Extended Abstract: Analysis of 1000 Arbiter PUF based RFID Tags

Christine Utz, Johannes Tobisch, Georg T. Becker
Horst Görtz Institute for IT Security, Ruhr-Universität Bochum

***Abstract*—In this extended abstract a large-scale analysis of 4-way Arbiter PUFs is performed with measurement results from 1000 RFID tags. Arbiter PUFs are one of the most important building blocks in PUF-based protocols and have been the subject of many papers. However, in the past often only software simulations or a limited number of test chips were available for analysis. Therefore, the goal of this work is to verify earlier findings in regard to the uniqueness and reliability of Arbiter PUFs by using a much larger measurement set. Furthermore, we used machine learning algorithms to approximate and compare the internal delay differences of the employed PUF. One of the main research questions in this paper is to examine if any "outliers" occurred, i.e., if some tags performed considerably different. This might for example happen due to some unusual manufacturing variations or faults. However, our findings are that for all of the analyzed tags the parameters fell within the range of a Gaussian distribution without significant outliers. Hence, our results are indeed in line with the results of prior work.**

I. Introduction

Physical Unclonable Functions (PUFs) are a hardware primitive with a variety of interesting use cases, e.g., secure key storage or authentication without the need for secure non-volatile memory. The first proposed PUF was the optical PUF in 2001 [1] followed by the first electrical PUF, the Arbiter PUF, in 2002 [2]. The principle behind the Arbiter PUF is to let two signals traverse a certain number of delay stages and output which one was faster. Arbiter PUFs have an exponential number of challenge-and-response pairs and can therefore be used in challenge-and-response protocols for authentication purposes. However, research has shown that Arbiter PUFs can easily be cloned in software using machine learning techniques. To prevent such attacks, several more complex schemes based on Arbiter PUFs have been proposed. The most prominent example is the XOR Arbiter PUF, in which the outputs of individual Arbiter PUFs are XORed to yield the final response [3]. While this construction increases the machine learning complexity, it cannot prevent these attacks [4], especially if reliability information is taken into consideration as well [5]. Machine learning attacks have been demonstrated on simulated data, but have also been repeatedly verified using data acquired from ASICs [6], [7], [8], [5].

Currently, new protocols such as the one by Yu *et al.* [9] are being developed to overcome the problems of machine learning attacks and the reliability-based machine learning attack in particular. But this increased machine learning resistance usually comes at the cost of a restricted number of authentications or the need for a secret software model. All in all, despite powerful attacks against Arbiter PUFs, there is still a lot of interest in this type of PUF.

There have been several academic test chips featuring Arbiter PUFs which were used to examine the reliability and uniqueness of the Arbiter PUF [6], [10], [11], [12], [13]. Many of these were only manufactured in relatively low number of chips, with the notable exception of [11] which used 192 test chips. We wanted to complement the previous work on ASIC PUFs by performing a large-scale analysis of 998 RFID tags equipped with a 64-stage 4-way PUF that we were able to buy from a commercial supplier[1]. All previous results suggest that the delay values of the Arbiter PUFs as well as their reliability follow a Gaussian distribution. We wanted to verify this earlier finding by testing a large set of Arbiter PUFs for "outliers", i.e., a potentially rare behavior that falls outside the assumed Gaussian distribution. In order to do so, we used these 1000 RFID tags equipped with a 64-stage 4-way PUF and analyzed their reliability and uniqueness. Furthermore, we used machine learning algorithms to approximate the internal delay distribution of the individual stages to check if these values are also Gaussian distributed. Unfortunately, the nature of the examined 4-way PUF is not ideal for such an analysis and it would be interesting to see such an analysis for other test chips in the future. Nevertheless, the results of our analysis clearly support the assumption that the internal delay differences per stage are Gaussian-distributed without any outliers that fall outside this distribution.

In the next section we will give a brief description of the analyzed 4-way PUF and how an Arbiter PUF can be modeled. After that we present the results of our analysis and end the paper with a brief conclusion.

II. The PUF-based RFID Tags

The RFID tags examined in this paper are the same as the ones used in [5]. The main feature of these PUF-based RFID tags is that each tag can be authenticated based on a built-in 4-way Arbiter PUF. The structure of this 4-way PUF is depicted in Figure 1. A 64-bit master challenge is sent from the reader to the tag. This master challenge is fed into a 64-bit Linear-Feedback Shift Register (LFSR) that generates subsequent challenges. Each 64-bit challenge is then sent to what we call the *mixer function*. The mixer function generates four sub-challenges by shuffling the 64-bit challenge similar to a Lightweight PUF [14]. Each of these four sub-challenges is 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.

