complete Interpose PUF. The attack algorithm is described in Algorithm 2.

The initial modeling of the lower layer and the heuristic to create a training set for the upper layer enable us to train a model for the upper layer and thereby launch a divide-and-conquer attack on the complete Interpose PUF. In this attack, we are able to model the upper and lower layer separately from each other. As can be seen from Fig. 2 and Theorem 1, an initial accuracy of around 90% $=: \varepsilon$ and an application of the above heuristic will result in a training set for the upper layer of around $2\varepsilon - 1 = 80\%$. Please note that the centering of the initial accuracy of the lower layer model at around 90% (as seen in Fig. 2) is only an artifact of our termination criterion from our Logistic Regression (LR) implementation. Of course, it is also possible to increase the initial accuracy close to 100% and conduct the attack with just training a single model for the lower layer, heuristically creating then a training set for the upper layer, and hereafter training a model for the upper layer. However, for performance reasons, we opted for an iterative approach. We simply terminate each run of the LR phase earlier and repeat the process of training and re-training the upper and lower layer, until a high accuracy is achieved. In this process, while the initial training set for the lower layer was created using randomly guessed interpose bits, all following training phases of the lower layer use the upper layer model to predict interpose bits (cf. lines 2 and 7 in Algorithm 2).

For a $(k_\text{up}, k_\text{down})$-iPUF with challenges of n-bit length, we conclude that launching the divide-and-conquer attack on the iPUF roughly requires the same computational effort as training a model for a $\max\{k_\text{up}, k_\text{down}\}$-XOR Arbiter PUF, although several iterations of the attack[8] are required. This provides an informal reduction of the iPUF security to the security of the XOR Arbiter PUF in the case that only non-reliability-based attacks are applied. This reduction is supported by both theoretical considerations and empirical results as presented in Sec. 4.

One caveat of our reduction lies in the nature of the heuristic in Algorithm 1: the training set for the upper layer is at most half the size of all challenges available to the attacker. While for $k_{\text{down}} > k_{\text{up}}$, this does not pose any challenge to the attacker, but for designs with $k_{\text{up}} = k_{\text{down}}$, this effectively forces the attacker to collect twice as many challenge-response pairs, compared to attack an XOR Arbiter PUF. On the other hand, relying on strict lower bounds for the number of challenge-response pairs is anyhow problematic, as Tobisch and Becker [TB15] have shown.

As noted in Sec. 3.1, the lower layer can randomly be trained in a half-inverted fashion, which will result in a training set with very *low* accuracy for the upper layer of around 10%. This in turn will result in the training of a model for the upper layer that will predict the *negated* response of the actual model, and both effects will cancel out. Therefore, the total accuracy of the trained model will not be affected by the random choice of the model for the lower layer, and indeed the attacker (within our attacker model) has no way of knowing which option is the correct one.

4 Results and Performance Analysis

This section presents a summary of empirical results obtained with our implementation of the divide-and-conquer attack presented in Sec. 3. An overview of the attacks can be found in Tab. 1.

As reported in other works [TB15, WBM+20], the training of models for large instances of XOR Arbiter PUFs is not always successful. Indeed, the non-convexity of the loss function and the consequences on the success probability of the LR algorithm for XOR Arbiter PUFs was already observed by Sölter [Söl09]. To reflect the time unsuccessfully spent training a model, we define for chosen security parameters $n, k_{\text{up}}, k_{\text{down}}$, target reliability, training set size N, and employed computing resources the *time until first success* as the expectation of time spend until a model with prediction accuracy higher than 95%, relative to the PUF's reliability, is obtained. To empirically approximate the time until first success of our attacks, for each group of experiments we computed the mean time of unsuccessful runs t_{fail}, and the mean time of successful runs t_{success}, as well as the relative frequency of successful runs h_{success}. Assuming a Geometric distribution, we compute the expected number of required trials until success as $n_1 = 1/h_{\text{success}}$ and the expected time until first success t_1 as,

$$t_1 = (n_1 - 1) \cdot t_{\text{fail}} + t_{\text{success}};$$

for $h_{\text{success}} = 0$ we set $t_1 = \infty$. We point out that different instances of XOR Arbiter PUFs may differ in their resistance to modeling attacks [TB15], and t_1 only refers to the average time until success, not ruling out the possibility that some instances of the given size may be harder or easier to model. All results shown in this work are with respect to the time until first success.

We studied n-bit challenge $(k_{\text{up}}, k_{\text{down}})$-Interpose PUFs for sizes $(1, k)$ and (k, k) for $k \leq 8$ and analyzed how the time to first success changes for different choices of security parameters n and k as well as training set size N. For performance reasons, choices different from $n = 64$ were only studied for the relatively low choices of $k_{\text{up}}, k_{\text{down}} \leq 4$. Training set sizes were guessed using Tobisch and Becker's [TB15] results and optimized

[8]Note that for both PUFs, the re-training performance is much higher than the initial training performance.

empirically. For results presented here, the choice of training set size which empirically resulted in lowest t_1 was chosen. As all training times refer to wall-clock time, attack times across different CPUs are not comparable. We conducted all modeling attacks for Interpose PUFs with varying reliability between 70% and 100%.

In Fig. 3, we summarize the required time until first success for smaller Interpose PUF sizes and different choices for the used challenge length. It can be seen that the required time increases approximately polynomial with the number of used challenge bits, which is in line with results reported both in the practical and theoretical realm of XOR Arbiter PUF attacks [RSS+13, GTS15b].

