Why Attackers Lose: Design and Security Analysis of Arbitrarily Large XOR Arbiter PUFs

Nils Wisiol, Christoph Graebnitz, Marian Margraf, Manuel Oswald, Tudor A. A. Soroceanu, and Benjamin Zengin

Freie Universität Berlin, Germany
{nils.wisiol,christoph.graebnitz,marian.margraf, manuel.oswald,tudor.soroceanu,benjamin.zengin}@fu-berlin.de

Abstract

In a novel analysis, we formally prove that arbitrarily many Arbiter PUFs can be combined into a stable XOR Arbiter PUF. To the best of our knowledge, this design cannot be modeled by any known oracle access attack in polynomial time.

Using majority vote of arbiter chain responses, our analysis shows that with a polynomial number of votes, the XOR Arbiter PUF stability of almost all challenges can be boosted exponentially close to 1; that is, the stability gain through majority voting can exceed the stability loss introduced by large XORs for a feasible number of votes. Considering state-of-the-art modeling attacks by Becker and Rührmair et al., our proposal enables the designer to increase the attacker's effort exponentially while still maintaining polynomial design effort. This is the first result that relates PUF design to this traditional cryptographic design principle.

1 Introduction

The notion of Physically Unclonable Functions (PUFs) and a first integrated circuit (IC) design that can serve as a PUF was proposed by Pappu et al. and Gassend et al. respectively [15, 30]. Most significantly, their contribution aimed at introducing the notion of PUFs as a mean of authenticating ICs without the need to store a secret key. While many IC authentication protocols rely on a secret key, removing the dependence on a secret from the protocol eliminates all attacks that rely on extracting the key from the device, an attack method that receives broad attention by security researchers.

To replace the secret key and corresponding cryptographic protocols, PUF ICs produce different response behavior on different chips, although the IC design remains the same. This can be achieved by a design that allows nano-scale chip imperfections to influence the output of the IC. As an example for using PUFs as a mean of authentication, an IC can be authenticated answering a number of challenges with the correct response that was prerecorded at the server. Some more tangible use cases are using PUFs as part of secure key cards as described in Gassend et al. [16] or for ICs in the context of radio-frequency identification (RFID) presented by Devadas et al. [9].

The promising PUF approach to IC authentication subsequently received much attention. Gassend et al. [17] and Lim et al. [22] implemented the designs to be used in IC authentication. They also presented successful modeling attacks on their implementations. Armknecht et al. [1, 3] formalized important security features of PUFs.

Arbiter PUFs, which are based on chip-individual delays, became one of the most important objects of PUF implementation and security research. Suh and Devadas [36] proposed to XOR different outputs of one Arbiter PUF to obfuscate the result. Thereafter, Devadas [8] proposed using multiple Arbiter PUFs to construct an XOR Arbiter PUF. In an XOR Arbiter PUF,

k Arbiter PUF responses are evaluated in parallel on k different Arbiter PUFs. The final response is then returned as the XOR of all single Arbiter PUF responses. However, as shown by Rührmair et al. [32] in a comprehensive overview of feasible attacks on various types of PUFs, XOR PUFs can be attacked if k is small. As all known Arbiter PUF implementations suffer from a portion of challenges that do not provide stable responses [6], large XOR Arbiter PUFs are infeasible to build, as their stability will decrease exponentially with growing k as an induction argument shows.

Our contribution provides a formally proven PUF design that allows to increase the number k of arbiter chains arbitrarily while maintaining the overall PUF stability. As we show empirically, a growing parameter k in turn enables the designer to increase the effort for all known oracle attacks exponentially in k, while maintaining polynomial design effort. Note, however, that our design comes at the cost of implementing a volatile memory-solution for majority vote counters, which may introduce new vulnerabilities to side-channel attacks.

The paper is structured as follows: Sec. 2 gives a detailed and motivated description of our proposed design. Sec. 3 compares our approach for secure PUFs to other proposals in the literature. After introducing general preliminaries in Sec. 4 and the necessary background in Sec. 5, we analyze PUF response stability theoretically. Subsequently, Sec. 7 provides both physical and oracle-access security considerations. Finally, Sec. 8 concludes the paper.

2 XOR Majority Vote Arbiter PUF

Our contribution proposes a modification of the well-known XOR Arbiter PUF design that is due to Devadas [8]. The design implements k Arbiter PUFs with n stages each in parallel and XORes the individual response bits. The XOR operation guarantees that the final output bit is influenced by all k individual response bits. This fact provides some evidence to show that the effort an attacker has to make to predict PUF responses is high. So far this idea was successful, as the best known attacks on XOR Arbiter PUFs have an exponential run time in k [32]. However, as hardware implementation of Arbiter PUFs are noisy, it was infeasible to implement XOR Arbiter PUFs with k larger than 12 [41].

In order to reduce the noise of Arbiter PUFs, responses can be determined by a majority vote process [26] before XORing the individual response bits as shown in Fig. 2.1. We show that with a feasible, i.e. polynomial, number of votes in the majority vote process, we can achieve a response stability as high as desired.

However, being closely related to controlled PUFs [18], this design comes at the cost of storing intermediate voting counts in volatile memory. We discuss in Sec. 7.1 how this may jeopardize hardware security goals of the PUF. Also, Majzoobi et al. [25] discussed the influence of voltage and temperature on responses of an FPGA-based Arbiter PUF implementations. One problem that may arise when implementing the XOR Majority Vote Arbiter PUF in hardware is to make sure that rapidly generated responses are statistically independent from each other.

3 Related Work

The employed concept of majority voting before the XOR operation is not new to PUF literature. Majzoobi et al. [25, 26] studied the impact of majority voting on the Arbiter precision empirically, relating the number of votes to a Gaussian distribution that can approximate the delay values. Rührmair et al. [31] and Ganji et al. [12] used majority voting to increase precision of measurements for a fixed number of votes, but did not formally quantify the stability

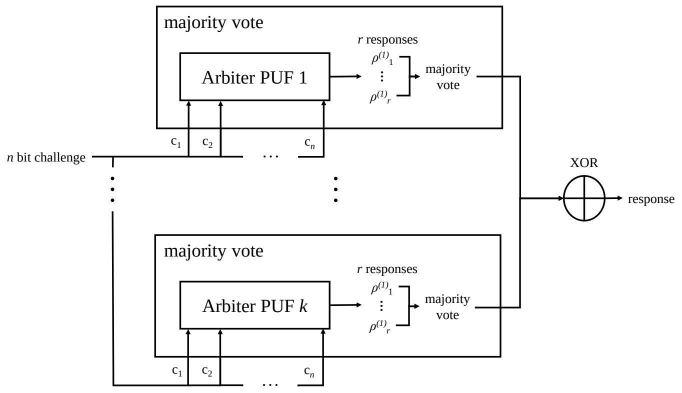

Figure 2.1: A detailed logic block diagram of the proposed XOR Majority Vote Arbiter PUF design. For each arbiter chain, r responses to the given challenge are evaluated and passed to a majority vote. The result is then XORed and returned.

gain. In contrast, our work aims to study the relation of stability gain by majority vote and stability loss through the XOR operation.

