Active and Passive Side-Channel Attacks on Delay Based PUF Designs

Georg T. Becker, Raghavan Kumar

***Abstract*—Physical Unclonable Functions (PUFs) have emerged as a lightweight alternative to traditional cryptography. The fact that no secret key needs to be stored in non-volatile memory makes PUFs especially well suited for embedded systems in which securely generating and storing secret keys is difficult and expensive. Compared to traditional cryptography, PUFs are often believed to be more resistant to implementation attacks.**

In this paper we will take a closer look at this assumption. Using a controlled Arbiter PUF as an example, we show that just like traditional cryptography strong PUFs are susceptible to implementation attacks. By combining machine learning with with side-channel analysis we are able to attack designed based on Arbiter PUFs that on are resistant to normal machine learning attacks. We use two different side-channels for our attacks: a passive power side-channel and an active fault attack based on altering the supply voltage of the controlled PUF. Even in the presence of considerable noise both attacks can accurately model the Controlled Arbiter PUF. Hence, the assumption that PUFs are generally more resistant against side-channel attacks is not necessarily true and side-channel resistance needs to be considered when PUF designs are evaluated.

***Index Terms*—Side-channel analysis, machine learning, Physical Unclonable Function, Arbiter-PUF, fault attack, CPA**

I. Introduction

Physical Unclonable Functions (PUF) have gained widespread attention in the research community as a new cryptographic primitive for hardware security applications. PUFs make use of the fact that two manufactured computer chips are never completely identical due to process variations. A PUF exploits these process variations to ensure that each chip has a unique behavior so that each chip can be uniquely identified. There are many applications for which PUFs can be used. Two prominent examples are the use in challenge-and-response protocols as well as for secure key generation and storage. The advantage of using a PUF to generate cryptographic keys is that the PUF ensures that each chip will have its own unique secret without the need to program it first. Furthermore, securely storing a cryptographic key in embedded devices in a way that they are resistant to physical attacks such as probing and reverse-engineering is extremely difficult. In PUFs on the other hand, no key needs to be stored in non-volatile memory since the secret is instead derived from physical characteristics which are hard to monitor.

PUFs can be classified into two categories: *weak PUFs* and *strong PUFs*. In a weak PUF, the number of challenges the PUFs can accept is very limited so that an attacker can try all possible challenges and store the responses. This way an attacker could easily forge the PUF by replacing the PUF with a simple memory look-up. A strong PUF on the other hand has a challenge space that is large enough so that it is computationally infeasible to try and store all possible challenges. Strong PUFs can be used in challenge-and-response protocols as well as for key generation. A weak PUF on the other hand cannot be used for challenge-and-response protocols. But they can still be used for key generation since it is usually sufficient to generate a limited number of different keys for each chip. Note that the terminology strong PUF and weak PUF might falsely give the impression that a strong PUF is "better" than a weak PUF. However, this terminology only defines the challenge space without judging the PUFs performance or other security properties.

Current PUF designs face two big problems that are related: they suffer from unreliability and are prone to machine learning attacks. In an ideal case, a PUF always generates the same response for a given challenge. However, due to environmental effects and thermal noise, the response to the same challenge can vary. Therefore, error correction codes are usually needed if the PUF is used for key generation and challenge-and-response protocols need to allow a few false response bits. The second problem is that even strong PUFs can be modeled in software and the needed parameters to model a specific PUF instance can be determined using machine learning techniques if challenge and response pairs are known to the attacker [21].

The use of strong PUFs in challenge-and-response protocols has gained a lot of attention despite their weakness towards machine learning attacks. PUFs have become cryptographic building blocks and many protocols have been proposed including protocols with theoretical security proofs [20]. Usually three possible advantages are listed for the use of PUFs in challenge-and-response protocols: One argument for PUFs is that some PUFs have a lower area or power overhead compared to cryptographic algorithms, especially if the PUFs are compared to cryptographic algorithms that include countermeasures against hardware attacks. Secondly, the secret (key) does not need to be programmed first, ensuring that every PUF instance has a unique key even before the first power-up.

G.T. Becker is with the Horst Görtz Institute for IT-Security at the University of Bochum and R. Kumar is with the Department of Electrical and Computer Engineering, University of Massachusetts at Amherst. The results presented here can also be found in Becker's PhD dissertation [3].

This is especially useful if the PUF is used in a piracy protection scheme. And last but not least, PUFs are often believed to be more secure against implementation attacks (physical attacks) such as probing and side-channel attacks. Resistance against implementation attacks is an extremely interesting property as it has been shown many times how difficult it is to protect embedded systems against these types of attacks. A very careful design and the combination of different countermeasures is needed to ensure a reasonable resistance against implementation attacks. But this increases development costs and design overhead significantly. If PUFs would indeed be very resistant towards these implementation attacks, this would make them a very promising alternative to traditional cryptography. However, this resistance has never been thoroughly analyzed and proven.

A. Related Work

It has already been shown that PUFs, when used for key generation, can be attacked with side-channel attacks by attacking the digital error-correction used in these systems [12], [16]. These attacks do not directly attack the used PUF itself but the digital post-processing. Nevertheless, they still indicate that using PUFs does not necessarily solve the problem of implementation attacks as other parts of the system might still be vulnerable. Recently, Merli *et al.* successfully attacked a ring-oscillator (RO) PUF using an EM attack that directly targeted the PUF and not the error correction [15]. RO-PUFs only have a limited number of challenges and responses and are therefore weak PUFs. The only side-channel attacks that directly targets a strong PUF is the recent work by Delvaux *et al.* [6]. The authors used the large unreliability of the PUF as side-channel leakage and for the first time modeled an arbiter PUF based on this information. However, for the attack to work, the attacker needs to know the unreliability of single response bits in the presence of thermal noise. This means that the attack actually not only uses the reliability information but also directly the response bit, i.e. whether a response is biased towards 1 or 0. But if the response bits are known, traditional machine learning algorithms can be used, which are still more efficient [6]. Hence, while the paper is interesting from a theoretical point of view, it has only limited practical relevance.

In parallel to our work, Rührmair *et al.* [22] have also proposed a combined machine learning and side-channel attack using the power and timing side-channel. In their work they chose an XOR-Arbiter PUF as their target architecture. However, their assumed power model is very idealistic. They assume that they can directly read out the number of responses that are "one" from the power traces without any noise. However, such a power model is very unrealistic in practice as discussed in Section III-A. How their approach works in a noisy environment which resembles a more realistic environment is still an open question. Besides the power side-channel they have also propose the use of a timing side-channel. This timing side-channel is very interesting. In their proof of concept implementation they used a dedicated circuit in the FPGA to measure the timing differences. In practice such an on-chip measurement mechanism won't be present. However, their approach could be extended to be used as a fault attack for XOR-Arbiter PUFs. In [22] the authors also propose some countermeasures how this timing side-channel attack can be prevented.

B. Contribution and Organization

In this paper we will take a closer look at the side-channel resistance of strong PUFs against passive and active attacks. In particular, two different classes of implementation attacks are considered, power side-channel analysis and fault attacks. As our target we chose the Arbiter PUF, as it is the most widely discussed strong PUF in the literature. We show that it is possible to attack the Arbiter PUF even in scenarios where machine learning attacks are infeasible, i.e. when the attacker does not directly have access to the individual response bits. We performed a hybrid machine learning power side-channel attack on a controlled PUF design based on an Arbiter PUF on simulated traces and showed that the attack is successful in the presence of considerable noise. By comparing our power side-channel attack on PUFs with successful CPA attacks on block ciphers from the literature, we show that with comparable noise levels a power side-channel attack on the controlled PUF would be successful as well. Furthermore, we propose a fault attack on the same design that is based on changing the supply voltage. We use the information which challenges become unreliable to perform a hybrid fault and machine learning attack that can accurately model a controlled Arbiter PUF.

Hence, our results raise doubt whether the assumption that PUFs are less prone to implementation attacks than traditional cryptographic algorithms holds true in general.

A detailed description of how Arbiter PUFs work and how they can be modeled is presented in the beginning of the next Section. In Subsection II-C a controlled PUF designed based on an Arbiter PUF is introduced under the assumption that the PUF achieves a reliability close to 100%. This secure design will be used as the target design for the power side-channel attacks in Section III and the fault attacks in Section V. In the last Section the implications of the results presented in this paper are discussed.