[1]The PUF-based RFID tags have been recalled by the manufacturer and are no longer available.

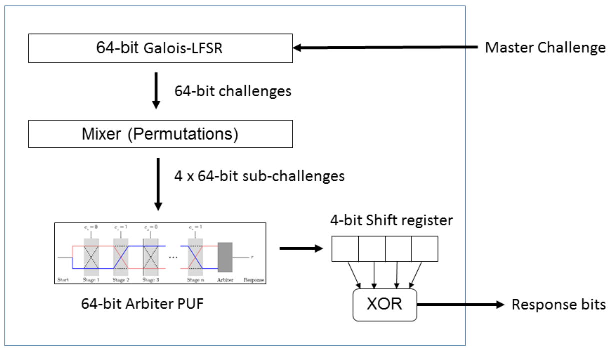

Fig. 1. Internal structure of the PUF tag. A 64-bit Galois LFSR is used to generate challenges from a master challenge. Each challenge is fed into the mixer function which, generates four sub-challenges by permuting the input challenge. These four sub-challenges are successively fed into the 64-stage Arbiter PUF. The four responses of the Arbiter PUF are then XORed to provide a response bit that is sent to the reader.

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 responses match, the tag is authenticated. The RFID tags can be used with such a protocol although they also allow for an offline authentication based on encrypted PUF parameters. For more details about this offline authentication protocol we would like to refer the reader to [5]. For the purpose of this paper, only the architecture of the PUF is relevant and not the authentication protocol.

III. Model of the Arbiter PUF

In an Arbiter PUF, two race signals are fed through a series of multiplexer stages and their final arrival time is measured using an arbiter. This arbiter either responds with a '0' if the top signal is faster or a '1' if the bottom signal is faster. The exact path the race signals take is determined by a challenge that is applied at the multiplexer stages. A challenge signal of '1' switches the top and bottom signals while a '0' does not switch the signals. This way 2^n challenges can be applied to an n-stage Arbiter PUF.

A delay stage i (consisting of two multiplexers) can be modeled by two parameters, the delay differences $\delta_{0,i}$ and $\delta_{1,i}$, corresponding to the delay difference added when the challenge bit is '0' and '1', respectively. If these two parameters are known for every delay stage, the final delay difference for every challenge can be computed by taking the switching into account: A '1' switches the top and bottom signal, which is equivalent to changing the sign of the delay difference. The delay difference ΔD_i after the i-th stage can therefore be computed recursively according to the following equation, where c_i denotes the i-th challenge bit:

$$\Delta D_i = \Delta D_{i-1} \cdot (-1)^{c_i} + \delta_{c_i,i} \tag{1}$$

The final response r is determined simply by the sign of ΔD_n, a positive delay difference results in a '1', a negative in a '0'. This recursive model requires $2 \cdot n$ parameters to describe an Arbiter PUF, however, there is a more efficient method to model an Arbiter PUF with only $n+1$ parameters.

$$\begin{aligned} w_1 &= \delta_{0,1} - \delta_{1,1} \\ w_i &= \delta_{0,i-1} + \delta_{1,i-1} + \delta_{0,i} - \delta_{1,i} \\ w_{n+1} &= \delta_{0,n} + \delta_{1,n} \end{aligned} \tag{2}$$

Additionally, the challenge vector $\vec{c} = c_1, ..., c_n$ has to be transformed into the feature vector $\vec{\Phi} = (\Phi_1, .., \Phi_{n+1}) \in \{-1, 1\}^{n+1}$:

$$\begin{aligned} \Phi_i &= \prod_{l=i}^{n} (-1)^{c_l} \quad \text{for} \quad 1 \leq i \leq n \\ \Phi_{n+1} &= 1 \end{aligned} \tag{3}$$

These transformations allow us to compute the delay difference ΔD_n with a simple scalar multiplication:

$$\Delta D_n = \vec{w}^{\intercal} \vec{\Phi} \tag{4}$$

This method is the common way to model an Arbiter PUF. We also used it in our experiments when performing machine learning attacks.

