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] and Ulrich Rührmair[4]

[1] Technische Universität Berlin
{nils.wisiol,jean-pierre.seifert}@tu-berlin.de,niklas.pirnay@campus.tu-berlin.de
[2] Freie Universität Berlin {christopher.muehl,marian.margraf}@fu-berlin.de
[3] University of Connecticut {phuong_ha.nguyen,marten.van_dijk}@uconn.edu
[4] LMU München and University of Connecticut 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 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 Overview and Motivation

It is long known that *"classical keys"*, i.e., secret digital numbers stored permanently in non-volatile memory, constitute an Achilles heel of modern security hardware. Sophisticated adversaries often will not target the used cryptographic primitives such as RSA or AES themselves, which have proven surprisingly attack resilient over the years. Instead, they will frequently aim for the employed secret keys directly, utilizing a variety of different physical or malware-based techniques [And08].

In said situation, both so-called Weak PUFs and Strong PUFs [RH14] can constitute a useful tool for system designers. To start with, Weak PUFs (for example SRAM PUFs [GKST07, HBF08] or DRAM PUFs [TKXC15, XSA+16]) allow the establishment of system-specific digital keys in hardware that has no NVM on board. The keys are derived from the individual, permanent manufacturing variations in the Weak PUF structure itself (such as the SRAM or DRAM cells). These variations determine, for example, the power-up states of the SRAM cells, or the exact decay behavior in DRAM cells, providing each SRAM or DRAM array with some easily measurable, unique properties. The so obtained secret keys can be processed in the system by standard cryptographic techniques — while only being derived and present in digital form when actually needed. This promises a security gain for the resulting hardware systems against various physical and software attacks [RH14].

Strong PUFs [RH14] take this approach yet one step further. In opposition to Weak PUFs, manufacturing variability imposes an exponentially large amount of individual behavior on them, captured as pairs of input and output, so-called *challenge-response pairs* (CRPs). *New*, unknown CRPs shall be hard to predict numerically for an adversary without physically possessing the Strong PUF, even if they know a large number of *other* CRPs of this very PUF. Research over the last decades has shown that such *"unpredictable"* CRPs can be used directly (and without post-processing by a crypto engine) in a variety of protocols: for example, Strong PUF CRPs can be sent in the clear from a prover to a verifier in remote identification schemes [PRTG02] or can be employed in plain form to enable advanced cryptographic protocols such as remote key exchange, oblivious transfer, or secure multi-party computation [TŠ07, Rüh10, BFSK11, DSFK+14]. Strong PUFs may, of course, also be employed for internal key derivation, just like Weak PUFs; but, successfully implemented, they promise a broader application spectrum. Together with the fact that they do not require any isolation in hardware[1], this turns the quest for secure and practical Strong PUFs into a highly worthwhile and relevant research task.

Despite these clearly defined advantages, Strong PUFs (just like any other known security primitive) cannot establish flawless or perfect security, though. One first class of known, impactful attacks concerns some of the cryptogaphic protocols built on Strong PUFs [RvD12, RvD13]. Secondly, so-called *"modeling attacks"* have plagued silicon Strong PUF hardware essentially from its very beginning [GLC+04, LLG+05]. Their basic idea is to collect a number of CRPs from a given PUF, and to apply machine learning techniques in order to extrapolate the Strong PUF's behavior on its (exponentially many) other challenges. Almost all known silicon PUF designs have been attacked by various machine learning algorithms in the past, at least up to certain sizes and complexities (see, e.g., [GLC+04, LLG+05, ÖHS08, MKP08b, RSS+13, RS14, Bec15, XRHB15, GYG+16, YHL16, Del19, WBM+19]). This has led to a perpetual battle between silicon Strong PUF designers and attackers over the years, with no clear and definite outcome yet.

One of the most recent designs that have been suggested in this context is the so-called Interpose PUF or iPUF, for short [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. It constitutes a lightweight design with almost no or very little digital computation (in the form of XOR operations and an 'interpose' operation). Secondly, it contains some novel design elements that practically thwart the main ML-attacks on Arbiter PUF variants that exist up to this moment. This has made the iPUF one of the most promising Strong PUF design proposal to-date.

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 only uses CRPs 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, respectively, 1, i.e., the reliability corresponds to the stability and 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] attempts in-depth mathematical arguments for formally showing the security of the iPUF against the above two mentioned 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

[1] A Strong PUF's CRP-interface is public by definition in the above-mentioned protocols, and assumed to be accessible by friends and foes alike [RH14].

(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.

