Efficient Power and Timing Side Channels for Physical Unclonable Functions

Ulrich Rührmair[1,⋆], Xiaolin Xu[1,⋆], Jan Sölter[3], Ahmed Mahmoud[1], Mehrdad Majzoobi[4], Farinaz Koushanfar[4], and Wayne Burleson[2]

[1] Technische Universität München, 80333 München, Germany
[2] University of Massachusetts Amherst, Amherst, MA 01003, USA
[3] Freie Universität Berlin, 14195 Berlin, Germany
[4] Rice University, Houston, TX 77005, USA
ruehrmair@in.tum.de, xiaolinx@umass.edu, jan_soelter@yahoo.com
ahmed.mahmoud@tum.de, m.majzoobi@gmail.com
fk1@rice.edu, burleson@umass.edu

Abstract. One part of the original PUF promise was their improved resilience against physical attack methods, such as cloning, invasive techniques, and arguably also side channels. In recent years, however, a number of effective physical attacks on PUFs have been developed [17,18,20,8,2]. This paper continues this line of research, and introduces the first power and timing side channels (SCs) on PUFs, more specifically on Arbiter PUF variants. Concretely, we attack so-called XOR Arbiter PUFs and Lightweight PUFs, which prior to our work were considered the most secure members of the Arbiter PUF family [28,30]. We show that both architectures can be tackled *with polynomial complexity* by a combined SC and machine learning approach.

Our strategy is demonstrated in silicon on FPGAs, where we attack the above two architectures for up to 16 XORs and 512 bits. For comparison, in earlier works XOR-based Arbiter PUF designs with only up to 5 or 6 XORs and 64 or 128 bits had been tackled successfully. Designs with 8 XORs and 512 bits had been explicitly recommended as secure for practical use [28,30].

Together with recent modeling attacks [28,30], our work shows that unless suitable design countermeasures are put in place, no remaining member of the Arbiter PUF family resists all currently known attacks. Our work thus motivates research on countermeasures in Arbiter PUFs, or on the development of entirely new Strong PUF designs with improved resilience.

Keywords: Physical unclonable functions (PUFs), side-channel attacks, power side channel, timing side channel, modeling attacks, machine learning, hardware security.

1 Introduction

One part of the original PUF promise was their improved resilience against many classical attack forms, in particular physical attacks. This included cloning, invasive techniques, and arguably also side channels (SC). Regarding the latter, recall that Strong

⋆ These two authors contributed equally.

L. Batina and M. Robshaw (Eds.): CHES 2014, LNCS 8731, pp. 476–492, 2014.

PUF based identification schemes [22] do not require a standard key that is processed bit by bit, a fact that arguably led to hopes about improved SC resilience within the community.

Recent years have put these assumptions to the test, but sometimes with a negative outcome. Let us start with non-physical attacks: Firstly, machine learning (ML) based modeling attacks have proven a more efficient threat than originally assumed. When the first of these attacks were put forward in 2004 [9], it was supposed that they could be thwarted by adding simple non-linear elements to Arbiter PUF designs, for example XOR gates or feed-forward loops. However, by improved ML algorithms, Rührmair et al. in 2010 and 2013 [28,30] also tackled XOR-based Arbiter PUFs up to 64 or 128 bits and 5 XORs, and Feed-Forward Arbiter PUFs up to essentially arbitrary sizes. As a second, non-physical attack form, PUF protocol attacks have been devised in recent years. Since they are not in the focus of this work, we refer interested readers to the literature on this topic [26,25].

Also dedicated physical attacks on PUFs have been devised lately. For example, the physical unclonability of PUFs, one of their core properties, has been investigated more closely. It is obvious that complex three-dimensional objects like PUFs cannot be cloned atom by atom by current fabrication technology. Generating a *perfect clone* thus to date is infeasible. However, *functional clones* are easier to construct, i.e., PUFs that merely agree with the original in their challenge-response behavior. In a breakthrough effort, Helfmeier et al. [8] in 2013 were indeed able to functionally clone SRAM PUFs by tuning the power-up states of SRAM cells. Soon after, invasive attacks on SRAM PUFs have been presented by Nedospasov et al. [20] in 2013. The authors apply semi-invasive, single-trace, backside readout of logic states to to obtain the responses of SRAM PUFs. This compromises any secret keys that would be derived from these responses.

Around the same time, first side-channel attacks on PUFs have been investigated. In 2011, Merli et al. [17] demonstrated SC attacks on the error correcting (EC) module of PUFs. Their attack is indirect in the sense that it does not target the PUF itself, but a specific EC module of the PUF, working only for certain modules. Furthermore, Merli et al. reported electromagnetic analyses on ring oscillator PUFs in two consecutive works in 2011 and 2013 [18,19]. Also in 2013, Delvaux et al. [2] exploited the instabilities of Arbiter PUF responses as side channel, implementing an idea originally suggested by Rührmair et al. in [28]. While the work of Delvaux et al. is quite fascinating due to the fact that it does not use any machine learning algorithms, it must be said that it performs slightly worse than pure machine-learning based modeling *without* side channels [28,30,2].

We continue this line of research, and introduce in this paper the first power and timing side channel attacks on PUFs. Our approach constitutes one of the first physical attacks on Strong PUFs [24,27,29] that can *notably increase* attack performance in comparison with existing, non-physical methods, specifically with pure modeling attacks [28,30].

In greater detail, we devise power and timing SCs for XOR Arbiter PUFs and Lightweight PUFs that provide the adversary with information about the *cumulative* number of zeros and ones in the outputs of the k parallel Arbiter PUFs before the XOR

gate. We then adapt existing machine learning (ML) techniques to efficiently exploit this information. This "hybrid" attack form can tackle XOR Arbiter PUFs and Lightweight PUFs with a *polynomial complexity* in their number of XORs, bitlengths, and number of required CRPs, while pure modeling attacks on these two PUFs have *exponential complexity* [28,30]. We provide a full proof of concept on FPGAs, attacking XOR Arbiter PUFs and Lightweight PUFs for up to 16 XORs and 512 bits. Comparably large sizes of these two PUFs had hence never been realized before in silicon; in earlier works, already XOR Arbiter PUFs with 8 XORs and 512 bits had been explicitly suggested as secure [28,30].