II. Target PUF Design

Arbiter-PUFs are the most popular strong PUF design in the literature. However, they can easily be modeled using machine learning attacks if the attacker knows challenge and response pairs. To overcome this problem, *controlled PUFs* [7] were proposed in which the direct challenge and response pairs of the Arbiter PUF are never revealed. Therefore controlled PUFs are secure against

machine learning attacks. In this paper we chose such a controlled PUF as our target. However, our method is sufficiently general to be applied to other designs based on an Arbiter PUF.

A. Arbiter PUF

The basic idea of the Arbiter PUF is to apply a race signal to two identical paths and determine which of the two paths is faster. The two paths have an identical layout so that the delay difference ΔD between the two signals mainly depends on process variations. This dependency on process variations ensures that each chip will have a unique delay behavior. The Arbiter PUF gets a challenge as its input which defines the exact paths the race signal takes. Figure 1 shows the schematic of an Arbiter PUF. It consists of a top and bottom signal that is fed through delay stages. Each individual delay stage consists of two 2-bit multiplexers (MUX) that have identical layouts and that both get the bottom and top signals as inputs. If the challenge bit for the current stage is '1' the multiplexers switch the top and bottom signals, otherwise the two signals are not switched. Each individual transistor in the multiplexers has a slightly different delay characteristic due to process variations and hence the delay difference between the top and bottom signal are different for a '1' and a '0'. This way the race signal can take many different paths: an n-stage Arbiter PUF has 2^n different paths the race signals can take. However, challenges that only differ in a few bits have a very similar behavior so that an Arbiter PUF does not necessarily have 2^n unique challenges. An Arbiter at the end of the PUF determines which of the two signals was faster. The Arbiter consists of two cross-coupled AND gates which form a latch and will have an output of '1' if the top signal arrives first and '0' if the bottom signal is the first to arrive. The Arbiter can have a slight bias so that the PUF result might be slightly biased towards '0' or '1'.

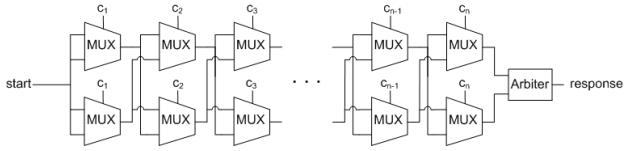

Fig. 1. Schematic of an n-bit Arbiter PUF.

B. Modeling an Arbiter PUF

Arbiter PUFs can easily be modeled in software if the delays that are added by each individual stages are known. Each stage has four delay values: the delay for the top and bottom signal for challenge bit '1' and for challenge bit '0'. Since we are actually not interested in the total delay but only in the delay difference, we can reduce these four values to two values per stage i, the delay difference $\delta_{1,i}$ between the top and bottom signal for challenge bit '1' and the delay difference $\delta_{0,i}$ for the challenge bit '0'. The delay difference is positive if the top signal is faster and negative if the bottom signal is faster. The total delay difference ΔD for a given challenge $\vec{C} = c_1, ...c_n$ can be computed easily by adding up the individual delays for each stage. If the challenge bit is '1' the wires are switched and the top signal becomes the bottom signal and vice versa. The switching of the wires can be modeled by simply multiplying the current delay difference with minus one. The time difference ΔD_i between the top and bottom signal after stage i can therefore easily be expressed recursively with the following equation:

$$\Delta D_i = \Delta D_{i-1} * (1 - 2 * c_i) + \delta_{c_i,i}$$

The final time difference between the two signals is simply time difference ΔD_n after the last stage n and the response bit r is defined by:

$$r = \begin{cases} 1 & \text{if } \Delta D_n > 0 \\ 0 & \text{if } \Delta D_n < 0 \end{cases}$$

This way the PUF can be modeled with $2 * n$ delay values. However, a more efficient approach to model an n-stage Arbiter PUF that only requires $n + 1$ parameters is used in practice. A PUF instance is described by the delay vector $\vec{w} = (w_1, ..., w_{n+1})$ with:

$$w_1 = \delta_{0,1} - \delta_{1,1}$$

$$w_i = \delta_{0,i-1} + \delta_{1,i-1} + \delta_{0,i} - \delta_{1,i} \text{ for } 2 \leq i \leq n$$

$$w_{n+1} = \delta_{0,n} + \delta_{1,n}$$

The delay difference ΔD_n at the end of the Arbiter is the result of the scalar multiplication of the transposed delay vector $\vec{w}$ with a feature vector $\vec{\Phi}$ that is derived from the challenges:

$$\Delta D_n = \vec{w}^T \vec{\Phi}$$

The feature vector $\vec{\Phi}$ is derived from the challenge vector $\vec{c}$ as follows:

$$\Phi_i = \prod_{l=i}^{n} (1 - 2c_l) \text{ for } 1 \leq i \leq n$$

$$\Phi_{n+1} = 1$$

Modeling a PUF in this way can significantly decreases the simulation time and also reduces the parameters that need to be known to $n + 1$. It was shown in the past how these parameters can be computed (or approximated) easily using different machine learning techniques. In practice, only a few hundred challenge and response pairs are needed to model an Arbiter PUF with a predication rate very close to the reliability of the attacked PUF [11]. To make machine learning attacks more difficult, some designs try to add a non-linear component to the Arbiter PUF e.g. by using feed-forwards or by XORing the responses of several Arbiter PUFs. These designs make machine learning attacks more difficult, but not necessarily computational infeasible. It has been shown that it is still possible to achieve a prediction accuracy of above 98% percent using machine learning techniques such as Evolution Strategies (ES) or Linear Regression (LR) against feed-forward

PUFs, lighweigth PUFs and XOR arbiter PUFs[21]. However, with increasing parameter sizes the attack becomes increasingly difficult and in [21] the authors report that XOR-Arbiter PUFs with at least 8 XORs were out of the reach with their attack.

C. Controlled PUF Design

Since current PUF designs can be attacked using machine learning if the attacker has access to challenge and response pairs, so called *Controlled PUFs* have been proposed to prevent these attacks. The term controlled PUF was introduced by Gassend *et al.* in [7] and the main idea is to add additional circuitry that prevents an attacker to apply arbitrary challenges and, most importantly, hides the individual response bits from the attacker. The main idea of most controlled PUFs is to instead of applying the challenges to the PUF directly, a master challenge is sent to the chip. From this master challenge, a challenge generator generates n individual challenges that are applied to the PUF. The n individual response bits of this PUF are not directly returned as outputs but instead are first applied to a cryptographically secure one-way function (e.g. a hash function). Figure 2 shows an example implementation of a Controlled PUF design and which is used as a case study in the remainder of this chapter. By

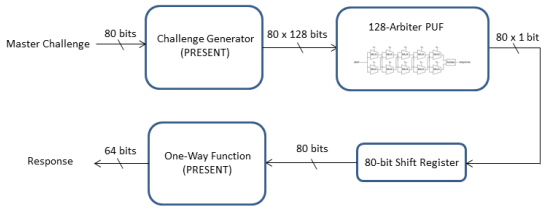

Fig. 2. The controlled PUF design. An 80-bit master challenge is applied to the controlled PUF from which 80 individual sub-challenges are derived using the challenge generator. These 80 sub-challenges are applied to the 128-bit Arbiter PUF and the 80 PUF responses are stored in a shift register. This 80-bit string is hashed using a cryptographically secure one-way function and the resulting 64-bit hash value is provided as the final response of the controlled PUF.

never revealing the individual response bits to the outside, machine learning attacks are not feasible any longer[1].

Please note that such a design only works if the used PUF is very reliable as a single false response bit will cause the current authentication attempt to fail. If the PUF is reliable enough, one can simply repeat the authentication process several times in case the authentication failed. Another option is to use error correction codes as mentioned in the original controlled PUF proposal. Since there already have been side-channel attacks on error correction codes we omit these error corrections in our analysis. Error correction does not have much impact on the power side-channel attack discussed in this paper. How error correction will affect the active attack will be discussed in Section V-B.

