The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs

Georg T. Becker(✉)

Horst Görtz Institute for IT-Security, Ruhr Universität Bochum, Bochum, Germany
georg.becker@ruhr-uni-bochum.de

Abstract. In this paper we demonstrate the first real-world cloning attack on a commercial PUF-based RFID tag. The examined commercial PUFs can be attacked by measuring only 4 protocol executions, which takes less than 200 ms. Using a RFID smartcard emulator, it is then possible to impersonate, i.e., "clone" the PUF. While attacking the 4-way PUF used by these tags can be done using traditional machine learning attacks, we show that the tags can still be attacked if they are configured as presumably secure XOR PUFs. We achieved this by using a new reliability-based machine learning attack that uses a divide-and-conquer approach for attacking the XOR PUFs. This new divide-and-conquer approach results in only a linear increase in needed number of challenge and responses for increasing numbers of XORs. This is in stark contrast to the state-of-the-art machine learning attacks on XOR PUFs that are shown to have an exponential increase in challenge and responses.

Hence, it is now possible to attack XOR PUF constructs that were previously believed to be secure against machine learning attacks. Since XOR Arbiter PUFs are one of the most popular and promising electrical strong PUF designs, our reliability-based machine learning attack raises doubts that secure and lightweight electrical strong PUFs can be realized in practice.

Keywords: PUFs · Machine learning · Real-world attacks · XOR PUFs

1 Introduction

Physical Unclonable Functions (PUFs) have gained extensive research attention since they were first proposed in 2001 [21]. PUFs use the inherent manufacturing differences within every physical object to give each physical instance a unique identity. While the first proposal was based on light scattering [21], in the same year the first electrical PUF, the Arbiter PUF, was proposed by Gassend *et al.* [8]. PUFs have some unique characteristics that make them an interesting research target for lightweight authentication schemes as well as anti-counterfeiting solutions. Furthermore, PUFs have gained a lot of attention as a secure key generation and storage mechanism. One of the key features is their "unclonability", the fact that it should be impossible to build two physical instances of a PUF that have the same characteristics.

T. Güneysu and H. Handschuh (Eds.): CHES 2015, LNCS 9293, pp. 535–555, 2015.
DOI: 10.1007/978-3-662-48324-4_27

PUFs are usually divided into two categories: *weak PUFs* and *strong PUFs*. A strong PUF can be queried with an exponential number of challenges to receive an exponential number of responses. They can be used in authentication protocols as well as for key generation and storage. Weak PUFs on the other hand only have a very limited challenge space and can only be used for key generation and storage. In practice, PUFs have two main drawbacks. One drawback is that PUFs are susceptible to environmental conditions and noise and therefore their responses are unreliable. To counter this unreliability, either error correction is needed or some false response bits need to be tolerated by the used PUF protocol. The other major drawback is that existing electrical strong PUFs can be simulated in software and the required parameters for such a software model can be approximated using machine learning techniques. This is particularly true for the Arbiter PUF. Several constructions based on the Arbiter PUF have been proposed such as the XOR PUF [31] and the Feed-Forward PUF [15]. They have have in common that some non-linearity is added to make machine learning attacks more difficult. While it is possible to attack XOR PUFs using machine learning for small numbers of XORs, XOR PUFs are widely assumed to be secure against machine learning attacks if enough XORs are used [12,28,29]. This makes the XOR PUF one of the most promising strong PUF designs.

PUFs have already made the step out of the scientific research labs into commercial products. For example, NXP and Microsemi use PUF-based key storage in some of their products. But PUFs are not only used for secure key storage in commercial products. There are also PUF-based RFID tags available. These extremely lightweight tags are promoted as a secure alternative to memory tags and are proposed as an anti-counterfeit solution for medical drugs and luxury products. They can furthermore be used for access control and payment applications. In this paper we take a closer look at such commercial PUF-based RFID tags. We show that it is possible to perform a machine learning attack on these RFID tags with measurement times below a second. It is then possible to "clone" the RFID tags using an RFID smartcard emulator.

While the examined PUF architecture in these tags is extremely weak and can be attacked in seconds using traditional machine learning attacks, we also show that PUF constructions that are supposedly secure can be attacked. We achieve this by using a new reliability-based machine learning attack. This new attack scales very well with the number of XORs and therefore increasing the number of XORs cannot defeat it.

1.1 Related Work

There has been extensive research on finding different electrical PUFs. Popular weak PUF proposals are for example ring-oscillator PUFs [31], SRAM-PUFs [9], sense-amplifier based PUFs [10,23] or bus-keeper PUFs [30]. Most strong PUF designs are variants of the Arbiter PUF such as the XOR PUF [31], Feed-Forward PUF [15] or the Lightweight PUF [18]. Compared with the number of papers that either propose new PUFs or discuss their performance in terms of reliability and uniqueness, e.g., [14,16,33] relatively little research has focused on

machine learning attacks. Most of our knowledge of machine learning attacks on PUFs is based on the 2010 CCS paper by Rührmair *et al.* [29]. In their paper Rührmair *et al.* showed that XOR PUFs, Feed-Forward PUFs and Lightweight PUFs can be attacked using ES and LR-based machine learning algorithms. However, their results showed that the required number of challenge and response pairs (CRPs) to model an XOR PUF grows exponentially with an increase in the number of XORs. Therefore, they concluded that XOR PUFs can withstand these machine learning attacks if enough XORs are used. Their initial analysis was conducted on simulated data but they later verified the results by using measurements taken from an ASIC [26].

Besides the search for PUFs which can withstand machine learning attacks, protocol and system level countermeasures have been proposed to combat machine learning. The first system level countermeasure is the idea of controlled PUFs [7]. In a controlled PUF, the PUF responses are not directly revealed but instead only the hash value of several PUF responses is transmitted. Since PUFs are unreliable, such controlled PUFs also need an error correction mechanism. Other more lightweight proposals are the Reverse Fuzzy Extractor [13] or the the Slender PUF protocol [17,25]. A good overview of different proposals and their security can be found in [6]. Another line of research examines different side-channel attacks on PUFs, e.g., in [3,5,19,24,32]. PUFs have also gained attention as building blocks for cryptographic protocols with formal definitions of strong PUFs [1,4,20]. However, it was pointed out that most existing PUFs do not match the formal PUF models [27].

1.2 Contribution and Organization

The main contribution of the paper can be summarized as follows:

1. We present the first cloning attack on a commercial strong PUF-based RFID tag, demonstrating the gap between the promised and achieved security with strong PUFs in practice. The employed 4-way PUF can be attacked using machine learning with measurement times of less than 200 ms. An RFID smartcard emulator with hardware costs of less than $25 can then be used to "clone" the PUF.
2. A new reliability-based machine learning attack on XOR PUFs is introduced. Even XOR PUFs with parameters previously considered computationally infeasible to attack using machine learning cannot withstand this new attack. Our results show that the widely believed assumption that the number of required responses for a machine learning attack on XOR PUFs increases exponentially with the number of XORs is wrong. Hence, plain XOR PUFs cannot be used in practice, regardless of the parameters used.