Organization of this Paper Section 2 provides the necessary background and methodology. Sections 3 and 4 describe the design and implementation of our power and timing side channels, respectively. Section 5 details our adaptation of logistic regression to incorporate SC information. Section 6 lists silicon results on FPGA implementations and provides an asymptotic peformance analysis. We conclude the paper in Section 7.

2 Background, Methodology, and Definitions

Background on XOR Arbiter PUFs and Lightweight PUFs. Together with SRAM PUFs, the Arbiter PUF family [7,31] is arguably the best studied PUF design, and also the most popular implementation of so-called "Strong PUFs" [24,27]. Nevertheless, a large number of its members have been attacked successfully by so-called modeling attacks in recent works [28,30]. The currently *only* remaining Arbiter PUF variants which partly resist modeling, since they cause exponential modeling efforts (i.e., exponential training times of the ML algorithm), were so-called XOR Arbiter PUFs [9,31] and Lightweight PUFs [11].

In an XOR Arbiter PUF, k Arbiter PUFs are used in parallel, and the same, multi-bit challenge is applied to all of them. The final, one-bit response is defined as the XOR of all the parallel k outputs [9,31]. In a Lightweight PUF [11,28], again k Arbiter PUFs are used in parallel, but different challenges $C^1, \ldots, C^k$ are applied to them, all of which are generated by some "input mapping" from a single, global challenge C (see [11] for the details of the mapping). The k outputs of the single Arbiter PUFs are used (without error correction) as input to a postprocessing function, which XORs subsets of them together in order to produce an m-bit output string (see again [11] for details). From a machine learning and modeling perspective, the optimal bit security is achieved if *all* of the k outputs are XORed to produce a *single bit output* [28,30]. Therefore earlier works [28,30] focused exactly on this case and on this special architecture of the Lightweight PUF, and so do we in this paper. If nothing else, this evaluates the maximally achievable bit security in a Lightweight PUF architecture. Using the same Lightweight PUF variant as [28,30] also allows a fair comparison with our results.

FPGA Implementations. We implemented the above XOR Arbiter PUFs and Lightweight PUFs on Xilinx Spartan-6 FPGAs. In order to balance FPGA routing asymmetries, a lookup table (LUT) based programmable delay line (PDL) has been implemented [13,10,15]. This is the standard approach for realizing Arbiter PUFs on

FPGAs, and ensures a balanced output between zeros and ones in each single Arbiter PUF. For each CRP, majority voting over five repeated measurements of the response to the same challenge was performed in order to determine the final response. The challenges were generated by an n-bit maximal-length linear feedback shift register (LFSR) with polynomial $f = 1 + x^1 + x^3 + x^4 + x^{64}$.

Machine Learning Definitions and Computational Resources. Following [28,30], we use the following definitions throughout the paper: The prediction error ϵ is the ratio of incorrect responses of the trained ML algorithm when evaluated on the test set. The prediction rate is $1 - \epsilon$. For all ML experiments throughout this paper, each test set consisted of 10,000 randomly chosen CRPs. The term N_{CRP} (or simply *"CRPs"*) denotes the number of CRPs employed in an attack, i.e., the size of the training set. We used an Intel Xeon X5650 processor at 2.67GHz with 48 GB of RAM in all of our ML experiments, having a value of a few thousand Euros. All computation times (= "training times") are calculated for one core of one processor of this hardware.

3 Power Side Channels on XOR-Based Arbiter PUFs

3.1 Basic Idea of the Power Side Channel

Currently known pure modeling attacks on XOR-based Arbiter PUFs require training times of the ML algorithm that are exponential in the number of XORs [28,30]. This makes it difficult to tackle XOR-based Arbiter PUFs with more than five or six single parallel Arbiter PUFs, and with bitlengths longer than 128, by pure modeling attacks [28,30]. XOR-based Arbiter PUF architectures are therefore the currently most secure designs from the Arbiter PUF family. Our side-channel attacks now take a novel route: They gain additional information from the physical implementation of XOR-based Arbiter PUFs, and use this information to improve the ML computation times (i.e., training times) from exponential to polynomial.

One straightforward power side channel is to apply power (i.e., current) tracing to determine the transition from zero to one of the latches (i.e., the arbiter elements) in the single Arbiter PUFs. The power tracing is based on measuring the amount of current drawn from the supply voltage during any latch transition to one. We implemented a first SPICE simulation to validate this approach, and to verify the power consumption of an arbiter circuit with different loading outputs. Only one latch (i.e., arbiter circuit) is used in the simulation, but with three different outputs loading scenarios (i.e., floating output, output connected to one gate, and output connected to four gates). Figure 1 illustrates the results, and shows the different amount of current drawn for the three different output loading scenarios. The reason for having different values for the different loadings 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 loading gates. Taking this phenomenon into consideration, the amount of charges normally drawn in case of a floating load should be subtracted.

In XOR-based architectures with k parallel single Arbiter PUFs, the current that is drawn *in sum* and *altogether* in principle tells the (cumulative) number of latches that

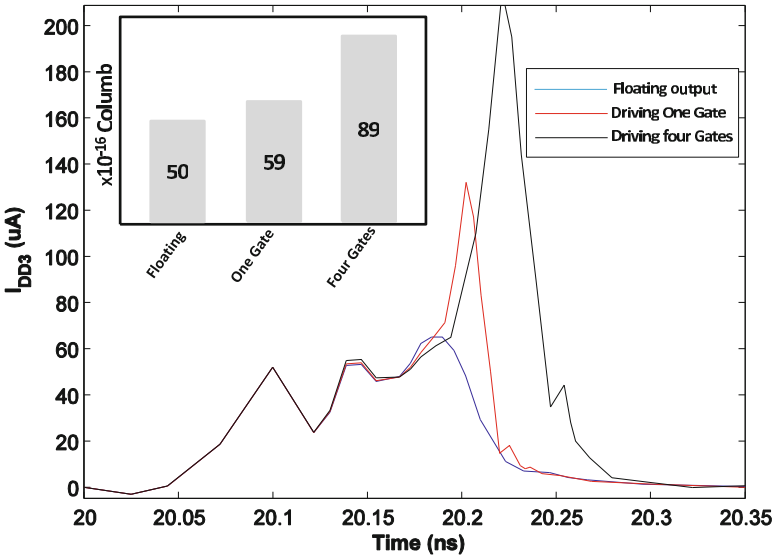

Fig. 1. The power tracking side-channel analysis for a latch that had a transition to 1, with different driving loads, in SPICE simulation. The inset is the amount of drawn charges, which is calculated from the area under each curve. The amount of charges is linearly proportional with the number of gates. The amount of charges normally drawn for a floating load should be subtracted.

