Combined Modeling and Side Channel Attacks on Strong PUFs

Ahmed Mahmoud
Institute for Nanoelectronics
TU München
80333 München, Germany
ahmed.mahmoud@tum.de

Ulrich Rührmair
TU München
Arcisstr. 21
80333 München, Germany
ruehrmair@in.tum.de

Mehrdad Majzoobi
ECE Department
Rice University
Houston, TX 77005, USA
mehrdad.majzoobi@rice.edu

Farinaz Koushanfar
ECE Department
Rice University
Houston, TX 77005, USA
farinaz@rice.edu

ABSTRACT

Physical Unclonable Functions (PUFs) have established themselves in the scientific literature, and are also gaining ground in commercial applications. Recently, however, several attacks on PUF core properties have been reported. They concern their physical and digital unclonability, as well as their assumed resilience against invasive or side channel attacks. In this paper, we join some of these techniques in order to further improve their effectiveness. The combination of machine-learning based modeling techniques with side channel information allows us to attack so-called XOR Arbiter PUFs and Lightweight PUFs up to a size and complexity that was previously out of reach. For Lightweight PUFs, for example, we report successful attacks for bitlengths of 64, 128 and 256, and for up to nine single Arbiter PUFs whose output is XORed. Previous work at CCS 2010 and IEEE TIFS 2013, which provides the currently most efficient modeling results, had only been able to attack this structure for up to five XORs and bitlength 64.

Our attack employs the first power side channel (PSC) for Strong PUFs in the literature. This PSC tells the attacker the number of single Arbiter PUF within an XOR Arbiter PUF or Lightweight PUF architecture that are zero or one. This PSC is of little value if taken by itself, but strongly improves an attacker's capacity if suitably combined with modeling techniques. At the end of the paper, we discuss efficient and simple countermeasures against this PSC, which could be used to secure future PUF generations.

Keywords

Physical unclonable functions, side channel attacks, modeling attacks, hardware security

1. INTRODUCTION

Most modern cryptographic and security schemes are built on the concept of a secret key. This forces current hardware to contain a piece of digital information that is, and remains, unknown to the adversary. This requirement can be difficult to uphold in practice: Physical attacks like invasive, semi-invasive or side-channel attacks, as well as software attacks like malware, can lead to key exposure and full security breaks.

Indeed, one of the main motivations in the development of *Physical Unclonable Functions (PUFs)* was their promise to better protect secret digital keys in vulnerable hardware systems. A PUF is an (at least partly) disordered physical system P that can be excited with external stimuli or so-called challenges c. It reacts with corresponding responses r, which depend on the challenge and on the micro- or nanoscale structural disorder that is present in the PUF. It is assumed that this disorder cannot be cloned or reproduced exactly, not even by the PUF's original manufacturer, and that it is unique to each PUF. Each PUF P thus implements a unique and individual function g that maps challenges c from an admissible challenge set to responses $r = g(c)$. The tuples (c, r) are thereby usually called the challenge-response pairs (CRPs) of the PUF.

Due to its complex internal structure, a PUF can avoid some of the shortcomings of classical digital keys. It is usually harder to read out, predict, or derive PUF-responses than to obtain digital keys that are stored in non-volatile memory. The PUF-responses are only generated when needed, which means that no secret keys are present permanently in an easily accessible digital form.

These facts have been exploited in the past for different security protocols. Prominent examples include schemes for identification [21, 6] or various forms of (tamper sensitive) key storage and applications thereof, such as intellectual property protection or read-proof memory [8, 13, 32]. Simultaneously, the use of so-called Strong PUFs [1] in advanced cryptographic protocols has been investigated. Various protocols were suggested, including schemes for identification [21], key exchange [21, 5, 1], oblivious transfer [22, 1], or bit

[1]For a differentiation between different PUF types, including so-called Weak PUF and Strong PUFs, we refer to [28, 26].

commitment [20, 1]. Using Strong PUFs in these protocols has the advantage that no permanently stored digital secret keys and standard computational assumptions (such as the hardness of factoring) are involved. Overall, the assumed security advantages of PUFs have attracted considerable attention within the security community over the last decade.

Recent Attacks on PUFs.

In recent years, however, an increasing number of attacks on PUFs have been published. Some of them were specifically developed for this new primitive, while others are an adaption of known strategies to the PUF case. Not all attacks apply to every PUF design, and, more generally, to the two major PUF types of *"Weak PUFs"* and *"Strong PUFs"* [28, 26], in the same manner.

To start with, one of the best-known PUF attacks are machine-learning (ML) based modeling attacks. They almost exclusively apply to Strong PUFs, however, since this PUF type has a publicly accessible CRP interface, which allows the simple collection of the large numbers of CRPs that are required in this attack type (see [28, 30]). This attack form has been discovered already in the early days of the field [12], and has been the subject of a relatively large number of subsequent publications [12, 19, 15, 14, 10, 2]. In two recent works from CCS 2010 [28] and IEEE T-IFS 2013 [30], a rather large number of Strong PUF designs have been examined and attacked successfully. The architectures that exhibited the highest ML-resilience in these two investigations were the XOR Arbiter PUF [31, 28] and the Lightweight PUF [14]. They could only be be attacked efficiently if they are composed of up to five or six single Arbiter PUFs with bitlength 64 or 128 each [28, 30].