Besides these main results, our analysis of the RFID tags also gives important insight into how many CRPs an attacker can collect in practice. In the security analysis of many designs less than a million challenge and responses are used. In contrast, the measurements of the PUF-based RFID tags show that it is quite realistic that an attacker can collect billions of responses.

The reliability-based machine learning attack introduced in this paper is based on a machine learning algorithm called CMA-ES. This machine learning algorithm together with the Arbiter PUF is introduced in the background section. In Sect. 3, the targeted commercial PUF-based RFID tags are discussed in detail, while in Sect. 4 the cloning attack on these tags is discussed. In Sect. 5 the new reliability-based machine learning attack on XOR PUFs is introduced. Finally, the implications of these attacks are discussed in the last section.

2 Background

The PUF tags that we examine use an n-way PUF, which is a variant of the Arbiter PUF. In this Section we will first introduce the Arbiter PUF and explain how the Arbiter PUF can be modeled in software and then how the needed parameters can be approximated using machine learning.

2.1 Arbiter PUF

The schematic of an Arbiter PUF can be seen in Fig. 1. An Arbiter PUF consists of a top and bottom signal that are fed through k delay stages. Each individual delay stage consists of two 2-bit multiplexers (MUXes) that have identical layouts and that both get the bottom and top signals as inputs. Since the layout of the two paths is identical, one would expect the introduced delay to be identical as well. However, in practice each transistor in the multiplexers has slightly different delay characteristics due to process variations. Hence, the delay introduced by the multiplexers is different for the top and bottom signal. Since each chip has different process variations, these delay differences are unique for every chip. If the challenge bit c_i for stage i is '1', the multiplexers switch the top and bottom signals, if it is '0' the two signals are not switched. This way, the race signal can take different paths: a n-stage Arbiter PUF has 2^n different paths the race signals can take. An arbiter at the end of the PUF determines which of the two signals is faster. The arbiter has an output of '1' if the top signal arrives first and '0' if the bottom signal is the first to arrive.

In order to increase the resistance of Arbiter PUFs against machine learning attacks adding a non-linear element to the PUF design was proposed. One of the most common methods to add non-linearity to a PUF design is the XOR PUF. In an n-XOR PUF, n Arbiter PUFs are placed on the chip. Each of the Arbiter

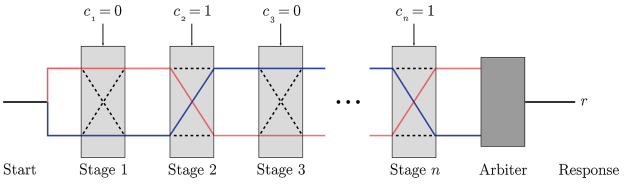

Fig. 1. Schematic of an n-bit Arbiter PUF.

PUFs receives the same challenges and the responses of the n PUFs are XORed to build the final response bits. While the machine learning resistance increases by XORing more PUFs, adding additional PUF instances obviously also increases the area overhead of the design. Furthermore, the XOR PUFs become more unreliable the more PUFs are XORed. This limits the number of XORs that can be used in practice. The response of an n-stage Arbiter PUF is determined by the delay difference between the top and bottom signal. This delay difference is the sum of the delay differences of the individual stages. The delay difference of each stage depends on the corresponding challenge. Hence, there are two delay differences per stage i, the delay differences $\delta_{1,i}$ corresponding to a challenge of '1' and $\delta_{0,i}$ corresponding to a challenge of '0'. The most efficient way to model a k-stage Arbiter PUF is by computing a delay vector $\vec{w} = (w_1, ..., w_{k+1})$ from these stage delay differences as follows:

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

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

$$w_{k+1} = \delta_{0,k} + \delta_{1,k} \tag{1c}$$

The delay difference ΔD 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 challenge c:

$$\Delta D = \vec{w}^T \vec{\Phi} \tag{2a}$$

$$r = \begin{cases} 1, & \text{if } \Delta D < 0. \\ 0, & \text{if } \Delta D > 0. \end{cases} \tag{2b}$$

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

$$\Phi_i = \prod_{l=i}^{k} (-1)^{c_l} \text{ for } 1 \leq i \leq k \tag{3a}$$

$$\Phi_{k+1} = 1 \tag{3b}$$

It was shown in the past how the delay vector $\vec{w}$ can be approximated efficiently using different machine learning techniques.

2.2 Evolution Strategies

Evolution Strategies (ES) are widely used machine learning techniques that are inspired by evolution theory. In evolution, a species can adapt itself to environmental changes by means of natural selection, also called *survival of the fittest*. In every generation, only the fittest specimen survive and reproduce, while the weak specimen die and hence do not reproduce. Since the specimen of the next generation inherit the genes of the fittest specimen of the previous generation, the species continuously improves. In ES-based machine learning attacks on PUFs, the same principle of survival of the fittest is used. As discussed, a PUF instance

can be described by its delay vector $\vec{w}$. The goal of a machine learning attack on an Arbiter PUF is to find a delay vector $\vec{w}$ that most precisely resembles the real PUF instance. The main idea of an ES machine learning attack is to generate random PUF instances (i.e., random delay vectors $\vec{w}$) and check which instances are the fittest, i.e., which PUF instances resemble the real PUF model the most. The fittest PUF instances are kept as *parents* for the next *generation* while the other PUF instances are discarded. In the next generation, *children* are generated using the parent's delay vector together with some random *mutations*, i.e., some random modifications of the delay vector. From these child instances the fittest instances are determined again and kept for the next generation as parents. This process is repeated for many generations in which the PUF instances gradually improve and resemble the real PUF behavior more and more.

In order to perform an ES machine learning attack it needs to be possible to describe a PUF instance by a vector $\vec{w}$. Furthermore, a fitness test is needed that, given delay vectors $\vec{w}$, can determine which instances, i.e., which delay vectors, are the fittest. Since Arbiter PUFs can be modeled using the delay vector w, if an ES machine learning attack is feasible depends on whether or not a good fitness test for these PUF models exist. Typically, the used fitness test for an Arbiter PUF is the model accuracy between the measured responses $\vec{r}$ of the physical PUF and the computed responses $\vec{r'}$ of the PUF instance under test. The PUF instances with the highest model accuracies are considered the *fittest*.