are zero, and the (cumulative) number that are equal to one. Please note, however, that it does *not* tell us *which* of the k parallel Arbiter PUFs had *which* output. If it did, CRPs from every single Arbiter PUF could be collected, and every single Arbiter PUF could be machine learned separately. As this is not possible, a more complicated strategy is required, in particular a way to exploit the cumulative number of zeros and ones beneficially in the ML process, as detailed in Section 5. But before we move on to the details of the ML process, we discuss the exact implementation of the side channels in this and the next section.

3.2 Practical Implementation of the Power Side Channel

Measurement Noise. To further validate the practicality of our power SC, we had to move beyond the simplifications of SPICE simulations, most notably the absence of supply and measurement noise and real process variations. We extracted the power trace of 30 sub-response patterns from Lightweight PUFs on FPGA (see Figure 2). However, we found that the 30 power traces are difficult to be differentiated from each other (as are their power consumptions). In other words, in practical implementations, a straightforward identification of the desired power side channel information from the measured power (current) traces appears infeasible.

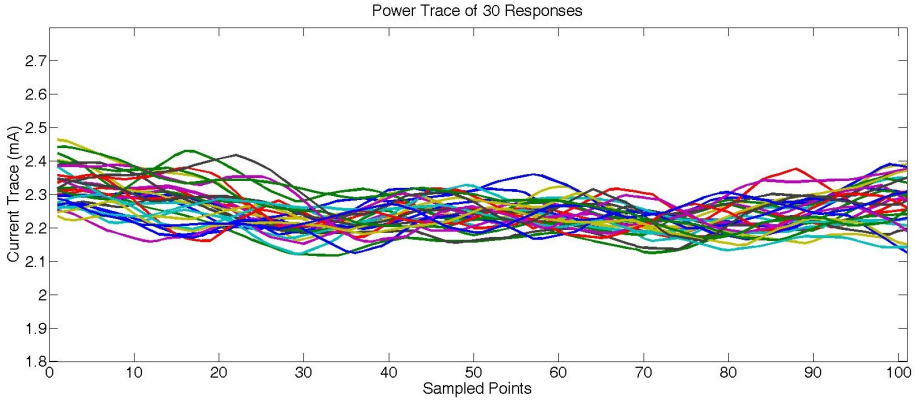

Fig. 2. Power trace of 30 different sub-responses, collected from FPGA, illustrating the difficulty of differentiating them from each other

There are two reasons for this problem:

1. In real silicon Arbiter and Lightweight PUFs, the final XOR function usually consumes no more than 5% silicon resource of the whole design. Thus, it is difficult to extract the power consumption of XOR function, which consumes much less power compared with the whole circuits;
2. Unlike a simulated PUF, measuring real silicon PUF circuit is always impacted by the noise from supply voltage and measurement, which plays a negative role in extracting the desired power information.

To overcome this problem and maintain the feasibility of our power side channel, we developed a new, statistical signal processing strategy.

Our main objective is to extract the subtle power consumption of XOR gates and transform it into a recognizable format, which is correlated with the cumulative number of one or zero sub-responses. Even though the extra power consumed by active XOR gates is not directly extractable, it does really affect the whole power consumption. Thus, it should change the probability distribution functions (PDF) of the measured power leakage, if it can extract the probability distribution of leaked power information, the cumulative of one sub-responses can be inferred. For this purpose, we apply a "challenge-dependent responses estimation" method to calculate the PDF of every power trace collection.

The "challenge-dependent responses estimation" is implemented by comparing the power trace just before and after the generation of response to distinguish subtle changes. In the experiment, we measure the power trace of a single PUF response for totally m times, and record all of them in parallel. If denoting the generation time of the ith PUF response R_i as t_i, we can then filter out the two adjacent sections of power trace (length of which is T_i^{before} and T_i^{after}) just before and after time t_i. Assume that $T_i^{before} = T_i^{after}$, then we divide each time slice into n parts with the collected power trace (current trace) data. Based on the divided current trace data, we can calculate the power consumption of each n part before and after the generation of response R_i.

By denoting power consumption of all the $2 * n$ parts of the ith PUF response under the lth measurement (totally m measurements are did as described above, thus, $l \in (1...m)$) as P_{lij}^{before} and P_{lij}^{after} respectively ($j \in (1...n)$), two matrices including the power consumption information of the i_{th} response are obtained as:

$$M_i^{before} = \begin{pmatrix} P_{11}^{before} & P_{12}^{before} & P_{13}^{before} & ... & P_{1n}^{before} \\ P_{21}^{before} & P_{22}^{before} & P_{23}^{before} & ... & P_{2n}^{before} \\ ... & ... & ... & ... & ... \\ P_{m1}^{before} & P_{m2}^{before} & P_{m3}^{before} & ... & P_{mn}^{before} \end{pmatrix} \tag{1}$$

$$M_i^{after} = \begin{pmatrix} P_{11}^{after} & P_{12}^{after} & P_{13}^{after} & ... & P_{1n}^{after} \\ P_{21}^{after} & P_{22}^{after} & P_{23}^{after} & ... & P_{2n}^{after} \\ ... & ... & ... & ... & ... \\ P_{m1}^{after} & P_{m2}^{after} & P_{m3}^{after} & ... & P_{mn}^{after} \end{pmatrix} \tag{2}$$