Our main criticism of the analysis presented in the iPUF paper is that it suggests that its analysis covers all known classical ML algorithms, while in fact only the above mentioned specific attacks are analysed and simulated, and no actual impossibility result is mathematically proven. In particular its Definitions 1 and 2 on 'equivalence' only point out a measure with respect to how many Arbiter PUFs in an Arbiter PUF based design effectively contribute to the response bit. They show that in this 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 that their security is equivalent as this paper demonstrates. The iPUF paper mistakenly suggests this security equivalence. In similar vein, Section 6.5 of [NSJ+19] analyses LR attacks and claims a general statement while only a 'holistic' LR approach is analysed and not one that attempts to use LR as a component in a new and more involved strategy as is done in this paper.

This paper introduces a new *tailor-made* (non-reliability based) classical ML strategy for the iPUF design. In fact, this shows that the $(k_{\text{up}}, k_{\text{down}})$-iPUF is at most as secure as a $\max\{k_{\text{down}}, k_{\text{up}}\}$-XOR Arbiter PUF. First, this shows the contradiction mentioned above. Specifically, the iPUF paper suggested that increasing k_{up}, even if it stays smaller than k_{down}, will improve security – but this is not true. Our result shows that only a parameter setting of the form $(k_{\text{up}} = 1, k_{\text{down}})$-iPUF or $(k_{\text{up}}, k_{\text{down}} = 1)$-iPUF in light of classical ML attacks can be of interest. Second, our new strategy allows the iPUF to be attacked up to substantial levels of size and complexity. In fact, these levels come uncomfortably close to the real, practical stability limits that any iPUF design necessarily is bounded by. This implies that there is only little or no room for mitigating our attacks in practice by straightforwardly making the iPUF larger (in terms of the number of employed Arbiter chains and the length of these chains themselves). Third, with respect to reliability based ML attacks, the iPUF paper only analyses the currently best known one of Becker [Bec15]; since this attack was very successful, little research has been done in order to improve this attack. In light of our result, we expect that an efficient taylor-made reliability based attack may not be out of reach.

As a result of the attacks and experiments given in this paper, we conclude that new advanced designs are necessary. They might try to mix some existing and useful design elements from the iPUF (in particular, the so-called interpose trick which defeats the best known reliability-based ML attack of [Bec15]) with new ideas to achieve long-term resilience.

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 found and accepted. We believe that the same will take place in the silicon Strong PUF area in the next years — with a hard-to-predict outcome. The construction of a long-term secure and highly efficient lightweight design still must be considered a partly open problem, especially in the face of the results of this paper.

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 $k_{\text{down}} = 8$ with prediction accuracies above 95%. In the hardest-to-learn cases studied, i.e., for the $(8, 8)$-64-bit-iPUF, our attacks require up to 150 million CRPs[2], and computation times between one and two 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 existing attacks on the iPUF had reached up to $(4, 4)$-iPUFs only [SBC19].

- While our attacks are still exponential in the number of employed CRPs (like other known classical ML attacks), they very strongly push the limit of attackable iPUF sizes in practice. They so come uncomfortably close to those sizes of the iPUF that are just about practical regarding their noise levels — recall that the noise of an $(k_{\text{up}}, k_{\text{down}})$-iPUF also increases exponentially in k_{up} and k_{down}.
- 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 is a false statement as it only holds true for a specific subset of classical ML attacks. Using our novel attack methods, we now empirically show that the Interpose PUF is at most as secure as a $\max\{k_{\text{up}}, k_{\text{down}}\}$-XOR Arbiter PUF.[3]
- 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].
- We introduce the "mean time to first success" metric for attack times on XOR Arbiter PUFs, taking into account the success rate of the LR algorithm.

1.3 Related Work and Brief Overview of PUF Modeling Attacks

Since the introduction of the first silicon Strong PUF in 2002, the so-called Arbiter PUF [GCvD02], the secure realization of Strong PUFs has been the subject of intense research, with attacks and countermeasures quickly superseding each other. 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) [MKP08a] remained secure for several years, but were eventually tackled up to substantial sizes by evolution strategies, and by a tailor-made variant of logistic regression at CCS 2010 [RSS+10], and also in follow-up works operating on "real", silicon CRPs [RSS+13]. Recently, a specialized attack has yet further improved the modeling performance on the LS PUF [WBM+19], and so has the use of strong computing resources [TB15].

Other promising Strong PUF variants, such as the (XOR) Bistable Ring PUF [CCL+11, XRHB15], the Current Mirror PUF [KB14], or the Voltage Transfer Characteristics PUF [VK15] were also tackled successfully up to certain sizes, using support vector machines [XRHB15], genetic algorithms [GYG+16], simulated annealing and ant colonies [YHL16], or even attacks without a mathematical PUF-model [GTFS16]. The abovementioned empirical methods were complemented by formal proofs, for example in the PAC-framework [GTS16, GTS15a].

[2]Which, could be collected from a silicon Strong PUF implementation with 1MHz CRP-frequency in merely 150 seconds.

