Security Analysis of Delay-Based Strong PUFs with Multiple Delay Lines

Anita Aghaie*, Amir Moradi†, Johannes Tobisch‡ and Nils Wisiol§
* Ruhr University Bochum, Horst Görtz Institute for IT-Security, Bochum, Germany (e-mail: anita.aghaie@rub.de)
† Universität zu Köln, Germany (e-mail: amir.moradi@uni-koeln.de)
‡ Max Planck Institute for Security and Privacy, Bochum, Germany (e-mail: johannes.tobisch@mpi-sp.org)
§ Technische Universität Berlin, Germany (e-mail: nils.wisiol@tu-berlin.de)

***Abstract*—Using a novel circuit design, we investigate if the modeling-resistance of delay-based, CMOS-compatible strong PUFs can be increased by the usage of multiple delay lines. Studying a circuit inspired by the Arbiter PUF, but using four instead of merely two delay lines, we obtain evidence showing that the usage of many delay lines does not significantly increase the security of the strong PUF circuit. Based on our findings, we suggest future research directions.**

I. Introduction

Since the introduction of delay-based strong Physical Unclonable Functions (PUFs), several generalizations of the original design [1] have been considered to increase security against modeling attacks. In this work, we add to this line of research by considering delay-based strong PUFs composed of multiple delay lines.

Our results extend the literature on the question if the machine learning resilience of delay based PUFs can be increased. The design [2] and attack [3], [4] of the Interpose PUF as well as recent attacks on other ad-hoc composite PUF designs [5] have demonstrated that composition of building blocks (alone) may not be sufficient to stop modeling attacks. In follow-up work, the PUF-G Framework [6] and attacks based on neural networks [3] were proposed to assess the security of composite designs. This work, on the other hand, takes a different approach and studies if the machine learning resilience of delay-based PUFs can be increased by modifying the building block, as opposed to embedding it in a larger design. We answer this question to the negative for the novel concept of delay-based PUFs based on multiple delay lines and give some insight in the modeling of delay-based PUFs.

In more detail, our contributions are as follows. First, we present an extension of the Arbiter PUF design inherently different from other Arbiter PUF variants present in the literature. Second, we apply the Arbiter PUF's additive delay model to our design and demonstrate an exponential blowup in required modeling parameters, which mitigates modeling attacks based on this mathematical model. Third, we empirically show that naive modeling attacks based on general-purpose neural networks also cannot be used to model our novel PUF design. Fourth, we extend our analysis to obtain a model that circumvents the exponential blowup in parameter count and demonstrate successful modeling attacks. We further argue that this attack can be used as evidence that PUFs built from multiple delay lines cannot mitigate modeling attacks reliably. We conclude with suggestions for future research for which we do not have evidence that our attack strategy will work.

II. Design

We consider *Multiple Permuted Delay Line PUFs* (MPDL PUFs), a generalization of the Arbiter PUF to a circuit that uses more than two delay lines. An MPDL PUF that has m delay lines can have $\lceil \log_2 m \rceil$ output bits $o_0, \dots o_{\lceil \log_2 m \rceil - 1}$, encoding the index of the fastest delay line. The fastest signal is detected using a circuit similar to the arbiter element of the Arbiter PUF. Alternatively, an MPDL PUF can have exactly one output bit o_s which is defined as the XOR of above output bits, i.e. $o_s = \bigoplus_{l=0}^{\lceil \log_2 m \rceil - 1} o_l$. In an MPDL PUF, the delay lines can be switched by 2-to-2 multiplexer elements controlled by challenge bits as well as by a fixed permutation that is given by the design specification.

We first restrict our attention to a specific instance of the MPDL PUF, the *Beli PUF*: It uses four delay lines, i.e. two output bits o_0, o_1 and one output bits $o_s = o_1 \oplus o_2$, respectively. To achieve a symmetric structure on n input bits, we arrange Beli PUF into $n/2$ blocks of four delay lines, using two challenge bits each. For filling in the permutations $\mathcal{P}_1, \dots, \mathcal{P}_{n/2}$, we opt to use $\mathcal{P} = (i_0, i_1, i_2, i_3) \mapsto (i_0, i_2, i_1, i_3)$. By exchanging the top two lines with the bottom two lines, this permutation guarantees that our design does not degenerate into a design reminiscent of two Arbiter PUFs. Using the same permutation in all places simplifies the analysis, but is without loss of generality. The detailed response behavior of the Beli PUF is shown in Table I, where we denote the delays on the four delay lines with $d_0, \dots, d_3$. The schematics of Beli PUF are shown in Figure II.