Based on the power trace processing above, we now denote the power information of a single PUF response with two matrix: M_i^{before} and M_i^{after}. Assuming that we totally collect K response bits, then the power consumption matrix for all responses can be described as (for brevity, "b" means before and "a" means after):

$$M^{before/after} = \begin{pmatrix} M_1^{b/a} & M_2^{b/a} & M_3^{b/a} & ... & M_K^{b/a} \end{pmatrix} \tag{3}$$

Due to the existence of environmental and measurement noise, the m parallel segmentations of measured power trace (such as $P_{11}^{before}, P_{21}^{before} ... P_{m1}^{before}$ in Equation 1, and $P_{11}^{after}, P_{21}^{after} ... P_{m1}^{after}$ in Equation 2) consumption would build n PDF respectively. Since we divide power trace slice into 2 parts (before and after), thus totally $2 * n$ PDF are generated for each response. As we discussed, though there is no directly leaked power information that we can extract for the XOR function, it impacts the probability distribution of the whole power trace. To convert the PDF information into the cumulative number of one and zero responses, we applied histograms method to describe the PDF, and then implement basic calculus computation to get the cumulative distribution function (CDF):

$$C_j^{before/after}(x) = \sum_{x_j < x} PDF(X = x_j) = \sum_{x_j < x} p(x_j) \quad j \in (1..n) \tag{4}$$

Based on Equation.4, the original leaked power information can be transformed as CDF. To filter out the difference between two power trace segments: before and after time t_i, and erase the impact of environmental and measurement noise, we then calculate the mean-squared-error (MSE) following Equation 5:

$$MSE_j = E[(C_j^{before}(x) - C_j^{after}(x)^2],\ j \in (1..n) \tag{5}$$

then, all of the n MSEs are summed up for a final sub-response estimation: E_i, which reflects and amplifies the impact of active XOR gates on leaked power:

$$E_i = \sum_{j=1}^{n} MSE_j,\ j \in (1..n) \tag{6}$$

With the proposed "challenge-dependent responses estimation" method, the power trace of different challenge-dependent responses patterns are transformed into an estimated value: "E_i". Thus, we can deduce the pattern of CRPs and integrate them within our proposed ML attacks.

Determining the Generation time of PUF Response. In the previous paragraph, we applied the "challenge-dependent responses estimation" method to extract the power side channel information of active XOR gates, assuming that we know the generation time of the ith PUF response R_i as t_i. However, one additional problem is that in practice, t_i is not a direct known parameter. In this last paragraph, we will now detail how we overcame this final problem.

If we randomly set a t_{i_random} as the generation time of response R_i, the power information of a certain PUF response R_i can be described as:

$$P_i = P_{i\_noise}^{before} + P_{i\_oc}^{before} + P_{i\_XOR}^{before} + P_{i\_noise}^{after} + P_{i\_oc}^{after} + P_{i\_XOR}^{after}, \tag{7}$$

where $P_{i\_noise}^{b/a}$ denotes the environmental and measurement noise (as before, "b" abbreviates before and "a" after t_{i_random} here), $P_{i\_oc}^{b/a}$ stands for the power consumption of "other circuitry", again before and after t_{i_random}, while $P_{i\_XOR}^{b/a}$ denotes the similar power information of XOR functional circuitry. Since based on the measurement, we can roughly tell the range of a PUF response generation time, we would have several choices of t_{i_random}. To determine the exact generation time of each PUF responses, we move the t_{i_random} in the approximate time range, then we will get different power side channel informative patterns.

Since the PUF circuitry are measured for multiple times, and under the same environment, we can assume that for each response, we will have:

$$P_{i\_noise}^{before} \approx P_{i\_noise}^{after} \ \ and \ \ P_{i\_oc}^{before} \approx P_{i\_oc}^{after} \tag{8}$$

thus, if we measure the power trace of a single PUF response for multiple times, we get:

$$\sum P_{i\_noise}^{before} - \sum P_{i\_noise}^{after} \approx 0 \ \ and \ \ \sum P_{i\_oc}^{before} - \sum P_{i\_oc}^{after} \approx 0 \tag{9}$$

Based on this algorithm, it is clear that only when t_i is set as the correct generation time, the E_i in Equation 6 is maximized.

4 Timing Side Channels on XOR-Based Arbiter PUFs

As with our power side channel, the objective of the timing side channel is providing additional information about the individual response bits (i.e., PUF output bits) even though the response bits are XOR'ed together for providing the output. Assume that k response bits $\{r_1, \ldots, r_k\}$ are XOR'ed to form a single output bit b_{out}. (Note that a k-input XOR shall consist of several stages of smaller XOR gates. For the sake of demonstration, assume that the delay of the response bit r_i, denoted by t_{r_i} follows a certain order, say $t_{r_1} \leq t_{r_2} \cdots \leq t_{r_{k-1}} \leq t_{r_k}$). Our timing side-channel approach is based on a delay measurement circuit, which can be used to characterize the delay length of different patterns of k response bits $\{r_1, \ldots, r_k\}$.

4.1 Timing Characterization Method

Every ASIC manufactured chip undergoes a set of structural and functional tests which measure/ evaluate the IC's physical and logical properties respectively. Measuring the delay of certain combinational paths in the circuit is a part of standard structural testing. Since the internal combinational paths are typically inaccessible, the timings are indirectly inferred from the FF outputs using clock sweeping. The FF values can be set using a testing scan-chain while all the FFs are connected to the global chip clock. The pertinent chip is referred to as Circuit Under Test (CUT). The frequency of this clock is swept in a continuous monotonic fashion from a high to low value while the path under measurement is toggled using the logic at the input FF. When the frequency is higher than the path delay, the output FF does not have enough time to settle which is called a "fail". Once the frequency approaches the path delay, the output FF sets to the correct value (from the initial reset dictated by the scan chain) which is the "pass" state. The frequency at which this transition occurs denotes the path delay and this overall testing method is called pass/fail timing test.