[3]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 the security reason to support this suggestion.

Despite the success of these attacks, it should not be overlooked that the best known *"pure"* modeling strategies (which use *nothing else* than sheer CRPs, and do *not* employ stability information of the individual CRPs) on several silicon Strong PUF architectures still take either exponential time, or exponential numbers of CRPs, or both. This includes, for example, XOR Arbiter PUFs, LS PUFs, XOR Bistable Ring PUFs, or also the iPUF itself. On the other hand, also the practical output stability of XOR-based Strong PUF constructions is exponentially bad in the number of XORed sub-responses. This poses hard limits on the scalability of such architectures, and turns their practical security in parts a race between efficient ML algorithms and noise-free implementations.

In order to deal with said exponential modeling costs, side channel information has been used in connection with ML. This has boosted attack complexity from exponential to polynomial on various occasions. For example, certain power and timing side channels for the fist time allowed attacking XOR Arbiter PUFs with merely polynomial complexity [RXS+14]. The most easily applicable "polynomial" attack method on XOR Arbiter PUFs today seems the adversarial exploitation of the output stability of PUF-CRPs as an additional source of information in so-called "reliability attacks" [Bec15].

The described research landscape clearly illustrates the difficult quest for secure Strong PUF designs. From the most popular and often-quoted architectures, only few have survived the test of time in that *no* modeling attacks were *ever* published: Pappu et al.'s specific optical PUF architecture [PRTG02]; Controlled PUFs and variants thereof [GCVDD02, GDC+08, HRvD+17]; and so called SHIC PUFs [RJB+11] (with the latter being the only known Strong PUF with information-theoretic and provable security against modeling. However, all three designs exhibit some known practicality issues: Pappu et al.'s optical PUF requires expensive external precision measurement; Controlled PUFs assume strong physical adversarial access restrictions during their extensive internal and digital post-processing steps; and SHIC PUFs possess relatively large areas (up to 1cm^2) and small read-out speeds (down to 100 bits/second).

This leaves the realization of efficient and secure Strong PUFs based on digital circuits 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 $(8, 8)$-iPUFs, by employing novel, more efficient algorithms and relatively large numbers of CRPs.

1.4 Organization of this Paper

In Sec. 2, we give a short overview over used PUF architectures and employed methodology. Sec. 3 introduces and discusses our attack strategy in detail. We present our empirical results in Sec. 4. We conclude the paper in Sec. 5.

2 Background and Methodology

2.1 Arbiter PUF, Additive Delay Model, and Noise/Reliability

An *Arbiter PUF* is a special circuit that takes advantage of production imperfections, which lead to varying signal propagation delays for different realizations of the same circuit blueprint [GCvD02]. In an Arbiter PUF circuit, after an input (*challenge*) is applied, a rising edge will be sent through it. Before entering the first of n stages, the rising edge signal will be split in two. Both rising edges will traverse the PUF stage by stage. At each stage, depending on the challenge bit applied, the signals will or will not be interchanged. At the end of the Arbiter PUF, an *arbiter* element measures if a signal reaches the top or bottom input first, determining the output (*response*) bit of the circuit.

The behavior of an n-bit Arbiter PUF can be modeled using the *additive delay model*. In this model, we use $\{-1,1\}$ to model the bit values[4] 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, Gassend et al. [GLC+04] were the first to study it's accuracy.

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 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*. Like above, the relation of c and x will not be made explicit if the context is unambiguous. 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.

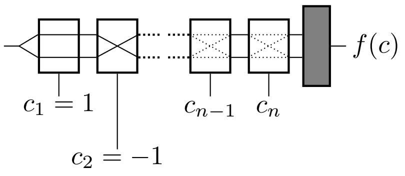
In this work, a reliability of 1.0 is referring to simulation with zero noise; other values of reliability are rounded to the nearest $1/10$. We focused our attention on Interpose PUFs with a reliability on or above 70%.

The reliability of Arbiter PUFs has been studied in detail by Delvaux and Verbauwhede [DV13]. They model evaluation noise of any arbiter chain by adding a Gaussian noise value ΔD_{noise} to the accumulated delay difference ΔD_{model}. This results in the arbiter chain's response to be modeled as $\operatorname{sgn}(\Delta D_{\text{model}} + \Delta D_{\text{noise}})$, i.e., challenges with values of ΔD_{model} closer to zero will be more likely to result in noisy responses. This artifact was verified by experiments with arbiter chains implemented in ASIC.