It is the subject of many publications to build (provable) secure strong PUFs and likewise it is subject of research to attack published PUF designs.

Based on various non-linear characteristics of current-voltage behavior, several strong PUF candidates have been proposed [20,21,40]. All of them empirically show some resistance against modeling attacks using support vector machines. However Guo et al. [19] published an attack using a genetic algorithm to efficiently learn the non-linear current mirror PUF [21] with accuracy up to 99%.

For XOR Arbiter PUFs, state-of-the-art attacks are due to Becker at al. [4] and Rührmair et al. [32]; both have been empirically shown to run in polynomial time. We discuss those attacks in detail in Sec. 7. Similarly, an attack published by Ganji et al. [13] has been proven to run in polynomial time, but only under the precondition that there is only a constant number of parallel Arbiter PUFs.

Tajik et al. [37,38] and Ganji et al. [11] showed that PUFs, including XOR Arbiter PUFs, can be efficiently modeled if additional information about the chip is retrieved in an attack that requires physical access to the PUF. We are convinced that these attacks can very easily be adopted to also work against our proposed XOR Majority Vote Arbiter PUF. However, this work focuses on the theoretical feasibility of building an arbitrarily large, stable XOR Arbiter PUF. As any hardware-based attack inevitably also depends on the chosen implementation, we do not consider their attacks in detail here.

Spenke et al. [35] take an entirely different approach to secure Arbiter PUFs by including the FPGA PUF definition into the challenge sent to the PUF. At the cost of prolonged evaluation of challenges, they increase the available challenge space and modeling effort. Their approach appears hard to be modeled, as detailed knowledge of the FPGA design is necessary.

4 Preliminaries

An electric *Physically Unclonable Function* (PUF) is a circuit that, although it follows the same design, behaves differently on different hardware. Based on the circuit behavior, the underlying hardware shall be identifiable. The notion of behavior is formalized by challenge-response pairs, with the individual PUF behavior causing a certain response for each challenge given. Usually, both challenge and response are digital data. The idea of PUFs stems from a non-electric implementation [30], but nowadays virtually all research focuses on electric PUFs.

An *Arbiter PUF* is a PUF based on electrical signal delays that vary from chip to chip. It was first introduced by Gassend et al. [15]. Two electrical signals start at the same time and travel through a layout of n *stages*. At each stage, one bit of the n-bit challenge determines whether the signals travel straight through or are interchanged. With the hardware implementation in mind, the stages are sometimes called *multiplexers*. At the end of this *arbiter chain*, an arbiter determines on which connector a signal arrives first and outputs the result as the *response*.

Arbiter PUFs are *strong* PUFs, as they offer a challenge space that has exponential size in the design parameter n. In contrast, PUFs with smaller challenge space are called *weak* PUFs.

This work uses Gaussian random distributions. We denote a random variable X chosen by a Gaussian distribution with mean μ and variance σ^2 by $X \sim \mathcal{N}(\mu, \sigma^2)$. For such X, we have $\Pr[X < x] = \Phi(\frac{x-\mu}{\sigma})$ where Φ is the Cumulative Distribution Function (CDF) of the standard normal distribution, $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t$. The CDF can be written in terms of the error function erf, defined by $\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} \mathrm{d}t$; we obtain $\Pr[X < x] = \frac{1}{2} + \frac{1}{2} \operatorname{erf}(\frac{x-\mu}{\sqrt{2}\sigma})$.

5 Background

This section provides the notions and models used in theoretical analysis throughout this work. Centerpiece of our analysis is a combination of a linear function that models noise-free arbiter chain delays and a stochastic process that models measurement and evaluation noise. This model was introduced by Majzoobi [24] and was later used by Delvaux and Verbauwhede [6] as the foundation for a noise-based side-channel attack.

5.1 Noise-Free Delay Value Model

The final delay difference of a single arbiter chain is the sum of all single delay differences and interchange effects along the way. An important insight for modeling arbiter chains however is that it does not matter *which signal* arrives first. In fact, the signals are indistinguishable. It only matters if *a signal* arrives on the top or bottom path first.

The delay difference of each stage depends on the challenge bit for this stage, as straight and crossing paths have different layout, and thus different nanoscale imperfections. The impact of the delay difference however depends on all following challenge bits: if the signals are later interchanged again, the impact of the stage's delay difference is also inverted. Hence, we obtain

Theorem 1. *The* deterministic noise-free total delay difference *of an arbiter chain with n stages on input*[1] $c = (c_1, .., c_n) \in \{-1, 1\}^n$ *is*

$$\Delta D_{\text{Model}}(c) = \sum_{i=1}^{n} \left(\delta_{c_i}^{(i)} \prod_{j=i}^{n} c_j \right), \tag{5.1}$$

[1]For the sake of easier notation, we chose to model challenges as vectors in $\{-1, 1\}^n$ rather than $\{0, 1\}^n$. If desired, all results can be transformed into $\{0, 1\}^n$ challenges by "encoding" inputs bits with a function $\rho : \{0, 1\} \to \{-1, 1\}$, where $\rho(0) = 1$ and $\rho(1) = -1$. This way, we can write $\rho(b) = (-1)^b$ and have the convenient property $\rho(b_1 b_2) = \rho(b_1) \oplus \rho(b_2)$, where $\oplus$ denotes addition modulo 2. Any output of our model can be transformed by ρ^{-1}.

where $\delta_1^{(i)}, -\delta_{-1}^{(i)} \in \mathbb{R}$ denote the delay difference in the i-th stage in the cases $c_i = 1$ and $c_i = -1$ respectively. For $x = (1, x_1, ..., x_n) \in \{1\} \times \{-1, 1\}^n$ with $x_i = \prod_{j=i}^n c_j$ for $i \geq 1$, the total delay difference ΔD_{Model} is a linear function in x.