For different choices of the number of employed arbiter chains k_{up} and k_{down}, we observed an exponential increase in the number of required challenge-response pairs and required attack time until first success, as shown in Fig. 4. Note that shown training set sizes produced the best result among several guessed choices, but *do not* constitute strict lower bounds. Careful optimization may lead to fewer required challenge-response pairs or shorter time to first success.

In all of our experiments we observed that lower reliability of the Interpose PUF does not have a big impact on the required training time.

For the choice of training set size and smaller choices of $k_{\text{up}}, k_{\text{down}}$ we observed a saturation threshold, beyond which adding more challenge-response pairs to the training set would increase training time instead of decreasing it. This may very well be related to implementation details of the Logistic Regression learner, including whether or not mini batches are used. For Interpose PUF sizes larger than $(6, 6)$, we were not able to confirm or refute this observation due to limitations in computational power.

While the attack as given in Alg. 2 is using an infinite loop, practical experiments were limited to at most five iterations, after which the learning attempt was given up. For Interpose PUF sizes larger than $(7, 7)$, we empirically observed that this is barely of any use, and limited the number of iterations to two.

Finally, the memory footprint of the attacks is manageable and proportionate to the training set size. The storage of 100 million CRPs requires about 6GB of memory; our attack needs a peak memory of about two times the training set size. This implies that all attacks requiring 100 million CRPs or less can be carried out on an up-to-date laptop. Attacks on larger instances require up to 300 million CRPs and 750 million CRPs and thus allocate a total of about 36GB and 90GB of memory, respectively.

Details on memory consumption of our attack implementation can be found in Table 1. Also note that memory consumption depends on many implementation details. Our implementation currently does not swap out memory and, as a time-memory trade-off, uses 1 byte to store 1 challenge bit. We naively store both the upper and lower layer training set separately, which results in storage of heavily redundant data.

5 Variants of the Interpose PUF

Given the successful attacks on the original Interpose PUF design, it is natural to ask if the design can be augmented to achieve better security. To facilitate a swift design process, and in contrast to the tailor-made attack presented in Sec. 3 and 4, this section uses a generic deep-learning modeling approach similar to the one used by Santikellur et al. [SBC19].

Excluded from discussion in this section are intermediate calculations on interpose bits, as they may increase hardware attack surface, as well as interpose positions different from $n/2$. For different choice of interpose positions, we refer to the original iPUF authors [NSJ+19]. We also stress that the interpose position can have significant impact on behavior and machine-learning resistance of the PUF construction, and that results presented here are only valid for interposing bits in the middle of the Arbiter PUF.

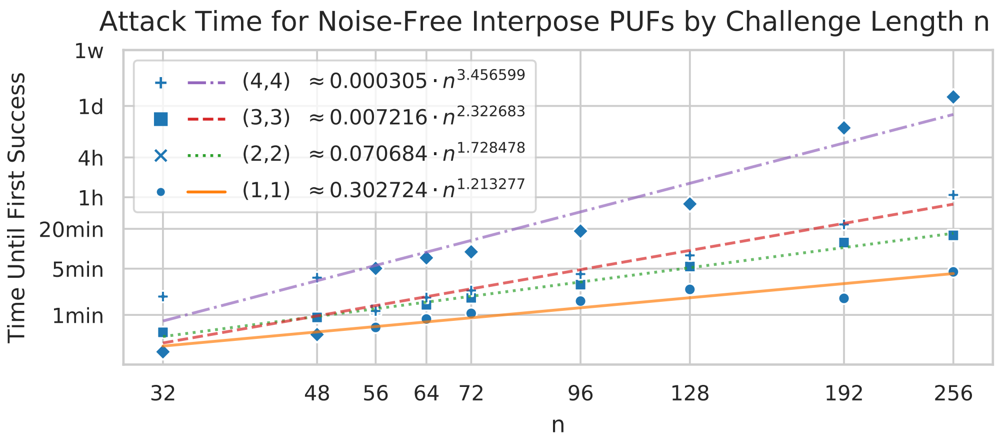


Figure 3: Attack run time for different challenge lengths; times shown refer to time until first success in single-threaded runs. Every data point shows the best obtained time until first success for various choices of guessed amounts of challenge-response pairs. Interpolations given were computed using regression for a, b on $a \cdot n^b$ using all available data.

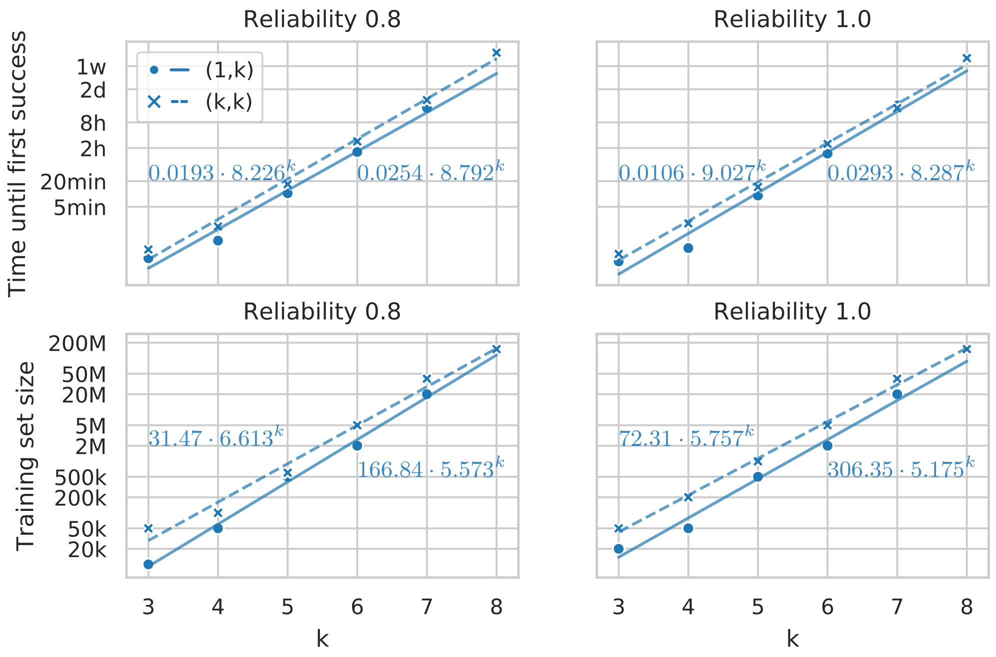


Figure 4: Attack run time and best-performing number of CRPs for Interpose PUFs of different reliability and varying number of employed arbiter chains. Interpolations given were computed using regression for a, b on $a \cdot b^k$ using all available data.