IV. Analysis

Previous research based on academic test chips and simulations showed that the delay distribution of an Arbiter PUF follows a Gaussian distribution. Similarly, the observed noise also seems to be Gaussian-distributed (e.g., see [15]). The main purpose of this paper is to verify the previous findings using a larger data set of PUF responses. In particular, we wanted to check if there are any "outliers" within this data set that deviate from this Gaussian model. Most of the analysis was performed on the raw response bits to determine the PUF's reliability, bias and uniqueness. We additionally performed machine learning attacks on all of the 998 tags. The goal of this analysis was to verify that all internal stage delays are independently and identically distributed with a Gaussian distribution. We wanted to determine if some stages dominate the response more than others or if single stages possibly dominate the PUF response.

In the first step we measured $51,200$ challenge and response pairs (CRPs) for functional 998 PUF tags (the communication with two tags did not work) twice and computed the reliability and uniqueness for each PUF tag. Histograms of the distribution of these values are depicted in Figure 2, with a fitted Gaussian distribution. Both the uniqueness as well as the reliability are Gaussian-distributed without any unusual outliers. Therefore our results clearly support earlier findings and no PUF instances with an unusually low uniqueness or reliability were found. The mean uniqueness these tags offered was very close to the ideal with 49.92% and a standard derivation of 0.234%. The mean reliability of the tags under nominal conditions was 86.7% with a standard derivation of 1.08% (temperature or voltage variations were not possible with our setup). Note that the low reliability stems from the fact that four bits are XORed in a 4-way PUF. If one considers the four XORs, the reliability of 86.7% for the 4-way PUF is equivalent to 96.3% reliability for a single Arbiter PUF response.

In the next experiment we investigated the internal delay differences of the PUF. The question we wanted to answer was whether some stages dominate the PUF responses more than

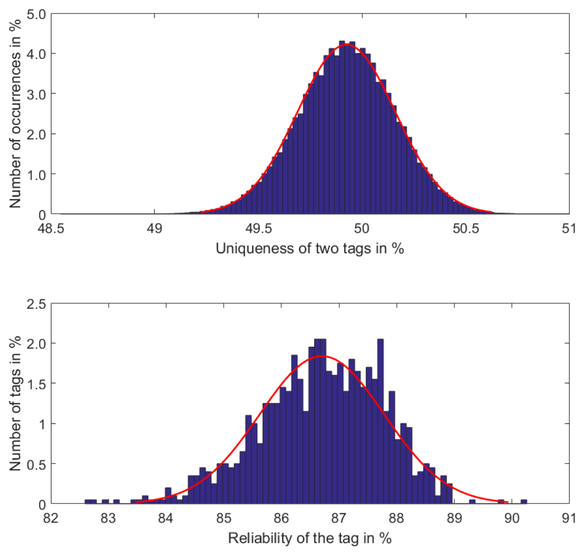

Fig. 2. Top: Histogram of the uniqueness values of the 998 PUF tags when compared with each other. The overlay is the corresponding Gaussian distribution ($\mu = 49.92\%, \sigma = 0.234\%$). Bottom: Histogram of the reliability values of the 998 PUF tags, overlaid with the corresponding Gaussian distribution ($\mu = 86.7\%, \sigma = 1.08\%$).

others and if there are outliers within their internal distribution. Measuring the internal delays of the PUFs on the RFID tags was not possible. Hence, we tried to approximate these internal delays using machine learning algorithms instead. Essentially, what we were doing is performing machine learning attacks to determine the delay vector $\vec{w}$. Attacking a 64-stage 4-way PUF using machine learning is relatively easy and many different machine learning algorithms can be used for this purpose, e.g. Logistic Regression, SVM or Evolution Strategies. We used CMA-ES machine learning as it was also used in [5] to learn each PUF instance, but other algorithms would have worked as well. It is important to note that these machine learning algorithms do not actually learn the exact internal delay differences but only their *ratio*. If we take a look at Equation 4, one can see that multiplying the vector $\vec{w}$ with a constant does not change the sign of the delay difference D_n and therefore neither that of the response. Since CMA-ES is non-deterministic, running the machine learning algorithms multiple times will therefore result in different vectors $\vec{w}$. However, the ratio of these vectors will be (nearly) identical.

Multiplying the vector $\vec{w}$ with -1 flips the sign of the delay difference and hence also the response bit. However, since four responses are XORed in a 4-way PUF, multiplying the vector $\vec{w}$ with -1 actually does not change the response bit: All four response bits that are XORed are flipped, but these flips cancel each other out. Therefore the CMA-ES algorithm is as likely to converge to $-\vec{w}$ as to $\vec{w}$.