Proof. A rigorous proof for (5.1) can be obtained through induction over n.

We can rewrite $\Delta D_{\text{Model}}(c)$ in a case-distinction-free form. The value $\delta_{c_i}^{(i)}$ can be written as the linear function $\delta^{(i)}(c_i) = \frac{1}{2}\left[(\delta_1^{(i)} - \delta_{-1}^{(i)})c_i + \delta_1^{(i)} + \delta_{-1}^{(i)}\right]$. Expanding this notation yields $\Delta D_{\text{Model}}(c) = \sum_{i=1}^n \left(w_i \prod_{j=i}^n c_j\right) + w_0$, where $w_i = \frac{1}{2}\left(\delta_{-1}^{(i)} + \delta_1^{(i)} + \delta_1^{(i-1)} - \delta_{-1}^{(i-1)}\right)$ and $w_0 = \frac{1}{2}\left(\delta_1^{(n)} - \delta_{-1}^{(n)}\right)$, setting $\delta_{c_i}^{(i)} = 0$ for $i = 0$. We can hence write $\Delta D_{\text{Model}}(x) = \sum_{i=0}^n w_i x_i = \langle w, x \rangle$. □

The noise-free Arbiter PUF responses can thus be modeled by a linear threshold function $\operatorname{sgn}(\Delta D_{\text{Model}}(x))$, where x can be easily computed from c. Many machine learning modeling attacks rely upon this fact [32].

5.2 Noise Model

Implementation of arbiter chains in hardware produce noisy responses. To obtain valid predictions for behavior and security of Arbiter PUFs, this noise needs to be modeled. For any challenge $c \in \{-1, 1\}^n$, we thus model the PUF delay value as

$$\Delta D(c) = \Delta D_{\text{Model}}(c) + \Delta D_{\text{Noise}}, \tag{5.2}$$

where ΔD_{Noise} is a Gaussian random variable with zero mean and a variance σ_{Noise}^2 depending on measurement conditions and implementation. Note that this model does not make any assumptions about distribution or model of $\Delta D_{\text{Model}}(c)$. Delvaux and Verbauwhede [6] were the first to propose this model.

5.3 Stability Analysis

The delay-value based model (5.2) allows us to analyze the stability per given challenge. In fact, measurements show that stability of Arbiter PUF responses significantly change depending on the challenge given [35].

Definition 2. For a given challenge $c \in \{-1, 1\}^n$ with model delay difference $\Delta D_{\text{Model}}(c)$ and ΔD_{Noise} following a normal distribution, we define $\text{Stab}(c)$ to be

$$\Pr_{\Delta D_{\text{Noise}}}\left[\operatorname{sgn}(\Delta D_{\text{Model}}(c) + \Delta D_{\text{Noise}}) = \operatorname{sgn}(\Delta D_{\text{Model}}(c))\right],$$

the *stability* of c.

The idea of stability is also captured by the notions of *reliability* [4], *repeatability* [6], or *robustness* [2] in other literature, although those notions might not formally be equivalent.

The stability of any challenge with known model delay difference can be expressed in terms of the error function erf. It only depends on the absolute value of $\Delta D_{\text{Model}}(c)$ and σ_{Noise}; the higher the model delay difference, the higher the stability. The lowest stability of $\frac{1}{2}$ is realized when the model delay difference is exactly zero.

Corollary 3. *For a PUF instance, a challenge with model delay difference $\Delta D_{\text{Model}}(c)$, and normally distributed noise ΔD_{Noise} with zero mean and variance σ^2_{Noise}, the probability that PUF response and model response agree is* $\text{Stab}(c) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{|\Delta D_{\text{Model}}(c)|}{\sqrt{2}\sigma_{\text{Noise}}}\right)$.

While in most use cases the stability of a particular challenge for a given arbiter chain is irrelevant, it is interesting how many "stable" or "unstable" challenges a given arbiter chain instance has. Sec. 6.1 formalizes this question and gives an exact answer by providing the probability distribution of $\text{Stab}(c)$ for uniformly random challenges c.

6 Stability

Our proposed PUF design utilizes both majority vote to increase stability, and XORing of responses, which is known to decrease stability. This section demonstrates that the stability gain overpowers the stability loss, that is, we show that our PUF design can scale to an arbitrarily large number of arbiter chains that are XORed while maintaining overall stability with a polynomial number of votes. We analyze our approach theoretically and (briefly) practically.

6.1 Theoretical Analysis

We analyze the stability of the Majority Vote XOR Arbiter PUF design starting from the basic building blocks, arbiter chains, and then extend the analysis step by step to cover Majority Vote Arbiter Chains and XOR Majority Vote Arbiter PUFs.

6.1.1 Arbiter Chain Stability

Let n be the number of stages in an arbiter chain, let ΔD_{Noise} be a random variable normally distributed with mean zero and variance σ^2_{Noise}.

From Corollary 3, we have for a challenge c that $\text{Stab}(c) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(|\Delta D_{\text{Model}}(c)|/(\sqrt{2}\sigma_{\text{Noise}})\right)$. We can see that the challenge stability and thus the distribution of stability of any Arbiter PUF depends on the distribution of $\Delta D_{\text{Model}}(c) = w_0 + \sum_{i=1}^{n}\left(w_i \prod_{j=i}^{n} c_j\right)$.

Under the assumption that the hardware-intrinsic imperfections formalized by the w_i follow a normal distribution, it can be proven that $\Delta D_{\text{Model}}(c)$ can also be approximated by a normal distribution using the Berry-Esseen Central Limit Theorem (CLT) [5, 10, 42].

What's more, the CLT error bound itself is a random variable depending on the choice of the w_i, with a narrowing variance and lowering mean as n becomes larger. That is, the approximation of ΔD_{Model} becomes better for increasing n.

This fact theoretically supports the common assumption [6, 14] about Gaussian delay values in Arbiter PUF research.