Table 1: Overview of Divide-and-Conquer-Attacks on 64-bit $(k_{\text{up}}, k_{\text{down}})$-Interpose PUFs. For each size and reliability, the best-performing number of CRPs is shown, defined as the setting that gave the shortest time to first success with our software; success is defined as final prediction accuracy above 95%. We used two different Intel® Xeon® CPU types, namely Gold 6130 at 2.1GHz (⋆) and E5-2630 v4 at 2.2GHz (●). For larger experiments, we allowed the use of up to 10 parallel threads to achieve faster training times; note however that for purely technical reasons the speed-up we observed did not exceed 4. Training times across different CPU types are not compared against each other. Additionally to the number of CRPs shown in the table, the attacker was provided with a test set containing an additional 10^4 challenge-response pairs.

$(k_{\text{up}}, k_{\text{down}})$	CRPs	rel.	Mem. (GB)	Time (# Threads)	Success Rate	Samples
(1, 5)	500k	0.8	<1	10.36min (1/⋆)	1.00	100
(1, 5)	500k	0.9	<1	8.70min (1/⋆)	1.00	100
(1, 5)	500k	1.0	<1	9.14min (1/⋆)	1.00	100
(1, 6)	2M	0.8	<1	1.62h (1/⋆)	1.00	57
(1, 6)	2M	1.0	<1	1.48h (1/⋆)	1.00	70
(1, 6)	5M	0.9	<1	1.42h (1/⋆)	1.00	55
(1, 7)	20M	0.8	2.5	17.54h (1/●)	0.97	39
(1, 7)	20M	0.9	2.5	16.17h (1/●)	1.00	33
(1, 7)	20M	1.0	2.5	20.07h (1/●)	1.00	31
(1, 9)	750M	1.0	91	approx. 8w (8/⋆)	0.26	23
(5, 5)	600k	0.8	<1	16.95min (1/⋆)	0.85	195
(5, 5)	600k	0.9	<1	16.13min (1/⋆)	0.88	191
(5, 5)	1M	1.0	<1	14.59min (1/⋆)	0.98	93
(6, 6)	5M	0.7	<1	3.79h (1/⋆)	0.63	54
(6, 6)	5M	0.8	<1	2.86h (1/⋆)	0.78	58
(6, 6)	5M	0.9	<1	2.62h (1/⋆)	0.83	58
(6, 6)	5M	1.0	<1	2.50h (1/⋆)	0.75	53
(7, 7)	40M	0.7	4.9	1.73d (10/●)	0.40	100
(7, 7)	40M	0.8	4.9	1.11d (10/●)	0.62	100
(7, 7)	40M	0.9	4.9	23.38h (10/●)	0.68	100
(7, 7)	40M	1.0	4.9	17.21h (10/●)	0.74	100
(8, 8)	150M	0.7	17.9	∞ (10/●)	0.00	43
(8, 8)	150M	0.8	17.9	2.07w (10/●)	0.25	48
(8, 8)	150M	0.9	17.9	1.59w (10/●)	0.33	55
(8, 8)	150M	1.0	17.9	1.54w (10/●)	0.35	49
(8, 8)	300M	0.7	35.8	18.96w (8/⋆)	0.04	26
(8, 8)	300M	0.8	35.8	2.73w (8/⋆)	0.30	10
(8, 8)	300M	0.9	35.8	1.64w (8/⋆)	0.42	26
(8, 8)	300M	1.0	35.8	2.53w (8/⋆)	0.28	99

5.1 Design Details and Motivation

This section extends the Interpose PUF design in somewhat natural way by iterating the idea of interposition. Subsequently, weaknesses of this iteration are discussed and mitigated using novel ideas. Along this path, the five iPUF-variants `Domino-iPUF`, `XOR-iPUF`, `XOR-Domino-iPUF`, `Tree-iPUF`, and `XOR-Cascaded-iPUF` are derived. While the first two variants are relatively straightforward extensions of the iPUF, the latter three are more complex and are therefore explicitly depicted in Figure 5.

The `Domino-iPUF` reiterates the iPUF's design such that the number of layers, i.e. the number of sequential interpositions, is increased. Thus, instead of two layers it consists of three layers, where the first one's output is interposed into the second one, whose output is in turn interposed into the third layer. Note that the Divide-and-Conquer attack could be extended to be applied to this iPUF variant. We refer to the number of arbiter chains employed in the upper, middle, and lower layer by $k_{\text{up}}, k_{\text{middle}}$, and k_{down}, respectively.