As we will see, none of the specifics of Beli PUF are essential to our security analysis and results can be applied to MPDL PUFs in general with little restriction.

For Beli PUF, and for MPDL PUFs in general, XOR-versions similar to XOR Arbiter PUFs are conceivable, where the same design is instantiated multiple times and the outputs are defined as the XOR of output bits across the individual instances. We denote such Beli PUFs with k individual instances 1-bit k-XOR Beli PUF and 2-bit k-XOR Beli PUF, respectively.

TABLE I

Output of Beli PUF where $\theta_{i,j} = \operatorname{sgn}(d_i - d_j)$ and * denotes "any value", meaning that the output values in that row are independent of the value in the starred column.

fastest	outputs			individual signals					
	o_0	o_1	o_s	$\theta_{0,1}$	$\theta_{0,2}$	$\theta_{0,3}$	$\theta_{1,2}$	$\theta_{1,3}$	$\theta_{2,3}$
d_0	1	1	1	-1	-1	-1	*	*	*
d_1	1	-1	-1	1	*	*	-1	-1	*
d_2	-1	1	-1	*	1	*	1	*	-1
d_3	-1	-1	1	*	*	1	*	1	1

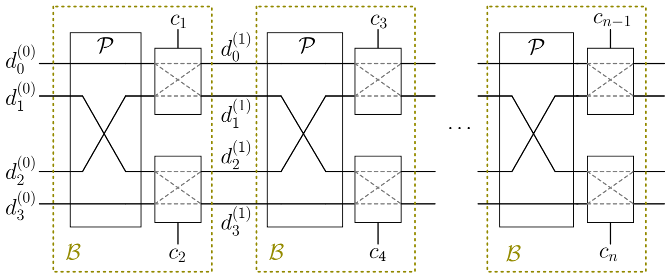

Fig. 1. Beli PUF structure. Permutation network of the Beli PUF. TODO change from 0-index to 1-index

III. Model Based on Additive Delay Model

For MPDL PUFs, models based on the additive delay model similar to the Arbiter PUF [7] can be derived. We do so concretely for the Beli PUF.

Let $d_l^{(j)}$ denote the delay after the j-th stage ($1 \leq j \leq n/2$) on line l. Let $c \in \{-1, 1\}^n$ be the challenge given to Beli PUF. In the j-th stage, we use c_{2j-1} for the top switch and c_{2j} for the bottom switch. We denote the internal delays of the switch box that receives challenge bit c_i with d_i^{TT},d_i^{BT}, d_i^{BB}, d_i^{TB} for the delay introduced by a signal traveling from top input to top output, bottom input to top output, bottom input to bottom output, and top input to bottom output, respectively.

As the top switch receives the input delays $d_0^{(j-1)}$ and $d_1^{(j-1)}$, the challenge bit c_{2j-1} and has the internal delays $d_{2j-1}^{\mathrm{TT}}, d_{2j-1}^{\mathrm{BT}}, d_{2j-1}^{\mathrm{BB}}, d_{2j-1}^{\mathrm{TB}}$, we obtain for the output delays of the top switch that

$$d_0^{(j)} = \begin{cases} d_0^{(j-1)} + d_{2j-1}^{\mathrm{TT}} & (c_{2j-1} = -1), \\ d_1^{(j-1)} + d_{2j-1}^{\mathrm{BT}} & (c_{2j-1} = 1), \end{cases}$$

$$d_1^{(j)} = \begin{cases} d_1^{(j-1)} + d_{2j-1}^{\mathrm{BB}} & (c_{2j-1} = -1), \\ d_0^{(j-1)} + d_{2j-1}^{\mathrm{TB}} & (c_{2j-1} = 1). \end{cases}$$