On our FPGA testbed, the pass/fail timing tests have to be implemented by reconfiguration. We adopt the measurement circuitry from [14,15] that is demonstrated in Figure 3. Note that because of the timing uncertainty around the FF metastability point, the toggle between the pass/fail states appears with a certain property. Thus, error density estimation followed by smoothing methods are used for inferring the exact toggle point from a set of stochastic measurements.

To estimate the probability of error at a certain clock frequency, an error histogram accumulator is realized using two counters. The first one is an error counter whose value increments by one each time an error occurs. The second one counts the clock cycles; after 2^N clock cycles, this counter clears (resets) the error counter and then restarts again, where N is the binary counters' size. The error counter value is stored in the memory one clock cycle before it is reset. Now, the stored number of fails normalized to N would yield the error probability value for each target frequency.

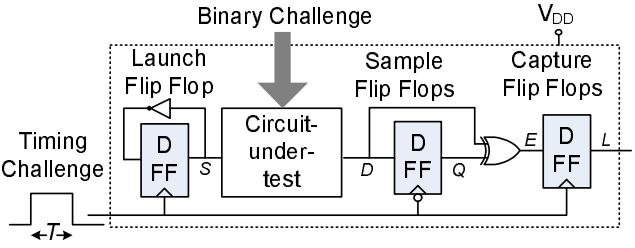

Fig. 3. The timing signature extraction circuit

Next, we linearly and continually sweep the input clock frequency: in T_{sweep} seconds from $f_i = \frac{1}{2T_i}$ to $f_t = \frac{1}{2T_t}$, where $T_t < t_p < T_i$. For each frequency sweep, a separate set of registers count the number of clock pulses. We use this counter as an accurate

timer which records the frequency of the timing errors. This counter value is retrieved every time the content of the error counter is written into memory. The system described above can be configured and utilized for extracting the delays of any CUT implemented on FPGA. We use this adaptation of pass/fail timing test to measure the delay between the FF storing the challenge input, to the output of the PUF which shall be stored in an output register. To prevent attacks, this output is measured after XORing the arbiter values. Note that the scanning for extracting delay values could also be performed in parallel to reduce the characterization time [14,15].

4.2 Characterization Accuracy

The resolution of the delay measurement, i.e., the measured delay's accuracy, is a function of a few factors: (i) the clock noise and skew, (ii) the sweeping frequency resolution, and (iii) the number of pulses at each frequency. The output of the characterization circuit is a binary zero/one (pass/fail) value. A real-valued output can be measured by repeating several (same width) clock pulses to the circuitry and accumulating the number of ones at the output. The resulting value, when normalized, shows the probability at which the timing errors occur for each input clock's pulse width. The more the input clock pulse is repeated, a higher sampling resolution and accuracy can be achieved.

For now assume that the clock pulse (of width T) is sent to the CUT for M times. Because of clock skew and phase noise, the characterization circuitry receives a clock pulse with width $T_{eff} = T + T_j$, where T_j is the additive jitter. Suppose that T_j is a random variable with a zero mean and symmetric distribution around its mean. The output probability is a continuous and smooth function of T_{eff}; thus, approximating the probability by averaging shall be an asymptotically unbiased estimator as $M \to \infty$. Lastly, the minimum measurable timing is a function of the maximum clock speed at which the FFs can be run (maximum clock frequency). During a linear frequency sweep, a longer sweep time increases both items (ii) and (iii) and thus the characterization accuracy.

4.3 Parameter Extraction

Thus far, we have described a system that measures the probability of timing errors for various clock pulse widths. The error probability can be fully represented by a set of few parameters; the parameters are directly related to the CUT delay and FF setup and hold times. It can be shown that the probability of timing errors shall be written as the sum of shifted Gaussian CDFs [14,15]. The central limit theorem can determine the Gaussian nature of the error probabilities which can be explained by Equation 10 showing the parameterized error probability function.

$$f_{\mathbf{D},\Sigma}(t) = 1 + 0.5 \sum_{i=1}^{|\Sigma|-1} -1^{\lceil i/2 \rceil} \left[Q(\frac{t - d_i}{\sigma_i}) \right] \tag{10}$$

where $Q(x)$=$\frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right)$ and $d_{i+1} > d_i$. To estimate the timing parameters, f is fit to the set of measured data points (t_i,e_i), where e_i is the error value recorded when the pulse width is t_i.

4.4 Side Channel Timing Analysis of XOR'ed Outputs

The pass/fail timing measurement above is able to estimate the delay of the overall PUF path (after XOR'ing). As we sweep the clock, we eventually get to a stable regime, i.e., the regime where the overall output does not change any more. However, before getting to this stable regime, there are clock periods for which only a few XOR inputs (i.e., response bits) change. Sweeping the clock frequency could yield the information about the approximate timing of the XOR inputs: every time one of the inputs to the XOR network, i.e., an arbiter output, changes, there will be a toggle. Even though it is not possible to distinguish the response bit that has changed, it is possible to estimate the number of flipping XOR inputs with a good probability. This number shall be vague if the timings of two or more response bits coincide. Since the probability of such a coincidence is rather low, in most instances clock sweeping shall yield an approximation of the number of flipped XOR inputs, i.e., the cumulative number of zeros and ones among the single Arbiter PUF responses $r_1, \ldots, r_k$.

5 Adapting Machine Learning Algorithms to Side Channel Information

The question how (and if at all) SC information on the cumulative number of zeros and ones can be efficiently exploited in PUF modeling turned out to be highly non-trivial. Eventually, we found a gradient based optimization similar to the logistic regression (LR) algorithm of [28,30]. The following treatment assumes some familiarity with this algorithm and with the work in [28,30].