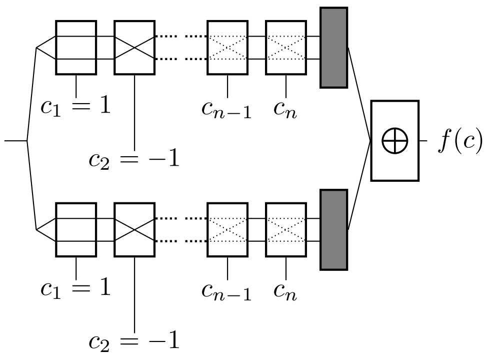
2.2 XOR Arbiter PUF and Interpose PUF

To decrease the attack surface of Arbiter PUFs (see Section 1.3), it was proposed [SD07] to evaluate several Arbiter PUFs in parallel and only output the XOR of the individual response bits. (Note that in this context, we refer to the employed individual Arbiter PUFs as *arbiter chains*.) The resulting *XOR Arbiter PUF* hence has two security parameters, namely the number of challenge bits, in our work usually n, and the number of employed

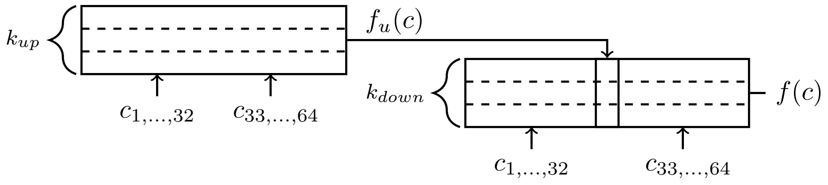
[4] In $\{-1,1\}$ notation, the XOR operation is represented by multiplication. Note that in order to obtain an isomorphism φ to the group $\mathbb{F}_2 = (\{0,1\},+)$, we must assign $\varphi(0) = 1$ and $\varphi(1) = -1$, i.e. representing TRUE by -1 and FALSE by 1. We can then write $\varphi(a) =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



(a) Schematic representation of an Arbiter PUF with challenge length n.



(b) Schematic representation of a 2-XOR Arbiter PUF with challenge length n.



(c) Schematic representation of a 64-bit Interpose PUF. Both upper layer and lower layer consist of k_{up} and k_{down} arbiter chains, respectively.

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

arbiter chains, here usually k, entering in the final XOR of their individual response bits. A 2-XOR Arbiter PUF is schematically displayed in Fig. 1b.

Given the existing attacks on XOR Arbiter PUFs detailed in Section 1.3, Nguyen et al. [NSJ+19] attempted to mitigate these known weaknesses in a new design attempt. Their Interpose PUF or iPUF architecture essentially consists of a smart combination of two XOR Arbiter PUFs. It is defined by a challenge length n, the number k_{u} of XORs in the first XOR Arbiter PUF and the number k_{d} of XORs in the second XOR Arbiter PUF. The first XOR Arbiter PUF, called *upper layer* has challenge length n and consists of k_{u} arbiter chains. When challenged with an input, its 1-bit response is interposed in the middle bit position of the second XOR Arbiter PUF (*lower layer*), resulting in a challenge length a total $n+1$ bit for the lower layer and k_{d} independent arbiter chains. A schematic representation of an Interpose PUF is displayed in Fig. 1c.

By virtue of the interposed bit on the lower layer, LR attacks 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 reliability-based attack is also mitigated by the iPUF design [NSJ+19].

To prove the security of $(k_{\text{up}}, k_{\text{down}})$-iPUF with respect to classical modeling attacks, the authors in [NSJ+19] studied its challenge-response behavior and claimed that the security of $(k_{\text{up}}, k_{\text{down}})$-iPUF is equivalent to that of $(k_{\text{up}}/2 + k_{\text{down}})$-XOR Arbiter PUF. In this paper, we show that the claim on the security based on the challenge-response behavior is not correct, i.e., the security of $(k_{\text{up}}, k_{\text{down}})$-iPUF is only as most that of a $\max\{k_{\text{up}}, k_{\text{down}}\}$-XOR Arbiter PUF.

2.3 Numeric Generation of CRP-Sets

All results presented in this work are based on numerically simulated CRPs. Our simulations use the additive delay model at their core, whose exactness has been proven on earlier occasions (see Section 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; also all arbiter chains are assumed to be unbiased. More complex XOR Arbiter PUFs and Interpose PUFs are constructed out of these single arbiter chain simulations in the straightforward manner.

In the generation of noisy CRPs, or of CRP-sets with reliability smaller than 100%, for each evaluation an independent amount of noise is drawn from a Gaussian distribution of zero mean and prescribed variance is drawn and added to the modeled delay difference. In all our experiments, we seeded the pseudo-random number generator (numpy's implementation of the Mersenne Twister) used to generate uniformly random challenges, Gaussian weights, and Gaussian noise to obtain fully reproducible results.

Our software for simulating and attacking as well as for management of our experiments is free open-source software available on `github.com/nils-wisiol/pypuf`. Written in python, we rely heavily on numpy to achieve high performance. Further implementation details that affect performance are discussed in Sec. 4.