Similar formulae can be derived for the bottom switch in the j-th stage. By using the fact that for $c_i \in \{-1, 1\}$, we have $c_i = -1 \iff 1/2 + 1/2 c_i = 0$ and $c_i = 1 \iff 1/2 - 1/2 c_i = 0$, we can write the output delays of the j-th stage, $d_0^{(j)}, d_1^{(j)}, d_2^{(j)}, d_3^{(j)}$ without case distinction as

$$\begin{aligned} d_0^{(j)} &= \frac{1 - c_i}{2}\left(d_0^{j-1} + d_{2j-1}^{\mathrm{TT}}\right) + \frac{1 + c_i}{2}\left(d_1^{j-1} + d_{2j-1}^{\mathrm{BT}}\right), \\ d_1^{(j)} &= \frac{1 - c_i}{2}\left(d_1^{j-1} + d_{2j-1}^{\mathrm{BB}}\right) + \frac{1 + c_i}{2}\left(d_0^{j-1} + d_{2j-1}^{\mathrm{TB}}\right), \\ d_2^{(j)} &= \frac{1 - c_i}{2}\left(d_2^{j-1} + d_{2j}^{\mathrm{TT}}\right) + \frac{1 + c_i}{2}\left(d_3^{j-1} + d_{2j}^{\mathrm{BT}}\right), \\ d_3^{(j)} &= \frac{1 - c_i}{2}\left(d_3^{j-1} + d_{2j}^{\mathrm{BB}}\right) + \frac{1 + c_i}{2}\left(d_2^{j-1} + d_{2j}^{\mathrm{TB}}\right). \end{aligned}$$

To obtain expressions for the delay in each line, we iteratively replace $d_i^{(j-1)}$ in $d_i^{(n)}$. However, in contrast to the similar procedure on the Arbiter PUF circuit, in the Beli PUF model this leads to expressions of exponential length in n for $d_i^{(n)} - d_j^{(n)}$. This behavior is caused by the introduction of the permutation $\mathcal{P}$. (If choosing $\mathcal{P}$ as the identity permutation, we obtain two Arbiter PUFs, and the Arbiter PUF additive delay model with a linear number of terms applies.)

To obtain a complete model of the Beli PUF from the expressions for d_0, d_1, d_2, d_3, we define the delay differences $\theta_{i,j} = \operatorname{sgn}(d_i - d_j)$ and observe that $\theta_{i,j} = 1$ if and only if $d_i > d_j$. The output bits o_0 and o_1 can be computed as a Boolean function of $\theta_{0,2}, \theta_{0,3}, \theta_{1,2}, \theta_{1,3}, \theta_{2,3}$ and $\theta_{0,1}, \theta_{0,3}, \theta_{1,2}, \theta_{1,3}, \theta_{2,3}$, respectively, based on the observations on $\theta_{i,j}$ given in Table I. For output bit o_0, we find that d_2 has the lowest delay if and only if $\max\{-\theta_{0,2}, -\theta_{1,2}, \theta_{2,3}\} = -1$ ("2 faster than 0 and 2 faster than 1 and 2 faster than 3"). d_3 has the lowest delay if and only if $\max\{-\theta_{0,3}, -\theta_{1,3}, -\theta_{2,3}\} = -1$ ("3 faster than 0 and 3 faster than 1 and 3 faster than 2"). As o_0 is -1 if and only if either d_2 or d_3 has the lowest delay, we obtain $o_1 =$

$$\min\left\{\max\{-\theta_{0,2}, -\theta_{1,2}, \theta_{2,3}\}, \max\{-\theta_{0,3}, -\theta_{1,3}, -\theta_{2,3}\}\right\}$$

A similar formulae can be derived for o_1, as o_1 is -1 if and only if either d_1 or d_2 have the lowest delay. Finally, in the case of 1-bit Beli PUF, the output can be modeled as $o_s = o_0 \cdot o_1$.

We empirically evaluated the length of the model expressions using Sage for $n \leq 18$, generating expressions for Beli PUF's d_0, d_1, and $d_0 - d_1$. (By symmetry, these findings extend to other delay lines in Beli PUF as well.) In Figure 2, a comparison of length of expressions with the Arbiter PUF model and the algebraic maximum length is given.

Due to the exponential size, it is infeasible for an attacker to recover all coefficients of this model and thus bars them from using the Beli PUF additive delay model for modeling attacks, which seemingly gives rise to hope that Beli PUF could resist modeling attacks.