An important observation is that in this iterated design, the influence of every additional layer on the final response halves. Consequently, an attacker having the exact knowledge of the lower x layers achieves an expected prediction accuracy of at least $1 - \frac{1}{2^{(1+x)}}$ by guessing the interpose bit on the highest known layer. This accuracy will be achieved independently of the total number of layers. Hence, when maintaining an interpose position at $n/2$, layer numbers greater than three increase the design's security only marginally and can thus be disregarded. [9]

The other straightforward extension, the `XOR-iPUF`, modifies original $(k_{\text{up}}, k_{\text{down}})$-iPUFs where $k_{\text{up}} = k_{\text{down}}$ such that the Arbiter PUF chains' outputs in the upper layer are interposed separately, each into one corresponding chain in the lower layer, instead of being XORed and then interposed into every lower layer's chain. In fact, this modification is an effective countermeasure against our tailor-made attack. (An alternative perspective on this is to consider this iPUF variant as the XOR of k separate $(1,1)$-iPUFs; hence the name `XOR-iPUF`.)

To increase the depth of the `XOR-iPUF` we contrived the `XOR-Domino-iPUF` (see Figure 5a). It combines both of the previously described designs. Thus, it is the XOR of k separate $(1,1,1)$-`Domino-iPUFs`, where $k = k_{\text{up}} = k_{\text{middle}} = k_{\text{down}}$.

Nevertheless, the number of layers is still subject to the limitation due to sharply dropping influence on the response bit. This motivated the design of the `Tree-iPUF` shown in Figure 5c. As upper layers influence exponentially more leaves, the drop in influence is compensated. Compared to `XOR-iPUF` of same number of layers, it reduces the number of employed arbiter chains while being immune to the Divide-and-Conquer attack. Being a binary tree, this iPUF augmentation consists of $2^{d+1} - 1$ (XOR) Arbiter PUFs, where d is the depth of the tree, i.e. the distance between the first and the last layer. Except for the PUFs in the last layer, the output of every PUF is interposed to two corresponding PUFs in the subsequent layer. The output bits of the last layer are combined via XOR into the final response. All nodes in the tree are k-XOR Arbiter PUFs.

A yet different design that combines both the concept of interposition and the idea of more layers, while maintaining high influence of all building blocks, is the `XOR-Cascaded-iPUF` (see Figure 5b). It is an iterated Interpose PUF design that addresses the above-mentioned problem of the influence loss by reusing each interpose bit: Each layer's output is used for interposition into the subsequent layer as well as for the final response which constitutes the parity of every layer's output bit (including the lowest layer's) and so is influenced by all layers equally. We refer to the number of layers by *length* l; each layer is comprised of a k-XOR Arbiter PUF. Again, we stress that similar to

[9] For this reason, other interpose/feed-in points than the ones examined by us might be interesting in the long term, and should be considered in future analyses. They may exhibit increased security and ML-resilience, even though we did not follow this route in this paper. Please compare also the caption of Figure 5.

Table 2: Overview over Deep-Learning-Attacks on derivatives of the Interpose PUF design. Each derivative has been tested for a selection of different parameters with the challenge-length fixed to 64 bit. For each type and parameter, 100 simulations and attacks were conducted (experiments marked with † have at least 75 samples). Success was defined as prediction accuracy above 90%, relative to the PUF's reliability. Note that attack times are all below comparable times for modeling the original Interpose PUF of similar sizes, even though a generic, non-specialized attack methodology was used.

PUF Type	Size	Arb. Chains	CRPs	Rel.	Mem. (GB)	Time	Succ. Rate
Domino-iPUF	$k = (3, 3, 3)$	9	2M	0.8	2.5	55.6min	1.00
Domino-iPUF	$k = (4, 4, 4)$	12	20M	0.8	12.7	1.0d	0.88
XOR-iPUF	$k = 4$	8	10M	0.8	9.7	4.3h	1.00
XOR-iPUF	$k = 5$	10	40M	0.8	37.8	2.8d	1.00
XOR-Dom.-iPUF	$k = 3$	9	2M	0.8	2.2	15.6min	1.00
XOR-Dom.-iPUF	$k = 4$	12	40M	0.8	37.8	2.0d	1.00
Tree-iPUF	$d = 2, k = 2$	14	5M	0.8	10.4	10.7h	1.00
Tree-iPUF	$d = 3, k = 1$	15	5M	0.8	10.0	8.8h	1.00
XOR-Casc.-iPUF	$l = 2, k = 4$	8	5M	0.8	7.3	16.2h	1.00
XOR-Casc.-iPUF	$l = 3, k = 3$	9	10M	0.8	10.4	8.4h	1.00
XOR-Casc.-iPUF	$l = 2, k = 5$	10	20M	0.8	46.0	2.1d†	0.99
XOR-Casc.-iPUF	$l = 5, k = 2$	10	10M	0.9	9.7	2.7h	1.00

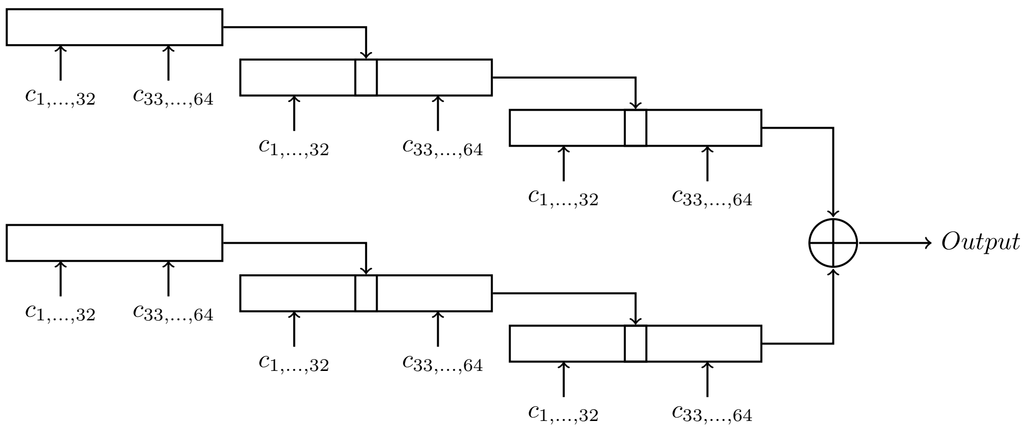
the `Domino-iPUF`, the security of the `XOR-Cascaded-iPUF` depends very strongly on the interpose/feed-in points in the lower layers; optimizing these may result in vastly different security properties, even though we did not follow this route in this paper.