Of course, the proposed design adds a non negligible overhead to the PUF design, as a challenge generator and one-way function is needed. However, there exist several secure lightweight encryption algorithms that could be used for this purpose. The challenge generator generates n sub-challenges for a single master challenge. There are two main requirements for the challenge generator:

1) A master challenge should not generate sub-challenges that are similar i.e. sub-challenges that have large sequences of bits that are equal between these sub-challenges.
2) Two master challenges should not generate sub-challenges that are similar to each other.

An example of a secure challenge generator would be a block cipher in counter mode with the key as the master challenge. The fact that the same design can be used as a challenge generator and as a one-way function greatly reduces the introduced area overhead of a controlled PUF. Many lightweight block ciphers exist such as PRESENT [4] or NSA's lightweight block ciphers [2] that could be used for this purpose. However, the results presented in this paper are sufficiently general that other approaches such as using a hash function or LFSRs as the challenge generator can be used instead.

It is also important to note that the physical security requirements for the challenge generator and the one-way function are much lighter than for the case that they are used with a traditional secret key that needs to be protected. The challenge generator only processes known values, hence it does not need to be resistant against information leakage. The one-way function also has reduced physical security requirements since it does not process a constant secret. For each master challenge, the input to the one-way function is different and unpredictable. Hence, differential attacks such as DPA and CPA are not feasible. Therefore only implementation attacks that can reveal information with a single input, e.g. a simple power analysis, need to be considered. But defending against these types of side-channel attacks is much easier than defending against differential side-channel attacks and most hardware block-ciphers are secure against simple power analysis. Hence, to attack such a system using implementation attacks, the attacker would need to directly attack the Arbiter PUF and not the digital post-processing.

In the following we will assume that a design as depicted in Figure 2 is used with a block cipher such as PRESENT as the challenge generator and one-way functions as well as an 80-bit shift register to store the 80 individual response bits. However, since our attacks will actually directly target the 128-bit Arbiter (or the registers storing the Arbiter response) and not the digital pre- or post-processing the results are also valid for other designs that use an Arbiter PUF with known challenges.

[1]This assumes that n is chosen sufficiently large and the PUF has enough entropy that brute-force attacks are not possible.

III. Power Side-Channel Attack on Arbiter PUFs

In this section we will take a closer look at the resistance of Arbiter PUFs towards passive power side-channel attacks. We assume that the attacker does not have access to the challenge and response pairs because a controlled PUF design as described in section II-C is used. The question is if the attacker can gain enough information out of power side-channel measurements of the PUF to successfully model it. In the following we will first take a closer look at the power consumption of the Arbiter PUF and then show how this information can be used to attack controlled PUFs in a combined machine learning and power side-channel attack.

A. Power Consumption of Arbiter PUFs

To evaluate information leakage of an Arbiter PUF in the power consumption we performed some simulations to test the power behavior of an 128-bit Arbiter PUF. Figure 3 shows the power consumption of the tested 128-Bit Arbiter PUF in 45nm technology. The PUF is the same design as described in Section II-C with one addition: the result of the PUF is stored in a register. It is a reasonable assumption to assume that the response bit is stored in a register, since the PUF response needs to be processed further. There are two points of interest in the power traces. One point of interest is when the Arbiter at the end of the PUF determines whether the output is 1 or 0 and the second point of interest is when the result is stored in the register. In this simulation the register was reset before the PUF execution. Figure 4 shows the correlation of 2000 power traces with the correct response bits as well as the correlation with response bits with a prediction accuracy between 50% to 90%. There is a strong correlation of the correct response bits and the power traces at two time instances. At around 1350 ns when the PUF evaluates the response the power traces show a correlation of -0.32 and when the response bit is stored in the flip flop at 1600 ns the correlation is close to 1.

The reason why the correlation is negative at 1350 ns is because the power consumption is actually higher if the response bit is 0 instead of 1 in this design due to the implementation details of the Arbiter PUF. The fact that a correlation of close to 1 is achieved during the storing of the result in the register is not very surprising since it is well known that a register has a large power consumption when the output value changes. For prediction accuracies below 100% the correlation coefficient decreases linearly. Hence, the correlation coefficient directly relates to the model accuracy and it is possible to distinguish PUF models with a high accuracy from PUF models with a low accuracy using the correlation coefficient.

In this noise-free simulations it is actually possible to directly read out the response bit from the power traces. However, in practice this will not be possible due to various noise sources.

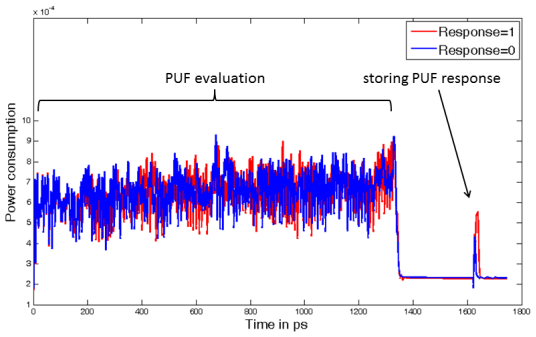

Fig. 3. Two power traces of an 128-bit Arbiter PUF for two different challenges, one challenge with a response of 1 and one with a response of 0.

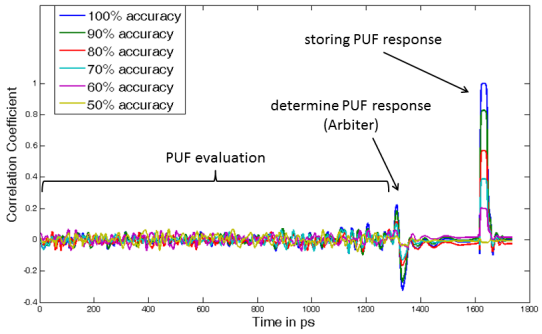

Fig. 4. The correlation of responses with different accuracies with 2000 simulated power traces of a 128-bit Arbiter PUF.

In our controlled PUF design the response of the PUF is stored in an 80-bit shift register. The assumption that the result of the arbiter is stored in a n-bit shift-register is very reasonable if a design is used in which the PUF is called n times before the result is processed. Using a shift-register is the most common and most efficient approach to store data in this case. The power consumption of a shift register follows a Hamming distance model: If two consecutive bits are different, then the stored values in the registers change and generate a large power consumption. On the other hand, if two consecutive bits are the same then the values do not change and hence consume only a small amount of power. The power consumption of the shift-register is therefore directly proportional to the Hamming distance between the current state of the shift register with the previous state.

During the evaluation of the Arbiter on the other hand the power consumption is independent of the previous response bit. This is due to the fact that the PUF is always set to the same state before the race signal is applied. Hence, the data-dependent power consumption during the end of the evaluation phase of the Arbiter PUF depends on a Hamming weight model. We have actually omitted a third time instance in which the power consumption directly depends on the response bit. When the Arbiter PUF is reset to the initial state to prepare for the next race signal there is again a data-dependent power consumption comparable to the power consumption during evaluation.

It should also be noted that after the power-up the shift register will contain all zeros. In this specific case the power consumption of storing the first response bit also follows the Hamming weight model. An attacker could power off the device between each measurement if he wanted to use the Hamming weight power model together with the power consumption of the register. In our attack, the Hamming distance power model on the shift register outperformed the Hamming weight power model so that this seems to be unnecessary or counter-productive for most attacks.

In practice, the side-channel measurements will contain noise from various sources. The different noise sources are typically physical noise, measurement noise, model matching noise, and algorithmic noise. Physical noise sums up noise sources such as supply noise, thermal noise, and temperature differences. Measurement noise is added by the measurement setup and includes noise added by the digital sampling or low pass filters that are inevitably added by the measurement setup. The assumed power model such as Hamming distance or Hamming weight is only an approximation of the real power consumption of the device. For example, in the Hamming distance power models it is assumed that a switching of 1 to 0 has the same power consumption as the switching from 0 to 1. In practice, a switching from 1 to 0 might have a slightly different power consumption than the switching from 0 to 1. This mismatch between the real power consumption and the used power model is called model matching noise. Algorithmic noise describes the power consumption caused by parts of the chip that are not part of the power model of the attack but run in parallel. In the case of the controlled PUF design, algorithmic noise could for example be the power consumption of the challenge generator if it runs in parallel to the PUF, state machines or unrelated parts of the chip such as communication logic.