Let $r_i(C) \in \{0,1\}$ be the output of the i^{th} Arbiter PUF within a k-XOR Arbiter PUF (or within a Lightweight PUF with k parallel Arbiter PUFs) to a challenge C. The side-channel information then yields the number n of individual Arbiter PUFs with output one: $n = \sum_i r_i(C)$. It lies in contrast to the general setting of binary outputs in LR on an interval scale. Therefore, instead of optimizing the binary class probabilities [28,30], we rely on minimizing the squared error between a side-channel model $f(\boldsymbol{w}, C)$ and the actual outputs n:

$$l(\mathcal{M}, \boldsymbol{w}) = \sum_{(C,\, t) \in\, \mathcal{M}} (f(\boldsymbol{w}, C) - n)^2.$$

The corresponding gradient

$$\nabla l(\mathcal{M}, \boldsymbol{w}) = \sum_{(C,\, r) \in\, \mathcal{M}} 2\left(f\left(\boldsymbol{w}\right) - n\right) \nabla f(\boldsymbol{w}) \tag{11}$$

is highly similar to the gradient in LR. We again applied the RProp update scheme (as in [28,30]) to find a solution $\hat{\boldsymbol{w}}$ with minimal error l.

Assuming the standard linear additive delay model [9,6,28,30], one obtains the following model of the side-channel information:

$$f(\boldsymbol{w}, C) = \sum_i \Theta(\boldsymbol{w}_i^T \boldsymbol{\Phi}_i).$$

Note that the model only depends on the direction, but not on the length $\|\boldsymbol{w}_i\|$ of the weight vectors. That is, any two solutions $\boldsymbol{w}_i$ and $\alpha\boldsymbol{w}_i, \alpha \in \mathbb{R}^+$ are equivalent. Therefore we might substitute the Heaviside function by the differentiable logistic sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$ to enable gradient based optimization. This is a reasonable substitution as $\lim_{\|\boldsymbol{w}\|\to\infty} \sigma(\boldsymbol{w}^T\boldsymbol{\Phi}) = \Theta(\boldsymbol{w}^T\boldsymbol{\Phi})$ and, as noted above, a valid solution is unaffected by scaling of $\boldsymbol{w}$.

As this substitution makes the model differentiable, we obtain the following gradient to insert in Equation 11:

$$\nabla f(\boldsymbol{w}_j) = \sigma(\boldsymbol{w}_j^T\boldsymbol{\Phi}_j)(1 - \sigma(\boldsymbol{w}_j^T\boldsymbol{\Phi}_j))\boldsymbol{\Phi}_j. \tag{12}$$

This gradient of an individual Arbiter PUF's weight vector $\boldsymbol{w}_j$ depends only on the value of the weight vector itself, being in strong contrast to the case without side-channel information [28,30]. The decoupling of individual Arbiter PUF updates thus drastically simplifies the ML problem, provided that side-channel information is available.

In addition to the above new regression, we applied a two step optimization methodology: First we optimized the PUF model based on the above process and gradient, using the side-channel information, until a fraction of $f = 0.95$ percent of the final XOR Arbiter output was correctly reproduced. Secondly, we further refined and optimized the model with the "standard" LR algorithm applied in [28,30] for 1000 iterations. This led to very low error rates around 2% or below. For all experiments, we used hundred times more CRPs than free parameters in the model, i.e.,

$$N_{CRP} \approx 100 \times \text{bitlength} \times \text{no. of XORs}.$$

Note that the above equation merely describes a linear CRP consumption in the problem parameters. This is in stark contrast to the exponentially growing complexities of pure ML attacks on XOR Arbiter and Lightweight PUFs [28,30].

While our approach in the first step of the above methodology mostly converged to the global minimum, in a few cases it got stuck (i.e., the performance after 5000 iterations was worse than 5% remaining missclassifications). In this case, we restarted the algorithm with a different random initialization of $\boldsymbol{w}$.

6 Results and Asymptotic Performance Analysis

We applied our adapted ML methods (see Section 5) to CRP data and SC information gathered from FPGAs (see Sections 2, 3, and 4), both for power and timing SCs. The results are presented in Tables 1 and 2. The attacks perform extremely efficiently, as we were able to successfully attack XOR Arbiter PUFs and Lightweight PUFs for up to 16 XORs and for bitlengths of up to 512 (timing SCs) and 128 (power SCs). No implementations of comparable sizes of these two PUFs in silicon had ever been considered or reported before. Furthermore, pure modeling attacks thus far had only been able to tackle the two PUFs for up 5 or 6 XORs and bitlength 64 [28,30]. Both facts illustrate the impact and reach of our new method.

Tables 1 and 2 already indicate that the CRP requirments and computation times grow very mildly, with the same holding for the prediction errors. In order to quantify

Table 1. Effectiveness of *timing* side-channel attacks on the XOR Arbiter PUF and Lightweight PUF (LW PUF), all carried out on FPGA implementations

No. of XORs	Bit Length	CRPs ($\times 10^3$)	Prediction Rate XOR Arb. PUF	Training Time XOR Arb. PUF	Predict. Rate LW PUF	Training Time LW PUF
8	64	26	98.5%	2 min	98.5%	1 min
	128	51.6	97.5%	12 min	98.2%	9 min
	256	103	97.7%	1:35 hrs	97.8%	1:00 hrs
	512	205	97.4%	16:50 hrs	97.5%	3:30 hrs
12	64	39	98.1%	16.5 min	98.5%	2 min
	128	77.4	97.4%	38.5 min	97.9%	24.1 min
	256	154.5	97.1%	3.8 hrs	97.3%	1.75 hrs
	512	308	96.92%	56.25 hrs	97.11%	9.55 hrs
16	64	52	98%	37 min	98%	7 min
	128	103.2	97.5%	2 hrs	97.5%	51.7 min
	256	206	97.3%	15.1 hrs	96.9%	4.8 hrs
	512	410	96.5%	102 hrs	96.7%	20.2 hrs

Table 2. Effectiveness of *power* side-channel attacks on the XOR Arbiter PUF and Lightweight PUF (LW PUF), all carried out on FPGA implementations