With the probability distribution of $\Delta D_{\text{Model}}(c)$ for uniformly and randomly chosen c, we can approximate the probability that a chosen challenge has stability below a given threshold. In the following, we assume $\Delta D_{\text{Model}} \sim \mathcal{N}(0, \sigma^2_{\text{Model}})$ as a simplification although we actually have $\text{E}[\Delta D_{\text{Model}}] = w_0$, as w_0 was in turn drawn from a distribution with mean zero. The simplification that ΔD_{Model} follows a normal distribution simplifies the analysis drastically.

Lemma 4. *For any given PUF instance with n stages, any probability $z \in [\frac{1}{2}, 1]$, and measurement and hardware conditions described by $\Delta D_{\text{Noise}} \sim \mathcal{N}(0, \sigma^2_{\text{Noise}})$, the probability that a*

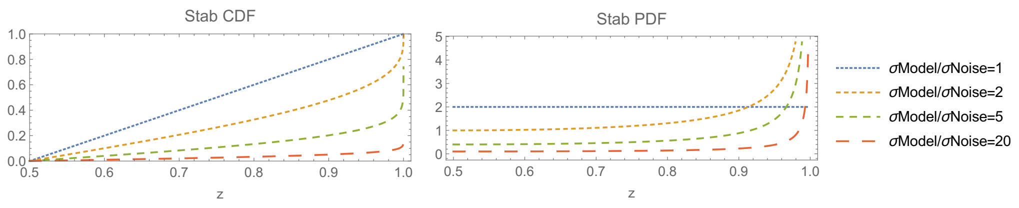

Figure 6.1: Cumulative Distribution Function (CDF) and Probability Density Function (PDF) of the distribution of challenge stability under the assumption of normally distributed model delay values, both shown for ${}^{\sigma_{\text{Model}}}/_{\sigma_{\text{Noise}}} \in \{1, 2, 5, 20\}$. The larger σ_{Model} is when compared to σ_{Noise}, the higher the probability for high stability becomes. The CDF $\Pr[\text{Stab}(c) < z] = \text{erf}\left(\frac{\sigma_{\text{Noise}}}{\sigma_{\text{Model}}} \text{erf}^{-1}(2z-1)\right)$ from Lemma 4 is a central tool in our analysis; the PDF helps us to interpret measured stability frequency in simulations (see Fig. 6.2).

uniformly, randomly chosen challenge c has stability lower than $z \in [\frac{1}{2}, 1]$ is approximated by

$$\text{Stab}_{\text{CDF}}(z) = \Pr_{c \sim \{-1,1\}^n} [\text{Stab}(c) < z] = \text{erf}\left(\frac{\sigma_{\text{Noise}}}{\sigma_{\text{Model}}} \text{erf}^{-1}(2z-1)\right).$$

The proof follows from the CLT approximation of ΔD_{Model} together with the essential Gaussian distribution fact that $\Pr[|\Delta D_{\text{Model}}(c)| < x] = \text{erf}\left({}^{x}/_{\sqrt{2}\sigma_{\text{Model}}}\right)$. The CLT also yields explicit error bounds, if needed.

In the simplest case, we can choose the threshold independently of n to be a constant, $z = 99\%$ to obtain the probability that a randomly chosen challenge has stability below 99%

$$\Pr_{c \sim \{-1,1\}^n} [\text{Stab}(c) < 99\%] \approx \text{erf}\left(1.64 \frac{\sigma_{\text{Noise}}}{\sigma_{\text{Model}}}\right).$$

For any given constant threshold z, the probability that a uniformly and randomly chosen challenge has stability below z, hence depends only on ${}^{\sigma_{\text{Noise}}}/_{\sigma_{\text{Model}}}$. A numeric evaluation can be found in Fig. 6.1. We can interpret $\Pr_c[\text{Stab}(c) < z]$ as cumulative density function. The derivative then gives the probability density function, as shown in the same figure. We can see from the probability density that, while the majority of challenges will have high stability, there is a significant (polynomial) portion of challenges that have stability close to $\frac{1}{2}$.

6.1.2 Majority Vote Arbiter Chain

The stability of an arbiter chain can be boosted by majority voting.

Definition 5. We define for a challenge $c \in \{-1,1\}^n$ the majority vote stability using r votes

$$\text{Stab}_{\text{MV}}^{(r)}(c) = \Pr_{\Delta D_{\text{Noise}} \in \mathbb{R}^r} [\text{majority vote result} = \text{sgn}(\Delta D_{\text{Model}}(c))].$$

The next theorem shows that for any monotone increasing $t(n)$, we can boost the stability for any challenge c that satisfies $\text{Stab}(c) \geq \frac{1}{2} + \frac{1}{t(n)}$ with a polynomial number of votes exponentially close to 1. (How close exactly will be formalized by the choice of $t'(n)$.) We will later show that this prerequisite is fulfilled by "most" challenges, see Lemma 7. For any remaining challenges that do not fulfill the boosting requirement, we know that their stability will be increased through our process, although not up to the desired value of $1 - \frac{1}{2^{t'(n)}}$. Hence the choice

$t(n)$ can be thought of as the parameter that determines which ratio of challenges should be boosted up to exponentially stability, whereas $t'(n)$ determines the minimum stability for these challenges.

Theorem 6. *Consider an arbiter chain with n stages. Let $t(n)$ be any monotone increasing polynomial with $t(n) > 2$. Then for any polynomial $t'(n)$ and all challenges c satisfying* $\mathrm{Stab}(c) \geq \frac{1}{2} + \frac{1}{t(n)}$ *there exists a polynomial $r(n)$ such that when using $r(n)$ votes ($r(n)$ odd for all n), we have* $\mathrm{Stab}_{\mathrm{MV}}^{(r)}(c) > 1 - \frac{1}{2^{t'(n)}}$.

Proof. We consider the probability $q(n) = 1 - \mathrm{Stab}_{\mathrm{MV}}^{(r)}(c)$ that the result of majority voting does not match the model value, that is, the minority of votes show the model value. We sometimes suppress dependencies on n in the notation.