5.2 Empirical Results of Deep Learning Modeling Attacks

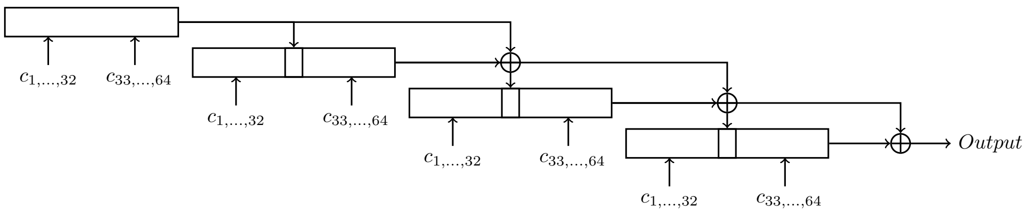
We empirically tested the above introduced five Interpose PUF design derivatives to gain insight in their machine-learning resistance. The designs were parameterized using 64-bit challenge-length and a number of arbiter chains in between 9 and 16, corresponding to an (1,8)-Interpose PUF, and an (8,8)-Interpose PUF, respectively. As attack strategy, following Santikellur et al. [SBC19], a Multi-Layer-Perceptron (MLP) modeling algorithm was chosen, as it requires little customization and no precise model of the concept class to be learned.

As can be seen from our results in Table 2, none of the discussed designs showed increased machine-learning resistance, even when using a generic attack not specialized for the concept class under attack. While some designs may possess practical parameter choices for which modeling is hard, all designs studied in this section were easier to model than the original Interpose PUF of comparable size.

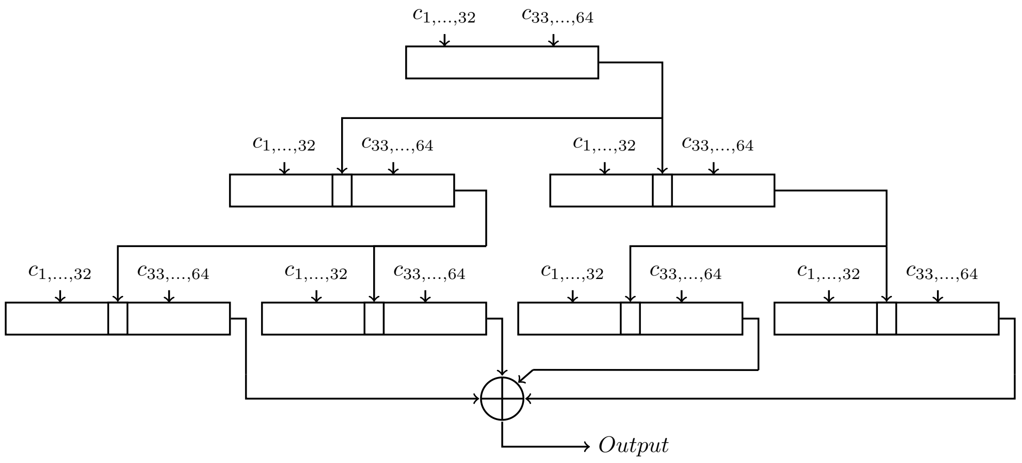
Our results show that the machine-learning hardness of the Interpose PUF at least cannot easily be increased by all too naive augmentation to the original design (albeit there were variants that we did not explore yet; please see comment in the caption of Figure 5). Furthermore, the modeling based on MLP has proven to be an easy-to-use and powerful tool for preliminary analysis of Strong PUF designs. We hence recommend the MLP-based attack to be part of every Strong PUF security analysis.



(a) `XOR-Domino-iPUF`: An iteration of the Interpose PUF design, but instead of having the same interpose bits everywhere, we evaluate interpose chains separately and return the parity of all responses. Each layer shown consists of an k-XOR Arbiter PUF.



(b) `XOR-Cascaded-iPUF`: A variant of the Interpose PUF design, where intermediate results have full influence on the response. We refer to the number of layer by *length* l; each layer consists of a k-XOR Arbiter PUF.



(c) `Tree-iPUF`: A binary-tree of depth d, where each node is a k-XOR Arbiter PUF. Intermediate nodes receive one interpose bit from the layer above and insert their result into two "child" nodes the the layer below. The responses of the leafs are XORed into the final output bit.

Figure 5: Variants of the Interpose PUF design. We stress that the exact ML-resilience of the presented variants may eventually depend notably on the exact interpose (or feed-in) points. All shown variants naturally follow the original iPUF design in the sense that intermediate responses are interposed/fed-in in the *middle* of new challenges (as in the iPUF). We remark that we did not optimize the interpose/feed-in points, or systematically study their effect on the ML-resilience of the resulting architectures in this paper yet.