IV. Implementation and Metrics

Beli PUFs can be implemented on FPGAs in a way similar to the Arbiter PUF. However, to obtain the final response bit(s) of the Beli PUF, a variant of the arbiter element is needed. One option is to implement six arbiter elements, as motivated by the model equations for o_0 and o_1 given above, where each arbiter element determines the value of one of the involved $\theta_{i,j}$. Due to noise, it is possible that the determined $\theta_{i,j}$ values

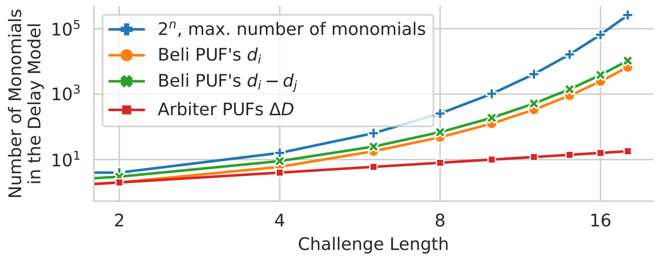

Fig. 2. Number of monomials for the Beli PUF's d_0, d_1 and their difference $d_0 - d_1$, as well as for the Arbiter PUF delay model, when written as multivariate polynomial of the challenge.

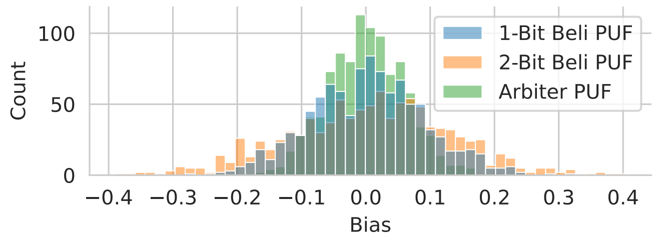

Fig. 3. Distribution of the bias of 1-Bit Beli PUF, 2-Bit Beli PUF, and Arbiter PUF for 1000 instanes each. Each bias has been estimated by querying 1000 uniform random, noise-free challenges.

are contradicting, in which case the final response of Beli PUF also may be noisy.

We compare the bias and bit sensitivity of Beli PUF simulations with metrics obtained for the Arbiter PUF simulation. As shown in Figure 3, the distribution of the bias of Beli PUF and Arbiter PUF is close to unbiased with only some variance. However, observe that the Arbiter PUF has a little smaller bias variance. For both designs, the bias can be further reduced by using an XOR-variant.

The bit sensitivity of Beli PUF is generally reduced and below the desirable value of $1/2$, which can be explained by the "minimum" operation on the four delay lines, as changed delay values in lines that do not involve the shortest delay do not affect the PUF's output. However, the bit sensitivity shows less variance across the bit position on the challenge than it is the case for the Arbiter PUF. The distributions of bit sensitivities for different challenge bit positions is displayed in Figure 4.

To measure the reliability of the Beli PUF, we first created several FPGA-implementations and then chose the candidate that showed the least bias across 100 tested chips. The reliability has then been measured by generating 1 million uniform random challenges and querying each FPGA on this challenge set 11 times. The selected Beli PUF implementation enjoys a high reliability of 98% on average, with little variance.

V. Generic Neural Network Attack

One way to avoid the exponential number of parameters in the additive delay model of the Beli PUF is to use general-

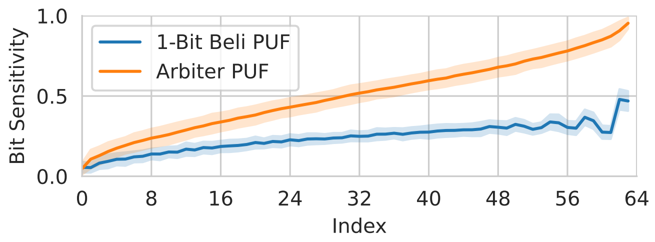

Fig. 4. Probability that flipping the input bit at the given index will change the PUF's output bit, as empirically estimated for 1-Bit Beli PUF and Arbiter PUF. Estimations are based on 100 simulated instances instances, where the probability for each index was approximated using 1000 uniform random noise-free challenges. The shaded area indicates the standard deviation interval of the bit sensitivity.