No. of XORs	Bit Length	CRPs ($\times 10^3$)	Prediction Rate XOR Arb. PUF	Training Time XOR Arb. PUF	Predict. Rate LW PUF	Training Time LW PUF
8	64	26	98.1%	3 min	98.4%	1.25 min
	128	51.6	98%	13 min	98.1%	9.25 min
12	64	39	98.3%	11 min	98.2%	3.5 min
	128	77.4	97.3%	47 min	97.8%	25 min
16	64	52	98%	38 min	98%	6.5 min
	128	103.2	97.5%	2:28 hrs	97.5%	46.5 min

this with yet more data points, we conducted comprehensive ML experiments on simulated CRPs and simulated SC data. The CRPs were generated by the linear additive delay model (LADM), similarly as in earlier ML experiments [28,30]. We executed these simulated attacks on XOR Arbiter PUFs and Lightweight PUFs for 2, 3, ..., 16 XORs, and with 64, 128, 256 and 512 bits. This means that we treated $2 \cdot 15 \cdot 4 = 120$ different architectures in sum, investing hundreds of hours of computation time. The generated data points are shown in Figure 4, and fully confirm the suspected mild, actually cubic growth. For those cases where we also had silicon data for comparison (see Tables 1 and 2), the silicon and the simulated attacks performed very similarly, confirming both earlier conjectures [6,28,30] on the validity of the additive linear delay model, as well as the accuracy of our side-channel measurements. The empirically estimated computational complexity of our attacks is hence $O(n^3)$, or, in other words, low-degree polynomial, in the problem size. Furthermore, as indicated already in Section 5, the number of used/required CRPs is merely linear in the same parameter.

Two important aspect should not go unnoticed. Firstly, our power side channel is more noisy than the timing side channel. This had the effect that we could only handle

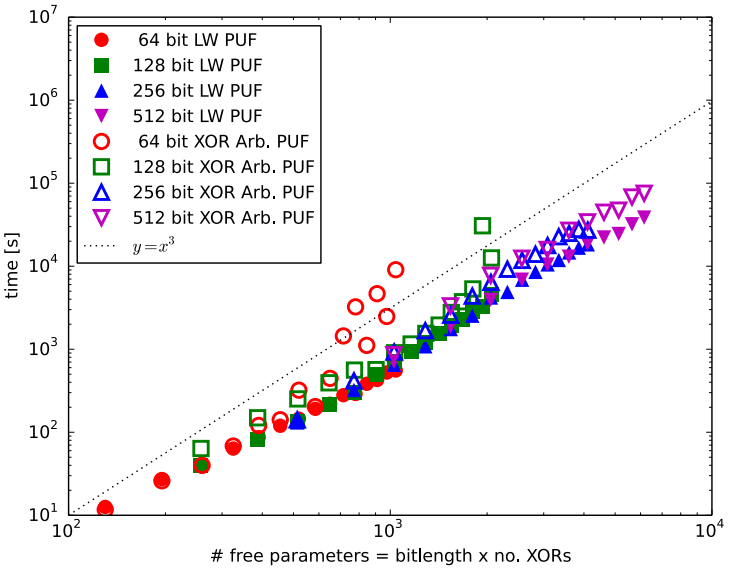

Fig. 4. The training times for our ML-algorithm on Lightweight PUFs (LW PUFs) and XOR Arbiter PUFs on a logarithmic scale. They show that the computational complexity regarding training times is cubic, i.e., $O(x^3)$.

bit lengths of up to 128 by use of the power SC. Improved, less noisy versions seem possible, but also non-trivial, and are left to future work.

Secondly, in the presence of side-channel information, our ML algorithms perform slightly faster on Lightweight PUFs than on XOR Arbiter PUF. Without side channels, the converse effect has been observed [28,30]. Intuitively, the challenge input mapping of the Lightweight PUF creates a more diverse and stable information basis for the ML algorithm, which leads to faster convergence. A full, rigorous mathematical analysis of this effect will be conducted in future work.

7 Summary and Conclusions

In this paper, we introduced and implemented the first power and timing side channels (SCs) on PUFs, more precisely on XOR Arbiter PUFs and Lightweight PUFs. These two PUF designs were chosen by us due to their particular relevance: The Arbiter PUF family is arguably the most studied electrical Strong PUF design, and said two PUFs are the most secure representatives of this family according to recent work [28,30]. Our two SCs consisted of (i) power tracing of the arbiter element (i.e., the latch) in Arbiter PUFs, and (ii) marking different response patterns with corresponding timing signatures. Both SCs tell us the *cumulative* number of zeros and ones in the outputs of the k parallel Arbiter PUFs within XOR-based Arbiter PUF variants, such as the XOR Arbiter PUF

or the Lightweight PUF. One main obstacle in exploiting the above SCs efficiently was that the attacker does not learn *which* of the single Arbiter PUF outputs is zero or one. This makes the cumulative information worthless at first sight. However, we were able to devise adapted, tailor-made ML algorithms, which can exploit the information very efficiently.

We carried out a full silicon proof of concept on FPGAs, attacking the two above PUFs for up to 16 XORs and bitlengths of 512 bits (by timing SCs) and 128 bits (by power SCs). Their smaller noise levels made timing SCs the yet more efficient tool, even though improved future versions of the power side channels seem possible. Interestingly, XOR-based Arbiter PUF variants had never even been *implemented* (left alone attacked) for comparable sizes in the literature, since already versions with 8 XORs and 512 bits had been recommended as practically secure against known attacks in earlier works [28,30]. This may illustrate the relevance and strength of our results. A close asymptotic analysis on simulated CRP data furthemore showed that our attacks have only *cubic complexity*. This is a drastic improvement over the exponential complexity of state-of-the-art, pure modeling attacks [28,30].

Our methods are the first physical attacks on Strong PUFs, i.e., on PUFs with many CRPs, that can notably increase attack performance. Overall, they imply that *as long as no suitable design countermeasures are put in place*, no currently existing architecture from the Arbiter PUF family can withstand all known attacks: "Standard" Arbiter PUFs as well as Feed-Forward Arbiter PUFs have been attacked by pure modeling attacks with polynomial complexity [28,30]; and XOR-based variants such as the XOR Arbiter PUF and the Lightweight PUF are susceptible to the methods presented in this paper, which have polynomial complexity, too.

We did not explicitly deal with design countermeasures in this paper for space reasons. However, one conceivable strategy against power SCs could consist of using two symmetric, inverted output signals with two latches. This construction could neutralize and balance power consumption, regardless of the PUF's output. Interestingly, this could even be used to detect and stabilize output errors in Arbiter PUF variants, even though we did not follow this route in in this paper. Countermeasure against our timing SCs would probably have to focus on the construction of an isochronous hardware. Implementing such strategies is left to future, follow-up works.