Summarizing the current state of the literature, XOR Arbiter PUFs and Lightweight PUFs composed of eight single Arbiter PUFs with bitlength 64 or 128, say, currently must be considered secure against modeling. At the same time, these architectures are just sufficienly stable to be used in practice — the authors of [28, 30] estimate the stability limits of these designs to be around eight single Arbiter PUFs which are XORed at the end of the structure. Recall in this context that XOR Arbiter PUFs and Lightweight PUFs cannot be made arbitrarily large — their stability descreases exponentially in the number of single Arbiter PUFs whose output is XORed [28, 30].

Other recent PUF attacks have in majority focused on so-called Weak PUFs. For example, Helfmeier et al. have reported cloning attacks on SRAM PUFs at HOST 2013 [9]. Nedospasov et al. very recently described successful invasive attempts on SRAM PUFs at FDTC 2013 [18]. Merli et al. attacked the error-correcting module of Weak PUFs at TRUST 2011 [16], and EM analyses on ring oscillator PUFs (RO PUFs) have been carried out by the same group at WESS 2011 [17].

Interestingly, comparable cloning, invasive or side channel attacks on typical Strong PUFs, such as optical PUFs or the Arbiter PUF and variants thereof, have not been reported to this date. In general, it appears more difficult to clone or invasively attack Strong PUFs. They possess very many CRPs and their response generation process is more complex: Typically, a quite large number of components interact in the generation of a single response. Successful physical cloning hence requires the accurate duplication of all of these components: For example successful tuning of *all* delay values in an Arbiter PUF; or precise positioning of *all* the scattering centers in an optical PUF.

Finally, a number of protocol attacks on advanced Strong PUF protocols such as key exchange or oblivious transfer have been presented recently at CHES 2012, IEEE S&P 2013, and other venues [27, 23, 24, 25]. They represent interesting and relevant strategies, but are beyond the topic of this paper, which focuses on hardware attacks.

Our Contributions.

In this work, we combine ML with side channel information to substantially improve the reach of modeling attacks on electrical Strong PUFs. We thereby target XOR Arbiter PUFs and Lightweight PUFs, which are currently considered the two securest electrical Strong PUFs [28, 30]. Our approach for the first time allows us to reach and exceed the practical stability limit of these architectures, which was estimated to be around eight single Arbiter PUFs in [28, 30]: We are now able to successfully attack XOR Arbiter PUFs and Lightweight PUFs of bitlengths 64 and 128 that are composed of up to ten single Arbiter PUFs. The attacks are executed on simulated CRPs generated by the established additive delay model for Arbiter PUFs; recent publications have shown that these synthetical CRPs are very close to "real" silicon CRPs, and can be used without much loss for ML-based security analyses [30]. For a detailed discussion on synthetic CRPs, PUF noise, practical stability limits, and ML performance, we refer the reader to [30], Section II-G.

Our attacks in practice require physical access to the PUF, which is part of the established Strong PUF attack model [28, 30]. They involve the collection of a substantial amount of CRPs (up to the order of millions), but lead to very short computation times for the machine learning algorithms themselves (up to several hundred seconds on a standard INTEL Quadcore). At the cost of more CRPs, the attacks could be extended relatively easily to XOR-based structures with more than ten XORed single Arbiter PUFs, but we did not follow this route in this paper.

Our power side channel tells the attacker the number of zeros and the number of ones that enter the large, final XOR gate of the XOR Arbiter PUF or Lightweight PUF. Taken for itself, this side channel appears relatively worthless, since the attacker still does not know which of the single Arbiter PUF outputs was a one and which a zero. But by waiting for the "good" cases where either *all* responses of the single Arbiter PUFs were a zero or *all* were a one, the attacker can gain useful information. Suitable combination with ML techniques leads to a drastic improve of modeling performance and accuracy, as shown in the sequel.

The presented approach is one of the first side channels for Strong PUFs reported in the literature. The only other known side channel for Strong PUFs is the exploitation of unstable CRPs in modeling attempts. Without implementing it, this idea has been mentioned briefly in [28], and has been carried out for the first time in two very interesting works by Delvaux and Verbauwhede [2, 3]. One current practical problem is that the efficiency is worse than in pure machine-learning based modeling attacks: More CRPs are required, and the remaining prediction error is slightly worse. This may be improved in future versions of the attacks. Currently, our PSC seems the only side channel that can substantially improve the reach and performance of Strong PUF modeling attacks.

At the end of the paper, we also discuss efficient countermeasures against our power side channel. They could be implemented easily in new, future generations of Arbiter PUFs and variants to thwart our attacks.

Organization of this Paper.
Section 2 provides some background on arbiter-based PUFs and previous machine learning attacks thereon. Section 3 describes a power tracking side channel. In sections 4 and 5, we discuss the effect of the power side channel on the security of Lightweight and XOR Arbiter PUFs, respectively. Potential design countermeasures against our power side channel are discussed in Section 6. Our work is summarized in Section 7.

2. BACKGROUND

2.1 Different Delay-based PUF Types

Delay-based PUFs were among the first electrical PUFs that have been introduced [6, 7, 11, 31] , and are currently among the most widespread and best investigated PUFs. They are based on a "race" between two competing signal paths, which is called by a final arbiter element (mostly a latch). Three delay-based PUFs that are relevant for this work are described in the sequel.