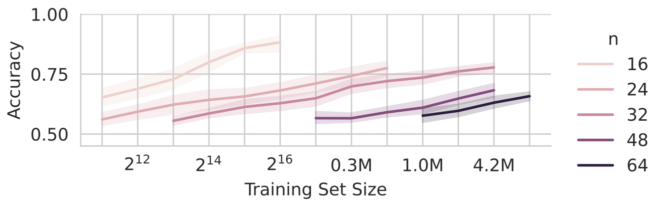

Fig. 5. Prediction accuracy of a general-purpose MLP modeling attack on 1-bit Beli PUF, simulated with various challenge sizes for 10 instances using noise-free CRP data sets of size N. The shaded area indicates a 95% confidence interval across attacked PUF instances.

purpose neural networks for modeling. Such networks have been successfully used in the modeling attacks on XOR Arbiter PUFs and on the Interpose PUF [7].

For attacking Beli PUF, we employed a Multilayer-Perceptron (MLP) Model which uses the ReLU activation function on its four hidden layers. Each hidden layer uses 256 neurons.

The resulting model accuracy indicates that, while Beli PUFs of smaller challenge lengths such as 16 bit can be modeled with high accuracy, the prediction accuracy of the modeling attack for 24 bit Beli PUFs stays below 80%, as shown in Figure 5. Increasing the training set size does not improve the result, which indicates that the used generic model is not suitable to efficiently model the Beli PUF.

For comparison, similar neural network attacks on 4-XOR 64-bit Arbiter PUFs require merely 150,000 CRPs to achieve near-perfect prediction accuracy.

VI. Efficient Modeling Attack

In contrast to the generic attack, here we motivate an attack based on a variant of the physically inspired Beli PUF model (Section §IV). While we demonstrate the attack specifically for the Beli PUF design, the idea is applicable to all MPDL PUFs.

In any MPDL PUF, the paths the signals take through the circuit are fully defined by a given challenge. For a given path P_i, the total signal delay can be computed from the physical parameters $d_j^{\mathrm{TT}}, d_j^{\mathrm{TB}}, d_j^{\mathrm{BT}}, d_j^{\mathrm{BB}}$ as $d_i = \sum_{(X,j)\in P} d_j^X$.

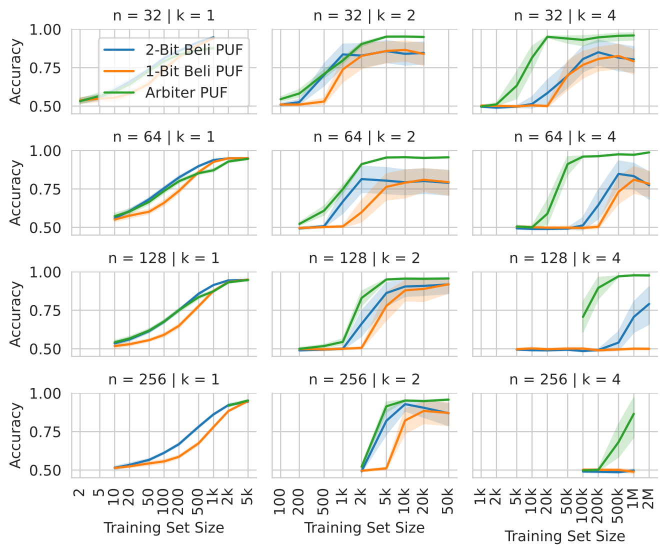

Fig. 6. Accuracy obtained when attacking the 1-bit Beli PUF, 2-bit Beli PUF, and XOR APUF. For each combination of challenge length n, number of CRPs N, number of XORs k, and PUF type, we ran at least 10 attacks with individual PUF simulation and attack initialization each. The accuracy shown is evaluated on an independent 1,000 CRPs test set. The shaded area indicates a 95% confidence interval across attacked PUF instances.