We get $q = \sum_{j=0}^{(r-1)/2} \binom{r}{j} \mathrm{Stab}(c)^j \left(1 - \mathrm{Stab}(c)\right)^{r-j}$. For any $0 \leq j \leq \frac{r-1}{j}$, we set $m = \frac{r}{2} - j$ and obtain $\mathrm{Stab}(c)^j \left(1 - \mathrm{Stab}(c)\right)^{r-j} = \left(\mathrm{Stab}(c)(1 - \mathrm{Stab}(c))\right)^{r/2} \cdot \left(\frac{1-\mathrm{Stab}(c)}{\mathrm{Stab}(c)}\right)^m$, where $\left(\frac{1-\mathrm{Stab}(c)}{\mathrm{Stab}(c)}\right)^m < 1$. Hence, $q < \left(\mathrm{Stab}(c)\left(1 - \mathrm{Stab}(c)\right)\right)^{\frac{r}{2}}$. By definition of Stab and the prerequisite we have $q < \left(1 - \mathrm{erf}^2\left(\frac{|\Delta D_{\mathrm{Model}}|}{\sqrt{2}\sigma_{\mathrm{Noise}}}\right)\right)^{\frac{r}{2}}$ and with $r(n) = 2 \cdot \left\lceil \ln 2 \cdot t(n)^2 \cdot t'(n) \right\rceil + 1$, we obtain $q < 2^{-t'(n)}$. Finally, we have $\mathrm{Stab}_{\mathrm{MV}}^{(r)}(c) = 1 - q > 1 - \frac{1}{2^{t'(n)}}$. □

Theorem 6 shows that certain challenges can be boosted to stability exponentially close to 1 with a polynomial number of votes in the majority vote process. For applications of this work, it is essential that the portion of challenges that cannot be boosted is negligible. The next lemma shows that we can expect the number of challenges not satisfying the prerequisites of Theorem 6 to be small.

Lemma 7. $\Pr[\mathrm{Stab}(c) < \frac{1}{2} + \frac{1}{t(n)}] < \frac{4}{\sqrt{\pi}} \cdot \frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}} \cdot \frac{1}{t(n)}$.

The proof follows from Lemma 4 along with the standard bounds $\frac{2}{\sqrt{\pi}} \cdot x \cdot e^{-x^2} < \mathrm{erf}(x) < \frac{2}{\sqrt{\pi}} \cdot x$ (for $x > 0$) and $\mathrm{erf}^{-1} x < x$ (for $x \in (0, 1/2)$).

The probability distribution of the stability in the challenge space is shown in Figure 6.2. In the PDF plot can be seen that the probability density for low stabilities does not converge to zero.

6.1.3 XOR Arbiter PUF

The boosted stability from Sec. 6.1.2 allows us to use a polynomial number $k(n)$ of arbiter chains and output the XOR of their responses while maintaining high stability.

Definition 8. Consider k arbiter chains with majority vote with r votes each. Let $\rho_i(c)$ with $1 \leq i \leq k$ be the final output of the i-th chain on input $c \in \{-1, 1\}^n$. Let $\Delta D_{\mathrm{Model}}^{(i)}(c)$ be the noise-free delay difference of the i-th chain on input c. We define the *stability of the XOR Majority Vote Arbiter PUF* as $\Pr_{\Delta D_{\mathrm{Noise}}}\left[\bigotimes_{i=1}^k \rho_i(c) = \bigotimes_{i=1}^k \mathrm{sgn}\, \Delta D_{\mathrm{Model}}^{(i)}(c)\right]$, denoted by $\mathrm{Stab}_{\mathrm{XOR}}^{(r)}$, where $\otimes$ denotes the XOR operation.

The probability $\mathrm{Stab}_{\mathrm{XOR}}^{(r)}(c)$ is bounded from below by the probability that all individual arbiter chain response bits match their respective model value. Although this bound disregards the cases where any even number of response bits are flipped, it does still yield the desired exponential bound.

Theorem 9. *Let $t(n)$ be any monotone increasing polynomial with $t(n) > 2$. Then for any polynomial $t'(n)$ and for any challenge c with $\mathrm{Stab}_i(c) \geq \frac{1}{2} + \frac{1}{t(n)}$ simultaneously in all arbiter chains $1 \leq i \leq k$, we have for an XOR Majority Vote Arbiter PUFs with $r(n) = 2 \cdot \lceil \ln 2 \cdot t(n)^2 \cdot t'(n) \rceil + 1$ votes and k arbiter chains $\mathrm{Stab}^{(r)}_{\mathrm{XOR}}(c) \geq 1 - \frac{1}{\frac{1}{k} 2^{t'(n)}}$.*

Proof. For any $1 \leq i \leq k$, we have $\mathrm{Stab}^{(r)}_{\mathrm{XOR}}(c) \geq \mathrm{Stab}^{(r)}_{\mathrm{MV}\,i}(c)^k \geq (1 - \frac{1}{2^{t'(n)}})^k \geq 1 - \frac{k}{2^{t'(n)}}$, using Bernoulli's inequality in the last step. □

As before with Theorem 6, we cannot expect all challenges to be boosted to this stability. Instead, we need to discuss how many challenges we can expect to fulfill the prerequisites of the boosting theorem.

Lemma 10. *For k given arbiter chains with challenge stability Stab_i, the probability for a uniformly and randomly chosen c to have $\mathrm{Stab}_i(c) \geq \frac{1}{2} + \frac{1}{t(n)}$ simultaneously for all chains $1 \leq i \leq k$ is greater than $1 - \frac{4}{\sqrt{\pi}} \cdot \frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}} \cdot \frac{k}{t(n)}$, under the condition that $\frac{4}{\sqrt{\pi}} \cdot \frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}} < t(n)$.*

This lemma follows directly from the probability bound of Lemma 7 and Bernoulli's inequality.

Although we cannot boost *all* challenges to stability exponentially close to 1, Lemma 10 shows that the number of challenges we do not boost is (polynomially) converging to zero with growing n. While not an improvement, this is also not a significant degradation of stability, comparing to a single arbiter chain (see Lemma 7).

Putting all previous results together, we come to the following conclusion.

Corollary 11. *Choosing challenges randomly and uniformly, we have for the XORed output bit of k arbiter chains with n stages and majority vote with r votes each, that for any constants $\alpha \in [0, \frac{1}{2}]$ and $\alpha' > 1$, there exists a number of votes $r \in O(\alpha^2 \cdot \alpha' \cdot k^2 \cdot \log k)$, such that*

$$\Pr\left[\mathrm{Stab}^{(r)}_{\mathrm{XOR}}(c) \geq 1 - \frac{1}{2^{\alpha'}}\right] \geq 1 - \alpha,$$

i.e. the proportion of challenges defined by α has stability exponentially close in α' to 1.