Arbiter PUFs.
The basic Arbiter PUF (Arb-PUF) is discussed in [7, 11, 31]. It consists of a sequence of n stages, for example multiplexers. Two electrical signals race simultaneously and in parallel through these stages. Their exact paths are determined by a sequence of n external bits $b_1 \cdots b_n$ applied to the stages, whereby the i-th bit is applied at the i-th stage. If $b_i = 0$, then the paths run "in parallel" through the multiplexers, and if $b_i = 1$, they cross each other and change position. After the last stage, an "arbiter element" consisting of a latch determines whether the upper or lower signal arrived first and correspondingly outputs a zero or a one. The external bits are usually regarded as the challenge C of this PUF, i.e., $C = b_1 \cdots b_n$, and the output of the arbiter element is interpreted as their response R. See [7, 11, 31] for further details. The parameter n is often referred to as the bitlength of the Arbiter PUF.

XOR Arbiter PUFs.
One possibility to strengthen the resilience of arbiter architectures against machine learning attacks, which has been suggested in [12, 31], is to employ k individual Arb-PUFs in parallel, each with n stages (i.e., each with bitlength n). The same challenge C is applied to all of them, and their individual outputs t_i are XORed in order to produce a global response t_{XOR}. We denote such an architecture as k-XOR Arb-PUF. The case of a 2-XOR Arb-PUF is illustrated in Figure 1.

Lightweight Secure PUFs.
Another type of delay-based PUF, which we term Lightweight Secure PUF or Lightweight PUF for short, has been introduced in [14]. It is similar to the XOR Arb-PUF of the last paragraph. At its heart are k individual standard Arb-PUFs arranged in parallel, each with n stages (i.e., with bitlength n), which produce k individual outputs $r_1, \ldots, r_k$.

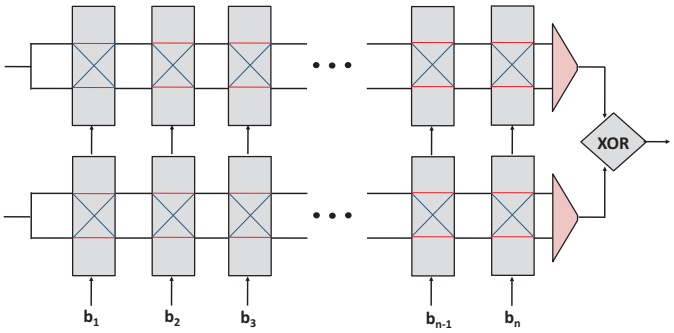

Figure 1: The architecture of a 2-XOR Arb-PUF. It consist of two single, parallel Arbiter PUFs. The outputs of their Arbiters (red triangles) are XORed to produce the final output.

These individual outputs are XORed to produce a multi-bit response $o_1, ..., o_m$ of the Lightweight PUF, according to the formula

$$o_j = \bigoplus_{i=1,\ldots,x} r_{(j+s+i) \bmod k} \quad \text{for } j = 1, \ldots, m. \qquad (1)$$

Thereby the values for m (the number of output bits of the Lightweight PUF), x (the number of values r_j that influence each single output bit) and s (the circular shift in choosing the x values r_j) are variable design parameters.

Another difference to the XOR Arb-PUFs lies in the k inputs $C_1 = b_1^1 \cdots b_n^1$, $C_2 = b_1^2 \cdots b_n^2$, ..., $C_l = b_1^l \cdots b_n^l$ which are applied to the k individual Arb-PUFs. Contrary to XOR Arb-PUFs, it does not hold that $C_1 = C_2 = \ldots = C_k = C$, but a more complicated input mapping that derives the individual inputs C_i from the global input C is applied. This input mapping constitutes the most significant difference between the Lightweight PUF and the XOR Arb PUF. We refer the reader to [14] for further details.

2.2 Previous Attacks on Delay-Based PUFs

In modeling attacks, an adversary collects a subset of all CRPs of the PUF, and uses them to predict or extrapolate the behavior of the PUF in the entire CRP space. This attack form can essentially only be applied to Strong PUFs, since they have many CRPs (only then extrapolation makes sense!), and since they have a publicly accessible CRP interface (meaning that everyone with physical access to the PUF or PUF embedding hardware can collect CRPs) (compare [28, 30]).

Usually, machine learning (ML) algorithms are employed as the weapon of choice for CRP prediction or CRP extrapolation, even though other techniques have been used at times, such as linear programming [19] or solving integer equations [2]. It is useful for ML performance to use a parametrized model of the PUF in this process. For Arbiter PUF, the standard a parametric model is the *"linear additive delay model"* [12, 28, 30]. It uses the delays in the Arbiter PUF's circuit elements as unknown variables, and sums up these delays to compute the overall delay and thus the PUF response at the end of the structure.

It has been shown in IEEE T-IFS 2013 [30] by comparison of ML performance on silicon and simulated CRP data that this model is very exact, and that it can be used to

PUF	Bit Length	Pred. Rate	No. of XORs	CRPs ($\times 10^3$)	Training Time
Arbiter PUF	64	99%	—	2.5	0.13 sec
	128			5.5	0.51 sec
XOR Arb PUF	64	99%	4	12	3:42 min
			5	80	2:08 hrs
			6	200	31:01 hrs
	128	99%	4	24	2:52 hrs
			5	500	16:36 hrs
			6	—	—
Lightweight PUF	64	99%	4	12	1:28 hrs
			5	300	13:06 hrs
			6	—	—
	128	99%	4	500	59:42 min
			5	1000	267 days
			6	—	—