It is usually assumed that each of these noise sources are independent and there is an additive effect between all the noise sources so that their overall effect can be approximated by a Gaussian distribution [23]. Physical noise and measurement noise can be reduced by averaging over several measurements. Algorithmic noise and model matching noise on the other hand can often not be reduced using averaging since the noise is usually constant for a given input. Only if the algorithmic noise is independent from the input to the target IP core, e.g. a global counter, can averaging be applied to reduce the signal-to-noise ratio.

Predicting the added noise level is very difficult since it depends on too many factors. Therefore different noise levels are used in the experiments to show that even in the presence of substantial noise a side-channel attack is possible. The power traces in the remainder of this work are simulated as follows: At first the power traces with the (idealized) used power model are computed. In a 1-bit Hamming distance power model this means that if the current response is different from the previous response a 1 is assigned as the power value, otherwise a 0. To this noise-free power model Gaussian noise with a mean value of μ and standard deviation of σ, denoted as $\mathcal{N}(\mu, \sigma^2)$, is added to simulate the various noise sources. Note that the mean value μ does not have any influence on the correlation coefficient and therefore in the following $\mathcal{N}(0, \sigma^2)$ is used.

The metric $\mathcal{N}(0, \sigma^2)$ might not be very intuitive to the reader and therefore a second metric is used as well. The power consumption during the rising edge of the clock is in many designs roughly proportional to the number of switching registers. To get a rough idea of how much noise $\mathcal{N}(0, \sigma^2)$ is we also represent the noise as the amount of switching registers that would add algorithmic noise equivalent to $\mathcal{N}(0, \sigma^2)$. The amount of noise added by n independently and randomly (with a probability of 50%) switching registers with an idealized power model is approximately Gaussian with a mean of $\mu = n * 1/2$ and a standard deviation of $\sigma = \sqrt{(n * 1/4)}$.

The question is how much noise is reasonable. Since there are so many factors influencing the power consumption and measurements it is difficult to predict the noise level and one will encounter many different noise levels in practice. To give the reader an idea of how much noise can roughly be expected we used results from successful CPA attacks on various different platforms as a comparison. Assuming a Gaussian noise distribution, it is possible to compute the approximate noise level in these attacks from the provided correlation coefficients (seee Appendix A for details).

To give the reader an idea of the typical amount of noise in a side-channel attack, Table I gives an overview of some successful CPA attacks found in the literature and their corresponding correlation coefficients and noise levels.

B. Combining CPA with Machine Learning

The previous section showed that we can distinguish PUF models that have a high model accuracy from PUF models with a lower model accuracy using the correlation coefficient as a metric. This indicates that power measurements can be useful in attacking a PUF. However, due to noise it is not possible to directly read out specific response bits from the power measurements. Therefore machine learning techniques such as SVM that require challenge and response pairs do not work. But there exists a machine learning technique that can easily be combined with correlation power analysis: Evolution Strategies (ES). The idea of ES is to randomly generate PUF models with different delay values and then test which of these models perform best, i.e. which are the *fittest*. The fittest models are then kept as *parents* for the next *generation*, while the other models are discarded. In the next generation, *children* are derived from these parents by randomly modifying the delay values of the parent models. In the next step the fitness of these children is evaluated and new parents for the next generation are chosen from these children. The idea behind this approach is that by always keeping only the best PUF models, the PUF models gradually become more accurate. This process is repeated and eventually the PUF models model the PUF with a very high accuracy.

Target Design	Power Model	cc	$\mathcal{N}(\mu_N, \sigma_N^2)$,	Noise Registers	Used Traces
PIC16F886 [18]	8-bit HD	≈ 0.75	$\approx \mathcal{N}(0, 5.1)$	≈ 12.5	100
Yubikey 2 [18] (EM)	8-bit HW	≈ 0.3	$\approx \mathcal{N}(0, 42)$	≈ 161	300
Virtex-4 Bitstream [17]	1-bit HD	≈ 0.08	$\approx \mathcal{N}(0, 39)$	≈ 155	60000
Virtex-5 Bitstream [17]	1-bit HD	≈ 0.04	$\approx \mathcal{N}(0, 156)$	≈ 623	90000
SHA-1 EEPROM [18]	8-bit HW	≈ 0.1	$\approx \mathcal{N}(0, 400)$	≈ 1600	2500
Kintex-7 FPGA [10]	8-bit HW	≈ 0.1	$\approx \mathcal{N}(0, 400)$	≈ 1600	6000
Yubikey 2 [18] (Power)	8-bit HW	≈ 0.06	$\approx \mathcal{N}(0, 1110)$	≈ 4430	6400
Stratix II Bitstream [18]	8-bit HD	≈ 0.05	$\approx \mathcal{N}(0, 1600)$	≈ 6400	400000
Mifare DesFire [19] (EM)	4-bit HD	≈ 0.01	$\approx \mathcal{N}(0, 10000)$	≈ 40000	250000

TABLE I

EXAMPLE CORRELATION COEFFICIENTS (cc) AND CORRESPONDING NOISE LEVELS, DEPICTED AS $\mathcal{N}(\mu_N, \sigma_N^2)$ AND THE NUMBER OF CORRESPONDING NOISE REGISTER, OF CPA ATTACKS ON DIFFERENT ARCHITECTURES: A PIC16F886 MICROCONTROLLER THAT IS USED IN AN RFID ACCESS CONTROL SYSTEM [18], THE YUBIKEY ONE-TIME PASSWORD TOKEN THAT USES AES [18], THE DS2432 AND DS28E01 SHA-1 HMAC PROTECTED EEPROM FROM MAXIM INTEGRATED [18], VIRTEX-4 AND VIRTEX-5 BITSTREAM ENCRYPTION BASED ON AES-256 [17], AN AES-128 IMPLEMENTATION ON A 22NM KINTEX-7 FPGA [10], AND AN EM ATTACK ON THE CONTACTLESS MIFARE DESFIRE SMARTCARD (POTENTIALLY WITH SOME SIDE-CHANNEL COUNTERMEASURES) [1]. PLEASE NOTE THAT THE NOISE LEVELS ARE ONLY APPROXIMATIONS BASED ON THE CPA VALUES PROVIDED BY THE CITED PAPERS AND ARE ONLY MEANT TO GIVE THE READER A ROUGH IDEA OF THE EXPECTED NOISE LEVELS IN A SIDE-CHANNEL ATTACK.

Using ES to attack PUFs is not new and ES has already been successfully applied to attack feed-forward Arbiter PUFs [21]. The advantage of ES is that it is not based on solving any equations. For ES to work it is only necessary to have a way to distinguish which PUF models from a given set of PUF models are the fittest. Typically, this is done by comparing the modeled response bits with the measured response bits. The models with the highest match rate (accuracy) are the fittest. But any other fitness test that can distinguish good PUF models from bad PUF models with a high probability can be used. As discussed, it is possible to use power measurements and the correlation coefficient to distinguish PUF models that have a high model accuracy from models with a small model accuracy. Therefore, correlation power analysis and ES can easily be combined: ES is used to generate potential PUF models and correlation power analysis is used to test the fitness of these models. ES also has the advantage that its random nature makes it quite resistant to noise. If a good PUF model is falsely discarded and instead a worse PUF model is chosen as a parent for the next generation, this will slow down the convergence to the optimal solution. But as long as more good PUF models are chosen as parents than bad models, the ES still works and can converge to a near optimal solution. This property is very helpful for noisy environments such as the discussed power side-channel.

There are many different variants of ES that mainly differ in how the child instances are derived from the parent instances. We tested several different methods and parameters. The optimal strategy depends on many aspects such as number of stages of the PUF, noise, available traces and the computation environment. We tested the (μ/γ)-ES approach without recombination with and without self-adoption. This approach has previously been successfully applied to feed-forward Arbiter PUFs in [21]. However, in our experiments the CMA-ES outperformed the (μ/γ)-ES and performed very well among all ranges of noise level. CMA-ES uses a weighted recombination approach with self-adoption. Details of this machine learning attack can be found e.g. in [8]. The same parameters as proposed in [8] were used with the exception that we increased the child-population since we are dealing with a very noisy environment.