This computation can be formalized as a dot product of a delay-indicator vector $x_i \in \{0,1\}^{4n}$, derived from the challenge c using the design specification of the circuit, as $d_i = \langle d, x_i \rangle$, where d is a list of all $4n$ physical parameters, $d = \left(d_1^{\mathrm{TT}}, d_1^{\mathrm{TB}}, \ldots, d_2^{\mathrm{TT}}, \ldots, d_n^{\mathrm{BB}}\right)$. This enables us to write all signal delays of the MPDL PUF as a function linear in the space of the the physical parameters. Using this observation, we can avoid the exponentially long model equations of Section §III. To model the Beli PUF response, we then use

$$\begin{aligned} o_0 &= \operatorname{sgn}\left(\min\{d_2, d_3\} - \min\{d_0, d_1\}\right), \\ o_1 &= \operatorname{sgn}\left(\min\{d_1, d_3\} - \min\{d_0, d_2\}\right), \quad \text{and} \\ o_s &= \operatorname{sgn}\left(\min\{d_1, d_2\} - \min\{d_0, d_3\}\right). \end{aligned}$$

These equations allow us to define a neural network that is able to model Beli PUF. Given the delay-indicator vectors derived from the challenges, this network can be trained to closely model Beli PUF responses.

Like done in attacks on the XOR Arbiter PUF, attacks on XOR Beli PUFs can be applied by adjusting the model to compute the product of the individual model output bits.

To confirm the validity of this Beli PUF model, we collected 100,000 CRPs from a FPGA implementation of a 1- and 2-bit Beli PUF with 64 challenge bits and trained models as outlined above, using 99,000 CRPs. The resulting models showed correct prediction in roughly 88% of cases on a test set of 1,000 CRPs. This provides practical evidence that our model and implementation behave similarly.

Using Beli PUF simulations, we ran attacks on 1-bit and 2-bit Beli PUFs as well as their XORed variants. As a baseline for comparison, we run the state-of-the-art attacks on XOR Arbiter PUFs, implemented using the same software stack. All attacks were conducted for 32, 64, 128, and 256-bit challenges.

The detailed results of our attacks are shown in Figure 6. Compared to the XOR Arbiter PUF baseline, our attacks on 1-bit and 2-bit Beli PUF generally show an increased data complexity. As expected, the results also indicate that slightly more data is required to train a model for the 1-bit Beli PUF, compared to its 2-bit version. Nevertheless, while differences in the data complexity exist, our results show that the Beli PUF cannot be expected to achieve a significant advantage over the attacker in terms of modeling attacks.

Reviewing our modeling and attack methodology, there is no indication that this kind of modeling attack strategy cannot be applied to other MPDL PUFs. Our results thus show that PUFs based on multiple delay lines cannot be expected to significantly increase security compared to XOR Arbiter PUFs.

VII. Future Work

The successful modeling attack in this paper demonstrates that adding multiple delay lines to a delay-based strong PUF design alone cannot increase security against modeling attacks. Hence, exploring other variations is necessary. In this work, we did not consider adding delay line switches which are controlled by internally generated bits, as done in the Feed-Forward version of the Arbiter PUF [1]. Seeing if Feed-Forward bits together with multiple delay lines can increase modeling attack resilience remains an open question.

We also did not consider composite PUF designs that use MPDL PUFs as building blocks and may benefit from the adapted bit sensitivity or high reliability. To facilitate future work, all source code used in this paper will be published under a free license.

References

[1] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, Sep. 2004. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.805

[2] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 243–290, Aug. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8351

[3] N. Wisiol, C. Mühl, N. Pirnay, P. H. Nguyen, M. Margraf, J.-P. Seifert, M. van Dijk, and U. Rührmair, "Splitting the Interpose PUF: A Novel Modeling Attack Strategy," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–120, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8584

[4] J. Tobisch, A. Aghaie, and G. T. Becker, "Combining Optimization Objectives: New Modeling Attacks on Strong PUFs," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 357–389, Feb. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8798

[5] J. Delvaux, "Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2043–2058, Aug. 2019.

[6] D. Chatterjee, D. Mukhopadhyay, and A. Hazra, "PUF-G: A CAD Framework for Automated Assessment of Provable Learnability from Formal PUF Representations," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, Nov. 2020, pp. 1–9.

[7] N. Wisiol, K. T. Mursi, J.-P. Seifert, and Y. Zhuang, "Neural-Network-Based Modeling Attacks on XOR Arbiter PUFs Revisited," Tech. Rep. 555, 2021. [Online]. Available: https://eprint.iacr.org/2021/555