Table 1: The performance of Logistic Regression on Arbiter PUFs, XOR Arbiter PUFs, and Lightweight PUFs, taken from Rührmair et al. [28]. The results were obtained on simulated CRPs generated by the additive linear delay model [12, 28, 30].

accurately examine the ML resilience of any Arbiter PUF variants, as the ones considered in this paper. The authors of IEEE T-IFS [30] also give a detailed discussion on the role of PUF-noise in simulated CRPs ([30], Section II-G). They argue why simulated, noise free CRPs, as the ones examined in this paper, are a good measure and benchmark for the ML-resilience of a PUF. In opposition to the occuring noise level, they are independent of the concrete implementation and environment, and hence indicate the *intrisic* hardness of a design. Furthermore, they represent the upper limit of the ML hardness of a given design – recall that error correction will usually be exectued on the PUF, and that this error correction cuts of information and ML-hardness from the PUF's output. Finally, physical adversarial access to the PUF is part of the established attack model for Strong PUFs. During this access period, the attacker can control the environmental parameters, and execute multiple measurements with majority voting. This can help him to collect CRP sets with very small error levels, as executed in [30]. Once more, this justifies the use of error-free synthetic CRPs as an "upper bound case" for ML experiments. For further details, please see [30].

While various PUF-models and ML algorithms have been applied to Arbiter PUFs and variants in the literature [12, 15, 19, 28], the strongest results up to this point were achieved by use of a special variant of Logistic Regression (LR) [28, 30]. For comparison with our findings, we summarize these known results from CCS 2010 [28] in Table 1.

The strong growth of the required CRPs and computation times for large numbers of XORs is an important aspect of the above data. The authors of [28] concluded that the growth is exponential in the number of XORs. At the same time, also the output instability of XOR Arbiter PUFs and Lightweight PUFs increases exponentially in the number of their XORs. This implies that there is a certain stability limit up to which they can be implemented in practice. In [28, 30], this stability limit was estimated around eight XORs. This is just beyond the reach of the current ML methods, which is five or six XORs as shown in Table 1.

This poses the question if and how XOR-based PUF architectures can be attacked up to this estimated limit in practical use cases (or even beyond). Note again that the attacker – by applying special environmental control and majority voting etc. – may be able to obtain CRP sets with less noise than the noise level of practical use cases, in which the environmental conditions may vary strongly (see discussion above and in [30], Section II-G).

The use of current ML algorithms alone had not been successful to this end [28, 30]. We now tackle this goal in this paper for the first time by a combined use of modeling and side channel attacks.

3. POWER CONSUMPTION AS A SIDE CHANNEL FOR ARBITER PUFS

The basic concept of our power side channel (PSC) is to apply power tracing to determine the transition from zero to one of the latches (=arbiter elements) that are part of Arbiter PUF based architectures. The power tracing technique is based on measuring the amount of current drawn from the supply voltage during any latch transition to one.

In order to lead a proof-of-concept for our approach, we implemented a SPICE simulation that uses only one latch with three different outputs loading (floating output, output connected to one gate, and output connected to four gates). Figure 2 illustrates our results, and shows the different amount of current drawn for the three different output loading. The reason for having different values for the different loading is that an additional amount of charges is required to charge the capacitance of each gate. Hence, the amount of drawn charges, which is the integration of the current curve, is linearly proportional with the number of gates. Taking into consideration, the amount of charges normally drawn in case of a floating load should be subtracted. Consequently, for extreme cases when all latches' output in the device are zeros or all are ones, this power tracing technique would allow the attacker to determine these cases very easily.

By applying this idea to XOR Arbiter PUFs and Lightweight PUFs, which utilize more than one single Arbiter PUF in parallel, one could determine the exact number of ones (and zeros) stored in the latches (arbiters) within the PUF. Following our above discussion, the power consump-

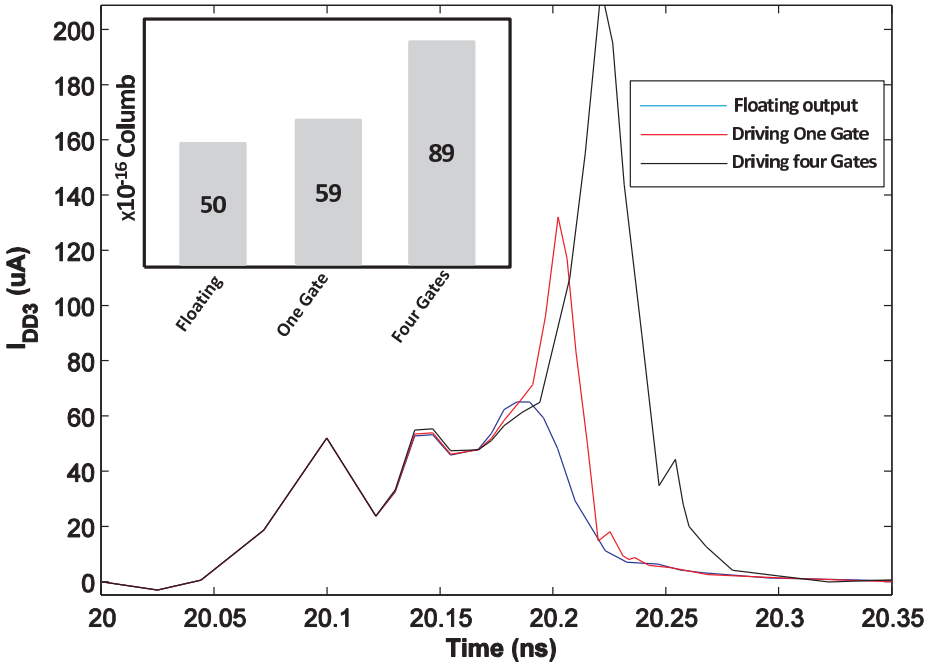

Figure 2: The Power Tracking SCA for a latch that had a transition to 1, with different driving loads. The inset is the amount of drawn charges, which is calculated from the area under each curve. The amount of charges is linearly proportional with the number of gates, noting that the amount of charges normally drawn for a floating load should be subtracted.