C. Results

As mentioned in Section III-A the design can be attacked using two different power models: A 1-bit Hamming weight (HW) power model in which each challenge is independent from the previous challenge and a Hamming distance power model (HD) that targets the shift registers where the responses are stored. Figure 5 shows the result of a combined correlation CMA-ES (referred to as CCMA-ES from here after) using a 1-bit HW power model and a noise of $\mathcal{N}(0, 25)$ which is equivalent to roughly 100 randomly switching registers. In Figure 6 the same attack is shown with a 1-bit HD power model. The CCMA-ES is a non-deterministic method, hence, if run with the same inputs, it can yield different results. In the figures 100 independent runs with the same PUF instance and power simulations were executed. The best run achieved an accuracy of 93.5% for the 1-bit HW power model compared to an accuracy of 95.6% for the 1-bit HD power model. However, while all runs converged to a solution close to the maximum with the 1-bit HW power model, with the 1-bit HD power model several runs did not converge to a solution close to the maximum. Hence, while the HD power model achieves higher accuracies if a run converges, there is a much higher chance that a run does not converge for the HD power model compared to the HW power model.

The reason for this is that a single miss-predicted response bit actually influences two bits in the HD power model(the current and next value). This leads to a relation between the prediction accuracy and correlation coefficient that unlike the Hamming weight power model is not linear. Figure 8 shows the relationship of accuracy versus correlation coefficient for the HW as well as HD power model. It is still true for the HD power model that a higher accuracy yields a higher correlation coefficient. However, this correlation coefficient increases slower for

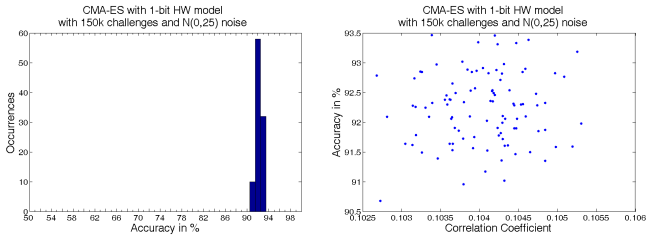

Fig. 5. Result of a CMA-ES with a 1-bit HW power model, 150k challenges and a noise of $\mathcal{N}(0, 25)$ which is equivalent to 100 switching registers.

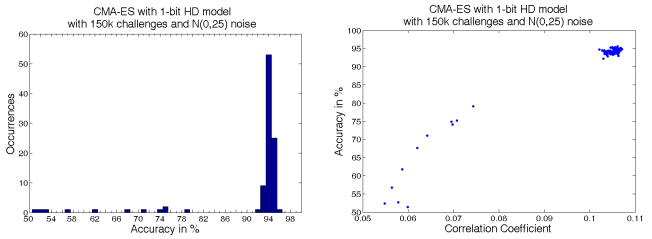

Fig. 6. Result of a CMA-ES with a 1-bit HD power model, 150k challenges and a noise of $\mathcal{N}(0, 25)$ which is equivalent to 100 switching registers.

low accuracies compared to higher accuracies. What this means in practice is that the HD power model does not perform as good as the HW power model while the prediction accuracy is fairly low. But for higher accuracies the HD model actually outperforms the HW model since for higher accuracies the correlation coefficient increases faster in the HD model compared to the HW model. This trend is independent of how many bits are used for the HW power model, but an 80-bit HD model shows a higher variance from the ideal curve than a 1-bit HD model. This explains why the HD model achieves higher accuracies while simultaneously having a lower rate of runs that converge.

If we assume that the result is stored in an 80-bit shift register then the power consumption during the storing of a response bit follows an 80-bit HD power model, not a 1-bit HD model. Using an 80-bit power model greatly increases the signal to noise ratio. Figure 7 shows the same attack as before with the 80-bit HD power model. In this case accuracies of up to 99.9% are achieved. Therefore this power model is recommended in practice, assuming the design allows it. It is also important to note that in the controlled PUF a single execution of the protocol requires 80 responses, hence, a single measurement actually contains 80 challenges. For the example of 150k challenges this means that only around 150,000/80=1875 measurements are needed.

Figure 9 shows the result for the CMA-ES attack with different noise levels and different number of traces with this 80-bit HD power model. One can see that noise can be counteracted by increasing the number of used challenges. Model accuracy is only one metric to determine the success of the attack. Another valid metric is the number of times the correct 80-bit string is predicted. The output of the

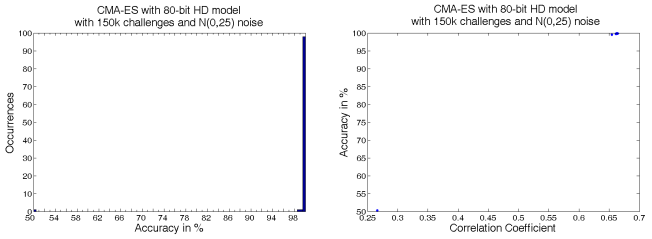

Fig. 7. Result of a CMA-ES with an 80-bit HD power model, 150k challenges and a noise of $\mathcal{N}(0, 25)$ which is equivalent to 100 switching registers.

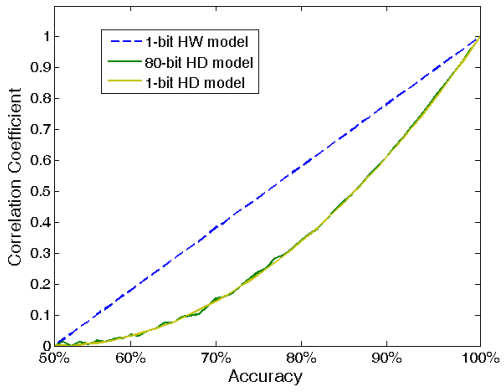

Fig. 8. Relation between the correlation coefficient and the prediction accuracy for the Hamming weight power model as well as the Hamming distance power model. For this simulation 1 million random response bits were used and no noise was added.

controlled PUF is the hash value of 80 response bits and therefore an attacker can verify if he has predicted the correct 80-bit string. To correctly predict these strings is the ultimate goal of the attacker since this will enable him to impersonate the PUF device.

Furthermore, if the attacker manages to predict the PUF with a high enough accuracy to find a single 80-bit match, the attacker can perform a second machine learning attack that uses this success metric to achieve accuracy exceeding 99%. The idea of this second machine learning attack is simple: Instead of still relying on the (noisy) power side-channel to evaluate the fitness of the PUF models, the attacker uses the number of string matches to determine the fitness. Otherwise the same CMA-ES is used as with the power model. Since this string match analysis is noise free, much larger accuracies can be achieved than with the power side-channel. Given enough challenges, accuracies beyond 99.99% are achieved. However, you cannot use this model until your PUF models have reached accuracies large enough that at least one string match is found. Hence, it is not possible to directly use this metric to attack the PUF without using a side-channel attack first.

The most effective attack is therefore a two step approach: In the first stage a combined machine learning and power side-channel attack is performed to model the PUF with a large enough accuracy to predict some of the 80-bit

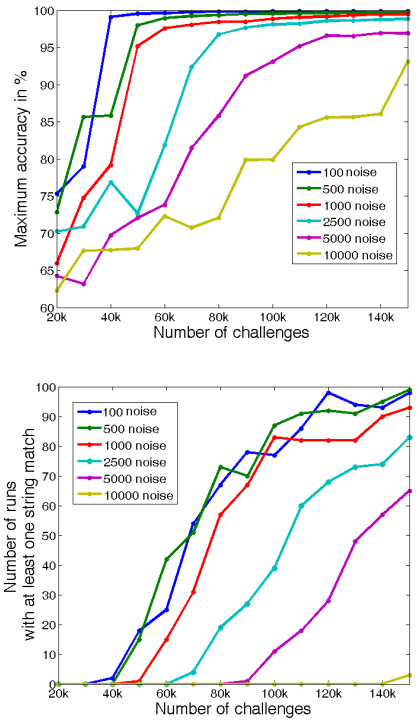

Fig. 9. Result of of 100 runs of an 80-bit CCMA-ES attack with different levels of noise. The noise level is expressed by the number of randomly switching registers that would generate the same amount of noise. On the left the maximum achieved accuracy with 100 runs is shown while on the right the number of runs that achieved an accuracy high enough to find at least one string match is shown.