Hence we only learn the ratio of the delay parameter $\vec{w}$ and we also cannot distinguish between $\vec{w}$ and $-\vec{w}$ in a 4-way PUF. Unfortunately the specific 4-way PUF used in the PUF tags further hampers our analysis: In the specific mixer function used in these tags, the first challenge $\vec{C_1} = c_{1,1}, c_{1,2}, c_{1,3}, .., c_{1,n}$ is the same as the second challenge in reverse order, i.e., $\vec{C_2} = c_{1,n}, c_{1,n-1}, .., c_{1,2}, c_{1,1}$. Similarly, the third challenge is identical to the fourth challenge in reverse order. For our machine learning algorithm, this has the peculiar effect that inverting the ordering of $\vec{w}$ results in the same response for the 4-way PUF.

All of this hampers our efforts to determine the internal delay differences of the individual PUFs since for the same PUF and the same measurements the non-deterministic machine learning attacks will result in a multitude of possible delay vectors $\vec{w}$ that are all equally likely to be correct. We therefore applied a post-processing to the delay vectors. In the first step we normalized each delay vector $\vec{w}$ found, i.e., we computed $\tilde{w_i} = w_i / \sum_{j=1}^{n} |w_i|$. After normalization, the machine learning algorithm can converge to four different solutions:

$$\begin{aligned} W_1 &= \tilde{w_1}, \tilde{w_2}, .., \tilde{w_n} \\ W_2 &= -\tilde{w_1}, -\tilde{w_2}, .., -\tilde{w_n} \\ W_3 &= \tilde{w_n}, \tilde{w_{n-1}}, .., \tilde{w_1} \\ W_4 &= -\tilde{w_n}, -\tilde{w_{n-1}}, .., -\tilde{w_1} \end{aligned} \quad (5)$$

To perform our machine learning attack we used all of the collected $51,200$ CRPs for each PUF. Note that even 1024 CRPs are enough to model the 4-way PUF with good accuracy [5]. We purposely used many more CRPs than necessary to obtain a very high accuracy and convergence rate in our machine learning runs. For every PUF we ran the CMA-ES algorithms until we had 50 successful attacks, each yielding a delay vector $\vec{w_i}$. A successful attack was defined by an accuracy larger than 80%. Once such a high accuracy is reached in a CMA-ES attack on an Arbiter PUF, the run usually converges close to the optimal solution. Hence, for this experiment we conducted nearly $50,000$ runs of the machine learning algorithm. In the first post-processing step these delay vector were normalized as described above. Figure 3 shows the 50 delay vectors $\vec{w_i}$ for one PUF as an example before normalization and after normalization was applied. One can clearly see that the machine learning runs converged to the four different solutions $W_1, .., W_4$ that are mirrored vertically at the x-axis and horizontally at stage 33. To get a single delay vector for each PUF, we performed another post-processing step: The result of the first machine learning run was chosen as the reference solution for each PUF and every other vector, if necessary, was reflected around the x-axis and vertically about stage 33. The result of this process is shown on the right of Figure 3. As one can see, after these post-processing steps there is only negligible variance in these vectors since the machine learning algorithm always converged to a near optimal solution. By computing the average of these 50 vectors we then derived the final delay vector for each PUF instance.

The reason why we computed these delay vectors was to determine if they indeed follow a Gaussian distribution or if some stages or PUFs exhibit unusual behavior that does not comply with the Gaussian assumption. For this we made a histogram of the delay distribution of this delay vector. Recall that we do not know if a delay value $\tilde{w_i}$ is indeed the delay for stage i or the mirrored solution, i.e., stage $n+1-i$. We therefore combined stage i and stage $n+1-i$ in the histogram depicted in Figure 4. All stage delays seem to indeed follow a Gaussian distribution centered around zero. Hence, our results clearly support earlier experimental and theoretical results. Not a single large outlier was detected among the 998 tags. The only stage with a different delay distribution than the other