tion will tell us the (cumulative) number of zeros and ones that are stored in the latches (or, in other words, the cumulative number of zeros and ones that is output by the single Arbiter PUFs before the large XOR gate). Please note, however, that we cannot derive which of the single Arbiter PUFs in an XOR Arbiter PUF have output one and which have output zero. We only know the overall number of zeros and ones before the XOR. Taken by itself, the side channel thus appears relatively worthless. We will nevertheless show that this basic information can boost PUF attacks significantly if it is used in the right manner.

4. ATTACKING LIGHTWEIGHT PUFS

Let us start by illustrating how combined power side channels and modeling can attack Lightweight PUFs.

4.1 Attack Strategy

As mentioned in the last section, our power side channel only tells us the cumulative number of zeros and ones before the XOR element. If we knew the exact output of each single Arbiter element before the XOR, this would be much more helpful: We could then machine learn each single Arbiter PUF separately. According to [28], this is possible with very high prediction rates and very few CRPs per Arbiter PUF. Once we have learned every single Arbiter PUF with very high accuracy, we could also predict the output of the entire structure exactly.

One way to resolve this problem is the following: Suppose that we want to break a Lightweight PUF with k parallel single Arbiter PUFs, and that we have the power side channel described in Section 3 available. We can then start collecting CRPs of the Lightweight PUF, using our power side channel for each collected CRP. We then simply wait for those "good" CRPs where the side channel tells us that either *all* k single Arbiter outputs were zeros, or *all* k single Arbiter outputs were one. We use only the "good" CRPs for machine learning, throwing away the others: As we know the output of each single Arbiter PUF for each of the good CRPs, we can apply the known ML techniques for learning each single Arbiter PUF separately on the basis of the good CRPs. Once we can predict each single Arbiter PUF with high accuracy, we can also predict the entire structure.

In order to decide if this strategy is viable, the critical question is: How frequent are the good CRPs, i.e., how often is it the case that all single Arbiter outputs are all zero or all one? Assuming an equally distributed output, this event will occur for a fraction of $2 \cdot \frac{1}{2^k} = 2^{k-1}$ cases, where k is the number of XORs. But note that around eight XORs are the current stability limit, whence k is relatively small in absolute terms, and so is the expression 2^{k-1}. This makes our strategy applicable in practice. Exact CRP requirements and prediction rates are given in the next section.

4.2 Results on Synthetic CRP Data

Table 4.2 shows the results of applying our technique to synthetic CRPs generated by the additive linear delay model [28]. [2] We waited for the cases where the single outputs are all zero or all one. The CRP numbers given in the table include all CRPs necessary for the attack, also those that

[2]As before, we stress that synthetic CRPs can be used well to evaluate ML performance on Arbiter PUF variants, as shown by the results of [30].

are not "good", and which are not used for training the ML algorithm. Please note that the collection of the required number of CRPs poses no problem in practice, since Arbiter PUFs operate at MHz CRP frequencies [11]; collecting the CRPs takes hence only few seconds or less.

Comparing our results to the success of ML attacks without side channels (Table 1), a strong performance gain in terms of computation times and reachable bitlengths of the PUF can be observed. It would even be possible to go strongly beyond nine XORs with still very small computational times by using enough CRPs.

Bit Length	Pred. Rate	No. of XORs	CRPs ($\times 10^3$)	Training Time
64	97.3%	4	40	18 sec
	96.6%	5	80	22 sec
	96.2%	6	200	44 sec
	96.8%	7	500	91 sec
	96.3%	8	1,000	98 sec
	96.0%	9	2,000	105 sec
128	97.4%	4	80	46 sec
	97.6%	5	200	122 sec
	97.4%	6	500	191 sec
	96.8%	7	1,000	200 sec
	96.4%	8	2,000	238 sec
	96.1%	9	4,000	303 sec
256	97.6%	4	200	200 sec
	97.4%	5	500	340 sec
	96.8%	6	1,000	400 sec
	96.4%	7	2,000	490 sec
	96.1%	8	4,000	580sec
	96.1%	9	8,000	700 sec

Table 2: The performance of LR plus power side channels on Lightweight PUFs. The training times are averaged over different PUF instances.

5. ATTACKING XOR ARBITER PUFS

5.1 Attack Strategy

Let us now discuss the effect of our side channel on the security of the (standard) XOR Arbiter PUF. It seems natural to transfer the approach of the last section to XOR Arbiter PUFs: Simply wait for the "good" CRPs, and use them to machine learn each single Arbiter PUF separately.