There exist many variants of ES machine learning algorithms which mainly differ in how many parents are kept in each generation, how the children are derived from the parents and how the random mutation is controlled. Typically, the mutation is done by adding a random Gaussian variable $N(0, \sigma)$ to each parameter. Different methods have been proposed for controlling the mutation parameter σ. The closer the PUF instances are to the optimal solution, the smaller σ should be. One approach to control σ is to deterministically decrease σ in every generation. In contrast, in self-adaption the mutation parameter adapts itself depending on how the machine learning algorithm is currently performing. For the reliability-based machine learning attack the Covariance Matrix Adaptation (CMA) ES machine learning algorithm with the default parameters suggested in [11] is used. CMA-ES uses recombination, i.e., one child instance depends on several parent instances. It also uses self-adaption, i.e., the mutation strength is not controlled deterministically but adapts itself depending on how the ES algorithm is performing.

3 The PUF-based RFID Tags

We ordered PUF-based RFID tags from an RFID company that — in addition to our main order of plain sticker type tags — provided us with samples in four different formats: smartcard format, sticker format, wristband and anti-metal label. Please note that the RFID company that manufactures the RFID tags is different from the manufacturer of the PUF ICs. A picture of the tags as well as the used RFID reader can be seen in Fig. 2. The sticker-type tags can be used

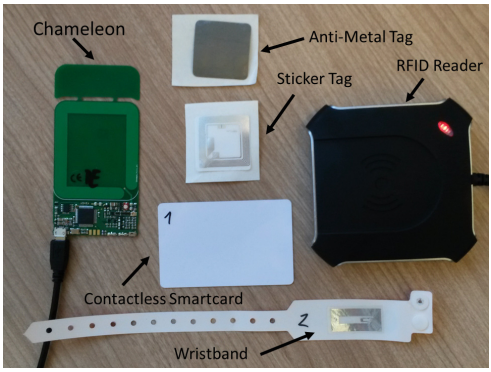

Fig. 2. The RFID smartcard emulator Chameleon and the target PUF-based RFID tags in different formats.

as an anti-counterfeit solution for medical drugs and luxury goods and cost only a few cents when ordered in large quantities. The RFID smartcard format as well as the wristband format can be used for access control, authentication and payment applications. The anti-metal label tags are designed in such a way that they also work when attached to metal and can for example be placed upon a smartphone. The brand names on the sample tags suggest that the smartcard and wristband tags were intended for payment and access control applications while the anti-metal tags featured the brand name of a loyalty program.

In the following the architecture of the tags is described in more detailed. How this architecture has been reverse-engineered is explained in detail in Sect. 4. The main feature of the PUF-based RFID tags is that each tag can be authenticated based on the built-in 4-way Arbiter PUF. The structure of the used 4-way PUF is depicted in Fig. 3. A 64-bit master challenge is sent from the reader to the tag. This master challenge is fed into a 64-bit LFSR that generates subsequent challenges. Each 64-bit challenge is then sent to what we call the *mixer function*. The mixer function generates four subchallenges by shuffling the provided 64-bit challenge similar to a Lightweight PUF. Each of these four subchallenges are fed to the same 64-bit Arbiter PUF. The resulting four response bits are XORed with each other to form a single response bit. Such a PUF, in which a single Arbiter PUF is used and n response bits are XORed, is called an n-way PUF. In the PUF protocol, 256 response bits are generated for each 64-bit master challenge.

Traditionally, the authentication process of an Arbiter PUF is based on a setup stage in which responses for randomly generated challenges are collected and stored in a database. During the authentication step, challenges from this database are selected and sent to the tag. The responses of the tag under test are then compared to the responses stored in the database. If the response matches, the tag is authenticated. However, such a system has some drawbacks. A large database of CRPs needs to be kept for every tag. If the tags are used as an

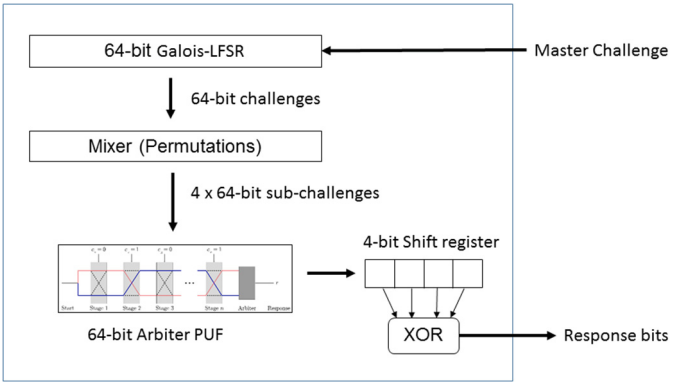

Fig. 3. Internal structure of the PUF tag. A 64-bit Galois LFSR is used to generate challenges from a master challenge. Each challenge is fed into the mixer function which generates four subchallenges by permuting the input challenge. These four subchallenges are successively fed into the 64-stage Arbiter PUF. The four responses of the Arbiter PUF are then XORed to provide a response bit that is sent to the reader.

anti-counterfeit solution, every verifier needs an up-to-date database with the CRPs from all tags. In practice, this requires an internet connection in most scenarios.

For the PUF-based RFID tags a different system that allows offline authentication was chosen. It is based on the idea that Arbiter PUFs can be modeled in software if the internal delay parameters are known. While this usually is an undesired property, it can be helpful to construct PUF protocols that do not need pre-collected CRPs. During a set-up phase, the internal parameters of the 4-way PUF are computed. We are not sure how this is done exactly for the tags under test since this step was already conducted by the manufacturer. One possible solution would be that during the set-up phase the individual PUF responses are directly transmitted without XORing them. With only a few hundred of these response bits it is possible to build an accurate model of the PUF parameters. After a PUF model is built, this phase should be disabled so that only the obfuscated outputs, i.e., the outputs of the 4-way PUF are transmitted.

A verifier that is in possession of these delay parameters does not need a database with stored CRPs to authenticate the tag. Instead, the responses for arbitrary challenges can be computed on-the-fly in software. The distribution of the PUF parameters is realized by storing the parameters on the tag in an encrypted form. A legitimate verifier who has the corresponding key can decrypt the encrypted delay parameters and use these parameters to verify the tag. Note that the encrypted PUF parameters are only stored and not encrypted on the tag. Hence, no encryption function and no secret key is needed on the tag itself. However, this method also has major drawbacks. The verifier needs to be a trusted entity since the security of the system depends on keeping the key secret. This can be problematic and trusted readers need to have the highest standards

Table 1. PUF-based RFID tag protocol

tags		**reader/tool**
ID, physical PUF, $enc_k(\vec{w})$		ID, k
	$\xleftarrow{auth}$	
	$\xrightarrow{UID, enc_k(\vec{w})}$	choose challenge C
$\vec{c} = LFSR(C)$	$\xleftarrow{C}$	$\vec{c} = LFSR(C)$
$\vec{r} \leftarrow PUF(\vec{c})$		$\vec{w} = dec_k(\vec{w})$
	$\xrightarrow{\vec{r}}$	$\vec{r'} \leftarrow PUF\_Model(\vec{c}, \vec{w})$
		if $HD(\vec{r}, \vec{r'}) \leq \tau$ accept

of physical attack security. Furthermore, a verifier can make a software clone of a PUF and hence the unclonability is not given any more. Therefore, such a system does not provided one of the key features, that the PUFs are unclonable, anymore.