2.4 Adversarial Model, Training and Test Set

As usual in the ML field against PUFs we assume that the adversary has unlimited access to the Strong PUF at hand for very long time periods. Furthermore, the CRP interface is unprotected. This leads the to a very large number of CRPs that can be collected[5].

Following this simple adversarial model, we supply our attack algorithm with a training set of predetermined size that contains N uniformly random challenges along with the respective responses. By the design of the Interpose PUF, these are the challenges to the upper layer and the responses of the lower layer of the target Interpose PUF. Additionally, we supply the attack algorithm with a test set of a fixed 10^4 challenges that are also uniformly random and chosen independently of the training set. For technical reasons, the size of the test set remains the same for all of our experiments and is not included in the total number of challenges given in Tab. 1. This test set is used by our attack for an early stopping whenever a high model accuracy is detected. For small Interpose PUF sizes, the early stopping rule has little absolute influence on the training times. However, for large Interpose PUF sizes, the test size is immaterial compared to the total number of challenge-response pairs required. All model accuracies reported in this paper are computed on a fresh and uniformly random challenge set, i.e., chosen independently of training and said test set, thus not seen before by the attacker. Its size consists of 10^4 CRPs.

2.5 Employed Computing Resources

The development and evaluation of our attack involved the modeling of over 200,000 simulated Interpose PUFs, and hence consumed an enormous amount of CPU time and was conducted using our Institute's High Performance Computing Center. However, in practice the evaluation would use only a single Interpose PUF instance, which can be attacked using a high-end workstation PC available for less than USD 1,000 or even with a regular laptop computer, depending on the Interpose PUF security parameters. Additionally, our implementation can be further optimized for performance and/or memory requirements.

[5] In [NSJ+19], regarding modeling attacks, the authors argued that the security of PUFs strongly depends on the number of CRPs accessed by the adversary. Based on the empirical lower bound of CPRs required for attacking XOR Arbiter PUFs in [TB15], the authors suggested 64-bit (1,10)-iPUF as a practical design because few billion CRPs may be needed to model it successfully.

3 Modeling the Interpose PUF

This section details our machine learning-based modeling algorithm for the Interpose PUF. First, we provide an intuition of the employed divide-and-conquer algorithm that separately models upper and lower layer of the Interpose PUF. Sec. 3.1 describes how we obtain an initial high-accuracy model for the lower layer of the Interpose PUF. Afterwards, Sec. 3.2 and Sec. 3.3 show how this paves the way to obtain a complete model of the full Interpose PUF. 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 already contains 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 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 [MKP08a, 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 CRP (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 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.*

3.2 Modeling of the Upper Layer

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 $(\varepsilon - 1/2) \cdot |C|$.*

[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].

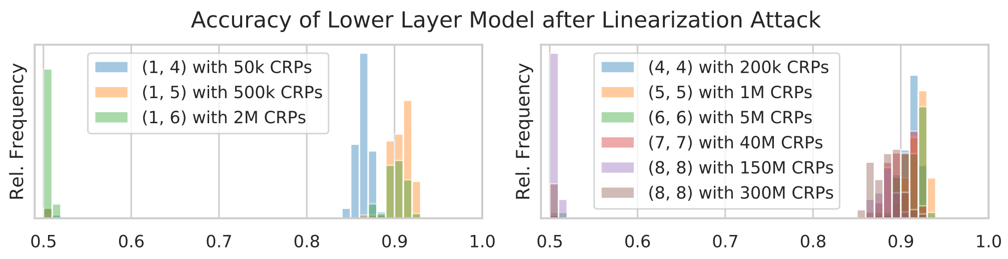


Figure 2: Accuracy of the lower layer model $\hat{f}_\mathrm{d}$ after training using the CRP set (C_d, R) with randomly guessed interpose bits for $(1, k)$ and (k, k)-Interpose PUFs, as captured after execution of line 3 of Alg. 2. Results shown are using the estimated best number of challenge-response pairs (see also Table 1). 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.)

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
```

If $\hat{f}_{\mathrm{d}}$ 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 $0 \le \alpha \le 1/2$. 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,

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

I.e., if the model predicts $\hat{f}_{\mathrm{d}}(c^{(+)}) \ne \hat{f}_{\mathrm{d}}(c^{(-)})$, then with probability at least $2\varepsilon - 1$ we have $f_{\mathrm{d}}(c^{(+)}) \ne f_{\mathrm{d}}(c^{(-)})$ 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 $\hat{f}_{\mathrm{d}}$ 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 $\Pr[f_{\mathrm{d}}(c^{(+)}) \ne f_{\mathrm{d}}(c^{(-)})]$ to be $1/2$ on average (with little variance). Our model will thus predict this situation correctly on at least an $2\varepsilon - 1$ fraction of the cases, hence we expect the total number of challenge-response pairs returned by Algorithm 1 to be at least $(\varepsilon - 1/2) \cdot |C|$. ☐

3.3 Divide-and-Conquer Attack

Algorithm 2 Divide-and-Conquer Interpose PUF Attack