Proof. Our main result follows from Theorem 6, Theorem 7, and Theorem 9 by setting $t(n) = \frac{4}{\sqrt{\pi}} \cdot \frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}} \cdot k \cdot \alpha$ and $t'(n) = \log_2 k + \alpha'$, where k is a monotone increasing polynomial in n. □

The choice of α affects the portion of challenges that will be boosted to exponential stability. Equivalently, α determines how many challenges we expect to fail to be boosted. The choice of α' sets up the boosting goal, i.e. how close to 1 we want the stability to be boosted to.

6.1.4 Number of Votes Required

For a real-world implementation, we need to know an upper bound on how many votes are required for the XOR Majority Vote Arbiter PUF to achieve the desired stability. In the simplest case, we require that all challenges c with $\mathrm{Stab}(c) \geq \frac{1}{2} + \frac{1}{10} = 60\%$ must have $\mathrm{Stab}^{(r)}_{\mathrm{XOR}}(c) > 99\%$. By Lemma 4, this captures most challenges, as $\Pr[\mathrm{Stab}(c) < \frac{1}{2} + \frac{1}{10}] = \mathrm{erf}(\frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}} \mathrm{erf}^{-1}(0.2)) < \mathrm{erf}(0.2 \frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}})$. As an example, we obtain $\Pr[\mathrm{Stab}(c) < \frac{1}{2} + \frac{1}{10}] < 3\%$ for $\frac{\sigma_{\mathrm{Noise}}}{\sigma_{\mathrm{Model}}} = \frac{1}{10}$. From Theorem 9, we hence have $t(n) = 10$ and

$$\mathrm{Stab}^{(r)}_{\mathrm{XOR}}(c) > 1 - \frac{1}{\frac{1}{k} 2^{t'(n)}} = 99\%$$

or $t'(n) = \log_2(100k)$. Along with the number of votes $r(n)$ as defined in Theorem 6, we obtain $r(n) = 2 \cdot \lceil \ln 2 \cdot t(n)^2 \cdot t'(n) \rceil + 1 \approx 1 + 200 \log(100k)$; or $r(n) \approx 1400$ for $k = 24$. Notice that the upper bound grows with $O(\log k)$. Our simulations show that in fact much lower numbers for $r(n)$ suffice, see Sec. 6.2. The large values of $r(n)$ are hence due to the many non-tight bounds we use in the proof of Theorem 6 and Theorem 9.

6.2 Simulation

In this section we present software simulation results that validate our stability results from Sec. 6. (Code at `https://github.com/nils-wisiol/pypuf/tree/2017-why-attackers-lose`.)

Number of Votes Required To show that a high total stability can be reached using majority vote we determined the minimum number of votes required for acceptable stability for various numbers of parallel arbiter chains. Fig. 6.2 shows detailed results of how many votes are needed for stable responses ($\Pr[\text{Stab}(c) \geq 95\%] \geq 80\%$), as determined by a binary search on the voting count for different k. The polynomially increasing number of votes required to fulfill the stability requirements for a useful PUF implementation shows that XOR Arbiter PUFs with high stability can be built arbitrarily large using majority vote.

Stability Distribution For a more precise exposition of the stability distribution of Majority Vote XOR Arbiter PUFs of size k we estimated stability values by a simulation. Our example uses $k = 32$ Majority Vote Arbiter Chains with $n = 32$ stages each. The stability distribution shown in Fig. 6.2 was achieved using $r \in \{51, 501\}$ votes. As estimated by the computation of the previous section, the number of votes needed to reach a stability of 95% with probability 80% is 51. In turn, approximately 501 votes are needed to achieve the stability of the single arbiter chain that was used to build the MV XOR Arbiter PUF. This shows that even large XOR Majority Vote Arbiter PUFs can become stable through a feasible number of votes.

7 Security Analysis

7.1 Resilience against Physical Attacks

Although the goal of PUFs was to increase the robustness against physical attacks, many are known, e.g. side channel [6, 28, 34], invasive [27, 29], and fault injection attacks [7, 37]. To our knowledge there is no known strong PUF design that is secure against physical attacks without classical hardware security measures that are also applied to secure flash memory. Hence, the XOR Majority Vote Arbiter PUF may only serve as a first step towards a secure strong XOR Arbiter PUF. Although resilient against all known machine learning attacks (see Sec. 7.2), the XOR Majority Vote Arbiter PUF is particularly vulnerable to the readout of the memory that stores the voting count, similar to Ring Oscillator PUFs (RO-PUFs) [23]. The contribution of this work is hence limited to showing one way to mitigate already-known modeling attacks at the additional cost of physical attack surface as already known in RO-PUFs.

Further research has to be done to combine the arbiter chains with tamper resistant volatile memory, to prevent a readout of the voting counts. Unfortunately, it is not straightforward to build such a PUF system, as it increases the complexity and cost of the PUF. Hu and Sunar [39, Sec. 13] are giving a state-of-the-art overview on tamper resistant memory.

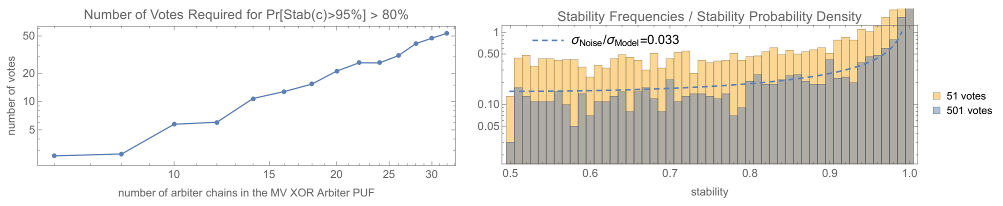

Figure 6.2: The graph (left) shows the minimum number of votes needed such that for a uniformly random challenge c we have $\Pr[\mathrm{Stab}(c) \geq 95\%] \geq 80\%$ for different k, as determined by a simulation (Sec. 6.2). The simulation uses arbiter chain length of $n = 32$, however we showed that the results are independent of n. This log-log graph confirms the result that the number of votes required grows polynomially. (Values were computed using `mv_num_of_votes.py .95 .80 32 32 2 .033 2000 200`.)
The histogram (right) shows the probability density of an XOR Majority Vote Arbiter PUF of size $k = 32$ and chain length of $n = 32$. We used 51 and 501 votes to boost stability to $\Pr[\mathrm{Stab}(c) \geq 95\%] \geq 80\%$ and to the stability of the building block arbiter chains, respectively. The dashed line shows the theoretical stability probability density for a single arbiter chain (i.e. before majority vote and XOR) as used in this simulation ($\sigma_{\mathrm{Noise}}/\sigma_{\mathrm{Model}} = 0.033$). The graph confirms that a Majority Vote XOR Arbiter PUF built from these arbiter chains and the given number of votes can not only achieve a decent stability (at 51 votes), but also reach the same stability as a single arbiter chain (at 501 votes). (The histogram shows data generated with pypuf's stability calculation, `stability_calculation.py 32 32` r `0.033 10000 200 0xbeef` for $r \in \{51, 501\}$.)