4 Attacking the PUF Tags

The first step of attacking the PUF tags was to reverse-engineer the used PUF design, which was not known to us in detail. The RFID company provided us with a software tool to test the tags. The authentication protocol used in the software tool is depicted in Table 1. The software tool starts the authentication process by sending a query *auth* to the PUF tag. The PUF tag answers with its UID and the encrypted PUF parameters. Then the tool generates a random master challenge C and sends it to the tag. The tag answers with the corresponding response string $\vec{r}$. The tool decrypts the encrypted PUF delay vector $\vec{w}$ using the secret key k and uses this delay vector to compute the expected response string $\vec{r'}$. If the mismatch between the received response string r and the computed response string $\vec{r'}$ is below a threshold τ, the PUF tag is authenticated. The tool also supports an "online" verification that works in a similar way. The difference is that the decryption of the PUF parameters and the response computation is performed by a trusted server and not by the software tool. The tags we received were already programmed and contained encrypted PUF parameters with the key from the test tool. The fact that the software tool also computes the PUF responses based on the encrypted PUF parameters made reverse-engineering of the PUF much easier. No hardware reverse-engineering was needed. Instead, the reverse-engineering was performed on the software tool using IDA Pro. Via this reverse-engineering, we were able to derive the structure of the 4-way PUF as already introduced in Fig. 3, including all parameters such as the exact LFSR and the details of the mixer function.

4.1 Machine Learning Attack

To collect the CRPs necessary to attack the 4-way Arbiter PUF, we used a Matlab script to send random challenges to the reader. A single protocol execution, which consisted of sending a 64-bit challenge and receiving a 256-bit response, took roughly 53 ms. However, the response collection can be sped up by not sending a new master challenge between each measurement. Without a new master challenge, the last LFSR state of the previous challenge is used as a new master challenge. Using this trick, the measurement time can be reduced to 43 ms per 256-bit response. Hence, with this setup more than 5,000 response bits can be collect in one second and more than 350,000 in one minute. We would like to stress that we see these numbers are of general interest as a reference when accessing the security of PUFs against machine learning. Given a setup like ours, it is quite reasonable that an attacker can collect billions of CRPs.

To attack the PUF tags we tested Logistic Regression (LR) in conjunction with RProp as proposed in [29] as well as a CMA-ES machine learning algorithm. Both algorithms can be used to attack the PUF but the LR algorithm greatly outperforms the CMA-ES algorithm. To evaluate the machine learning attack, a training set was used by the machine learning algorithm to train the PUF parameters and a separate set, the reference set, was used to evaluate the resulting model accuracy. We were able to reliably perform a LR machine learning attack using a training set consisting of data from only 4 protocol executions, i.e., $4 \cdot 256 = 1024$ CRPs with a resulting average model accuracy of 85.8 %. Please note that the achieved model accuracy is very close to the observed average reliability of the PUF tags of 87.5 % and therefore sufficient to impersonate the PUF. All of the 10 different tags we tested could be successfully attacked. The measurement time of 1024 CRPs is only 172 ms and the computation time was in average around 40 seconds on a laptop. When more than 1024 CRPs are used, the computation time actually decreases significantly to a few seconds.

Hence, the 4-way PUF employed on these PUF-based RFID tags does not provide any protection against machine learning attacks and it is trivial for an attacker to recover the PUF parameters needed to model the PUF.

4.2 Cloning a PUF Tag

The machine learning attack from the previous Section provided us with the PUF parameters that can be used to build a software model of the PUF. The remaining question is how we can actually use these PUF parameters to build a clone of the PUF tags. For this the freely programmable RFID smartcard emulator Chameleon was used. The Chameleon is based on an 8-bit ATXmega32A4U microcontroller clocked at 8 MHz and can be seen on the left side of Fig. 2. It has the size of a standard smartcard, costs less than $25 and is also available in a version powered by a small battery. The Chameleon is an open-source project and details of the Chameleon, the PCB layout and the firmware can be found online at GitHub [22]. One key feature of the Chameleon is that it is possible to set the UID, which cannot be done in most commercial RFID cards and tags.

We implemented the used communication protocol of PUF-based RFID tags on the ATXmega as well as a software model of the 4-way PUF. The computation of the PUF responses is time-critical, as there is a timeout if the computation of the PUF responses takes too long. To speed up the PUF computation we only used 5-bit precision integers for the PUF delay vector $\vec{w}$ since in this case no overflows occur on an 8-bit system. The delay vector has been computed using the machine learning attack described in the previous section.

The implementation was done using C and the computation of a 128-bit response block on the Chameleon took roughly 400,000 clock cycles which results in a computation time of 12.8 ms at a clock frequency of 32 MHz. In our experiments, this was fast enough not to trigger any timeouts. As mentioned, we only used 5-bit delay parameters for each PUF which reduced the model accuracy and resulted in a mismatch of around 20 % between the responses from the Chameleon and the responses computed by the demo tool. The mismatch of around 20 % in average is similar to the mismatch observed between the computed responses of the software tool and legitimate tags. This mismatch is well within the acceptable error rate of the software tool. If 16-bit precision numbers are used, the parameters derived by the machine learning attack actually have a higher model accuracy than the encrypted PUF parameters. We verified this attack using the test tool and the software declared the Chameleon an authentic PUF tag. This proves that cloning attacks on such PUF-based RFID tags can be performed in practice. Interestingly, the biggest challenge for an attacker in practice is actually to find a suitable programmable RFID chip in which the attacker can freely set the UID. However, in absent of freely programmable RFID smartcards, this can be simply solved by building your own RFID smartcard emulator.