Rather unexpectedly, this straightforward approach fully fails in the case of XOR Arb PUFs. The reason is that in a XOR Arb PUF with k parallel single Arbiter PUFs, the input challenge C is directly applied to all of the k single Arbiter PUFs without being changed. [3] In other words, the very same challenge C is applied at all k single arbiter PUFs. The "good" CRPs collected for the single Arbiter PUFs hence all have the same input challenge C. Furthermore, by definition, the good CRPs all have the same output for the k single Arb PUFs. The collection of good CRPs thus leads to a collection *of exactly the same CRPs* for all k single Arb PUFs. If these CRPs are fed into a machine learning algorithm, it develops exactly the same model for all of the k single Arb PUFs, and predicts the very same outputs for all k single Arbiter PUFs (while in reality, they are, of course, different). The morale here is that symmetry in the design (such as in the XOR Arbiter PUF) can sometimes increase security against side channels.

This leads to the question whether there are other fruitful combinations of power side channel information with ML techniques. This turned out to be a very hard problem. After investigating several potential alternative methods (and failing with them), we finally discovered an effective technique.

Let us start by comparing how logistic regression (and other supervised ML methods) perform on Arbiter PUFs and on XOR Arbiter PUFs. The standard arbiter PUF is relatively easy to learn for two reasons: First of all, the model of the Arbiter PUF is simple (linear additive delay model [12]). Secondly, when we compare the response of the modeled PUF to the responses of the original PUF we have two cleanly separated cases: (i) The response is correct, leading to a positive feedback for the ML algorithm. (ii) The response is not correct, leading to a negative feedback. The model parameters are changed in this case, and iteratively converge to a correct solution.

For XOR Arbiter PUFs, this picture changes strongly, as the final XOR gate "masks" the output of the single Arbiter PUFs. Assume as an example that $k = 5$. If the output of such a 5-XOR Arb PUF is a "one", there are three possible cases:

(i) There is exactly one single Arbiter PUF that has output one, and the remaining four have output zero.

(ii) There are exactly three single Arbiter PUFs that have output one, and two have output zero.

(iii) All five single Arbiter PUFs have output one.

In a classical XOR Arbiter PUF, we cannot distinguish between the three cases (i.e., we would not know the overall number of zeros and ones); and even if we knew, we would not know exactly which of the single Arbiter PUFs output a one and which a zero. The second problem remains, but the first problem can directly be solved via power side channel information. This leads to a very strongly improved ML convergence, as shown in the next sections. It helps us to tackle XOR Arbiter PUFs with extremely reduced computation times.

5.2 Results on Synthetic CRP Data

Table 5.2 shows the error ratio and the time taken for our attack on XOR arbiter PUFs explained in the previous subsection. Although we can in principle go further for higher number of XORs, there is a memory limitation problem that we still must to overcome. The reason is that the algorithm needs to work with the entire CRPs matrix which overfloats the memory limitation. This problem did not appear when attacking Lightweight PUFs, since we then initially divided the employed CRPs. We filtered them, using just the "good" CRPs in the learning process. In opposition to this, we needed to employ all the CRPs during the learning process for attacking XOR Arbiter PUFs. Experiments are on the way in our group to circumvent this problem by

[3] Please note that this is in opposition to the Lightweight PUF, where an input mapping is applied to the input challenge C. In this process k different challenges $C_1, \ldots, C_k$ are generated, which are then used as the k different inputs of the single Arb PUFs.

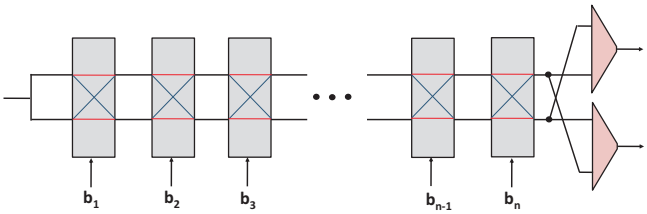

Figure 3: Standard Arbiter PUF with a differential output. Two arbiters are employed to generate two differential response bits, which have inverted input signals. The architecture would prevent the use of power tracking SCA. Asides, it also provides new error detection and correction capabilities.

hardware or software, and to fully unleash the power of our attack strategy with respect to XOR Arbiter PUFs. Table 5.2 gives some first indication of how strongly the training times decrease, compared to the pure machine learning attacks shown in Table 1.

Bit Length	Pred. Rate	No. of XORs	CRPs ($\times 10^3$)	Training Time
64	95%	4	40	10 sec
		5	80	37 sec
		6	200	160 sec
		7	500	247 sec
128	95%	4	80	34 sec
		5	200	170 sec
		6	500	374 sec

Table 3: The performance of LR plus power side channels on XOR Arbiter PUFs. The training times are averaged over different PUFs-instances.

6. COUNTERMEASURES

In order to immunize arbiter-based PUFs against power tracking SCA, one could add an additional, symmetric arbiter at the last stage of the PUF. The input signal for the added arbiter is inverted before it enters the arbiter (see Figure 3).

This idea behind the architecture is obviously to keep the number of zeros and ones in the entire PUF architecture constant, thereby preventing power tracking side channel attacks. A differential output of the described kind could also be desirable in fault tolerant applications, since it might allow error correction, and indicate unstable responses. Recall that the arbiter (=latch element) typically is one of the main sources of instability in an arbiter PUF: Signal pathes with a runtime difference that is on the order of the switching time of the latch often cause unstable responses. If the two differential latches have inconsistent content, this indicates the occurence of an unstable PUF response.

Use of k latches in such a differential design, together with standard majority voting over the latches' outputs, could be used as a simple and very efficient means for response stabilization and error correction.