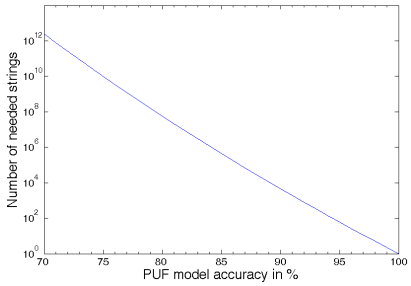

Fig. 10. The number of needed strings so that the probability of a match is at least 50%.

strings. In the second step, a machine learning attack using the number of string matches as a fitness function is used to achieve prediction accuracies beyond 99%. Figure 10 shows the required number of 80-bits strings for different PUF accuracies to find a match with a probability of at least 50%. This figure can be used to roughly determine the needed model accuracy that needs to be achieved using the power side-channel attack so that in a second step a CMA-ES based on string matches can be performed. For example, with an accuracy of 90% only about 10k traces are needed to find a string match.

We tested this two-step approach by adding noise to the power measurement equivalent to a million random switching registers and the first step based on the power

Noise Registers	Challenges	Traces	Accuracy
100	30,000	375	97
1,000	50,000	625	95
10,000	150,000	1,875	93
100,000	1,000,000	12,500	93.6
1,000,000	10,000,000	125,000	88.2

TABLE II
REQUIRED NUMBER OF CHALLENGES FOR DIFFERENT NOISE LEVELS TO ACHIEVE AN ACCURACY LARGE ENOUGH TO FIND A STRING MATCH WITH AN 80-BIT HD POWER MODEL. WITH SUCH A STRING MATCH A SECOND MACHINE LEARNING ALGORITHM ACHIEVED ACCURACIES BEYOND 99%.

Noise Registers	Challenges	Traces	Accuracy
100	80,000	1,000	89
500	450,000	5,625	90
1,000	750,000	9,375	89
5,000	4,000,000	50,000	87
10,000	7,500,000	93,750	88

TABLE III
REQUIRED NUMBER OF CHALLENGES FOR DIFFERENT NOISE LEVELS TO ACHIEVE AN ACCURACY LARGE ENOUGH TO FIND A STRING MATCH WITH A 1-BIT HW POWER MODEL. WITH SUCH A STRING MATCH A SECOND MACHINE LEARNING ALGORITHM ACHIEVED ACCURACIES BEYOND 99%.

consumption achieved a model accuracy of 88%, which was enough to launch a second machine learning attack based on string matches. This second attack then was able to model the PUF with an accuracy of 99.99%. For this attack 10 million challenges, for which only 128k measurements are needed, were used. Table II gives an overview of the required number of traces for different noise levels so that at least one string match is found. In each case it was then possible to achieve accuracies beyond 99% with the second machine learning attack. Table III shows the same analysis for the 1-bit HW power model. These noise levels are very large while the required number of traces are comparably small for a side-channel attack (see Table]I for comparison). Hence, it is reasonable to say that power side-channel attacks on controlled PUFs are successful even in the presence of considerable noise.

IV. Attacking other PUF designs

The main goal of this paper is to show that Arbiter-PUFs are indeed vulnerable to side-channel attacks. To do this, we used controlled PUFs as our main motivation example. Nevertheless, the same or similar methods can be applied to other constructs that rely on Arbiter-PUFs as a building block. For example, this attack is directly transferable to the PUF protocol proposed by van Herrewege et.al. [9]. In this protocol individual PUF responses are generated and then applied to a reverse fuzzy extractor to build a mutual authentication protocol. The only difference from an attacker's perspective is that the individual response bits are not applied to a hash function but to a reverse fuzzy extractor. Therefore, the attack can directly be applied to the reverse fuzzy extractor PUF protocol.

To a certain degree the attack can also be applied to the

Lightweight PUFs proposed in [14] and with large drawbacks to XOR-Arbiter PUFs [13]. In these PUF designs k individual Arbiter PUFs are present and then combined with each other using XORs. If the responses from the individual PUFs are stored in a register before they are combined, we can again use a hamming-distance power model. However, if the outputs of the Arbiters are directly combined through combinatorial logic the power model changes. How exactly needs to be evaluated, but it is likely that the power model follows a Hamming Weight model, mainly due to glitches.

The biggest difference between Lightweight PUFs and XOR-Arbiter PUFs from an power side-channel attack perspective is that in the Lighweight PUFs each of the k Arbiter PUF gets a different challenge. This allows a divide-and-conquer approach to attack the Lightweight PUF. We can attack each PUF individually and consider the responses of the other PUFs as noise. Due to its high noise resistance, it should therefore be possible to attack Lighweight PUFs using CCMA-ES. Increasing the number k of used PUF instances only has a small impact on the number of needed traces since we attack one PUF at a time.

However, this divide-and-conquer approach cannot be used for an XOR-Arbiter PUF, since every PUF instance gets the same challenge.When an attacker wants to attack a single PUF from an XOR-Arbiter PUF with n XORs, the responses of the other $k-1$ PUFs cannot be seen as independent noise. All n PUFs get the same challenge and hence the n responses are all related to each other. Hence, one needs to model all n PUF instances at the same time and therefore the attack complexity grows considerably when n is increased. The optimal strategy to attack XOR-Arbiter PUFs using a power side-channel attack is therefore an interesting research problem.

V. Fault Attack on Arbiter PUFs

So far we have seen that Arbiter PUFs are vulnerable to passive side-channel attacks. In this section we will take a closer look at their resistance against active attacks. It was often assumed that the fact that the PUF changes its behavior if it is being tampered with increases the security of PUFs against implementation attacks. However, in this Section we will see that the unreliability information can instead be used to successfully model a PUF.

A. Impact of Noise and Environmental Conditions on Arbiter PUFs

Arbiter PUFs have a problem with unreliability in practice, which means that the same challenge does not always generate the same response. There are two sources that can cause an Arbiter PUF to change its response bit: thermal noise and changes in environmental conditions. Thermal noise adds approximately Gaussian noise to the delay value of each individual execution of the PUF. If the delay difference between the top and bottom signal is small for a given challenge, this noise can switch the response bit by changing a previously positive delay difference into a negative delay difference and vise versa. The closer the delay difference is to zero, the more likely it is that the response bit changes.

The second reason why a response bit might flip is due to changes in the environment. For example, it is well known that changing the supply voltage or operation temperature changes the propagation delay and rise and fall time of a CMOS transistor. The magnitude of this change depends directly on the transistor sizing as well as the process variations. Hence, the Arbiter PUF behaves differently at different supply voltages. For each challenge the delay difference will either increase or decrease when the supply voltage is reduced. The amount of the increase or decrease depends on the PUF instance as well as the challenge.

This has some interesting implications. First of all, the attack by Delvaux et.al. [6] does not work with environmental noise in the same way as it does with thermal noise. The added delay is not a Gaussian random variable but a deterministic function of the supply voltage (or other environmental conditions) whose slope depends on the PUF instance as well as the challenge. Since the slope of the function is different for each challenge, two challenges that have the same delay difference might behave differently when the supply voltage is altered. One challenge might flip while the other does not. Nevertheless, if the delay difference is close to 0 it is much more likely that the response bit flips than if the absolute delay difference is large. In Figure 11 the delay differences for an 128-bit Arbiter PUF is shown. The delay difference is approximately Gaussian and ranges between -150ps and +150ps. When the supply voltage was altered from the nominal voltage of 1.1V to 1V and 1.2V roughly 6% of the response bits flipped. In Figure 11 the delay difference of the challenges that flipped are depicted in black. Only traces that were close to zero (between -12ps and +13 ps) flipped. The distribution of the delay difference of the flipped traces is again approximately Gaussian and the closer to zero the more likely that the response flipped. However, while a trace who's absolute delay difference was larger than 13ps never flipped, a lot of traces whose delay difference were less than 13ps did not flip. Hence, it is not possible to directly read out the delay difference from the information when a challenge flips. But we still get enough information to reliably model the PUF as we will see in the following.