5 Reliability-Based Machine Learning Attacks

In the previous section we have seen how to attack commercial PUF tags that are based on a 64-stage 4-way PUF. It is well known that small XOR PUFs can be attacked using machine learning and hence it is not a big surprise that the employed 4-way PUF can be attacked. In this section we therefore take a closer look at more secure architectures. In particular, we introduce a new reliability-based machine learning attack that scales much better with increasing XORs than the machine learning attacks from [29]. In traditional machine learning attacks, the attacker tries to model the PUF-based on the values of the response bits. However, not only the value of the responses contains useful information. The reliability of a response, i.e., how often the PUF evaluates to the same response bit for a given challenge, also holds valuable information. Delvaux *et al.* were the first to point this out [5]. They observed that the delay difference for a specific challenge of an Arbiter PUF is directly proportional to the unreliability of the corresponding response bit if the environmental conditions are kept stable. This is due to the fact that the various sources of noise add an approximately Gaussian delay $D_{noise} = norm(\mu, \sigma)$ to the delay difference ΔD. Hence, when also considering noise, Eq. 2 of the PUF model changes to:

$$\Delta D = \Delta D_{PUF} + D_{noise} = \vec{w}^T \vec{\Phi} + D_{noise} \tag{4a}$$

$$r = \begin{cases} 1, & \text{if } \Delta D_{PUF} + D_{noise} < 0. \\ 0, & \text{if } \Delta D_{PUF} + D_{noise} > 0. \end{cases} \tag{4b}$$

The key observation is that if the delay difference ΔD_{PUF} for a given challenge $\vec{\Phi}$ is very large, it is unlikely that the noise term D_{noise} changes the sign of ΔD. However, if the delay difference ΔD_{PUF} is close to zero, the chance that the response bit changes due to D_{noise} is much higher. Delvaux *et al.* used the exact reliability value of individual response bits to approximate delay differences for this response [5]. With the exact delay difference of individual responses the individual stage delays can be computed by solving a set of linear equations. However, Delvaux *et al.* pointed out that this approach is not as efficient as machine learning algorithms. Furthermore, their method cannot be applied to an XOR PUF, since you need to know the individual response bit for each Arbiter PUF as well as their exact reliability.

Becker *et al.* extended the idea of using unreliability and proposed a fault attack that is based on machine learning [3]. Changing the environmental conditions such as temperature or supply voltage has a similar effect as thermal noise, but is usually larger. The idea of the fault attack on controlled PUFs is to change the supply voltage for specific response bits and observe if the response of the controlled PUF changes. This information is then used in a CMA-ES based machine learning attack. This paper extends the basic idea behind this attack by adding a divide-and-conquer strategy in order to gain the ability to attack XOR PUFs. Furthermore, in this paper no active fault attack by changing the supply voltage is performed. Instead, only the inherent unreliability of the PUF is used. The measurements therefore can be conducted under the same environmental conditions. With this new strategy, we are able to break XOR PUFs that would be computationally infeasible to attack using traditional machine learning algorithms focusing on the output bits.

5.1 CMA-ES Attack Based on Reliability

The main idea of the reliability-based CMA-ES attack is to make repeated measurements for the same challenge so observe which response bits are stable and which response bits sometimes flip. If a response for a given challenge is unstable, it is likely that the corresponding delay difference ΔD_{PUF} is close to zero, i.e., $|\Delta D_{PUF}| < \epsilon$. But if a response bit has a high reliability for a given challenge, it is likely that the delay difference is large, i.e., $|\Delta D_{PUF}| > \epsilon$. We slightly modify the CMA-ES machine learning algorithm by using a fitness function that is based on this observation. The goal of the fitness function is to test which of a given set of PUF models $\vec{w}$ performs best, i.e., which is the fittest. In the first step the same challenge $\vec{\Phi}_i$ is sent to the PUF l times to collect l response bits $r_{i,1}, r_{i,2}, .., r_{i,l}$. Then the reliability h_i is computed for challenge $\vec{\Phi}_i$ using the following formula:

$$h_i = |\frac{l}{2} - \sum_{j=1}^{l} r_{i,j}| \tag{5}$$

To test the fitness of a given PUF model $\vec{w}$ the attacker first computes a hypothetical reliability $\tilde{h_i}$ for all challenges $\vec{\Phi_i}$ by testing if the corresponding absolute delay difference $|\Delta D_i|$ is larger than an error boundary ϵ:

$$\tilde{h_i} = \begin{cases} 1, & \text{if } |\vec{w}^T \vec{\Phi_i}| > \epsilon \\ 0, & \text{if } |\vec{w}^T \vec{\Phi_i}| < \epsilon \end{cases} \tag{6}$$

In the next step the attacker checks how well the hypothetical reliability vector $\tilde{h} = \tilde{h_1}, .., \tilde{h_n}$ matches the measured reliability vector $h = h_1, .., h_n$. This is done by computing the Pearson correlation coefficient between h and $\tilde{h}$. The correlation coefficient shows the linear relationship between the two vectors. It can therefore be used to test how well the hypothetical reliability vector $\tilde{h}$ of a PUF model $\vec{w}$ matches the observed reliabilities h. The higher the correlation coefficient, the *fitter* the PUF model is. The only other modification to the CMA-ES algorithm is that the parameter ϵ is an additional parameter that needs to be approximated by the machine learning algorithm. Hence, $k + 2$ instead of $k + 1$ parameters need to be approximated by the machine learning algorithm for a k-stage Arbiter PUF. Otherwise, the reliability CMA-ES works like a traditional CMA-ES attack: In each generation several PUF models are generated and then evaluated using the fitness test. The best models are kept and are used to generate the PUF models for the next generation. This way the PUF model gradually improves with each generation.

The attack was first tested using CRPs taken from the 4-way PUF of the commercial RFID tags. In a 4-way PUF, four individual response bits are XORed to form the final response bit. Hence, an attacker does not know the reliability for a given PUF response since it is only possible to measure the reliability of the final response bit. Therefore the fitness function needs to be adjusted. In a first step, the reliability values for the four subchallenges of the 4-way PUF are computed and then the reliability values of the four subchallenges are simply added up. Note that the more subchallenges are expected to be unreliable, the more likely it is that the measured response bit is unreliable as well. We performed the reliability-based CMA-ES attack on the 4-way PUF using the reliability information of $l = 5$ measurements. The attack was less efficient than a traditional CMA-ES attack and required around 4,000 CRPs. Nevertheless, the result shows that the reliability CMA-ES attack works in practice. For a larger number of XORs the reliability-based attack outperforms traditional machine learning algorithms significantly. To test this, two consecutive response bits of the 4-way PUF were XORed, effectively turning the 4-way PUF into an 8-way PUF. The number of needed CRPs for a traditional machine learning attack greatly increases when the number of XORs is increased, due to the increased non-linearity of the XORs as well as due to the fact that unreliability also increases. The 8-way PUF could be attacked with 300,000 CRPs using a LR machine learning algorithm. In comparison, the reliability-based CMA-ES attack still only needed 4,000 CRPs.