We believe that the PUF attacks presented in this and other papers should be interpreted in a balanced fashion. None of them "kills" the field in its entirety. In our opinion, they are part of a natural consolidation process in the PUF area, similar to the consolidation that classical security primitives have undergone already some time ago. The occurence of this process could be seen as indication that the field is becoming increasingly mature. One typical byproduct is the insight that certain aspects are not as simple as originally believed, which may be disappointing at first sight. Overall, however, a sound consolidation will be beneficial to the field, eventually creating more research opportunities than it destroys. This paper could be seen as one (of many) steps within this process.

Acknowledgements. The work at the University of Massachusetts Amherst was supported in part by SRC task 1836.074, US NSF grants 0923313 and 0964641, and US DHHS grant 90TR0003/01. The work at Rice University was supported in part by NSF

CCF-1116858:SHR:Small, NSF CNS-1059416:CI-ADDO-NEW: Trust-Hub, and ONR ONR N00014-11-1-0885 grants.

References

1. Bishop, C.M., Nasrabadi, N.M.: Pattern recognition and machine learning. Springer, New York (2006)
2. Delvaux, J., Verbauwhede, I.: Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In: HOST (2013)
3. Delvaux, J., Verbauwhede, I.: Attacking PUF-Based Pattern Matching Key Generators via Helper Data Manipulation. IACR Cryptology ePrint Archive, Report 2013/566
4. Delvaux, J., Verbauwhede, I.: Key-recovery Attacks on Various RO PUF Constructions via Helper Data Manipulation. IACR Cryptology ePrint Archive, Report 2013/610
5. Delvaux, J., Verbauwhede, I.: Fault Injection Modeling Attacks on 65nm Arbiter and RO Sum PUFs via Environmental Changes. IACR Cryptology ePrint Archive, Report 2013/619
6. Devadas, S.: Physical unclonable functions and secure processors. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 65–65. Springer, Heidelberg (2009)
7. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security, pp. 148–160 (2002)
8. Helfmeier, C., Nedospasov, D., Boit, C., Seifert, J.-P.: Cloning Physically Unclonable Functions. In: HOST 2013 (2013)
9. Lim, D.: Extracting Secret Keys from Integrated Circuits. MSc Thesis, MIT (2004)
10. Majzoobi, M., Koushanfar, F., Devadas, S.: FPGA PUF using programmable delay lines. In: IEEE Workshop Information Forensics and Security, WIFS (2010)
11. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight Secure PUFs. In: ICCAD, pp. 607–673 (2008)
12. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing techniques for hardware security. In: Proceedings of the International Test Conference (ITC), pp. 1–10 (2008)
13. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Techniques for Design and Implementation of Secure Reconfigurable PUFs. ACM Trans. Reconfigurable Technology and Systems 2(1) (2009)
14. Majzoobi, M., Dyer, E., Elnably, A., Koushanfar, F.: Rapid FPGA Characterization using Clock Synthesis and Signal Sparsity. In: International Test Conference (ITC), pp. 1–10 (2010)
15. Majzoobi, M., Koushanfar, F.: Time-Bounded Authentication of FPGAs. IEEE Transactions on Information Forensics and Security (TIFS) 6(3), 1123–1135 (2011)
16. Rostami, M., Majzoobi, M., Koushanfar, F., Wallach, D., Devadas, S.: Robust and Reverse-Engineering Resilient PUF Authentication and Key-Exchange by Substring Matching. IEEE Transactions on Emerging Topics in Computing (2014)
17. Merli, D., Schuster, D., Stumpf, F., Sigl, G.: Side-Channel Analysis of PUFs and Fuzzy Extractors. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 33–47. Springer, Heidelberg (2011)
18. Merli, D., Schuster, D., Stumpf, F., Sigl, G.: Semi-invasive EM attack on FPGA RO PUFs and countermeasures. In: ACM Workshop on Embedded Systems Security, WESS 2011 (2011)
19. Merli, D., Heyszl, J., Heinz, B., Schuster, D., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of RO PUFs. In: HOST 2013 (2013)
20. Nedospasov, D., Helfmeier, C., Seifert, J.-P., Boit, C.: Invasive PUF Analysis. In: Fault Diagnonsis and Tolerance in Cryptography, FDTC 2013 (2013)

21. Pappu, R.: Physical One-Way Functions. PhD Thesis, Massachusetts Institute of Technology (2001)
22. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way Functions. Science 297, 2026–2030 (2002)
23. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: IEEE International Conference on Neural Networks, pp. 586–591 (1993)
24. Rührmair, U., Devadas, S., Koushanfar, F.: Security based on Physical Unclonability and Disorder. In: Tehranipoor, M., Wang, C. (eds.) Introduction to Hardware Security and Trust, Springer, Heidelberg (2011)
25. Rührmair, U., van Dijk, M.: Practical security analysis of PUF-based two-player protocols. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 251–267. Springer, Heidelberg (2012)
26. Rührmair, U., van Dijk, M.: PUFs in Security Protocols: Attack Models and Security Evaluations. In: IEEE Symposium on Security and Privacy, Oakland 2013 (2013)
27. Rührmair, U., Holcomb, D.E.: PUFs at a glance. In: DATE 2014, pp. 1–6 (2014)
28. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling Attacks on Physical Unclonable Functions. In: ACM Conference on Computer and Communications Security (2010)
29. Rührmair, U., Sölter, J., Sehnke, F.: On the Foundations of Physical Unclonable Functions. Cryptology e-Print Archive (June 2009)
30. Rührmair, U., Sölter, J., Sehnke, F., Xu, X., Mahmoud, A., Stoyanova, V., Dror, G., Schmidhuber, J., Burleson, W., Devadas, S.: PUF Modeling Attacks on Simulated and Silicon Data. IEEE Transactions on Information Forensics and Security, IEEE T-IFS (2013)
31. Edward Suh, G.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: DAC 2007, pp. 9–14 (2007)