6 Conclusion

This paper introduced a novel, divide-and-conquer based attack methodology for the Interpose PUF (iPUF), and demonstrated successful modeling attacks on iPUFs of 64-bit challenge length and sizes of up to $k_{\text{up}} = k_{\text{down}} = 8$. Our attacks utilize CRPs-sets that could be physically collected from iPUFs with 1MHz CRP frequencies in five minutes or less, and employ computational power that is readily available to the general public for around 1,000 Euros (i.e., an 8-Core with 64 GB RAM). While our attacks are "merely" conducted on simulated CRPs, the very high accuracy of the underlying linear additive delay model has been confirmed on various occasions in the PUF-literature (see, e.g., [GLC+04, RSS+13]). Furthermore, our attacks are highly robust against random noise, as shown in this paper.

Also a few new, straightforward variants of the original iPUF design could not resist modeling attacks, as we proved in additional ML-experiments. To the contrary, those variants we analyzed did not even require specialized or tailor-made attacks; instead, generic deep learning methods were sufficient. While this does not rule out that yet other variants or additions might possess better resilience (please compare Footnote 9 and the caption of Figures 5), it still complements our above results, showing their profound and substantial nature. In sum, Interpose PUFs and examined variants *of the attacked sizes* must be considered insecure. Designers must either increase challenge length or the number of employed arbiter chains (or both) as countermeasures. We discuss these two options in the following.

While an increase of k_{up} or k_{down} in the iPUF design can indeed improve security drastically (i.e., it will exponentially increase the computation times of our attacks), it will likewise decrease the resulting reliability of the iPUF exponentially in the same parameters. This delimits the practically viable choices for k_{up} or k_{down}, i.e., for the number of parallel arbiter chains in an iPUF. Since the reliability of the Interpose PUF is implementation dependent, it is hard to state a sharp upper bound for which it will remain useful. Based on arbiter chain reliabilities reported in the literature [Bec15], which are around 96% for a single, 64-bit Arbiter chain, some quick simulations of ours estimate a 64-bit (1, 12)-Interpose PUF to have an average reliability of 72%; and an (1, 15)-Interpose PUF is estimated to have 68% (compare also Appendix A). Using heavy error correction and majority voting on repeated measurements of the same CRPs, both of which can be executed publicly and outside a trusted computing base, such error levels might still be just about tolerable, depending on the considered application.

Secondly, to avoid an exponential decrease in stability while still notably improving security, an increase in challenge length could be considered (for a discussion on reliability of such designs, please see Appendix A). For example, $(1, 10)$- or $(1, 12)$-iPUFs with challenge lengths of 512 bits are well out of reach for our current attacks, while their reliability levels might be similar to those of the same architectures with smaller challenge lengths (see above and Appendix A). It must be noted, however, that such an increase in challenge length will lead to a chip area consumption of around 25,000 gate equivalents (GEs). Such designs are possible on large RFID tags, systems on a chip, standard FPGAs and ASICs, but not an option for very small or inexpensive RFID tags, for example. In those situations where such numbers of GEs are realistic, the described $(1, 10)$- or $(1, 12)$-iPUFs of challenge lengths 512 are beyond the reach of the attacks conducted in this paper.

Future Work. An immediate starting point for future investigations is whether the current reach of our attacks (namely $(1, 9)$ and $(8, 8)$-iPUFs of challenge length 64) can be further extended, both with respect to larger challenge lengths n and bigger k_{up} and k_{down}. Can, for example, the $(1, 10)$-iPUF of challenge length 64 bits, which was recommended in the original iPUF paper [NSJ+19], be broken by our methods? All too strong expansion of our results may not be trivial, however, as machine learning efforts grow noticeably in all three said parameters n, k_{up} and k_{down}. Secondly, future research might investigate whether fully polynomial, reliability-based attacks on the iPUF are possible. The iPUF's main architectural principle, in which previous PUF-responses are inserted or "interposed" as middle bits in other PUFs, at least thwarts the existing reliability-based attack by Becker [Bec15, NSJ+19]. This is, and remains, one of the main benefits and contributions of the iPUF design. Still, it will have to be seen whether new, tailor-made reliability-based methods, which follow the spirit of our novel divide-and-conquer strategy, might not also apply to the iPUF in the long term.

A third promising topic would be to investigate under which circumstances the successful modeling of XOR Arbiter PUF like architectures in the presence of feature-noise is possible. (E.g., to examine how many unknown challenge bits, such as the interpose bit in the iPUF, can be compensated in modeling attacks.) Further, quantitative insights into the reach and limits of this method could not just enable new attacks, but might also lead to new, more secure designs. Fourthly, investigating variants of the architectures considered by us in this paper would seem worthwhile. As mentioned earlier, the interpose or feed-in point of the upper signals in the `(XOR) Domino-iPUF`, `XOR-Cascaded-iPUF` or `Tree-iPUF` potentially have a very strong impact on security, but we did not study this effect in detail in this paper. One final and quite central future subject could be the creation of generic, widely applicable test methodologies for the security of new Strong PUF designs; some first steps in this direction have already been taken [GFS19].

Overall, it seems that the area of Strong PUFs will likely remain a battleground between attackers and designers for many years to come. Similar to the field of block ciphers or hash functions, future research will successively create a deeper understanding of the possibilities that both attackers and designers have available on their sides — and will clarify which side will gain ground more quickly.

Acknowledgements

We thank the High Performance Computing Centers of Freie Universität Berlin and Technische Universität Berlin, which combined contributed more than 59 years of CPU time to the development of the Divide-and-Conquer attack presented in this work. This work has been supported by the German Ministry for Education and Research as BBDC 2 (ref. 01IS18025A). Ulrich Rührmair acknowledges funding via the BMBF-project PICOLA.

References

[Bec15] Georg T. Becker. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, Lecture Notes in Computer Science, pages 535–555. Springer Berlin Heidelberg, 2015.