```
 1: procedure Attack($n, k_u, k_d, C, R$)
 2:     $C_d \leftarrow$ Interpose($C$, random bits)          ▷ Guess training set for lower layer
 3:     $\hat{f}_d \leftarrow \mathrm{LR}^{k_d}_{n+1}(C_d, R)$          ▷ Train model for lower layer
 4:     while test accuracy below target do
 5:         $C_u, R_u \leftarrow$ Heuristic($C, R, \hat{f}_d$)          ▷ Create training set for upper layer
 6:         $\hat{f}_u \leftarrow \mathrm{LR}^{k_u}_{n}(C_u, R_u)$          ▷ (Re-)train upper layer
 7:         $C_d \leftarrow$ Interpose($C, \hat{f}_u(C)$)          ▷ Create training set for lower layer
 8:         $\hat{f}_d \leftarrow \mathrm{LR}^{k_d}_{n}(C_d, R)$          ▷ Re-train lower layer
 9:     end while
10:     return $\hat{f} : c \mapsto \hat{f}_d(c_1, \ldots, c_{n/2}, \hat{f}_u(c), c_{n/2+1}, \ldots, c_n)$ ▷ 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 $(k_{\text{up}}, k_{\text{down}})$-Interpose PUF with challenges of n bit length, we conclude that launching the divide-and-conquer attack on the Interpose PUF roughly requires the same computational effort as training a model for a $\max\{k_{\text{u}}, k_{\text{d}}\}$-XOR Arbiter PUF, although several iterations of the attack[8] are required. This provides a reduction of the Interpose PUF security to the security of the XOR Arbiter PUF, supported by both theoretical considerations and empirical results as presented in Section 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 a 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$^+$19], the training of models for large instances of XOR Arbiter PUFs is not always successful. Indeed, the non-convexity of the loss function 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

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

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 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 polynomial-time 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.

Memory footprint of the attacks is manageable and proportionate to the training set size, where 100 million CRPs require about 6GB of memory. Our attack needs a peak memory of about two times the training set size, implying 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 thus require about 36GB of memory, an amount currently easily to be found below USD 1,000. 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.

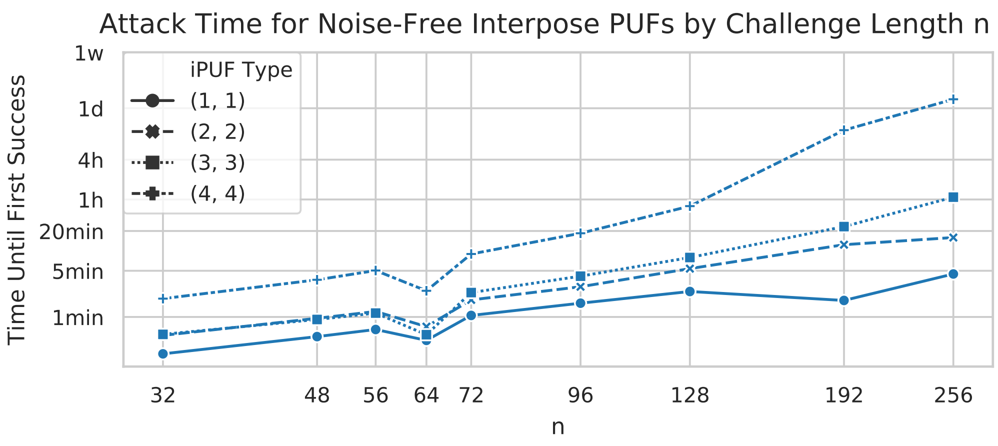


Figure 3: Attack results for different challenge lengths, times refer to time until first success in single-threaded runs. The drop at 64 bit is caused as a faster CPU was used. Every data point shows the best obtained time until first success for various choices of guessed amounts of challenge-response pairs.