7.2 Resilience against Machine Learning Attacks

To analyze the XOR Majority Vote Arbiter PUF and provide evidence for its security against machine learning attacks we use an oracle access model. An adversary, who wants to perform an attack on a PUF communicates with an oracle. Through the oracle, the adversary can obtain any number of genuine PUF responses to (adaptively) chosen challenges. Notice that responses to equal challenges do not necessarily have always the same value, as responses can be noisy.

In the following, we study the feasibility of two state-of-the-art attacks on the XOR Majority Vote Arbiter PUF.

Becker's Machine Learning Algorithm This section discusses an attack on XOR Arbiter PUFs presented by Becker [4, Sec. 5]. Becker uses an evolution strategy based machine learning evolution algorithm that assesses fitness based on the *reliability* of the responses of the individual arbiter chains. His notion of reliability is similar to our notion of stability in Sec. 6. Suppose an attacker sends the same challenge c to the oracle l times and collects l response bits $r(c)_1, r(c)_2, \ldots, r(c)_l$. Then the reliability h_i is computed by $h(c) = \left|\frac{l}{2} - \sum_{j=1}^{l} r(c)_j\right|$. This notion of reliability $h(c)$ maximizes in the case that $\mathrm{Stab}(c) = 1$, and minimizes in the case that $\mathrm{Stab}(c) = 0.5$.

To learn physically feasible constructions without majority vote of XOR Arbiter PUFs ($k \leq 12$) in reasonable time with a relatively small number of examples in the training set, Becker's attack uses information revealed by the reliability of individual challenges. A response for a given challenge is called *unstable* if it has a delay difference $\Delta D_{\mathrm{Model}}$ close to zero. Unfortunately, as we showed in Theorem 6 and Lemma 7, there is always a small number of responses

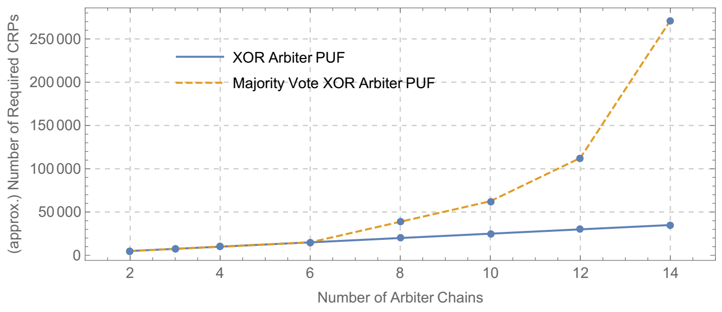

Figure 7.1: The required number of CRPs when using Becker's CMA-ES reliability attack against the Majority Vote XOR Arbiter PUF shows an exponential increase. In contrast, classic XOR Arbiter PUFs can be attacked with linear increase only and are limited by reliability decrease.
All measurements were obtained using an empirical binary search on attacks on simulated (Majority Vote) XOR Arbiter PUFs. (All values shown for $n = 32$.)

in a PUF, that can not be boosted to stability exponentially close to 1 with a polynomial number of votes. Hence, there are always challenges with stability $\mathrm{Stab}(c) < \frac{1}{2} + \frac{1}{t(n)}$ and so the prerequisite for Becker's attack is fulfilled.

Nevertheless, the simulation of Figure 7.1 show that a Majority Vote XOR Arbiter PUF is only learnable with Becker's algorithm if an exponential number of examples in the training set is used.

Rührmair's et al. Logistic Regression Algorithm In this section we briefly discuss an attack on XOR Arbiter PUFs presented by Rührmair et al. [32, 33]. They implemented and evaluated various machine learning methods in order to learn different types of PUFs, including basic Arbiter PUFs and XOR Arbiter PUFs. For both, the best results were delivered by the Logistic Regression (LR) algorithm with Resilient Propagation (RProp) as optimization method.

Due to its nature, the LR algorithm has to perform multiple iterations before it finds a minimum. One trial of the algorithm takes polynomial time. But the number of restarts grows exponentially in the number k of arbiter chains. This fact was already observed by Rührmair et al. and is insignificant when k is limited by the stability. However, in the majority vote scenario, the designer's choice of k can defeat the modeling attack runtime.

8 Conclusion

All known oracle attacks against stable XOR Arbiter PUFs based on machine learning are only successful if the number of arbiter chains is small. On the other hand, a large number of arbiter chains leads to an instability for most challenges due to noisy effects. Hence, such XOR Arbiter PUFs cannot be used in practice. Our work showed that with majority voting, we can boost the stability for almost all challenges exponentially close to 1, even for PUFs with a large number of arbiter chains. We formally proved that for this purpose only a feasible (polynomial) number of votes is needed to increase the stability gain through majority voting beyond the stability loss induced by the XOR operation. However, while the size of the arbiter chains will only slightly increase with the vote counter, our design comes at the cost of additional digital memory that holds volatile sensitive information and a prolonged evaluation run time, which may jeopardize security goals.

To support our theoretical stability claim we evaluated the number of votes necessary in a software simulation. We further showed based on empirical data that state-of-the-art attacks based on oracle access will not be able to model our design in polynomial time. Further research should focus on how the reliability-based CMA-ES attack scales with the XOR Arbiter PUF stability.

In asymmetric cryptography, one usually has methods to increase the security (e.g., extending the key length) if there are minor improvements for attacks (e.g., faster computers). To our knowledge, our design is the first result that relates PUF design to this traditional cryptographic design principle.