For the controlled PUF design from Section II-C to be useful an Arbiter PUF with a high reliability is needed. Ideally, the PUF should be resistant to thermal noise and changes in the environmental conditions. However, this is very difficult to achieve for all possible environmental conditions. Therefore, it is much more likely that the PUF will be designed to be reliable under normal operation conditions as defined in the specifications. For example, techniques such as the one described in [5] can help to counteract changes in temperature or supply voltage. Usually, these techniques are optimized for a small range of operating conditions and will not work outside the spec-

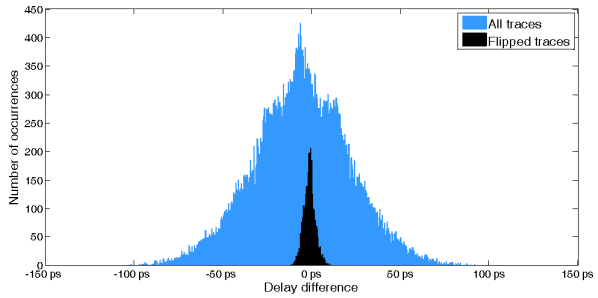

Fig. 11. The delay difference in pico seconds between the top and bottom signal after the last stage for 49k traces. Colored in blue are the delay differences of all traces and in black are the delay differences for the traces whose output flipped when the supply voltage was changed from 1.1V to 1V and 1.2V.

ification of the device. Furthermore, it is likely that the controlled PUF will depend on a stable voltage source or that the protocol allows to collect challenges and responses for different working conditions. Hence, even if a controlled PUF might be very reliable under normal working conditions, it is reasonable to assume that an attacker with access to the device can alter the environmental conditions to make the PUF unreliable. In particular, it is likely that an attacker with physical access to the PUF is able to apply a supply voltage that is outside of the specifications.

In the following, we will again assume that a controlled PUF design as described in Section II-C is used. We furthermore assume that the attacker can alter the voltage supply in a way that the chip is still functioning but some PUF responses will flip. The advantage of changing the supply voltage compared to e.g. changing the temperature is that it can be done easily in a fast and automatic fashion. It is possible to build a set-up that will alter the supply voltage of the chip for only a few clock cycles. This has the advantage that only one or a few sub-challenges of the controlled PUF design will be subject to the altered supply voltage. Other environmental changes, e.g. generating a strong magnetic field might have the same or similar effects. What kind of faults are feasible and effective besides voltage manipulations is an interesting future research direction. Basically any change to the PUF that can be applied to a single sub-challenge (or few sub-challenges) and that changes the delay difference so that only a subset of response bits flip can be used for our fault attack.

We performed Monte Carlo simulations using HSpice for an 128-bit Arbiter PUF in 45nm technology with a nominal voltage of 1.1V and with 1V and 1.2V and compared the response bits to determine which responses flipped when the voltage was altered. It turns out that when reducing the voltage from 1.1V to 1V only 1.9% of the responses flipped while increasing the voltage from 1.1V to 1.2V resulted in 4.5% of flipped bits. One reason why increasing the voltage had larger effects on the responses is that the standard cells we used were optimized for usage of 1.1V and 1V but not for 1.2V[2]. In total 6% of the challenges flipped and only 0.4% of the challenges flipped for both a voltage reduction as well as voltage increase. In the following, data from these simulations are used to test our fault attack.

B. Combined Machine Learning Fault Attack

The same CMA-ES machine learning method as in the hybrid power side-channel and machine learning attack is used for the combined machine learning fault attack. Recall that it is necessary to be able to distinguish PUF models that have a high model accuracy from PUF models with a low model accuracy for an ES machine learning attack to work. In Section III-B this was done using the power side-channel. Instead, in this attack the reliability information of the PUF under supply voltage manipulations is used to judge the PUF models. The basic idea is to take measurements of the same challenges with different supply voltages and check for which challenges the response changed. If the response bit for challenge i flipped we assign $F_i = 1$ otherwise $F_i = 0$. From Figure 11 we know that if a response bit flipped it is very likely that the corresponding delay difference was close to zero i.e. $|\Delta D| < \tau$. To test a PUF model we compute the delay difference $\Delta\hat{D}_i$ for each challenge i and assign:

$$\hat{F}_i = \begin{cases} 1 & \text{if } |\Delta\hat{D}_i| < \hat{\tau} \\ 0 & \text{if } |\Delta\hat{D}_i| > \hat{\tau} \end{cases}$$

One way to check the fitness of the PUF model is to compare the hypothesis vector $\hat{F}$ with the measured vector F by counting how often the two vectors are equal, i.e., $\sum |F_i - \hat{F}_i|$. Intuitively, the correct PUF model should have the smallest difference between the two vectors. However, using the number of mismatches is not very good to measure the fitness of the PUF instances. The number of flipped bits is relatively small and as seen in Figure 11, and not all bits flip if $|\Delta D| < \tau$. Hence, even for the correct PUF model there is a mismatch between F and $\hat{F}$, i.e. $\sum |F_i - \hat{F}_i| > 0$. A PUF model that results in a vector $\hat{F}_i = 0$ for all i might therefore have a smaller mismatch between $\hat{F}$ and F than a PUF model with a high model accuracy. Therefore, using the number of mismatches as the fitness test does not work in practice. Using the correlation coefficient between F and $\hat{F}$ on the other hand works very well. Hence, the correlation coefficient is used again to determine the fitness of the PUF models. Compared to the CMA-ES based on power side-channels, the fault based CMA-ES needs to model the parameter τ in addition to the $n+1$ delay values. But otherwise there is not much difference between the two attacks.

Figure 12 shows the result of the fault CMA-ES with 8k traces and +/- 0.1V supply voltage variation. The attack can model the PUF reliably with an accuracy of 97.7%.

[2]Note that this can also be due to the fact that the HSpice simulations are more accurate the closer you simulate to the nominal operation conditions.

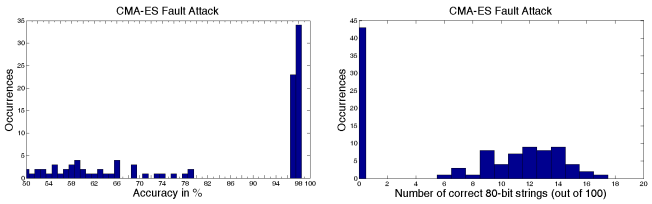

Fig. 12. The result of 100 runs of the CMA-ES Fault attack with 8k traces and +/− 0.1V supply voltage variation without additional noise. On the left the accuracy of the resulting PUF models are shown. On the right side the number of 80-bit strings that were correctly predicted by the PUF models are shown.

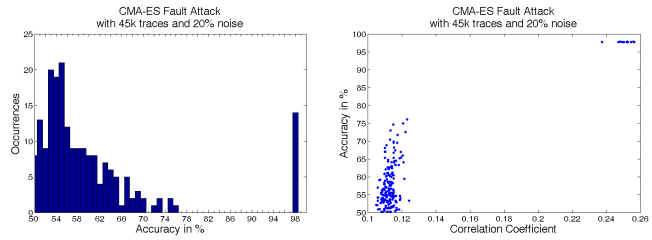

Fig. 13. The result of 100 runs of the CMA-ES Fault attack with 45k traces and +/− 0.1V supply voltage variation with 20% faulty responses.

From 100 runs 56 runs achieved an accuracy of at least 97%. These results represent an ideal case without any noise. However, when performing this attack in practice there might be measurement errors in the fault trace F. In the controlled PUF case we assume that we can alter the operating conditions for only a single sub-challenge while keeping the operating conditions constant for the other 79 sub-challenges. Since the 80 response bits are fed through a hash function we can see if at least one response bit flipped by checking if the hash value changed. But since a hash function is used, it is not possible to determine which response bit has changed or how many. It can therefore happen that a response bit flip is detected although it actually corresponds to a flip in one of the 79 sub-challenges that are not targeted. We call this case a "false positive" since F_i is set to one although no flip actually happened for the targeted challenge.

Besides a "false positive" there could also be a "false negative" in which a response bit that is supposed to flip does not flip, e.g. because the fault setup did not work correctly. It is likely that the closer $|\Delta D|$ is to zero the more unlikely it is that a false negative happens for this challenge. False negatives as well as false positives can be reduced by repeating the measurement several times. It could also happen that the targeted challenge flips due to noise instead of the voltage alteration. However, if a challenge flips due to noise it is likely that the delay difference is close to zero as well. Hence, whether the response flips due to noise or voltage alteration does not matter.

C. Results

We tested how the attack works in the presence of these noise sources. We randomly changed 20% of the bits of F which represents a false positive rate of 20% and a false negative rate of 20%[3]. Figure 13 shows the result of a CMA-ES attack with 45k challenges with 20% overall noise. The attack still works and achieves the same accuracy as the noiseless case. However, the number of challenges needed to attack the PUF increased from 8k challenges to 45k challenges and more runs do not converge than in the noise-free case. Nevertheless, the number of challenges is still fairly low considering the rather large amount of noise added. Figure 14 shows the success of the attack with different noise levels and different numbers of challenges.