[CCL+11] Qingqing Chen, György Csaba, Paolo Lugli, Ulf Schlichtmann, and Ulrich Rührmair. The bistable ring puf: A new architecture for strong physical unclonable functions. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 134–141. IEEE, 2011.

[DV13] Jeroen Delvaux and Ingrid 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*, pages 137–142. IEEE, 2013.

[GCvD02] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on*

Computer and Communications Security, CCS '02, pages 148–160, Washington, DC, USA, 2002. ACM.

[GFS19] Fatemeh Ganji, Domenic Forte, and Jean-Pierre Seifert. PUFmeter a Property Testing Tool for Assessing the Robustness of Physically Unclonable Functions to Machine Learning Attacks. *IEEE Access*, 7:122513–122521, 2019.

[GLC+04] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience*, 16(11):1077–1098, 2004.

[GTFS16] Fatemeh Ganji, Shahin Tajik, Fabian Fäßler, and Jean-Pierre Seifert. Strong Machine Learning Attack Against PUFs with No Mathematical Model. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, Lecture Notes in Computer Science, pages 391–411. Springer Berlin Heidelberg, 2016.

[GTS15a] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Let me prove it to you: RO PUFs are provably learnable. In *ICISC 2015*, pages 345–358. Springer, 2015.

[GTS15b] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Why Attackers Win: On the Learnability of XOR Arbiter PUFs. In Mauro Conti, Matthias Schunter, and Ioannis Askoxylakis, editors, *Trust and Trustworthy Computing*, Lecture Notes in Computer Science, pages 22–39. Springer International Publishing, 2015.

[GTS16] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Pac learning of arbiter PUFs. *Journal of Cryptographic Engineering*, 6(3):249–258, 2016.

[GTS18] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. A Fourier Analysis Based Attack Against Physically Unclonable Functions. In Sarah Meiklejohn and Kazue Sako, editors, *Financial Cryptography and Data Security*, volume 10957, pages 310–328. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[GYG+16] Qingli Guo, Jing Ye, Yue Gong, Yu Hu, and Xiaowei Li. Efficient attack on non-linear current mirror PUF with genetic algorithm. In *2016 IEEE 25th Asian Test Symposium (ATS)*, pages 49–54. IEEE, 2016.

[HRvD+17] C. Herder, L. Ren, M. van Dijk, M. Yu, and S. Devadas. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Transactions on Dependable and Secure Computing*, 14(1):65–82, January 2017.

[HYKD14] C. Herder, M. Yu, F. Koushanfar, and S. Devadas. Physical Unclonable Functions and Applications: A Tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, August 2014.

[KB14] Raghavan Kumar and Wayne Burleson. On design of a highly secure puf based on non-linear current mirrors. In *2014 IEEE international symposium on hardware-oriented security and trust (HOST)*, pages 38–43. IEEE, 2014.

[LLG+05] Daihyun Lim, Jae W Lee, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.

[MKP08] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Lightweight Secure PUFs. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '08, pages 670–673, Piscataway, NJ, USA, 2008. IEEE Press.

[NSJ+19] Phuong Ha Nguyen, Durga Prasad Sahoo, Chenglu Jin, Kaleel Mahmood, Ulrich Rührmair, and Marten van Dijk. The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 243–290, August 2019.

[RBK10] Ulrich Rührmair, Heike Busch, and Stefan Katzenbeisser. Strong PUFs: Models, Constructions, and Security Proofs. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security: Foundations and Practice*, Information Security and Cryptography, pages 79–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[RH14] U. Rührmair and D. E. Holcomb. PUFs at a glance. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.

[RJB+11] Ulrich Rührmair, Christian Jaeger, Matthias Bator, Martin Stutzmann, Paolo Lugli, and György Csaba. Applications of High-Capacity Crossbar Memories in Cryptography. *IEEE Transactions on Nanotechnology*, 10(3):489–498, May 2011.

[RS14] Ulrich Rührmair and Jan Sölter. PUF modeling attacks: An introduction and overview. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 348. European Design and Automation Association, 2014.

[RSS+10] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249. ACM, 2010.

[RSS+13] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, and Srinivas Devadas. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, 2013.

[Rv12] Ulrich Rührmair and Marten van Dijk. Practical Security Analysis of PUF-Based Two-Player Protocols. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, Lecture Notes in Computer Science, pages 251–267. Springer Berlin Heidelberg, 2012.

[RvD13a] Ulrich Rührmair and Marten van Dijk. On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols. *Journal of Cryptographic Engineering*, 3(1):17–28, 2013.

[RvD13b] Ulrich Rührmair and Marten van Dijk. PUFs in Security Protocols: Attack Models and Security Evaluations. In *2013 IEEE Symposium on Security and Privacy*, pages 286–300, May 2013.

[RXS+14] Ulrich Rührmair, Xiaolin Xu, Jan Sölter, Ahmed Mahmoud, Mehrdad Majzoobi, Farinaz Koushanfar, and Wayne Burleson. Efficient Power and Timing Side Channels for Physical Unclonable Functions. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems*

– *CHES 2014*, Lecture Notes in Computer Science, pages 476–492. Springer Berlin Heidelberg, 2014.

[SBC19] Pranesh Santikellur, Aritra Bhattacharyay, and Rajat Subhra Chakraborty. Deep Learning based Model Building Attacks on Arbiter PUF Compositions. page 10, 2019.

[SD07] G. Edward Suh and Srinivas Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 9–14, New York, NY, USA, 2007. ACM.

[SMCN18] Durga Prasad Sahoo, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, and Phuong Ha Nguyen. A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security. *IEEE Transactions on Computers*, 67(3):403–417, March 2018.

[Söl09] Jan Sölter. *Cryptanalysis of Electrical PUFs via Machine Learning Algorithms.* M.Sc. Thesis, Technische Universität München, München, 2009.