This result might be surprising at first, since the number of needed CRPs did not increase at all compared to the 4-way PUF. But in the fitness function of a reliability-based machine learning attack on an n-way PUFs, the reliability of each individual response is added and not XORed. Therefore, the number of XORs has only a small impact on the machine learning algorithm. As a matter of fact, we can still attack the PUF with 4,000 CRPs if a 20-way PUF is built by XORing 5 consecutive bits of the 4-way PUF. The reliability of a 20-way PUF is only around 63.1 % and hence not really useable in practice. Still, the reliability-based machine learning attack finds a PUF model with a model accuracy of 61.1 % for the 20-way PUF. The model accuracy of the underlying 4-way PUF is actually still around 87 % for this attack. Hence, unlike for traditional machine learning attacks, increasing the XORs in an n-way PUF does not increase the machine learning resistance.

5.2 Attacking XOR PUFs

The most popular strong PUF design is not the n-way PUF, but the XOR PUF which uses a different Arbiter PUF for each XOR. Every additional XOR also adds additional parameters that need to modeled in addition to the increased non-linearity of the XOR. This leads to an exponential increase in needed CRPs with each additional XOR. It is therefore assumed that if enough stages and XORs are used, XOR PUFs can withstand machine learning attacks [12,28]. However, in the following we show how even XOR PUF instances that are presumably secure can be attacked using the reliability-based machine learning attack.

For an n-XOR, k-stages PUF, $(k+1) \cdot n$ parameters need to be determined in a machine learning attack. However, in the reliability-based machine learning attack a divide-and-conquer strategy can be used to attack each Arbiter PUF individually. This reduces the parameters that need to be approximated in one machine learning run to only $k+1$. This is the main reason why the reliability-based machine learning attack significantly outperforms other machine learning attacks on XOR PUFs. The idea behind the divide-and-conquer approach is that the reliability of a response bit depends equally on each of the n employed Arbiter PUFs. Let us assume that for a PUF model $\vec{w}$ and challenge $\vec{\Phi}_i$ of one of the n Arbiter PUFs, the expected reliability is low, i.e., $\tilde{h}_i = 0$. Then the measured reliability h_i should also be low, since a bit flip of one of the response bits that are XORed directly results in a bit flip of the output of an XOR PUF. Hence, in this case the measured reliability h_i matches with the computed reliability $\tilde{h}_i$. If the computed reliability for the challenge $\vec{\Phi}_i$ is high, i.e., $\tilde{h}_i = 1$, and the observed reliability is also high, the computed and observed reliability vectors match each other. However, one of the other $n-1$ remaining Arbiter PUFs might be unreliable for this specific challenge. Hence, we might sometimes observe an unreliable response despite our hypothesis assuming a reliable response. But since we always guess the unreliable case correct, our hypothesis vector $\tilde{h}$ is still correlated with the observed reliability vector h, even if it is not a complete match. The unreliability introduced by the other $n-1$ Arbiter PUFs is therefore

nothing else but noise from a machine learning perspective. Since the CMA-ES machine learning algorithm in conjunction with the correlation coefficient as a fitness function is very robust to noise, the CMA-ES algorithm still finds an accurate PUF model for the Arbiter PUF under test. Hence, in this attack we do not target all Arbiter PUFs at once. Instead we model one Arbiter PUF at a time. This is the main reason why our attack scales so well with the number of XORs. Each additional Arbiter PUF only adds additional noise to the computation. Furthermore, the relative increase in noise by adding a single XOR decreases with the number of XORs. Hence, the machine learning attack complexity only increases linear with the number of added XORs.

The only remaining question is how we can target a specif Arbiter PUF from the set of Arbiter PUFs used in the XOR PUF. There are basically two ways to build an XOR PUF. Either the same challenge is applied to all Arbiter PUFs, or each Arbiter PUF gets a different challenge. The classic XOR PUF uses the same challenges for all Arbiter PUFs. But results from [29] suggest that using different challenges for each PUF makes machine learning attacks harder. Let us first consider the case that each of the individual Arbiter PUFs gets their own set of unique challenges. In this case an attacker can target a specific Arbiter PUF based on which challenges the attacker uses in the machine learning attack. Since each Arbiter PUF has a different set of challenges, one Arbiter PUF is attacked after the other. Hence, to attack an n-XOR PUF, n individual reliability-based machine learning attacks are performed. After PUF models for all n PUFs are found, the entire XOR PUF can be modeled by simply XORing the individual responses.

The CMA-ES machine learning algorithm is a non-deterministic algorithm and sometimes does not converge to a near-optimal solution. In this case the algorithm needs to be restarted. Machine learning runs that did not find an accurate PUF model, i.e., that did not converge, have a much smaller fitness value than successful runs. Therefore the correlation coefficient can be used to test if a machine learning run was successful. If it was successful, the next PUF can be targeted, otherwise the machine learning algorithm should be restarted again with the same challenges. To test this we simulated different XOR PUFs by assuming a random Gaussian distribution of the individual stage delay parameters δ of the individual Arbiter PUFs. Assuming a Gaussian distribution is the common approach (e.g. used in [29]) and resembles a best-case scenario from a security perspective. To model the impact of noise a random variable is added to each computed delay difference ΔD_{PUF} with a Gaussian distribution of $norm(0, \sigma_{noise})$. The challenges were generated randomly and all simulations were carried out using matlab. To speed up the computation, a Mex function was written in C for the computationally expensive part of the PUF computations and the attacks were run on a AMD Opteron cluster with 4 nodes and 64 cores each. Each attack only used 16 cores so that in total 16 attacks run simultaneously on the cluster. Now let us consider the case that the same challenge is applied to all Arbiter PUFs, as is done in the classic XOR-Arbiter PUF. In this case an attacker cannot target a specific Arbiter PUF using different challenges since all PUFs get the same challenges. However, the probabilistic nature of

Table 2. Results of a reliability-based CMA-ES on different simulated n-XOR, 128-stage PUFs with a noise level of $\sigma_{noise} = 1$. In the top rows different challenges were used for each Arbiter PUF and in the bottom rows the same challenges were used as done in the classic XOR PUF . The results are the average of 10 independent attacks. "Accuracy single Arbiter" is the maximum and minimum achieved model accuracy of a single Arbiter PUF.