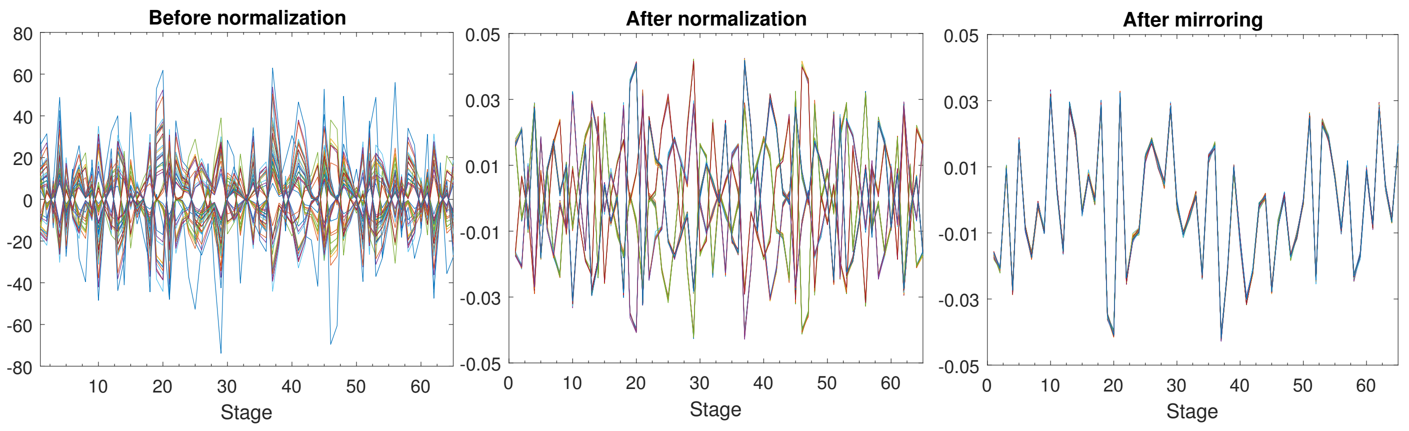

Fig. 3. The delay vectors of 50 successful CMA-ES machine learning runs for one PUF instance. On the left, the 50 delay vectors before normalization are plotted. In the middle, normalization of these 50 vectors was performed, resulting in four clearly visible solutions. On the right, mirroring was performed with the result that all 50 delay vectors are (nearly) identical.

stages is w_1/w_{65}. While this stage seems to also be Gaussian-distributed with a mean of 0, it clearly has a much smaller variance than the other stages.

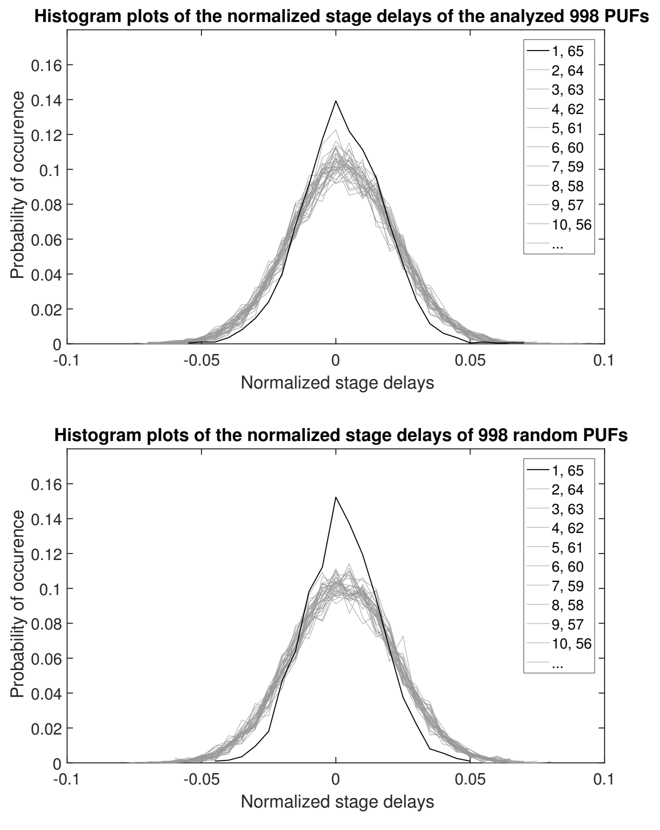

Fig. 4. Top: Histogram of the distribution of the normalized stage delays. Bottom: The same analysis repeated with simulated ideal PUF parameters that follow a Gaussian distribution.