9 Acknowledgments

The authors would like to thank Georg T. Becker, Yongzhi Cao, Fatemeh Ganji, Rainer Plaga, Jean-Pierre Seifert, and Shahin Tajik for many helpful discussions and insights. We further would like to thank the HPC Service of ZEDAT, Freie Universität Berlin, for computing time.

References

[1] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Francois-Xavier Standaert, and Christian Wachsmann. A formalization of the security features of physical functions. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 397–412. IEEE, 2011.

[2] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 685–702, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[3] Frederik Armknecht, Daisuke Moriyama, Ahmad-Reza Sadeghi, and Moti Yung. Towards a unified security model for physically unclonable functions. In *Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610*, pages 271–287, 2016.

[4] Georg T Becker. The gap between promise and reality: on the insecurity of XOR arbiter PUFs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 535–555. Springer, 2015.

[5] Andrew C. Berry. The accuracy of the Gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–122, 1941.

[6] 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.

[7] Jeroen Delvaux and Ingrid Verbauwhede. Fault injection modeling attacks on 65 nm arbiter and RO Sum PUFs via environmental changes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(6):1701–1713, 2014.

[8] Srini Devadas. Physical unclonable functions (pufs) and secure processors. In *Workshop on Cryptographic Hardware and Embedded Systems*, 2009.

[9] Srinivas Devadas, Edward Suh, Sid Paral, Richard Sowell, Tom Ziola, and Vivek Khandelwal. Design and implementation of PUF-based "unclonable" RFID ICs for anti-counterfeiting and

security applications. *2008 IEEE International Conference on RFID (Frequency Identification), IEEE RFID 2008*, pages 58–64, 2008.

[10] Carl-Gustaf Esseen. *On the Liapounoff limit of error in the theory of probability.* Almqvist & Wiksell, 1942.

[11] Fatemeh Ganji, Juliane Krämer, Jean-Pierre Seifert, and Shahin Tajik. Lattice basis reduction attack against physically unclonable functions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1070–1080. ACM, 2015.

[12] Fatemeh Ganji, Shahin Tajik, Fabian Fäßler, and Jean-Pierre Seifert. Strong machine learning attack against PUFs with no mathematical model. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9813:391–411, 2016.

[13] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. Why attackers win: On the learnability of XOR arbiter PUFs. In *Trust and Trustworthy Computing*, pages 22–39. Springer, 2015.

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

[15] 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*, pages 148–160. ACM, 2002.

[16] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Delay-based circuit authentication and applications. In *Proceedings of the 2003 ACM symposium on Applied computing - SAC '03*, page 294, New York, New York, USA, 2003. ACM Press.

[17] 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.

[18] Blaise Gassend, Marten Van Dijk, Dwaine Clarke, and Srinivas Devadas. Controlled physical random functions. *Security with Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, 10(4):235–253, jan 2007.

[19] Qingli Guo, Jing Ye, Yue Gong, Yu Hu, and Xiaowei Li. Efficient Attack on Non-linear Current Mirror PUF with Genetic Algorithm. *2016 IEEE 25th Asian Test Symposium (ATS)*, pages 49–54, 2016.

[20] Mukund Kalyanaraman and Michael Orshansky. Novel strong PUF based on nonlinearity of MOSFET subthreshold operation. *Proceedings of the 2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013*, pages 13–18, 2013.

[21] 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, may 2014.

[22] 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.

[23] Heiko Lohrke, Shahin Tajik, Christian Boit, and Jean-Pierre Seifert. No place to hide: Contactless probing of secret data on FPGAs. Cryptology ePrint Archive, Report 2016/593, 2016.

[24] Mehrdad Majzoobi, Eva Dyer, Ahmed Elnably, and Farinaz Koushanfar. Rapid FPGA Delay Characterization Using Clock Synthesis and Sparse Sampling. In *IEEE International Test Conference (ITC), Austin, TX.* 2010.

[25] Mehrdad Majzoobi, Akshat Kharaya, Farinaz Koushanfar, and Srinivas Devadas. Automated Design , Implementation , and Evaluation of Arbiter-based PUF on FPGA using Programmable Delay Lines. *IACR Cryptology ePrint Archive*, 2014:639, 2014.

[26] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. FPGA PUF using programmable delay lines. *2010 IEEE International Workshop on Information Forensics and Security, WIFS 2010*, 2010.

[27] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Semi-invasive EM attack on FPGA RO PUFs and countermeasures. *Workshop on Embedded Systems Security*, pages 1–9, 2011.

[28] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-channel analysis of PUFs and fuzzy extractors. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6740 LNCS, pages 33–47, 2011.

[29] Dmitry Nedospasov, Jean-Pierre Seifert, Clemens Helfmeier, and Christian Boit. Invasive PUF analysis. In *Proceedings - 10th Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2013*, pages 30–38, 2013.

[30] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.

[31] Ulrich Ruhrmair, J.L. Martinez-Hurtado, Xiaolin Xu, Christian Kraeh, Christian Hilgers, Dima Kononchuk, Jonathan J. Finley, and Wayne P. Burleson. Virtual Proofs of Reality and their Physical Implementation. In *2015 IEEE Symposium on Security and Privacy*, volume 2015-July, pages 70–85. IEEE, may 2015.

[32] 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.

[33] 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, nov 2013.

[34] 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. *Cryptographic Hardware and Embedded Systems*, (8731):476–492, 2014.

[35] Alexander Spenke, Ralph Breithaupt, and Rainer Plaga. An arbiter PUF secured by remote random reconfigurations of an FPGA. In *International Conference on Trust and Trustworthy Computing*, pages 140–158. Springer, 2016.

[36] 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*, pages 9–14. ACM, 2007.

[37] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. Physical Characterization of Arbiter PUFs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 493–509. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[38] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. Laser Fault Attack on Physically Unclonable Functions. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 85–96. IEEE, sep 2015.

[39] Mohammad Tehranipoor and Cliff Wang. *Introduction to hardware security and trust*, volume 9781441980. 2012.

[40] Arunkumar Vijayakumar and Sandip Kundu. A Novel Modeling Attack Resistant PUF Design based on Non-linear Voltage Transfer Characteristics. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, DATE '15, pages 653–658, New Jersey, 2015. IEEE Conference Publications.

[41] M. D. M. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, July 2016.

[42] Илья Сергеевич Тюрин. Уточнение верхних оценок констант в теореме Ляпунова. *Успехи математических наук*, 65(3):201–202, 2010.