[TB15] Johannes Tobisch and Georg 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*, pages 17–31. Springer, 2015.

[VK15] Arunkumar Vijayakumar and Sandip 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*, pages 653–658. EDA Consortium, 2015.

[WBM$^+$20] Nils Wisiol, Georg T. Becker, Marian Margraf, Tudor A. A. Soroceanu, Johannes Tobisch, and Benjamin Zengin. Breaking the lightweight secure puf: Understanding the relation of input transformations and machine learning resistance. In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications*, pages 40–54, Cham, 2020. Springer International Publishing.

[WM19] Nils Wisiol and Marian Margraf. Why attackers lose: Design and security analysis of arbitrarily large XOR arbiter PUFs. *Journal of Cryptographic Engineering*, 9(3):221–230, September 2019.

[XRHB15] Xiaolin Xu, Ulrich Rührmair, Daniel E Holcomb, and Wayne Burleson. Security evaluation and enhancement of bistable ring PUFs. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 3–16. Springer, 2015.

[YHL16] Jing Ye, Yu Hu, and Xiaowei Li. Poster: attack on non-linear physical unclonable function. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1753. ACM, 2016.

A Reliability Analysis of XOR Arbiter PUFs

We first discuss how the reliability of a single Arbiter PUF depends on its length n. Assume the approximate exactness of the linear additive delay model, where the response bit is computed as the sign of the inner product $\langle w, x \rangle$, with w being the weight vector of the Arbiter PUF. Ideally, one can assume that there is no measurement noise and the weight vector w is set to a fixed value upon manufacturing; we may model each entry of w to be drawn from a normal distribution $\mathcal{N}(0, \sigma_{\text{man}})$ upon manufacturing. Environmental noise such as voltage variations, temperature fluctuations, and aging of the circuit will modify each weight entry w_i by adding noise n_i drawn from some normal distribution $\mathcal{N}(0, \sigma_{\text{meas}})$; here, the n_i are drawn independently for each CRP measurement. Also, the latch may add noise n_L where n_L is drawn from some other normal distribution $\mathcal{N}(0, \sigma_{\text{latch}})$. Given this noise model, a response is now computed as the sign of $\langle w + n, x\rangle + n_L$. Due to n and n_L, the sign may flip in comparison to $\langle w, x\rangle$ (where the latter models the correct/ideal response). The reliability is measured as the probability the sign flips (implying that the response bit will be inverted). This probability is measured over all possible manufactured arbiter PUFs and over all possible measurement noise. Since vector x has its entries in $\{-1, +1\}$, the inner product $\langle w, x\rangle$ has distribution $\mathcal{N}(0, \sqrt{n} \cdot \sigma_{\text{man}})$. Similarly, in $\langle w + n, x\rangle + n_L = \langle w, x\rangle + \langle n, x\rangle + n_L$ the component $\langle n, x\rangle + n_L$ has distribution $\mathcal{N}(0, \sqrt{n} \cdot \sigma_{\text{meas}} + \sigma_{\text{latch}})$. The probability of changing the sign is equal to the probability that value $\langle w, x\rangle$, which is drawn from $\mathcal{N}(0, \sqrt{n} \cdot \sigma_{\text{man}})$, flips its sign due to a shift $\langle n, x\rangle + n_L$, which is drawn from $\mathcal{N}(0, \sqrt{n} \cdot \sigma_{\text{meas}} + \sigma_{\text{latch}})$. This probability depends on the standard deviation $\sqrt{n} \cdot \sigma_{\text{man}}$ in terms of its equivalence in number of standard deviations $\sqrt{n} \cdot \sigma_{\text{meas}} + \sigma_{\text{latch}}$. In other words, the probability is a function of

$$\gamma = \frac{\sqrt{n} \cdot \sigma_{\text{man}}}{\sqrt{n} \cdot \sigma_{\text{meas}} + \sigma_{\text{latch}}} = \frac{\sigma_{\text{man}}}{\sigma_{\text{meas}} + {\sigma_{\text{latch}}}/{\sqrt{n}}}.$$

The larger γ, the less likely the sign will flip upon measurement of an arbitrary Arbiter PUF, hence, the more reliable. Since γ increases in n, we conclude that the reliability will improve for increasing n. We notice that this improvement has a limit, i.e., $\gamma \uparrow \frac{\sigma_{\text{man}}}{\sigma_{\text{meas}}}$ meaning that the latch noise will have a lesser and lesser effect for increasing n. For a detailed discussion on reliability, see Section V.A in [HRvD+17].

We review the following formula to describe the reliability of a given k-XOR Arbiter PUF [NSJ+19]. We assume that all Arbiter PUFs have the same reliability rate p where $p \in (0.5, 1]$ and the delays in the components (i.e., stages) in all Arbiter PUFs are independent of each other. For a given challenge c, the output of an Arbiter PUF instance would be flipped due to measurement noise with probability $1 - p$. The output of the XOR Arbiter PUF for a given challenge c will not be flipped if there is an even number of flipped APUF outputs due to the noise. The reliability of XOR Arbiter PUF p_{XOR} can be calculated as follows:

$$\begin{aligned} p_{\text{XOR}} &= \sum_{i=0, i \text{ is even}}^{k} \binom{k}{i} \times (1-p)^i \times p^{k-i} \qquad (1) \\ &= \sum_{i=0, i \text{ is even}}^{k} \binom{k}{i} \times p^i \times (1-p)^{k-i} \\ &= \frac{(p + (1-p))^k + (p - (1-p))^k}{2} = \frac{1 + (2p-1)^k}{2}. \end{aligned}$$