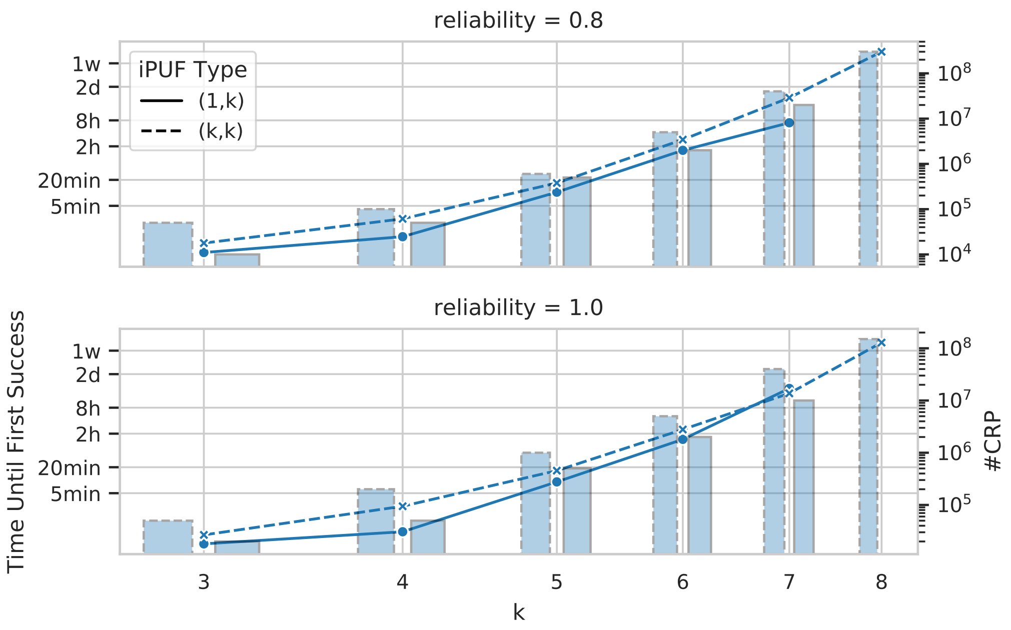


Figure 4: Attack results for PUFs of different reliability and varying number of employed arbiter chains. Note that the run times shown refer to a number of different CPUs and multi-threading settings. Training set size was chosen as the best of several tries.

Table 1: Overview of Divide-and-Conquer-Attacks on 64-bit $(k_{\mathrm{up}}, k_{\mathrm{down}})$-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 ($\star$) and E5-2630 v4 at 2.2GHz ($\bullet$). 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.

$(k_{\mathrm{up}}, k_{\mathrm{down}})$	# CRPs	rel.	Mem. (GB)	Time (# Threads)	Success Rate	# Samples
(1, 5)	500k	0.8	0.4	10.36min (1/$\star$)	1.00	100
(1, 5)	500k	0.9	0.4	8.70min (1/$\star$)	1.00	100
(1, 5)	500k	1.0	0.4	9.14min (1/$\star$)	1.00	100
(1, 6)	2M	0.8	0.5	1.62h (1/$\star$)	1.00	57
(1, 6)	5M	0.9	0.9	1.42h (1/$\star$)	1.00	55
(1, 6)	2M	1.0	0.5	1.48h (1/$\star$)	1.00	70
(4, 4)	200k	0.7	0.3	2.95min (1/$\star$)	0.97	91
(4, 4)	100k	0.8	0.3	1.71min (1/$\star$)	0.98	83
(4, 4)	200k	1.0	0.3	2.03min (1/$\star$)	0.91	91
(5, 5)	600k	0.8	0.4	16.95min (1/$\star$)	0.85	195
(5, 5)	600k	0.9	0.4	16.13min (1/$\star$)	0.88	191
(5, 5)	1M	1.0	0.4	14.59min (1/$\star$)	0.98	93
(6, 6)	5M	0.7	0.9	3.79h (1/$\star$)	0.63	54
(6, 6)	5M	0.8	0.9	2.86h (1/$\star$)	0.78	58
(6, 6)	5M	0.9	0.9	2.62h (1/$\star$)	0.83	58
(6, 6)	5M	1.0	0.9	2.50h (1/$\star$)	0.75	53
(7, 7)	40M	0.7	5.1	1.73d (10/$\bullet$)	0.40	100
(7, 7)	40M	0.8	5.1	1.11d (10/$\bullet$)	0.62	100
(7, 7)	40M	0.9	5.1	23.38h (10/$\bullet$)	0.68	100
(7, 7)	40M	1.0	5.1	17.21h (10/$\bullet$)	0.74	100
(8, 8)	150M	0.7	18	∞ (10/$\bullet$)	0.00	43
(8, 8)	150M	0.8	18	2.07w (10/$\bullet$)	0.25	48
(8, 8)	150M	0.9	18	1.59w (10/$\bullet$)	0.33	55
(8, 8)	150M	1.0	18	1.54w (10/$\bullet$)	0.35	49
(8, 8)	300M	1.0	36	1.92w (8/$\star$)	0.32	72

5 Conclusion

In this paper, we introduced novel attack methodology applicable to the Interpose PUF and demonstrated practicality for Interpose PUFs of 64 bit challenge length for sizes up to $k_{\text{up}} = k_{\text{down}} = 8$ using computational power readily available to the general public. While entirely conducted on simulations, we also demonstrated that the inevitable presence of noise in real-world challenge-response pairs cannot prevent a successful attack. We must conclude that Interpose PUFs of attacked sizes must be considered insecure and designers must increase the security parameters.