7. SUMMARY AND CONCLUSIONS

In this paper, we investigated the reach of combined power side channel (PSC) and modeling attacks on electrical Strong PUFs architectures. We focused on XOR Arbiter PUFs and Lightweight PUFs, which are presumably considered the most secure electrical Strong PUF architectures [28, 30], at the least among the class of delay-based PUFs. Previous examinations had shown that these two PUFs could only be attacked successfully for around five XORed single Arbiter PUFs, and for bitlengths of 64 bits or 128 bits [28, 30]. This implied that XOR Arbiter PUFs or Lightweight PUFs with bitlenghts of 64, 128 or 256 bits and eight single Arbiter PUFs had to be considered both practical (i.e., sufficiently stable) and secure previous to our work [28, 30].

The new side channel we suggested and examined in this work is power tracing of the arbiter element (i.e., the latch) in Arbiter PUFs and variants thereof, such as the XOR Arbiter PUF and Lightweight PUF. We led a proof of concept by SPICE simulations for the viability of this approach. The power side channel (PSC) tells us the cumulative number of zeros and ones before the XOR gate. Taken by itself, our PSC is almost worthless, since the attacker does not learn *which* of the single Arbiter PUF outputs is zero or one. If combined with machine learning based modeling techniques, it can strongly boost performance, however.

First of all, it allowed us to tackle Lightweight PUFs with up to nine XORs and bitlengths of up to 256 bits (see Table 4.2). Our strategy was to wait for the "good" CRPs in which either *all* outputs of the single Arbiter PUFs are a zero, or *all* these outputs are a one. In these special cases, the ML problem of attacking Lightweight PUFs reduces to the problem of machine learning single Arbiter PUFs – which can be done extremely efficiently both on simulated and silicon data, as shown earlier [28, 30]. The described strategy leads to a high, even exponential CRP consumption in the number of XORs (since the good cases are exponentially rare), but to very small computation times and prediction errors. For the comparably small number of XORs used in practical Lightweight PUF architectures, the exponential number of required CRPs does not hinder the attacks and is still quite practical. Recall that implementations of Arbiter PUF variants would work at MHz CRP frequencies [11], whence the collection of the maximally required few millions of CRPs could be carried out in relatively short time. Furthermore, the bitlengths of the Lightweight PUFs no longer provides a good handle for increasing ML resilience under out attacks. This previously was the case with pure modeling attacks, as pointed out in [28].

The second examined case of XOR Arbiter PUFs turned out to be far more complicated. This was surprising, since Lightweight PUFs had proven more resilient to classical modeling attacks in previous works than XOR Arbiter PUFs [28, 30]. Their special "input mapping", by which they derive the challenges applied to each single Arbiter PUF, had complicated earlier machine learning attempts. With respect to our power side channel, however, it turns out that this input mapping makes attacks easier: It has the effect that all the "good" CRPs (see last paragraph) of the Lightweight PUF have different challenges, and can be used without further ado in machine learning algorithms. Due to the symmetric design of XOR Arbiter PUFs, however, all the "good" CRPs of XOR Arbiter PUFs have exactly the same challenges, and, by definition, also the same responses. They

hence do not differentiate between all the different single Arbiter PUFs within the structure! This prevents the above direct application of our above technique to XOR Arbiter PUFs. The morale here is that symmetry sometimes can be a useful design principle against PSCs. We thus developed a new strategy, in which the collected side channel information about the number of zeros and ones is fed directly into the used machine learning algorithm. It turned out that this can drastically improve the computation times of LR on XOR Arbiter PUFs compared to attacks that solely use ML techniques without side channel information (compare Tables 5.2 and 1).

Finally, we suggested a simple new differential architecture for the arbiter PUF as a countermeasure that could potentially immunize PUFs against the power tracking SCA. It consists of using two symmetric, inverted output signals with two latches, and essentially re-iterates our above credo that design symmetry can be helpful against PSC attacks. It is an important outcome of our work that this simple, but efficient countermeasure should be used in future PUF designs in order to prevent our attack. Asides, it can also be used to detect and stabilize output errors, even though we did not follow this route in detail in this paper.

The PUF-attacks presented in our and other recent papers could be seen as a natural consolidation process in the PUF area, similar to the detailed investigations that classical cryptoprimitives and security systems have already undergone in the last decades. We believe that this interplay between attacks and countermeasures will in the long term be to the benefit of PUFs. Similar as in the case of classical systems, it will further improve implementation security step by step, and will presumably lead to secure PUF constructions in the end.

8. REFERENCES

[1] Christina Bruzska, Marc Fischlin, Heike Schröder, Stefan Katzenbeisser: *Physically Unclonable Functions in the Universal Composition Framework.* CRYPTO 2011.

[2] Jeroen Delvaux, Ingrid Verbauwhede: *Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise.* HOST 2013.

[3] Jeroen Delvaux, Ingrid Verbauwhede: *Fault Injection Modeling Attacks on 65nm Arbiter and RO Sum PUFs via Environmental Changes.* IACR Cryptology ePrint Archive, Report 2013/619.

[4] Srinivas Devadas: *Physical unclonable functions and secure processors.* Invited talk, Workshop on Cryptographic Hardware and Embedded Systems (CHES 2009), September 2009.

[5] Marten van Dijk: *System and method of reliable forward secret key sharing with physical random functions.* US Patent No. 7,653,197, October 2004.