# XORs	Reliability	# CRPs	Accuracy reference set	Accuracy training set	#runs per XOR	Accuracy single Arbiter	Time
1	98.0 %	$20 \cdot 10^3$	99.0	98.3	8.7	98.3 %–99.3 %	0.9 h
4	92.5 %	$150 \cdot 10^3$	97.6 %	94.6 %	4.0	99.0 %–99.6 %	1.8 h
8	86.2 %	$300 \cdot 10^3$	95.3 %	89.0 %	3.4	98.6 %–99.7 %	3.3 h
16	76.0 %	$500 \cdot 10^3$	90.8 %	80.2 %	19.4	98.7 %–99.6 %	30.5 h
32[a]	63.7 %	$2000 \cdot 10^3$	83.6 %	68.4 %	9.5	99.1 %–99.6 %	60 h
4	92.5 %	$150 \cdot 10^3$	97.7 %	94.2 %	4.2	99.1 %–99.7 %	1.1 h
8	86.2 %	$300 \cdot 10^3$	95.7 %	89.1 %	7.2	99.1 %–99.7 %	3.4 h
16[b]	76.1 %	$500 \cdot 10^3$	90.0 %	80.1 %	30.6	98.7 %–99.6 %	34 h

[a] This row only uses the average from 3 independent attacks and not 10

[b] For the classic 16-XOR PUF a 2-step approach was used in which the first 13 PUFs were attacked using a reliability-based attack and the remaining 3 PUFs were attacked using an traditional CMA-ES. "#runs per XOR" is the average of the 13 PUFs determined by the reliability-based attack.

Table 3. Results of reliability-based CMA-ES attacks on a 8 XOR, 128 stages PUF with different noise values σ_{noise}. Again 10 attacks were performed per entry.

σ_{noise}	Reliability single Arbiter	Reliability XOR PUF	# CRPs	Accuracy reference set	#runs per XOR	Accuracy single Arbiter	Time
0.1	99.8 %	98.4 %	$2500 \cdot 10^3$	96.0 %	6.8	98.9 %–99.7 %	16.7 h
0.25	99.5 %	96.2 %	$1000 \cdot 10^3$	95.5 %	8.1	97.2 %–99.8 %	17.0 h
0.5	99.0 %	92.6 %	$500 \cdot 10^3$	94.7 %	7.5	98.9 %–99.8 %	6.6 h
1	98.0 %	86.2 %	$300 \cdot 10^3$	95.3 %	3.4	98.6 %–99.7 %	3.3 h
2	96.0 %	75.8 %	$200 \cdot 10^3$	94.5 %	1.6	98.8 %–99.6 %	1.2 h
4	92.1 %	62.7 %	$100 \cdot 10^3$	84.6 %	8.5	96.2 %–98.2 %	4.6 h

CMA-ES helps us in this case. The reliability-based machine learning algorithm will converge to one of the Arbiter PUFs of the XOR PUF since the correlation coefficient for a correct PUF model is higher than that for an inaccurate PUF model. If the machine learning algorithm would always converge to the same PUF, the attack would not be very helpful. However, due to the probabilistic nature of CMA-ES, the algorithm converges to different PUFs in different runs, even when called with the same inputs. The idea of the attack is to performed as many independent machine learning runs until all n distinct PUF models are found. Ideally, each of the n Arbiter PUFs should be equally likely to be found by a single run. In practice, some Arbiter PUFs are "easier" and some are "harder" to attack for given challenges and reliability vectors. Hence, the machine learning algorithm converges more often to some PUF instances than others.

In practice, the attacker does not necessarily need to find all of the n PUFs using the reliability-based machine learning attack. If only a few Arbiter PUFs remain, the attacker can find the remaining PUF models using a traditional LR or CMA-ES machine learning attack. When only a few PUFs remain, the chances are high that the machine learning attack converges to a PUF that has already been modeled. Therefore, this two step approach can considerably decreases the attack time for PUFs with many XORs such as a 16-XOR PUF. In general, unsuccessful runs can be aborted early to greatly decrease the computation time of the attack. To determine which runs are likely to be unsuccessful, the global mutation parameter σ in conjunction with the fitness value can be used. Furthermore, the hamming distance between responses from the model under test and the already computed PUF models can be used to detect runs that are converging to a PUF model that has already been found. These runs can also be aborted early to considerably speed up the computation time. The results of the reliability-based machine learning attack are summarized in Table 2. A noise level of $\sigma_{noise} = 1$ was used which resulted in an reliability of 98 % for a 128-stage Arbiter PUF. This is a conservative estimation of the reliability of Arbiter PUFs. For comparison, the observed unreliability of 64-stage Arbiter PUFs in [14] was around 97 % for nominal operation conditions. With this new attack even a 32-XOR PUF could be attacked with only 2 million CRPs, which would be impossible using LR. We also tested the attack for more reliable or less reliable PUFs. The results of this experiment are summarized in Table 3. For very reliable PUFs, more CRPs are needed, but the attack still works. To verify these results using real measurements, we emulated an XOR PUF structure by taking measurements from up to 8 different PUF RFID tags and XORed their corresponding output bits. This effectively turned the 4-way PUF tags into a mixture of an n-way and XOR PUF. In an 8-XOR-4-way PUF, 8 different 4-way PUFs are XORed, which results in a total of 32 XORs for a single response bit. The results of this experiment can be found in Table 4. The 8-XOR-4-way PUF can be successfully attacked using 400,000 CRPs. Hence, our reliability-based machine learning on XOR PUFs also works with real silicon data.

Table 4. Results of a reliability-based CMA-ES on an n-XOR 4-way PUF construction using $l = 5$ repeated measurements from the PUF-based RFID tags in which different challenges were used for each 4-way PUF.

# PUFs	# XORs	# CRPs	Reliability	Accuracy reference set	Accuracy training set	Accuracy 4-way	Time
1	4	$4 \cdot 10^3$	87.5 %	87.1 %	88.6 %	87.1	0.7 m
2	8	$10 \cdot 10^3$	80.0 %	78.5 %	80.3 %	88.0 %	1.6 m
4	16	$40 \cdot 10^3$	69.2 %	67.2 %	69.4 %	87.9 %	1.7 m
8	32	$400 \cdot 10^3$	56.3 %	55.6 %	56.4 %	87.5 %	13.1 m

This shows that the exponential increase in number of required responses for increasing XORs does not hold for the reliability-based machine learning attack. Unlike previously stated, plain XOR PUFs are insecure, regardless of the used parameters.

6 Discussion

In this paper we showed that the security of strong PUFs is still greatly lacking. This is true for both commercially available PUFs as well as strong PUF proposals by the scientific community. A lot of research effort has been focused on finding different PUF architectures. However, we are still far behind to understand the full power of machine learning attacks, in particular if more information than just the plain response bits is used. The newly proposed reliability-based CMA-ES attack is a prime example for this.