For sizes other than the ones we empirically tested, available evidence presented in this paper allows the conclusions that firstly, the challenge length, like in other Arbiter PUF-based designs, cannot be used to achieve exponential advantage over the attacker. Secondly, the number of employed arbiter chains does provide the designer with exponential advantage, but the increase of k_{up} and k_{down} is restricted by the exponential increase of unreliability caused by the underlying arbiter chain design. In light of our results it remains unclear if implementations of the Interpose PUF can be scaled to sizes that can be considered secure even for larger-scale versions of the Divide-and-Conquer attack. However, it must be noted that if implemented reliably, the iPUF still provides resilience against Becker's [Bec15] attack and hence constitutes a major improvement over the XOR Arbiter PUF.

We conclude stating that the field of Strong PUFs remains a cat-and-mouse game of designers and attackers, with the Interpose PUF being the latest major contribution to the defensive side. While security of the Interpose PUF is significantly reduce by the Divide-and-Conquer attack, we hope future designs can benefit from the interposing design and the methodology of testing a novel design against all known attacks; a well-established approach in block cipher design, where all known attacks are shown to be inapplicable for a given new cipher.

Remaining open questions are whether the Interpose PUF can be implemented large and reliably enough to defeat the Divide-and-Conquer attack, and whether or not changes to the Interpose PUF design can mitigate the Divide-and-Conquer approach presented here.

6 Acknowledgements

We thank the High Performance Computing Centers of Freie Universität Berlin and Technische Universität Berlin, which combined contributed more than 30 years of CPU time to this work. Nils Wisiol gratefully acknowledges funding by BMBF BBDC II 01/S18025A.

References

[And08] Ross Anderson. *Security engineering.* John Wiley & Sons, 2008.

[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.

[BFSK11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In *Annual Cryptology Conference*, pages 51–70. Springer, 2011.

[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.

[Del19] J. Delvaux. Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs. *IEEE Transactions on Information Forensics and Security*, 14(8):2043–2058, August 2019.

[DSFK+14] Dana Dachman-Soled, Nils Fleischhacker, Jonathan Katz, Anna Lysyanskaya, and Dominique Schröder. Feasibility and infeasibility of secure computation with malicious PUFs. In *Annual Cryptology Conference*, pages 405–420. Springer, 2014.

[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, New York, NY, USA, 2002. ACM.

[GCVDD02] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Controlled physical random functions. In *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pages 149–160. IEEE, 2002.

[GDC+08] Blaise Gassend, Marten Van Dijk, Dwaine Clarke, Emina Torlak, Srinivas Devadas, and Pim Tuyls. Controlled physical random functions and applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):3, 2008.

[GKST07] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, Lecture Notes in Computer Science, pages 63–80. Springer Berlin Heidelberg, 2007.

[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.

[HBF08] Daniel E Holcomb, Wayne P Burleson, and Kevin Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, 2008.

[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.

[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.

[MKP08a] 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.

[MKP08b] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Testing Techniques for Hardware Security. In *2008 IEEE International Test Conference*, pages 1–10, October 2008.

[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.

[ÖHS08] Erdinç Öztürk, Ghaith Hammouri, and Berk Sunar. Towards robust low cost authentication for pervasive devices. In *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 170–178. IEEE, 2008.

[PRTG02] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, September 2002.

[RH14] Ulrich Rührmair and Daniel E Holcomb. PUFs at a glance. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 347. European Design and Automation Association, 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.

[Rüh10] Ulrich Rührmair. Oblivious transfer based on physical unclonable functions. In *International Conference on Trust and Trustworthy Computing*, pages 430–440. Springer, 2010.

[RvD12] Ulrich Rührmair and Marten van Dijk. Practical security analysis of PUF-based two-player protocols. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 251–267. Springer, 2012.

[RvD13] 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.

[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.

[TKXC15] Fatemeh Tehranipoor, Nima Karimian, Kan Xiao, and John Chandy. DRAM based intrinsic physical unclonable functions for system level security. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 15–20. ACM, 2015.

[TŠ07] Pim Tuyls and Boris Škorić. Strong Authentication with Physical Unclonable Functions. In Milan Petković and Willem Jonker, editors, *Security, Privacy, and Trust in Modern Data Management*, Data-Centric Systems and Applications, pages 133–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[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$^+$19] 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. Technical Report 799, 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.

[XSA$^+$16] Wenjie Xiong, André Schaller, Nikolaos A Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. Run-time accessible DRAM PUFs in commodity devices. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 432–453. Springer, 2016.

[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.