[6] Blaise Gassend, Dwaine Clarke, Marten van Dijk, Srinivas Devadas: *Silicon physical random functions.* ACM Conference on Computer and Communications Security 2002: 148-160

[7] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten v. Dijk, Srinivas Devadas: *Identification and authentication of integrated circuits.* Concurrency and Computation: Practice & Experience, pp. 1077 - 1098, Volume 16, Issue 11, September 2004.

[8] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, Pim Tuyls: *FPGA Intrinsic PUFs and Their Use for IP Protection.* CHES 2007: 63-80

[9] Clemens Helfmeier, Dmitry Nedospasov, Christian Boit, Jean-Pierre Seifert: *Cloning Physically Unclonable Functions.* IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'13), 2013.

[10] Gabriel Hospodar, Roel Maes, Ingrid Verbauwhede: Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability. WIFS 2012: 37-42

[11] J.-W. Lee, Daihyun Lim, Blaise Gassend, G. Edward Suh, Marten v. Dijk, and Srinivas Devadas. *A technique to build a secret key in integrated circuits with identification and authentication applications.* In Proceedings of the IEEE VLSI Circuits Symposium, June 2004.

[12] Daihyun Lim: *Extracting Secret Keys from Integrated Circuits.* MSc Thesis, MIT, 2004.

[13] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, Pim Tuyls: *The Butterfly PUF: Protecting IP on every FPGA.* HOST 2008: 67-70

[14] Mehrdad Majzoobi, Farinaz Koushanfar, Miodrag Potkonjak: *Lightweight Secure PUFs.* IC-CAD 2008: 607-673.

[15] Mehrdad Majzoobi, Farinaz Koushanfar, Miodrag Potkonjak: *Testing techniques for hardware security.* In Proceedings of the International Test Conference (ITC), pages 1-10, 2008.

[16] Dominik Merli, Dieter Schuster, Frederic Stumpf und Georg Sigl: *Side-Channel Analysis of PUFs and Fuzzy Extractors.* Conference on Trust and Trustworthy Computing (TRUST 2011). Lecture Notes in Computer Science, 2011, Volume 6740/2011, 33-47.

[17] Dominik Merli, Dieter Schuster, Frederic Stumpf, Georg Sigl: *Semi-invasive EM attack on FPGA RO PUFs and countermeasures.* ACM Workshop on Embedded Systems Security (WESS'11), 2011.

[18] Dmitry Nedospasov, Clemens Helfmeier, Jean-Pierre Seifert, Christian Boit: *Invasive PUF Analysis.* Fault Diagnonsis and Tolerance in Cryptography (FDTC'13), 2013.

[19] Erdinc Öztürk, Ghaith Hammouri, Berk Sunar: *Towards robust low cost authentication for pervasive devices.* In PerCom, pages 170-178. IEEE Computer Society, 2008.

[20] Ravikanth Pappu: *Physical One-Way Functions.* PhD Thesis, Massachusetts Institute of Technology, 2001.

[21] Ravikanth Pappu, Ben Recht, Jason Taylor, Neil Gershenfeld: *Physical One-Way Functions*, Science, vol. 297, pp. 2026-2030, 20 September 2002.

[22] Ulrich Rührmair: *Oblivious Transfer based on Physical Unclonable Functions (Extended Abstract).* TRUST 2010.

[23] Ulrich Rührmair, Marten van Dijk: *Practical Security Analysis of PUF-based Two-Player Protocols.* CHES 2012.

[24] Ulrich Rührmair, Marten van Dijk: *On the Practical Use of Physical Unclonable Functions in Oblivious*

Transfer and Bit Commitment Protocols. Journal of Cryptographic Engineering (JCEN), 2013.

[25] Ulrich Rührmair, Marten van Dijk: *PUFs in Security Protocols: Attack Models and Security Evaluations*. IEEE Symposium on Security and Privacy (Oakland'13), 2013.

[26] Ulrich Rührmair, Srinivas Devadas, Farinaz Koushanfar: *Security based on Physical Unclonability and Disorder*. In M. Tehranipoor and C. Wang (Editors): "Introduction to Hardware Security and Trust". Springer, 2011.

[27] Ulrich Rührmair, Christian Jaeger, Michael Algasiner: *An Attack on PUF-based Session Key Exchange, and a Hardware-based Countermeasure: Erasable PUFs.* Financial Cryptography and Data Security 2011.

[28] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, Jürgen Schmidhuber: *Modeling Attacks on Physical Unclonable Functions*. ACM Conference on Computer and Communications Security, 2010.

[29] Ulrich Rührmair, Jan Sölter, Frank Sehnke: *On the Foundations of Physical Unclonable Functions.* Cryptology e-Print Archive, June 2009.

[30] Ulrich Rührmair, Jan Sölter, Frank Sehnke, Xiaolin Xu, Ahmed Mahmoud, Vera Stoyanova, Gideon Dror, Jürgen Schmidhuber, Wayne Burleson, Srinivas Devadas: *PUF Modeling Attacks on Simulated and Silicon Data*. IEEE Transactions on Information Forensics and Security (IEEE T-IFS), 2013.

[31] G. Edward Suh, Srinivas Devadas: *Physical Unclonable Functions for Device Authentication and Secret Key Generation.* DAC 2007: 9-14

[32] Pim Tuyls, Geert Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, Rob Wolters *Read-Proof Hardware from Protective Coatings.* CHES 2006: 369-383