Basically, to prevent a reliability-based CMA-ES attack, an attacker should not be able to send the same challenge twice and observe the reliability of the responses. This could be achieved if the challenges are generated by both the tag as well as the verifier. However, this typically means that the verifier needs a software model of the PUF. Proposal for such protocols have already been made, see for example the Slender PUF protocol [25]. However, several key features of PUFs are lost if a software model is needed. For example, the "unclonability" feature is lost since everyone in possession of this software model can create a software clone of the PUF. Furthermore, every entity that authenticates a PUF instance needs to be a trusted entity, since such a software model can be seen as the equivalent of a symmetric key. Such a PUF also violates the "unprotected challenge-response interface" requirement as defined in [28] and hence is not a strong PUF according to formal definitions. Another approach to prevent this attack might be the idea of a controlled PUF [7]. However, recently it was shown that it is possible to perform a reliability-based machine learning attack based on the helper data of error correction codes [2]. Since controlled PUFs rely on error correction code, this attack is directly applicable to controlled PUFs as well. Hence, simply using a controlled PUF does not solve this problem. How to build a strong PUF that resists the reliability-based machine learning attack is therefore an interesting open research problem.

Our attack on the commercial PUF-based RFID tags shows the real-world implications of this research. An attacker only needs to hold an RFID reader or a NFC enabled smartphone within ca. 5 cm of the tags for 200 ms to collect enough CRPs to build an accurate software model. We verified that it is possible to read out the smartcard format PUF tags when they are carried within a wallet in the back-pocket of a jeans. We showed that it is possible to clone the tags using a self-made RFID smartcard emulator from off-the-shelf components for less than $25. This allows an electrical "pickpocketing" attack that can be carried our in real-time. Hence, while a lot of hope was put into strong PUFs as a secure and lightweight authentication solution, it seems that both academia as well as industry are still far away from achieving these goals.

References

1. Armknecht, F., Maes, R., Sadeghi, A., Standaert, F.X., Wachsmann, C.: A formalization of the security features of physical functions. In: IEEE Symposium on Security and Privacy 2011 (SP), pp. 397–412. IEEE (2011)
2. Becker, G.T.: On the pitfalls of using arbiter pufs as building blocks. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **PP**(99), 1 (2015)
3. Becker, G.T., Kumar, R.: Active and passive side-channel attacks on delay based puf designs. IACR Cryptology ePrint Archive **2014**, 287 (2014)
4. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 51–70. Springer, Heidelberg (2011)
5. Delvaux, J., Verbauwhede, I.: 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
6. Delvaux, J., Gu, D., Schellekens, D., Verbauwhede, I.: Secure lightweight entity authentication with strong PUFs: mission impossible? In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 451–475. Springer, Heidelberg (2014)
7. Gassend, B., Clarke, D., Van Dijk, M., Devadas, S.: Controlled physical random functions. In: Proceedings of 18th Annual Computer Security Applications Conference 2002, pp. 149–160 (2002)
8. Gassend, B., Clarke, D., Van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM conference on Computer and communications security, pp. 148–160. ACM (2002)
9. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
10. Güneysu, T.: Using data contention in dual-ported memories for security applications. Sign. Proces. Syst. **67**(1), 15–29 (2012)
11. Hansen, N.: The CMA evolution strategy: a comparing review. In: Towards a New Evolutionary Computation, Studies in Fuzziness and Soft Computing, vol. 192, pp. 75–102. Springer, Heidelberg (2006)
12. Herder, C., Yu, M.D., Koushanfar, F., Devadas, S.: Physical unclonable functions and applications: a tutorial. Proc. IEEE **102**(8), 1126–1141 (2014)
13. Van Herrewege, A., Katzenbeisser, S., Maes, R., Peeters, R., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C.: Reverse fuzzy extractors: enabling lightweight mutual authentication for PUF-enabled RFIDs. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 374–389. Springer, Heidelberg (2012)
14. Katzenbeisser, S., Kocabaş, Ü., Rožić, V., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C.: PUFs: myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 283–301. Springer, Heidelberg (2012)
15. Lee, J.W., Lim, D., Gassend, B., Suh, G.E., Van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symposium on VLSI Circuits, 2004. Digest of Technical Papers, 2004. pp. 176–179. IEEE (2004)
16. Maiti, A., Casarona, J., McHale, L., Schaumont, P.: A large scale characterization of ro-puf. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2010, pp. 94–99. IEEE (2010)

17. Majzoobi, M., Rostami, M., Koushanfar, F., Wallach, D., Devadas, S.: Slender puf protocol: A lightweight, robust, and secure authentication by substring matching. In: IEEE Symposium on Security and Privacy Workshops (SPW) 2012, pp. 33–44, May 2012
18. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight secure pufs. In: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 670–673. IEEE Press (2008)
19. Merli, D., Heyszl, J., Heinz, B., Schuster, D., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of ro pufs. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2013, pp. 19–24 (2013)
20. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (Malicious) physically uncloneable functions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 702–718. Springer, Heidelberg (2013)
21. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297(5589), 2026–2030 (2002). http://www.sciencemag.org/content/297/5589/2026.abstract
22. chameleon Project: Chameleon mini, January 2015. https://github.com/emsec/ChameleonMini/wiki
23. Maes, P.T.R., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices. In: WISSec 2008 (2008)
24. Rührmair, U., Xu, X., Sölter, J., Mahmoud, A., Majzoobi, M., Koushanfar, F., Burleson, W.: Efficient power and timing side channels for physical unclonable functions. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 476–492. Springer, Heidelberg (2014)
25. Rostami, M., Majzoobi, M., Koushanfar, F., Wallach, D., Devadas, S.: Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching. IEEE Trans. Emerg. Top. Comput. **PP**(99), 1 (2014)
26. Rührmair, U., Solter, 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 Trans. Inf. Forensics Secur. **8**(11), 1876–1891 (2013)
27. Rührmair, U., van Dijk, M.: Pufs in security protocols: attack models and security evaluations. In: IEEE Symposium on Security and Privacy (SP) 2013, pp. 286–300. IEEE (2013)
28. Rührmair, U., Holcomb, D.E.: Pufs at a glance. In: Proceedings of the conference on Design, Automation & Test in Europe, p. 347. European Design and Automation Association (2014)
29. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 237–249. CCS 2010, ACM, New York, NY, USA (2010). http://doi.acm.org/10.1145/1866307.1866335
30. Simons, P., van der Sluis, E., van der Leest, V.: Buskeeper PUFs, a promising alternative to D flip-flop PUFs. In: HOST 2012, pp. 7–12. IEEE (2012)
31. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of the 44th annual Design Automation Conference, pp. 9–14. ACM (2007)

32. Tajik, S., Dietz, E., Frohmann, S., Seifert, J.-P., Nedospasov, D., Helfmeier, C., Boit, C., Dittrich, H.: Physical characterization of arbiter PUFs. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 493–509. Springer, Heidelberg (2014)
33. Yu, M.D., Sowell, R., Singh, A., M'Raihi, D., Devadas, S.: Performance metrics and empirical results of a puf cryptographic key generation asic. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2012, pp. 108–115. IEEE (2012)