However, this is to be expected: A closer look at Equation 2 reveals that $w_2, .., w_{n-1}$ consists of the sum of four (Gaussian-distributed) random variables, while w_1 and w_{65} only consist of the sum of two random values. Since the assumption is that all of these random values are independently and identically distributed (i.i.d.), the variance in w_1 and w_{65} is expected to be smaller than in the other stages. To verify this observation and our analysis in general, we generated 998 random PUFs in Matlab using an ideal i.i.d. Gaussian distribution. Using this software model, we generated $51,200$ CRPs and then performed the same machine learning analysis as with the real measurements. The result of this analysis is depicted in Figure 4 as a reference. As one can see, the distribution looks nearly identical to the one from the measurements. Only the variance of stage (1, 65) is actually smaller than what was observed in the measurement. One possible explanation for this is that in the delay differences in the last stage ($\delta_{0,65}, \delta_{1,65}$) the delay difference caused by the arbiter is included in addition to the delay difference of the multiplexer. This might explain why the variance is slightly higher than in the ideal model which does not consider delay differences caused by the arbiter. But the overall conclusion is that even in this large-scale analysis, the internal delay differences are Gaussian-distributed without any obvious outliers that do not comply with the Gaussian and i.i.d. assumption.

V. Conclusion

The main conclusion of this paper is simple: Even in a large-scale analysis based on 998 PUF tags, unusual outliers were observed in neither the reliability, uniqueness, nor the internal stage delay distributions. All results confirm the assumption that the stage delay values are Gaussian-distributed with approximately the same mean and variance (besides the last stage). We did not find any PUF instance that was significantly weaker than the others. This shows that — besides the problem of machine learning attacks — the Arbiter PUF has indeed very good characteristics when implemented with care.

As a second conclusion, we would like to note that the machine learning based analysis to determine the delay distribution of the PUF is a very useful tool that should be considered in future work when analyzing strong PUFs. A simple uniqueness and reliability analysis might not reveal if, for example, due to implementation issues, some stages are biased or effects exist that cause deviation from the Gaussian behavior. Therefore we encourage researchers to also include such a machine learning based analysis when examining PUF test chips.

References

[1] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, 2002.

[2] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 148–160. ACM, 2002.

[3] G. E. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 9–14. ACM, 2007.

[4] 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 2010, pages 237–249. ACM, 2010.

[5] G.T Becker. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, CHES 2015, pages 535–555. Springer, 2015.

[6] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas. A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, pages 176–179. IEEE, 2004.

[7] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. PUF Modeling Attacks on Simulated and Silicon Data. *IEEE Trans. Information Forensics and Security*, 8(11):1876–1891, 2013.

[8] G. Hospodar, R. Maes, and I. Verbauwhede. Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling Poses Strict Bounds on Usability. In *2012 IEEE International Workshop on Information Forensics and Security*, WIFS '12, pages 37–42, 2012.

[9] M. D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede. A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication. *IEEE Transactions on Multi-Scale Computing Systems*, PP(99):1–1, 2016.

[10] S.Katzenbeisser, Ü. Koçabas, V. Rozic, A.-R. Sadeghi, I.Verbauwhede, and C. Wachsmann. PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon. In *International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '12, pages 283–301, 2012.

[11] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. van der Leest. Experimental Evaluation of Physically Unclonable Functions in 65 nm CMOS. In *Proceedings of the 38th European Solid-State Circuit conference*, ESSCIRC 2012, pages 486–489, 2012.

[12] L. Lin, S. Srivathsa, D. K. Krishnappa, P. Shabadi, and W. Burleson. Design and Validation of Arbiter-Based PUFs for Sub-45-nm Low-Power Security Applications. *IEEE Transactions on Information Forensics and Security*, 7(4):1394–1403, 2012.

[13] S. V. S. Avvaru, C. Zhou, S. Satapathy, Y. Lao, C. H. Kim, and K. K. Parhi. Estimating Delay Differences of Arbiter PUFs Using Silicon Data. In *2016 Design, Automation & Test in Europe Conference & Exhibition*, DATE 2016, pages 543–546, 2016.

[14] M. Majzoobi, F.Koushanfar, and M.Potkonjak. Lightweight Secure PUFs. In *2008 International Conference on Computer-Aided Design*, ICCAD 2008, pages 670–673, 2008.

[15] J. Delvaux and I. Verbauwhede. Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise. In *IEEE International Symposium on Hardware-Oriented Security and Trust*, HOST 2013, pages 137–142, 2013.