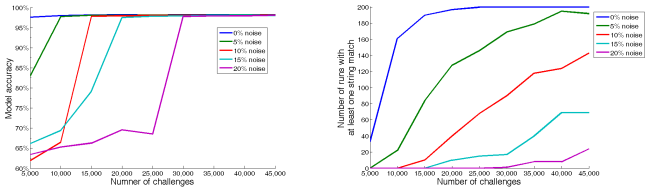

Fig. 14. Result of a Fault CMA-ES attack with 200 independent runs for different levels of overall-noise and for different numbers of challenges. On the left the highest achieved model accuracy is shown while the right figure shows the number of runs that had at least one string match.

As can be seen, the attack still works in the presence of considerable noise. Hence, fault attacks on controlled PUFs based on Arbiter PUFs are feasible. The presumed strength, that tampering with the PUF results in different responses, turns out to actually be a security vulnerability. One can therefore no longer state that Arbiter PUFs are more resistant to implementation and probing attacks than traditional cryptography.

D. Error Correction and Fault attacks

So far we have assumed that the used Arbiter-PUF is reliable enough that no error correction is needed. Current PUF design however are not reliable enough to be used without error correction. Therefore in many controlled PUF designs error correction codes are used to correct errors before the response bits are fed into the hash functions. In this case the described fault attack does not work, since the introduced error would be corrected again. However, if error correction is used, the controlled PUF does not only get a challenge c but also helper data h. This helper data is used to correct the response r before it is applied to the hash function. Error correction codes can correct up to d errors in the response. If there are more than d errors, the error correction fails.

To be able to perform a fault attack on a controlled PUF that uses error correction, the attacker therefore needs to manipulate h, so that an error at the targeted response bit does not get corrected. So an attacker needs to send a manipulated helper data h' instead of h that does not correct the introduced error. How easy it is to

[3]Since we change the bits independent of $|\Delta D|$ this can be seen as a worst case scenario.

send such manipulated helper data h' strongly depends on the used error correction and it is likely that the false-negative rate increases for such an attack. Furthermore, h' needs to be determined for every challenge. Hence, the attack complexity increases and much more challenge and responses need to be exchanged.

Hence, error correction increases the complexity of the attack but does not necessarily prevent the attack. Studying fault attacks in the presence of different error correction codes is therefore an interesting future research direction.

VI. Conclusion and implications of Side-Channel Attacks on Arbiter PUFs

The results presented in this paper show that — unlike often believed — Arbiter PUFs are not necessarily more resistant against side-channel attacks than traditional cryptographic algorithms. By combining side-channel analysis with machine learning techniques it is possible to attack controlled PUFs reliably even in the presence of considerable noise. This hybrid side-channel and machine learning approach works with passive power side-channels as well as with active fault attacks. Hence, controlled Arbiter PUFs are vulnerable to both active as well as passive side-channel attacks. It is therefore very important to also consider side-channel attacks when evaluating PUFs and protocols based on PUFs. Even designs that from a theoretical perspective would be secure against machine learning attacks are not necessarily secure once they are implemented. Which PUF structures and protocols are more resistant against side-channel attacks and how efficient side-channel countermeasures for PUFs look like is therefore an interesting new research area.

References

[1] L. Batina, B. Gierlichs, and K. Lemke-Rust. Comparative evaluation of rank correlation based dpa on an aes prototype chip. In *Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 341–354. Springer Berlin Heidelberg, 2008.

[2] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/.

[3] G. T. Becker. Intentional and unintentional side-channels in embedded systems. PhD Dissertation, University of Massachusetts Amherst, February 2014.

[4] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007*, pages 450–466. Springer, 2007.

[5] A. Cortez, V. van der Leest, R. Maes, G. Schrijen, and S. Hamdioui. Adapting voltage ramp-up time for temperature noise reduction on memory-based pufs. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, pages 35–40, 2013.

[6] J. Delvaux and I. Verbauwhede. Side channel modeling attacks on 65nm arbiter pufs exploiting cmos device noise. In *6th IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, June 2013.

[7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 149–160, 2002.

[8] N. Hansen. The cma evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[9] A. Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In A. Keromytis, editor, *Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 374–389. Springer Berlin Heidelberg, 2012.

[10] Y. Hori, T. Katashita, A. Sasaki, and A. Satoh. Sasebo-giii: A hardware security evaluation board equipped with a 28-nm fpga. In *Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on*, pages 657–660. IEEE, 2012.

[11] G. Hospodar, R. Maes, and I. Verbauwhede. Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 37–42. IEEE, 2012.

[12] D. Karakoyunlu and B. Sunar. Differential template attacks on puf enabled cryptographic devices. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.

[13] D. Lim. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology, USA, May 2004.

[14] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure pufs. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 670–673. IEEE Press, 2008.

[15] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl. Localized electromagnetic analysis of ro pufs. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*, pages 19–24, 2013.

[16] D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Side-channel analysis of pufs and fuzzy extractors. In J. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, editors, *Trust and Trustworthy Computing*, volume 6740 of *Lecture Notes in Computer Science*, pages 33–47. Springer Berlin Heidelberg, 2011.

[17] A. Moradi, M. Kasper, and C. Paar. Black-box side-channel attacks highlight the importance of countermeasures — an analysis of the xilinx virtex-4 and virtex-5 bitstream encryption mechanism. In *Topics in Cryptology Ű– CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2012.

[18] D. Oswald. Implementation attacks: From theory to pratice. Dissertation, Ruhr-Universität Bochum, September 2013. http://www.emsec.rub.de/media/attachments/files/2013/10/diss_david_oswald_2.pdf.

[19] D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power analysis and templates in the real world. In *Cryptographic Hardware and Embedded Systems (CHES)*, Lecture Notes in Computer Science, pages 207–222, 2011.

[20] U. Rührmair and M. van Dijk. Pufs in security protocols: Attack models and security evaluations. In *IEEE Symposium on Security and Privacy 2013 (SP 2013)*, 2013.

[21] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[22] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, and W. Burleson. Power and timing side channels for pufs and their efficient exploitation. Cryptology ePrint Archive, Report 2013/851, 2013. http://eprint.iacr.org/.

[23] F.-X. Standaert, E. Peeters, C. Archambeau, and J.-J. Quisquater. Towards security limits in side-channel attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 30–45. Springer Berlin Heidelberg, 2006.

Appendix A
Computing the corresponding noise level for a given correlation coefficient

The Pearson correlation coefficient is used to test the linear relation between two signals, in the case of a CPA the measured power values $\vec{x}$ and some computed hypothetical power values $\vec{h}$:

$$p = \frac{cov(\vec{h}, \vec{x})}{\sqrt{var(\vec{h})var(\vec{x})}}$$

with *var* indicating the sample variance and *con* the sample covariance. The correlation coefficient is directly related to signal-to-noise ratio of the measured signal. This makes comparing different attacks and measurement setups easy. Assuming an approximately with a Gaussian noise distribution $\mathcal{N}(\mu_N, \sigma_N^2)$[23], the power consumption follows a random variable X with $X = H + \mathcal{N}(\mu_N, \sigma_N^2)$, where H is the random variable of the assumed power model. In this specific case the correlation coefficient can be written as:

$$\begin{aligned} p &= \frac{cov(H, H + \mathcal{N}(\mu_N, \sigma_N^2))}{\sqrt{var(H)var(H + \mathcal{N}(\mu_N, \sigma_N^2)}} \\ &= \frac{cov(H, H) + cov(H, \mathcal{N}(\mu_N, \sigma_N^2))}{\sqrt{var(H)^2 + var(H)var(\mathcal{N}(\mu_N, \sigma_N^2))}} \\ &= \frac{var(H)}{\sqrt{var(H)^2 + var(H)\sigma_N^2}} \end{aligned}$$

Hence, it is possible to compute the expected correlation coefficient for a given noise distribution $\mathcal{N}(\mu_N, \sigma_N^2)$ and power model H. Similarly, it is also possible to derive the noise level given the measured correlation coefficients and the used power model H.